National Technical University of Athens

School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Computer Vision, Speech Communication and Signal Processing Group

Automatic Music Synthesis using Neural Networks and Machine Learning Techniques

DIPLOMA THESIS

of

Danae-Nikoleta Charitou

Supervisor: Petros Maragos
Professor NTUA

Co-supervisor: Athanasia Zlatintsi
Postdoctoral Researcher NTUA

Athens, October 2022

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Computer Vision, Speech Communication and Signal Processing Group

Automatic Music Synthesis using Neural Networks and Machine Learning Techniques

DIPLOMA THESIS

of

Danae-Nikoleta Charitou

Supervisor: Petros Maragos
Professor NTUA

Co-supervisor: Athanasia Zlatintsi
Postdoctoral Researcher NTUA

Approved by the examining committee on 31st October, 2022:

.....	.....	.....
Petros Maragos	Gerasimos Potamianos	Athanasios Rontogiannis
Professor NTUA	Associate Professor UTH	Associate Professor NTUA

Athens, October 2022

.....
Danae-Nikoleta Charitou
Electrical and Computer Engineer, NTUA

to my family

Abstract

Machine Learning has flourished over the last few years, resulting in the inevitable inclusion of Artificial Intelligence into our everyday life. The emulation of human mental acuity, achieved by Artificial Neural Networks, has made overwhelming progress concerning fundamental or even instinctive intellectual processes. On this ground, the interest of research community is now focused on more creative and generative functionalities, one of those being music synthesis. The process of creating musical pieces is considered a higher mental function that still remains unfathomed, even at a non-computational level. A musical composition constitutes a form of expressing various attributes, such as knowledge, experience, ideas, emotions. Therefore, this involving notion of subjectivity makes the problem of automatic music generation particularly complex.

Our approach in the research field of automatic music synthesis is based on Generative Adversarial Networks, one of the most prominent system architectures in the area of generative modeling with several applications in comparable problems of different data types, such as image, video and text. Initially, we examine the task of polyphonic music synthesis for multiple tracks, in terms of generation from scratch, that is without any human input or supplementary information. Afterwards, we extend our model in a human-AI cooperative framework by exploring the task of accompaniment generation, namely the generation process of the musical part which provides the rhythmic and/or harmonic support for the melody or main themes of a song, composed by human. The experimentation over the structure of individual networks, the architecture of the whole system, the training algorithm and various parameters with respect to the generated musical samples, allows us to investigate different aspects of the procedure that an Artificial Intelligence model follows in order to compose music, demonstrating at the same time the impact of the aforementioned components on the produced musical result. Finally, a set of objective metrics concerning musical features is established, while a user study is also conducted in the context of subjective evaluation. In this way, we show that our model is capable of creating novel aesthetic music characterized by tonal, temporal and harmonic structure, achieving competitive performance in comparison with the baseline implementation.

Keywords — Music, Synthesis, Machine Learning, Artificial Intelligence, Generative Adversarial Networks, Generation from scratch, Accompaniment Generation

Περίληψη

Τα τελευταία χρόνια ο κλάδος της Μηχανικής Μάθησης αναπτύσσεται με ραγδαίους ρυθμούς, καθιστώντας αναπόφευκτη την ενσωμάτωση της Τεχνητής Νοημοσύνης σε ποικίλες πτυχές της ανθρώπινης καθημερινότητας. Η μοντελοποίηση της ανθρώπινης ευφυΐας μέσω της δημιουργίας Τεχνητών Νευρωνικών Δικτύων σημείωσε σημαντική πρόοδο σε επίπεδο βασικών, ίσως και ενστικτωδών για τον άνθρωπο, λειτουργιών, στρέφοντας έτσι το ενδιαφέρον της επιστημονικής κοινότητας στην προσπάθεια προσέγγισης πιο παραγωγικών και δημιουργικών διαδικασιών. Μια εξ αυτών είναι και η σύνθεση μουσικής. Πρόκειται για μία ανώτερη νοητική λειτουργία, η οποία ακόμη και σε μη υπολογιστικό επίπεδο θεωρείται ανεξερεύνητη. Ένα μουσικό κομμάτι αποτελεί μια μορφή έκφρασης διαφόρων στοιχείων, όπως οι γνώσεις, η εμπειρία, τα ακούσματα, τα συναισθήματα, οπότε αυτή η έννοια της υποκειμενικότητας που εμπλέκεται, καθιστά το πρόβλημα δημιουργίας μουσικής με αυτόματο τρόπο ιδιαίτερα περίπλοκο.

Η δική μας προσέγγιση στην ερευνητική περιοχή της αυτόματης σύνθεσης μουσικής στηρίζεται στα Παραγωγικά Αντιπαραθετικά Δίκτυα (Generative Adversarial Networks), τα οποία αποτελούν την κατ' εξοχήν αρχιτεκτονική υπολογιστικού συστήματος όσον αφορά tasks που περιλαμβάνουν κάποια διαδικασία δημιουργίας και είναι ευρέως διαδεδομένα σε αντίστοιχα προβλήματα διαφορετικού τύπου δεδομένων, όπως η εικόνα, το βίντεο και το κείμενο. Βάσει, λοιπόν, του εν λόγω μοντέλου, εξετάζουμε σε πρώτο στάδιο την αυτόματη παραγωγή πολυφωνικής μουσικής για πολλαπλά όργανα χωρίς τη χρήση ανθρώπινης εισόδου ή συμπληρωματικών δεδομένων. Στην συνέχεια, επεκτείνουμε το σύστημά μας σε ένα συνεργατικό πλαίσιο ανθρώπου-μηχανής, μελετώντας τη διαδικασία αυτοματοποιημένης σύνθεσης του μουσικού τμήματος που αποτελεί την συνοδεία μιας κύριας μελωδικής γραμμής προερχόμενης από αυθεντική μουσική σύνθεση. Πειραματιζόμενοι με την δομή των επιμέρους δικτύων, την αρχιτεκτονική του συνολικού συστήματος, τον αλγόριθμο εκπαίδευσης αλλά και διάφορες παραμέτρους που χαρακτηρίζουν τα παραγόμενα μουσικά δείγματα, εξερευνούμε διαφορετικές πτυχές του τρόπου με τον οποίο μπορεί ένα μοντέλο Τεχνητής Νοημοσύνης να συνθέτει αυτόνομα μουσική, καταδεικνύωντας συγχρόνως την επίδραση αυτών των στοιχείων στο ακουστικό αποτέλεσμα. Τέλος, αξιοποιώντας ένα σύστημα ποσοτικών μετρικών που αφορούν μουσικά γνωρίσματα αλλά και διεξάγοντας μια ποιοτική μελέτη με την μορφή ακουστικού πειράματος, συμπεραίνουμε ότι το σύστημά μας κατέχει την δυνατότητα δημιουργίας νέων μουσικών συνθέσεων που χαρακτηρίζονται από τονική, χρονική και αρμονική δομή, επιτυγχάνοντας συγχρόνως ανταγωνιστικά αποτελέσματα συγκριτικά με την baseline αρχιτεκτονική.

Λέξεις Κλειδιά — Μουσική, Σύνθεση, Μηχανική Μάθηση, Τεχνητή Νοημοσύνη, Παραγωγικά Αντιπαραθετικά Δίκτυα

Acknowledgments

Με την παρούσα Διπλωματική Εργασία ολοκληρώνεται ο εξαετής κύκλος φοίτησής μου στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου και συνάμα η έρευνα που πραγματοποίησα στο Εργαστήριο Εργαστήριο Όρασης Υπολογιστών, Επικοινωνίας Λόγου και Επεξεργασίας Σήματος τον τελευταίο ενάμιση χρόνο. Με την αφορμή λοιπόν που μου δίνεται, θα ήθελα να ευχαριστήσω:

Τον επιβλέποντα καθηγητή κ. Πέτρο Μαραγκό, ο οποίος, μέσω των εποικοδομητικών του διαλέξεων στο πλαίσιο των προπτυχιακών μαθημάτων αλλά και του γενικότερου ερευνητικού του έργου, με ενέπνευσε να ασχοληθώ ενεργά με το συγκεκριμένο αντικείμενο και μου προσέφερε την ευκαιρία αλλά και την απαραίτητη καθοδήγηση για την εκπόνηση της διπλωματικής μου εργασίας στο εργαστήριο του.

Ιδιαιτέρως τη μεταδιδακτορική ερευνήτρια Δρ. Νάνσυ Ζλατίντση και το διδακτορικό φοιτητή Χρήστο Γαρούφη για την καθοδήγηση, τις συμβουλές, τις ευχάριστες συζητήσεις και την πολύτιμη βοήθεια που μου προσέφεραν καθ' όλη την διάρκεια της συνεργασίας μας.

Τον επιστημονικό συνεργάτη Κοσμά Κρίτση για τις συμβουλές και την βοήθεια που προσέφερε, τον διδακτορικό φοιτητή Παναγιώτη-Παρασκευά Φιλντίση για την τεχνική υποστήριξη αλλά και όλους όσους συμμετείχαν στο ακουστικό πείραμα της ποιοτικής μελέτης που διεξήχθη.

Τέλος, την οικογένειά μου και τους φίλους μου για την αμέριστη στήριξη που μου προσέφεραν καθ' όλη την διάρκεια της ακαδημαϊκής μου σταδιοδρομίας.

Δανάη-Νικολέτα Χαρίτου
Οκτώβριος 2022

Contents

List of Figures

List of Tables

Chapter 0

Extended Greek Abstract

0.1 Εισαγωγή

0.1.1 Ορισμός Ερευνητικού Προβλήματος

Τα τελευταία χρόνια το επιστημονικό πεδίο της Μηχανικής Μάθησης αναπτύσσεται με ραγδαίους ρυθμούς, καθιστώντας αναπόφευκτη την ενσωμάτωση της Τεχνητής Νοημοσύνης σε ποικίλες πτυχές της ανθρώπινης καθημερινότητας. Η μοντελοποίηση της ανθρώπινης ευφυΐας μέσω της σχεδίασης και υλοποίησης υπολογιστικών συστημάτων που δρουν αυτόνομα σημείωσε σημαντική πρόοδο σε επίπεδο βασικών, ίσως και ενστικτωδών για τον άνθρωπο, λειτουργιών, στρέφοντας έτσι το ενδιαφέρον της επιστημονικής κοινότητας στην προσπάθεια προσέγγισης πιο παραγωγικών και δημιουργικών διαδικασιών. Μια εξ αυτών είναι και η σύνθεση μουσικής, η οποία αποτελεί και το αντικείμενο της παρούσας Διπλωματικής Εργασίας.

Αυτόματη Σύνθεση Μουσικής

Η διαδικασία δημιουργίας νέων μουσικών κομματιών με αυτόματο τρόπο, δηλαδή με την ελάχιστη δυνατή ανθρώπινη παρέμβαση.

Το βασικό κίνητρο πίσω από την εφαρμογή τεχνικών Μηχανικής Μάθησης για την επίλυση του συγκεκριμένου ερευνητικού προβλήματος, έναντι άλλων μεθόδων που στηρίζονται κατά κύριο λόγο στην ανάπτυξη κανόνων και ειδικευμένων γραμματικών, έγκειται στην ικανότητα των μοντέλων Τεχνητής Νοημοσύνης να αντιμετωπίζουν με έναν αγνωστικό τρόπο την διαδικασία σύνθεσης και συγχρόνως την εγγενώς πολύπλοκη δομή της μουσικής πληροφορίας. Αυτή, λοιπόν, η γενικευμένη προσέγγιση συνιστά ένα πολύ σημαντικό εργαλείο στην μελέτη του τρόπου με τον οποίο μια υπολογιστική μηχανή μπορεί να αντιληφθεί την μουσική, ως μορφή έκφρασης διαφόρων εννοιών, ιδεών, εμπειριών, ακόμη και συναισθημάτων.

Σύμφωνα με τους Briot et al. [1], οι θεμελιώδεις πτυχές του προβλήματος υπολογιστικής αυτοματοποίησης της σύνθεσης μουσικής είναι οι ακόλουθες:

- **Στόχος Σύνθεσης:** Αφορά κυρίως το είδος και την δομή του παραγόμενου μουσικού περιεχομένου (π.χ. μονοφωνία, πολυφωνία, αντίστιξη, κ.λ.π), σε συνδυασμό με το πλαίσιο χρήσης και εφαρμογής των αποτελεσμάτων που προκύπτουν (π.χ. εκτέλεση από άνθρωπο ή μηχανή).
- **Μουσική Αναπαράσταση:** Γενικότερα, μια μουσική σύνθεση μπορεί να παρασταθεί στο πλαίσιο λειτουργίας ενός υπολογιστικού συστήματος με ποικίλους τρόπους κωδικοποίησης, όπως για παράδειγμα ως εικόνα, ως ηχητική πληροφορία ή ως κάποια συμβολική μορφή ειδικής σημειογραφίας. Όλες αυτές οι διαφορετικές μορφές αναπαράστασης εξυπηρετούν και διαφορετικού είδους λειτουργικότητες.
- **Αρχιτεκτονική Συστήματος:** Ο όρος αυτός αναφέρεται στην εσωτερική δομή του υπολογιστικού συστήματος που μοντελοποιεί την διαδικασία αυτόματης παραγωγής μουσικών συνθέσεων. Οι σχεδιαστικές επιλογές ποικίλουν, καθώς εξαρτώνται άμεσα από την μέθοδο αναπαράστασης της μουσικής πληροφορίας.
- **Μέθοδος:** Η στρατηγική εκπαίδευσης της υπολογιστικής μηχανής κατέχει καθοριστικό ρόλο στην διαμόρφωση του εν λόγω ερευνητικού προβλήματος. Κατά κύριο λόγο περιλαμ-

βάνει τον αλγόριθμο εκμάθησης αλλά και διάφορες επιπρόσθετες ρυθμιστικές παραμέτρους που εμπλέκονται στην διαδικασία παραγωγής νέου μουσικού περιεχομένου.

- **Λειτουργικό Πλαίσιο:** Οι λειτουργικότητες που πλαισιώνουν την διαδικασία αυτόματης σύνθεσης μουσικής καθορίζονται κατά κύριο λόγο από την συνολικότερη διαμόρφωση του εξεταζόμενου προβλήματος. Συνήθως αφορούν κάποια ευρύτερα χαρακτηριστικά του μοντέλου, όπως η μεταβλητότητα, ο βαθμός δημιουργικότητας και το επίπεδο αλληλεπίδρασης με άλλα συστήματα ή ανθρώπινους χρήστες άμεσα ή έμμεσα.

Η αυτόνομη δημιουργία μουσικών συνθέσεων αποτελεί αναμφίβολα ένα ιδιαίτερα δύσκολο πρόβλημα στο πεδίο υπολογιστικής μοντελοποίησης παραγωγικών λειτουργιών. Κατ' αρχάς, σε ανθρώπινο επίπεδο η σύλληψη νέων μουσικών ιδεών είναι άμεσα συνδεδεμένη με σύνθετες υποκειμενικές εμπειρίες και άλλες ασαφείς έννοιες, όπως τα συναισθήματα, οι οποίες δεν μπορούν εύκολα να γίνουν αντιληπτές από μια μηχανή.

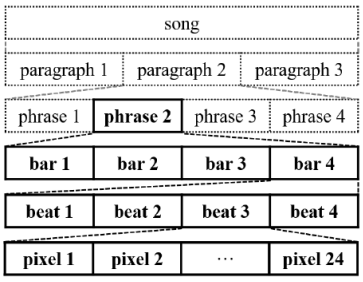


Figure 0.1.1: Ιεραρχική δομή ενός μουσικού κομματιού [2]

Μια ακόμη πρόκληση στο πλαίσιο του εν λόγω ερευνητικού προβλήματος αφορά την κατά κάποιο τρόπο ιεραρχική δομή που χαρακτηρίζει μια μουσική σύνθεση, η οποία αναπαρίσταται γραφικά στην Εικόνα 0.1.1 [2]. Έχει αποδειχθεί ότι ο ανθρώπινος εγκέφαλος κατά την ακρόαση μουσικής έχει την τάση να επικεντρώνεται σε δομικά μοτίβα που σχετίζονται με την συνοχή, τον ρυθμό, την ένταση και την συναισθηματική ροή [3, 4] και εμφανίζονται σε πολλαπλές χρονικές κλίμακες [5]. Συνεπώς, ένας μηχανισμός μοντελοποίησης όλων αυτών των χαρακτηριστικών αυτοαναφοράς αλλά και των διαφόρων χρονικών εξαρτήσεων ανάμεσα στις δομικές μονάδες θεωρείται αναγκαίος.

Μια επιπρόσθετη δυσκολία προκύπτει από το γεγονός ότι οι μουσικές συνθέσεις συνήθως αποτελούνται από πολλά και διαφορετικά μουσικά όργανα, καθένα από τα οποία διαθέτει τα δικά του χαρακτηριστικά και τις αντίστοιχες δυναμικές, αλλά όλα μαζί εκτυλίσσονται συλλογικά στον χρόνο κατά έναν αλληλένδετο τρόπο. Ακόμη, στο πλαίσιο της πολυφωνικής μουσικής, οι φθόγγοι καθενός από τα εμπλεκόμενα tracks παρουσιάζονται κατά κύριο λόγο σε διάφορους δομικούς σχηματισμούς, όπως συγχορδίες ή arpeggios, εισάγοντας επιπλέον αρμονικές εξαρτήσεις. Όλες αυτές, λοιπόν, οι αλληλοσυσχετίσεις, οι οποίες κατέχουν καθοριστικό ρόλο στην διαμόρφωση του τελικού ακούσματος, δεν μπορούν εύκολα να μοντελοποιηθούν από ένα υπολογιστικό σύστημα, ειδικότερα στην περίπτωση μιας αγνωστικής προσέγγισης.

Τέλος, η διαδικασία αξιολόγησης των μοντέλων παραγωγής μουσικής αποτελεί ένα ακόμη ιδιαίτερα σημαντικό ζήτημα. Από την μία πλευρά, έννοιες όπως η επίδοση και η βελτίωσή της δεν μπορούν να οριστούν με σαφήνεια σε σχέση με την ποιότητα των παραγόμενων αποτελεσμάτων, καθιστώντας μεθόδους, οι οποίες στηρίζονται στην χρήση αντικειμενικών μετρικών, ιδιαίτερα προβληματικές. Από την άλλη πλευρά, πρακτικές αξιολόγησης, οι οποίες βασίζονται κατά κύριο λόγο στην ανθρώπινη κρίση, θεωρούνται πιο προτιμητέες. Ωστόσο, χωρίς ομοφωνία ως προς την έννοια της δημιουργικότητας αλλά και δεδομένης της υποκειμενικής αντίληψης ως προς την μουσική, η οποία δεν μπορεί να περιγραφεί μέσω κανόνων [6], η σχεδίαση ενός ακουστικού πειράματος που δύναται να οδηγήσει σε αξιόπιστα επιστημονικά τεκμήρια, ενέχει αρκετές προκλήσεις.

0.1.2 Συνεισφορές της παρούσης Διπλωματικής Εργασίας

Στο πλαίσιο της παρούσης Διπλωματικής Εργασίας εξετάζουμε δύο διαφορετικές προσεγγίσεις ως προς την αυτόματη δημιουργία νέου μουσικού περιεχομένου, κάνοντας χρήση Τεχνητών Νευρωνικών Δικτύων και εφαρμόζοντας μεθόδους Μηχανικής Μάθησης. Έτσι, λοιπόν, οι συνεισφορές μας μπορούν να διαχωριστούν σε δύο μέρη ως εξής:

Unconditional Generation

- Βασιζόμενοι στο MuseGAN [2], αναπτύσσουμε ένα μοντέλο αυτόματης παραγωγής πολυφωνικών μουσικών φράσεων σε συμβολική αναπαράσταση, αποτελούμενων από 5 μουσικά όργανα: *Drums*, *Piano*, *Guitar*, *Bass* και *Strings*. Στην περίπτωση αυτή, η διαδικασία σύνθεσης δεν υπόκειται σε επιπλέον περιορισμούς, ούτε στηρίζεται σε συμπληρωματικά δεδομένα.
- Παραμετροποιούμε την υλοποίησή μας ως προς ένα σύνολο μεταβλητών που αφορούν μουσικά χαρακτηριστικά των παραγόμενων φράσεων, επιφέροντας έτσι μια τροποποίηση στην εσωτερική δομή των δικτύων που απαρτίζουν το σύστημά μας ανάλογα με την επιθυμητή έξοδο.
- Ενσωματώνουμε στο μοντέλο μας μηχανισμούς για καλύτερη παρακολούθηση και έλεγχο της διαδικασίας εκπαίδευσης.
- Αναπτύσσουμε μια εναλλακτική υλοποίηση των ήδη υπαρχόντων μουσικών μετρικών και στην συνέχεια επεκτείνουμε το σύστημα αξιολόγησής μας με 3 νέες προσθήκες, οι οποίες επικεντρώνονται σε τονικά χαρακτηριστικά και άλλα στοιχεία μουσικής υφής.
- Εκτελούμε ένα σύνολο πειραμάτων που στηρίζονται σε διαφορετικές παραμετροποιήσεις και χρησιμοποιούμε τις ποσοτικές μας μετρικές για την αξιολόγηση των αποτελεσμάτων.
- Διεξάγουμε μια ποιοτική μελέτη στην μορφή ακουστικού πειράματος με 40 συμμετέχοντες και αποδεικνύουμε ότι το μοντέλο μας σημειώνει σημαντικά καλύτερη επίδοση από το baseline όσον αφορά 3 μουσικά κριτήρια: *Musical Naturalness*, *Harmonic Consistency* και *Musical Coherence*.

Conditional Generation

- Επεκτείνουμε το μοντέλο μας σε ένα συνεργατικό πλαίσιο ανθρώπου-μηχανής προς την κατεύθυνση της αυτόματης παραγωγής μουσικής συνοδείας: δεδομένου ενός από τα εμπλεκόμενα tracks (προερχόμενου από ανθρώπινη μουσική σύνθεση), το σύστημά μας παράγει αυτόματα τα υπόλοιπα 4, θεωρώντας τα ως την ρυθμική και αρμονική του συνοδεία.
- Πειραματιζόμαστε με διάφορες παραλλαγές του μοντέλου μας, οι οποίες διαφοροποιούνται ως προς τις δομικές συνιστώσες που απαρτίζουν το σύστημα, τον αλγόριθμο εκπαίδευσης και το είδος του conditional οργάνου, δηλαδή εκείνου που αποτελεί την βάση της διαδικασίας σύνθεσης.
- Αξιολογούμε τα προκύπτοντα αποτελέσματα, κάνοντας χρήση του συστήματος μουσικών μετρικών αλλά και της ποιοτικής μελέτης μέσω του ακουστικού πειράματος. Με αυτό τον τρόπο εξάγουμε ενδιαφέροντα συμπεράσματα σχετικά με την επίδραση των διαφόρων τροποποιήσεων στην ποιότητα και την μουσικότητα των παραγόμενων συνοδειών.

0.2 Θεωρητικό Υπόβαθρο

Στην συνέχεια, θα εξετάσουμε συνοπτικά ορισμένα βασικά εργαλεία, τα οποία αποτελούν τα θεμέλια της προσέγγισής μας στο ερευνητικό πρόβλημα της Αυτόματης Παραγωγής Μουσικής.

0.2.1 Συνελικτικά Νευρωνικά Δίκτυα

Ο όρος *Τεχνητό Νευρωνικό Δίκτυο* αναφέρεται σε ένα υπολογιστικό σύστημα επεξεργασίας δεδομένων, το οποίο προσομοιώνει την λειτουργία του ανθρώπινου εγκεφάλου. Πρόκειται στην ουσία για ένα δίκτυο διασυνδεδεμένων δομικών μονάδων που ονομάζονται νευρώνες και είναι τοπολογικά οργανωμένοι σε επίπεδα. Κατ' αντιστοιχία με το βιολογικό πρότυπο, κάθε τεχνητός νευρώνας λαμβάνει ένα σύνολο αριθμητικών εισόδων από άλλους κόμβους του δικτύου, τις οποίες μετασχηματίζει βάσει ενός γραμμικού συνδυασμού με ανάλογα βάρη και ύστερα από την εφαρμογή μιας μη-γραμμικής συνάρτησης ενεργοποίησης παράγει την τελική έξοδο, η οποία τροφοδοτείται στην συνέχεια σε άλλους νευρώνες του δικτύου. Το βασικό χαρακτηριστικό των Τεχνητών Νευρωνικών Δικτύων έγκειται στην δυνατότητα εκπαίδευσής τους μέσα από μια διαδικασία μηχανικής μάθησης, η οποία στοχεύει στην σταδιακή βελτίωση της ικανότητάς τους να επιλύουν κάποιο συγκεκριμένο πρόβλημα. Σε επίπεδο υλοποίησης, αυτό επιτυγχάνεται μέσω ενός αλγορίθμου ακολουθιακού υπολογισμού των μεταβολών των βαρών κάθε νευρώνα του δικτύου, ο οποίος καλείται Backpropagation.

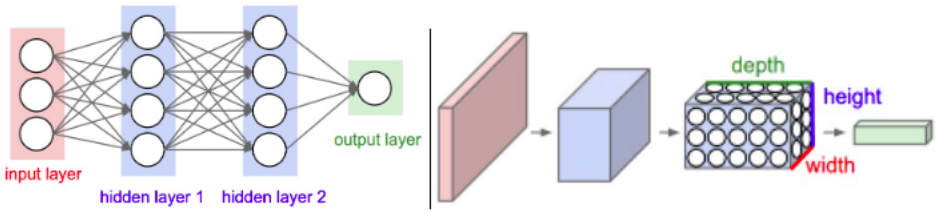


Figure 0.2.1: Αρχιτεκτονική Συνελικτικού Νευρωνκού Δικτύου σε σύγκριση με την γενική περίπτωση [7]

Τα *Συνελικτικά Νευρωνικά Δίκτυα* (CNNs) αποτελούν μία κατηγορία Τεχνητών Νευρωνικών Δικτύων ειδικά σχεδιασμένων για την επεξεργασία και την ανάλυση δεδομένων που χαρακτηρίζονται από κάποιου είδους χωρική τοπολογία πλέγματος και αναπαρίστανται συνήθως με την μορφή γενικευμένων πινάκων. Η λειτουργία τους βασίζεται στον μηχανισμό του βιολογικού συστήματος όρασης, ο οποίος είναι άμεσα συνδεδεμένος με την έννοια του πεδίου υποδοχής (receptive field). Σύμφωνα με τους Levine και Shefner [8], ένα πεδίο υποδοχής ορίζεται ως "μία περιοχή στην οποία ο οπτικός ερεθισμός οδηγεί σε αντίδραση ενός συγκεκριμένου αισθητήριου νευρώνα". Στο υπολογιστικό, λοιπόν, μοντέλο, η τοπολογική περιοχή της εισόδου που αντιστοιχεί στις αισθητήριες συνδέσεις κάθε κόμβου οριοθετείται από μια συγκεκριμένη δομή, η οποία ονομάζεται πυρήνας ή φίλτρο και εμπεριέχει στην ουσία τα βάρη των συνδέσεων. Νευρώνες με επικαλύπτοντα τοπικά πεδία υποδοχής ως προς την ίδια είσοδο διατάσσονται στο χώρο σε επίπεδα 3-διάστατων υπολογιστικών όγκων, κατ' αναλογία με τις κλάσεις οπτικών κυττάρων στον φλοιό του ανθρώπινου ματιού, οι οποίες επιτελούν στην ουσία την διαδικασία εξαγωγής χαρακτηριστικών από το οπτικό ερέθισμα.

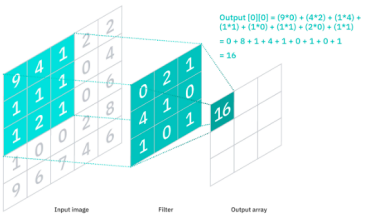


Figure 0.2.2: Συνέλιξη [9]

Η αλληλεπίδραση μεταξύ ενός φίλτρου και του γενικευμένου πίνακα εισόδου μοντελοποιείται υπολογιστικά με την πράξη της συνέλιξης. Όπως φαίνεται και στην Εικόνα 0.2.2, κάθε τιμή της τοπολογίας εξόδου, η οποία αποτελεί τον επονομαζόμενο χάρτη χαρακτηριστικών (feature map), προκύπτει ως το άθροισμα των στοιχείων του αντίστοιχου πεδίου υποδοχής σταθμισμένο κατά το διάνυσμα βαρών του εφαρμοζόμενου πυρήνα. Η διαδικασία αυτή, στην γενικότερη περίπτωση, επιφέρει διαστατική μείωση της εισόδου. Ωστόσο, έχουν αναπτυχθεί διάφορες παραλλαγές του τυπικού συνελικτικού τελεστή, οι οποίες στοχεύουν σε διαφορετικούς χωρικούς μετασχηματισμούς του αρχικού πλέγματος. Μια εξ αυτών αποτελεί και η λεγόμενη *transposed* συνέλιξη, η οποία επιτυγχάνει την δημιουργία ενός μεγαλύτερων διαστάσεων χάρτη χαρακτηριστικών, μέσω της εφαρμογής της κλασσικής μεθόδου σε μια κατάλληλα επαυξημένη εκδοχή του πίνακα εισόδου.

0.2.2 Παραγωγικά Ανταγωνιστικά Δίκτυα

Τα *Παραγωγικά Ανταγωνιστικά Δίκτυα* (Generative Adversarial Networks) αποτελούν μια κατηγορία υπολογιστικών συστημάτων Μηχανικής Μάθησης, τα οποία, όπως μαρτυρά και η ονομασία, στηρίζονται σε ένα μηχανισμό ανταγωνιστικής εκπαίδευσης 2 ανεξάρτητων νευρωνικών δικτύων με στόχο την στατιστική μοντελοποίηση της κατανομής ενός δοθέντος συνόλου δεδομένων. Πιο συγκεκριμένα:

- **Generator:** Το παραγωγικό δίκτυο G δημιουργεί νέα υποψήφια δείγματα, μετασχηματίζοντας ένα διάνυσμα τυχαίου θορύβου $\mathbf{z}$, το οποίο προέρχεται από έναν λανθάνοντα χώρο $p_{\mathbf{z}}$, στην μορφή των δεδομένων της επιθυμητής κατανομής. Στην ουσία, το δίκτυο αυτό αποτελεί την υπολογιστική υλοποίηση μιας παραμετρικής συνάρτησης $G = G(\mathbf{z}; \theta_g)$, η οποία απεικονίζει στοιχεία της κατανομής εισόδου $p_{\mathbf{z}}$ σε δείγματα της κατανομής εξόδου p_g.
- **Discriminator:** Το διαχωριστικό δίκτυο D αξιολογεί τα δεδομένα $\mathbf{x}$ που λαμβάνει ως είσοδο, προβλέποντας την κλάση στην οποία ανήκουν (αυθεντικά ή όχι). Κατ' αντιστοιχία με τον αντίπαλό του, το δίκτυο αυτό αποτελεί την υπολογιστική υλοποίηση μια παραμετρικής συνάρτησης $D = D(\mathbf{x}; \theta_d)$, η οποία αντιστοιχίζει τα δείγματα εισόδου $\mathbf{x}$ σε πραγματικές τιμές στο διάστημα $[0, 1]$. Στην ουσία, η έξοδος $D(\mathbf{x})$ αντιπροσωπεύει την πιθανότητα το $\mathbf{x}$ να προέρχεται από την κατανομή των πραγματικών δεδομένων p_d έναντι της p_g.

Η λειτουργία του συνολικού συστήματος, η οποία αναπαρίσταται διαγραμματικά στην εικόνα 0.2.3, μπορεί να μοντελοποιηθεί ως ένα παίγνιο αντιπαραθετικής μάθησης μεταξύ των δύο αντίπαλων μοντέλων, του Generator και του Discriminator. Μαθηματικά, η εν λόγω διαδικασία αποτυπώνεται στην ακόλουθη minimax συνάρτηση V, όπου $\mathbb{E}$ είναι ο τελεστής αναμενόμενης τιμής:

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_d}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \quad (0.2.1)$$

Παρατηρώντας την σχέση 0.2.1, διαπιστώνουμε ότι ο Generator στοχεύει στην ελαχιστοποίηση

του όρου $\log(1 - D(G(\mathbf{z})))$, η οποία συνεπάγεται ότι ο Discriminator δεν μπορεί να αναγνωρίσει τα υποψήφια δείγματα που παράγει ως μη αυθεντικά, δηλαδή $D(G(\mathbf{z})) \approx 1$. Με αυτό τον τρόπο, το παραγωγικό δίκτυο ωθείται στην ουσία σε μια προσπάθεια έμμεσης ανίχνευσης υποκείμενων γνωρισμάτων της επιθυμητής κατανομής δεδομένων, προκείμενου να καταφέρει να "ξεγελάσει" το αντίπαλο μοντέλο. Παράλληλα, ο Discriminator επιδιώκει την μεγιστοποίηση του αθροίσματος των δύο όρων $\log(D(\mathbf{x})) + \log(1 - D(G(\mathbf{z})))$, έτσι ώστε να μάθει να κατηγοριοποιεί σωστά και τα αυθεντικά $(D(\mathbf{x}))$ αλλά και τα συνθετικά δείγματα $(D(G(\mathbf{z})))$ που εξετάζει.

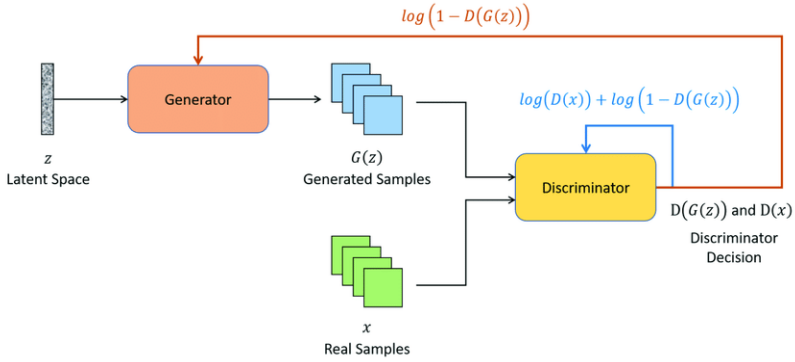


Figure 0.2.3: Παραγωγικό Ανταγωνιστικό Δίκτυο [10]

Σε κάθε βήμα του επαναληπτικού αλγορίθμου εκπαίδευσης, οι επιδόσεις των δύο εμπλεκόμενων μοντέλων ως προς τους ατομικούς τους στόχους καθορίζονται από τους προαναφερθέντες όρους της σχέσης 0.2.1, οι οποίοι στην ουσία αφορούν τις αποφάσεις του Discriminator σχετικά με την κλάση ταξινόμησης των ανάλογα εξεταζόμενων δειγμάτων. Βάσει των προβλέψεων αυτών, υπολογίζονται, μέσω μιας κατάλληλα επιλεγμένης συνάρτησης κόστους, οι αναγκαίες μεταβολές των βαρών για κάθε ένα από τα διαγωνιζόμενα δίκτυα και πραγματοποιείται η ενημέρωσή τους σύμφωνα με την μέθοδο οπισθοδιάδοσης (Backpropagation). Η διαδικασία αυτή τερματίζεται όταν επιτευχθεί η επονομαζόμενη *Ισορροπία Nash* [11], δηλαδή όταν η απόδοση των δύο αντίπαλων μοντέλων δεν μπορεί πλέον να βελτιωθεί περαιτέρω. Αυτό ιδανικά συμβαίνει όταν το classification rate του Discriminator προσεγγίσει το 50%, το οποίο ισοδυναμεί με τυχαίες προβλέψεις όσον αφορά την κλάση προέλευσης των εξεταζόμενων δειγμάτων και επομένως συνεπάγεται ότι ο αντίστοιχος Generator είναι σε θέση να δημιουργεί νέα συνθετικά δεδομένα που δεν μπορούν να διακριθούν από τα πραγματικά. Ωστόσο, στην πράξη η κατάσταση σύγκλισης σε ένα τέτοιας μορφής πλαίσιο μάθησης δεν μπορεί εύκολα να καθοριστεί και γι' αυτό παραμένει ανοιχτό πρόβλημα.

0.2.3 Αυτοκωδικοποιητές

Ο αυτοκωδικοποιητής (autoencoder) είναι ένας τύπος Τεχνητού Νευρωνικού Δικτύου, ο οποίος χρησιμοποιείται για εξαγωγή κωδικοποιημένων αναπαραστάσεων από δεδομένα που δεν επισημειωμένα με κάποιου είδους ετικέτα. Η διαδικασία αυτή λαμβάνει χώρα σε ένα μη επιβλεπόμενο πλαίσιο μάθησης, στο οποίο οι παραγόμενες κωδικοποιήσεις επικυρώνονται και βελτιώνονται επαναληπτικά βάσει της ποιότητας ανακατασκευής της αρχικής εισόδου, σύμφωνα με την γνωστό αλγόριθμο οπισθοδιάδοσης (Backpropagation). Όπως φαίνεται και στο διάγραμμα της Εικόνας 0.2.4, ένας αυτοκωδικοποιητής αποτελείται από δύο βασικές δομικές μονάδες:

- **Encoder:** Ο κωδικοποιητής μετασχηματίζει τα δεδομένα εισόδου σε μια συμπιεσμένη μορφή, γνωστή και ως *code*. Πιο συγκεκριμένα, το αρχικό διάνυσμα $\mathbf{x}$ διαβαίνει μέσα από μια σειρά νευρωνικών επιπέδων, τα οποία επιφέρουν σταδιακή μείωση της διαστατικότητάς του, καταλήγοντας σε ένα bottleneck. Το κρυφό αυτό επίπεδο απαρτίζεται κατά κανόνα από λιγότερους κόμβους συγκριτικά με το επίπεδο εισόδου, περιορίζοντας με αυτό τον τρόπο την πληροφορία που διασχίζει το συνολικό δίκτυο. Έτσι, λοιπόν, δημιουργείται μια χαμηλότερης τάξης αναπαράσταση της εισόδου σε ένα λανθάνοντα χώρο.
- **Decoder:** Ο αποκωδικοποιητής δρα ως "διερμηνέας" της παραγόμενης κωδικοποίησης, αποσυμπιέζοντας αυτή την κρυφή αναπαράσταση σε ένα διάνυσμα του χώρου εισόδου $\mathbf{x}'$. Με αυτό τον τρόπο, επιχειρεί στην ουσία να ανακατασκευάσει τα αρχικά δεδομένα, αξιοποιώντας τα στοιχεία που παρέχει η αντίστοιχη λανθάνουσα κωδικοποίηση. Δομικά ο αποκωδικοποιητής συνήθως αντικατοπτρίζει την αρχιτεκτονική του κωδικοποιητή, υπό την έννοια ότι αποτελείται από τα συμπληρωματικά νευρωνικά επίπεδα, τα οποία επιφέρουν σταδιακή διαστατική αύξηση, διατεταγμένα σε αντίστροφη σειρά.

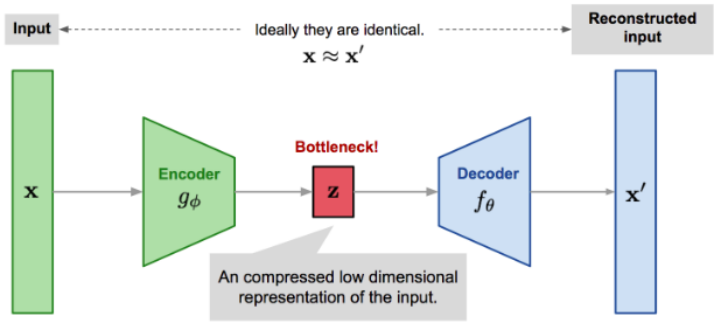


Figure 0.2.4: Αυτοκωδικοποιητής [12]

0.3 Baseline Project: MuseGAN

Το MuseGAN, το οποίο αποτελεί την συντομογραφία του **Mu**lti-track **se**quential **G**enerative **A**dversarial **N**etwork, είναι, όπως μαρτυρά και η ονομασία, ένα υπολογιστικό σύστημα αυτόματης παραγωγής πολυφωνικής μουσικής, αποτελούμενης από 5 διαφορετικά όργανα (*Bass*, *Guitar*, *Strings*, *Drums*, *Piano*), σε συμβολική αναπαράσταση. Το project[1] αυτό προτάθηκε από τους Dong at al. [2] στο *Association for the Advancement of Artificial Intelligence* (AAAI) Conference το 2018 και στηρίζεται στον μηχανισμό των Παραγωγικών Ανταγωνιστικών Δικτύων (GANs).

Figure 0.3.1: MuseGAN tracks [13]

0.3.1 Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του MuseGAN αποτελείται από δύο κύρια μέρη: ένα *multitrack* μοντέλο, το οποίο επικεντρώνεται στις αλληλεξαρτήσεις μεταξύ των διαφόρων μουσικών οργάνων και

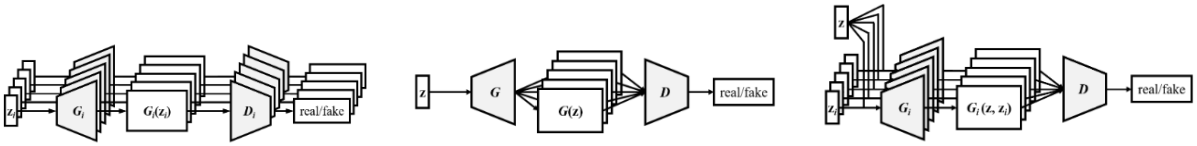
[1] Ο κώδικας της υλοποίησης, το dataset που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου, αλλά και ορισμένα ηχητικά αποσπάσματα είναι διαθέσιμα στην ιστοσελίδα του MuseGAN

ένα *temporal* μοντέλο, το οποίο διαχειρίζεται τις εμπλεκόμενες χρονικές συσχετίσεις. Πιο συγκεκριμένα:

Multitrack μοντέλα

Σύμφωνα με την ανθρώπινη εμπειρία, υπάρχουν δύο επικρατούσες προσεγγίσεις όσον αφορά την διαδικασία δημιουργίας νέων μουσικών συνθέσεων:

- Ένα σύνολο μουσικών που παίζουν διαφορετικά όργανα μπορούν να δημιουργήσουν νέο μουσικό περιεχόμενο σε ένα συνεργατικό πλαίσιο, αυτοσχεδιάζοντας ο καθένας πάνω στο δικό του track χωρίς κάποιο προκαθορισμένο διακανονισμό ή εκτενή προετοιμασία.
- Ένας συνθέτης δημιουργεί μουσική σε ένα πιο συντονισμένο αλλά και δομημένο πλαίσιο, έχοντας γνώση των αρχών αρμονίας και ενορχήστρωσης. Στην συνέχεια, οι μουσικοί που απαρτίζουν την ορχήστρα εκτελούν χωρίς παρεκκλίσεις τα αντίστοιχα μουσικά μέρη της σύνθεσης, τα οποία είναι οργανωμένα με τρόπο που εξασφαλίζει αρμονική συμφωνία και συνοχή.



(a) Jamming Model (b) Composer Model (c) Hybrid Model

Figure 0.3.2: Multi-track models [2]

Βάσει των προαναφερθεισών τεχνικών σύνθεσης, οι Dong et al. [2] σχεδιάζουν 3 διαφορετικά μοντέλα για τις αλληλεξαρτήσεις μεταξύ των εμπλεκόμενων οργάνων:

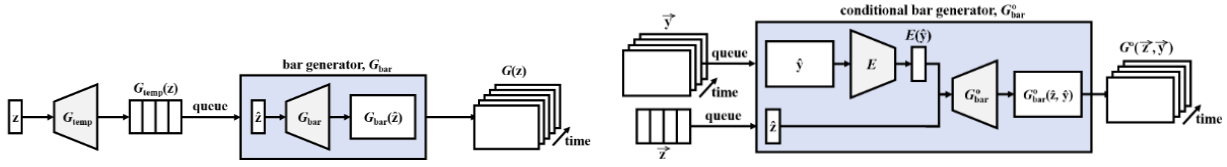
- **Jamming model:** Στο πλαίσιο του Jamming μοντέλου, πολλαπλά Παραγωγικά Ανταγωνιστικά Δίκτυα λειτουργούν ανεξάρτητα με στόχο την δημιουργία πολυοργανικής μουσικής. Πιο συγκεκριμένα, κάθε Generator G_i παράγει μουσικές φράσεις που αντιστοιχούν σε ένα συγκεκριμένο track από ένα ιδιωτικό τυχαίο διάνυσμα εισόδου z_i και λαμβάνει εποπτικά σήματα οπισθοδιάδοσης από τον αντίστοιχο Discriminator D_i.
- **Composer model:** Στο πλαίσιο του Composer μοντέλου, ένας ενιαίος Generator G παράγει πολυκαναλικά μουσικά αποσπάσματα, όπου το κάθε κανάλι αντιστοιχεί σε ένα από τα εμπλεκόμενα tracks. Η δομή αυτή απαιτεί ένα κοινό τυχαίο διάνυσμα εισόδου $\mathbf{z}$, το οποίο κατά κάποιο τρόπο αντιπροσωπεύει το πλάνο του συνθέτη, αλλά και ένα μοναδικό Discriminator, ο οποίος εξετάζει τα παραγόμενα μουσικά τμήματα συλλογικά, προκειμένου να αποφανθεί για την αυθεντικότητα της σύνθεσης.
- **Hybrid model:** Το hybrid μοντέλο, όπως υποδεικνύει και η ονομασία, αποτελεί ένα συνονθύλευμα των προαναφερθέντων αρχιτεκτονικών, το οποίο συνδυάζει σε επίπεδο υλοποίησης τον αυτοσχεδιασμό στο πλαίσιο του jamming με την οργάνωση και τον συντονισμό που επιβάλλει ένας συνθέτης. Όπως φαίνεται και στην εικόνα 0.3.2c, απαρτίζεται από πολλαπλούς Generators, κάθε ένας εκ των οποίων αντιστοιχεί σε ένα διαφορετικό track και ένα μοναδικό Discriminator. Κάθε Generator G_i λαμβάνει ως είσοδο ένα ιδιωτικό (intra-track) τυχαίο διάνυσμα $\mathbf{z}_i$, το οποίο αφορά το μουσικό μέρος του αντί-

στοιχου οργάνου και ένα κοινό (inter-track) τυχαίο διάνυσμα $\mathbf{z}$, το οποίο συντονίζει τους διάφορους μουσικούς G_i όπως ένας συνθέτης. Ο Discriminator D εξετάζει τα παραγόμενα μουσικά τμήματα συλλογικά, προκειμένου να αποφανθεί για την αυθεντικότητα της συνολικής σύνθεσης.

Temporal μοντέλα

Όλα τα multi-track μοντέλα που παρουσιάστηκαν προηγουμένως παράγουν πολυφωνική μουσική με χρονική διάρκεια ενός μέτρου, το οποίο αποτελεί και το βασικό δομικό στοιχείο των συνθέσεων γενικότερα. Προκειμένου λοιπόν να γίνει εφικτή η δημιουργία μουσικών δειγμάτων μεγαλύτερης χρονικής διάρκειας όπου τα διαδοχικά μέτρα συνδέονται μεταξύ τους με συνεπή τρόπο, οι δημιουργοί του MuseGAN [2] εφαρμόζουν 2 διαφορετικές μεθόδους για την μοντελοποίηση της χρονικής δομής:

- **Generation from Scratch:** Αυτή η μέθοδος στοχεύει στην δημιουργία μουσικών φράσεων καθορισμένου μήκους, εντάσσοντας την διαδοχή των μέτρων στο workflow του Generator με την μορφή μιας επιπρόσθετης διάστασης [14]. Πιο συγκεκριμένα, σε αυτή την περίπτωση το παραγωγικό δίκτυο αποτελείται από δύο μέρη:
  - *Temporal Structure Generator:* Ο G_{temp} μετασχηματίζει ένα τυχαίο διάνυσμα εισόδου $\mathbf{z}$ σε μια ακολουθία λανθανόντων μεταβλητών $\vec{\mathbf{z}} = \{\vec{\mathbf{z}}^{(t)}\}_{t=1}^{T}$ (το $T > 0$ συμβολίζει τον συνολικό αριθμό παραγόμενων μέτρων), η οποία αναμένεται να ενσωματώνει πληροφορία σχετικά με χρονικές συσχετίσεις.
  - *Bar Generator:* Ο G_{bar} μετασχηματίζει την προκύπτουσα ακολουθία μεταβλητών $\vec{\mathbf{z}}$ σε μια μουσική φράση T μέτρων με διαδοχικό τρόπο (bar by bar).
- **Track-conditional Generation:** Στο πλαίσιο της μεθόδου αυτής, ένα από τα εμπλεκόμενα tracks δίνεται ως είσοδος στο μοντέλο, το οποίο καλείται να παράξει αυτόματα τα υπόλοιπα, θεωρώντας τα ως την αρμονική και ρυθμική του συνοδεία. Πιο συγκεκριμένα, στην περίπτωση αυτή, το παραγωγικό δίκτυο αποτελείται στην ουσία από τον G_{bar}, ο οποίος παράγει τα διαδοχικά μέτρα της συνοδείας με ακολουθιακό τρόπο, λαμβάνοντας σε κάθε βήμα δύο εισόδους, το conditional track $\vec{\mathbf{y}}^{(t)}$ και το τυχαίο διάνυσμα $\mathbf{z}^{(t)}$, όπου t είναι ο δείκτης του τρέχοντος μέτρου. Ωστόσο, επειδή η ακολουθία μεταβλητών $\vec{\mathbf{y}} = \{\vec{\mathbf{y}}^{(t)}\}_{t=1}^{T}$ αναπαρίσταται σε ένα χώρο υψηλών διαστάσεων, ενσωματώνεται στην αρχιτεκτονική του συστήματος ένας κωδικοποιητής, ο οποίος μετασχηματίζει το conditional track σε ένα embedding στον χώρο του θορύβου, εξάγοντας inter-track χαρακτηριστικά από την high-order μορφή του [15].



(a) Generation from Scratch (b) Track-conditional Generation

Figure 0.3.3: Temporal models [2]

MuseGAN

Το MuseGAN είναι στην ουσία το αποτέλεσμα της ενοποίησης και επέκτασης των προαναφερθέντων multi-track και temporal μοντέλων. Η συνολική αρχιτεκτονική του συστήματος αναπαρίσταται γραφικά στο διάγραμμα της εικόνας 0.3.4. Όλες οι δομικές του συνιστώσες είναι υλοποιημένες ως βαθιά Συνελεκτικά Νευρωνικά Δίκτυα, εκ των οποίων τα διαχωριστικά αποτελούνται από τυπικά συνελικτικά επίπεδα που επιφέρουν διαστατική μείωση της εκάστοτε εισόδου, ενώ τα παραγωγικά από transposed συνελικτικά επίπεδα που επιτυγχάνουν το αντίστροφο αποτέλεσμα.

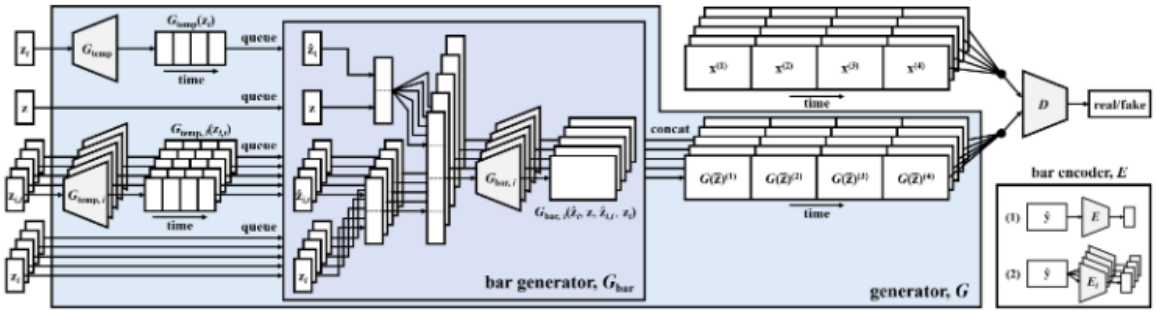
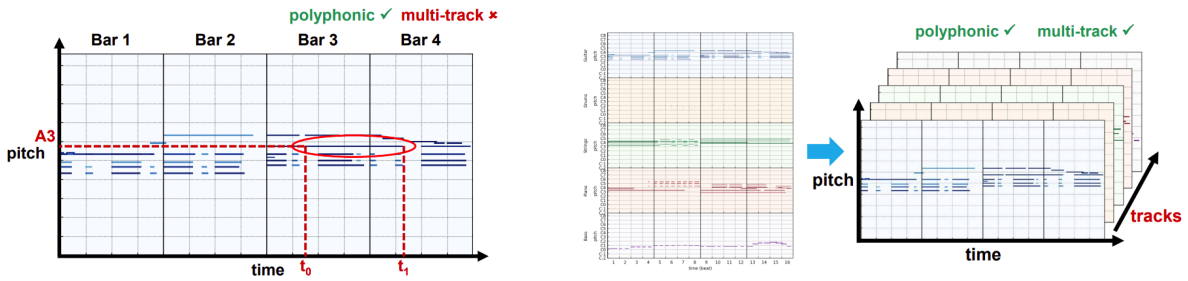


Figure 0.3.4: MuseGAN [13]

0.3.2 Δεδομένα Εκπαίδευσης

Μέθοδος Αναπαράστασης

Δεδομένου ότι τα CNNs προορίζονται για επεξεργασία δεδομένων στην μορφή γενικευμένων πινάκων καθορισμένου μεγέθους, τα μουσικά κομμάτια που χρησιμοποιούνται για την εκπαίδευση του MuseGAN αναπαρίστανται σε μια μορφή γραφικής παρτιτούρας συμβολικού format, η οποία καλείται *pianoroll*. Πρόκειται στην ουσία για ένα πίνακα δυαδικών τιμών, όπου ο οριζόντιος άξονας αντιπροσωπεύει τον αυξάνοντα χρόνο βάσει μιας συγκεκριμένης διακριτοποίησης, ενώ ο κατακόρυφος τους διάφορους φθόγγους ταξινομημένους ανάλογα με το τονικό ύψος. Η δυαδική τιμή 1 στο κελί $[i, j]$ του εν λόγω πίνακα υποδεικνύει ότι η νότα i εκτελείται στο timestep j.



(a) Pianoroll format με συμβολικό timing

(b) Muti-track pianoroll format

Figure 0.3.5: Αναπαράσταση μουσικών δεδομένων [13]

Εύκολα μπορεί κανείς να διαπιστώσει ότι η συγκεκριμένη μέθοδος μπορεί να αναπαραστήσει πολυφωνική μουσική που αντιστοιχεί σε ένα μοναδικό track. Προκειμένου λοιπόν να είναι

εφικτή η μοντελοποίηση μουσικής που αποτελείται από πολλά διαφορετικά όργανα, οι Dong et al. [2] χρησιμοποιούν την *multi-track pianoroll* αναπαράσταση. Όπως φαίνεται και στο σχήμα 0.3.5b, ένα multi-track pianoroll είναι ένα σύνολο πολλαπλών pianorolls, κάθε ένα εκ των οποίων αντιστοιχεί σε κάποιο track.

Dataset

Τα μουσικά αποσπάσματα που χρησιμοποιούνται για την εκπαίδευση του MuseGAN, προέρχονται από το Lakh MIDI Dataset (LMD) [16], μια από τις μεγαλύτερες συλλογές μουσικής σε συμβολική αναπαράσταση, η οποία δημιουργήθηκε από τον Colin Raffel και περιλαμβάνει 176.581 μοναδικά MIDI αρχεία.[2] Πιο συγκεκριμένα, γίνεται χρήση ενός συγκεκριμένου υποσυνόλου του, το οποίο ονομάζεται *LMD-matched* και αποτελείται από 45.129 αρχεία που έχουν αντιστοιχηθεί με καταχωρήσεις του Million Song Dataset (MSD) [17]. Το τελικό σύνολο παραδειγμάτων εκπαίδευσης, ύστερα από την μετατροπή των MIDI αρχείων σε multi-track pianorolls, καλείται **Lakh Pianoroll Dataset** ή **LPD** εν συντομία και μπορεί να βρεθεί στην ιστοσελίδα[3] του project.

Προεπεξεργασία Δεδομένων

Τα δεδομένα του *LMD-matched* υπόκεινται σε μια κατάλληλη διαδικασία προεπεξεργασίας, τα βήματα της οποίας απεικονίζονται γραφικά στο Διάγραμμα 0.3.6. Το τελικό σύνολο εκπαίδευσης αποτελείται από 50.266 μουσικές φράσεις 4 μέτρων, καθένα εκ των οποίων αντιστοιχεί χρονικά σε 96 διακριτά timesteps και έχει εμβέλεια 84 διαφορετικών νοτών.

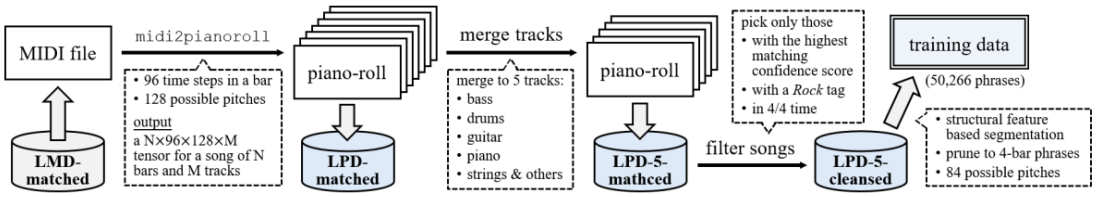


Figure 0.3.6: Προεπεξεργασία Δεδομένων Εκπαίδευσης [2]

0.3.3 Μετρικές Αξιολόγησης

Στο πλαίσιο της πειραματικής αξιολόγησης του MuseGAN, οι Dong et al. [2] προτείνουν 1 intra-track και 4 inter-track ποσοτικές μετρικές που αφορούν μουσικά χαρακτηριστικά και μπορούν να υπολογιστούν τόσο για αυθεντικά όσο και για παραγόμενα δείγματα:

- **Empty Bars (EB):** ποσοστό κενών μέτρων (%)
- **Used Pitch Classes (UPC):** μέσος αριθμός τονικών τάξεων[4] που χρησιμοποιούνται ανά μέτρο (από 0 έως 12)

[2] Ένα MIDI αρχείο εμπεριέχει στην ουσία ένα σύνολο εντολών που μπορούν να εκτελεστούν από διάφορες ηλεκτρονικές συσκευές, παράγοντας το αντίστοιχο ηχητικό αποτέλεσμα.

[3] https://salu133445.github.io/lakh-pianoroll-dataset/dataset

[4] Μια τονική τάξη (pitch class) ορίζεται ως το σύνολο όλων των νοτών που απέχουν μεταξύ τους έναν ακέραιο αριθμό οκτάβων.

- **Qualified Notes (QN):** ποσοστό "qualified" νοτών[5] (%)
- **Drum Pattern (DP):** ποσοστό νοτών σε μοτίβα ρυθμού 4/4[6] (%)
- **Tonal Distance (TD):** αρμονικότητα μεταξύ δύο μουσικών οργάνων[7]

0.4 Unconditional Generation

Ο όρος "Unconditional Generation" αναφέρεται στην διαδικασία αυτόματης δημιουργίας νέου μουσικού περιεχομένου από το μηδέν, δηλαδή χωρίς κάποια πρότερη γνώση ή συμπληρωματική πληροφορία σχετικά με τα παραγόμενα δείγματα. Η συγκεκριμένη μέθοδος παραγωγής αποτελεί την αρχική μας προσέγγιση στο ερευνητικό πρόβλημα της Αυτόματης Σύνθεσης Μουσικής όσον αφορά το πλαίσιο της παρούσης Διπλωματικής Εργασίας.

0.4.1 Μοντέλο

Στηριζόμενοι στο MuseGAN, αναπτύσσουμε ένα μοντέλο αυτόματης παραγωγής πολυφωνικών μουσικών φράσεων σε συμβολική αναπαράσταση, αποτελούμενων από 5 μουσικά όργανα (*Drums*, *Piano*, *Guitar*, *Bass* και *Strings*). Πιο συγκεκριμένα:

Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του συστήματός μας βασίζεται σε ένα συνελικτικό μηχανισμό Παραγωγικού Ανταγωνιστικού Δικτύου και είναι εμπνευσμένη από μια επόμενη μελέτη των Dong και Yang [19], η οποία επικεντρώνεται σε διαφορετικές μεθόδους μετατροπής πινάκων με πραγματικές τιμές σε δυαδικά pianorolls. Πρόκειται στην ουσία για μια δομική παραλλαγή του Hybrid μοντέλου στην αρχιτεκτονική του MuseGAN, καθώς συνδυάζει τις δύο θεμελιώδεις τεχνικές σύνθεσης (jamming, composing) αλλά με πιο ομοιόμορφο και συνάμα συμπαγή τρόπο. Όπως φαίνεται και στο διάγραμμα της εικόνας 0.4.1, απαρτίζεται από 2 βασικές δομικές συνιστώσες:

- **Generator:** Ο Generator αποτελείται από ένα *shared* (κοινό) δίκτυο G_s, ακολουθούμενο από M *private* (εξειδικευμένα) υποδίκτυα G_p^i $(i = 1, \dots M)$, κάθε ένα εκ των οποίων αντιστοιχεί σε ένα διαφορετικό μουσικό όργανο. Ο G_s παράγει αρχικά γενικευμένο μουσικό περιεχόμενο, το οποίο διαισθητικά αντιπροσωπεύει την κοινή μουσική ιδέα που διαμοιράζονται όλα τα εμπλεκόμενα tracks, όπως ένας συνθέτης συντονίζει τα διάφορα όργανα βάσει της συνολικής μουσικής δομής. Στην συνέχεια, κάθε G_p μετασχηματίζει αυτή την αφηρημένη μορφή στο τελικό pianoroll του αντίστοιχου track, σύμφωνα με τα δικά του

[5] Ένας φθόγγος χαρακτηρίζεται *qualified* εάν η χρονική του αξία είναι μεγαλύτερη ή ίση από 3 timesteps (π.χ. τριακοστό δεύτερο).

[6] Το dataset που χρησιμοποιείται για την εκπαίδευση του MuseGAN περιλαμβάνει pianorolls μόνο σε ρυθμό 4/4. Η μετρική `DP` υπολογίζει το ποσοστό των νοτών που εμφανίζονται στα ισχυρά μέρη κάθε μέτρου ανάλογα με το χρησιμοποιούμενο resolution.

[7] Το τονικό περιεχόμενο ενός μουσικού κομματιού μπορεί να αναπαρασταθεί σε συμπυκνωμένη μορφή μέσω ενός ειδικού περιγραφητή που καλείται *chroma vector*. Πρόκειται για ένα διάνυσμα 12 στοιχείων, το οποίο υποδεικνύει την ενέργεια κάθε τονικής τάξης στο ηχητικό σήμα. Η μετρική `TD` στηρίζεται στην προβολή των chroma vectors των αντίστοιχων tracks στον εσωτερικό χώρο ενός 6-διάστατου πολυτόπου. Βάσει αυτής της απεικόνισης, οι τονικές κλάσεις αντιστοιχίζονται στις κορυφές του και οι ισχυρές αρμονικές σχέσεις, όπως οι τρίτες και οι πέμπτες, εμφανίζονται ως μικρές Ευκλείδειες αποστάσεις [18].

μουσικά γνωρίσματα και χαρακτηριστικά, όπως υποδεικνύει το πλαίσιο αυτοσχεδιασμού στην jamming προσέγγιση. Δομικά η συγκεκριμένη αρχιτεκτονική διαφοροποιείται από την hybrid, λόγω της προσθήκης του *shared* δικτύου, το οποίο απαιτεί μόνο ένα ενιαίο τυχαίο διάνυσμα θορύβου στην είσοδο.

- **Discriminator:** Ο Discriminator καθρεπτίζει στην ουσία την δομή του Generator. Πιο συγκεκριμένα, αποτελείται από M *private* υποδίκτυα D_p^i $(i = 1, \dots M)$, κάθε ένα εκ των οποίων αντιστοιχεί σε διαφορετικό μουσικό όργανο, ακολουθούμενα από ένα *shared* δίκτυο D_s. Αρχικά, κάθε D_p εξάγει low-level χαρακτηριστικά από το αντίστοιχο track, τα οποία στην συνέχεια αξιοποιεί ο D_s για τον σχηματισμό μιας κοινής, high-level αναπαράστασης, βάσει της οποίας εκτελεί την τελική πρόβλεψη σχετικά με την αυθεντικότητα της συνολικής μουσικής σύνθεσης. Η βασική διαφορά ανάμεσα στο διαχωριστικό δίκτυο του μοντέλου μας και εκείνο του baseline συστήματος έγκειται στην προσθήκη των private υποδικτύων που επικεντρώνονται στα διαφορετικά μουσικά όργανα, καθώς το MuseGAN χρησιμοποιεί μόνο έναν κοινό Discriminator, ο οποίος αξιολογεί τα παραγόμενα tracks συλλογικά.

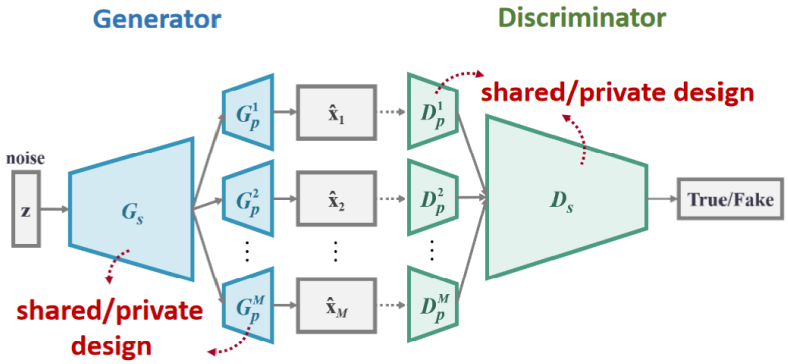


Figure 0.4.1: Αρχιτεκτονική του Unconditional Μοντέλου μας

Υλοποίηση

Όπως και στην περίπτωση του MuseGAN [2], όλες οι δομικές συνιστώσες του unconditional μοντέλου μας είναι υλοποιημένες ως Βαθιά Συνελικτικά Δίκτυα. Τα μεν διαχωριστικά δίκτυα αποτελούνται από διαδοχικά τυπικά συνελικτικά επίπεδα, έτσι ώστε να επιτυγχάνεται η αναγκαία διαστατική μείωση που απαιτεί η απεικόνιση ενός pianoroll σε μια στατιστική πρόβλεψη πραγματικής τιμής. Από την άλλη, τα παραγωγικά αποτελούνται από *transposed* συνελικτικά επίπεδα έτσι ώστε να είναι εφικτός ο μετασχηματισμός του αρχικού διανύσματος τυχαίου θορύβου στην τελική μορφή μιας πολυφωνικής μουσικής σύνθεσης.

Όπως αναφέρθηκε προηγουμένως, το baseline project είναι ειδικά σχεδιασμένο για επεξεργασία και δημιουργία δεδομένων με συγκεκριμένη δομή, όσον αφορά την διακριτοποίηση του χρόνου, το εύρος νοτών ή ακόμη και το πλήθος των μέτρων που συνιστούν μια μουσική φράση. Προκειμένου, λοιπόν, να αντιμετωπίσουμε αυτόν τον περιορισμό και να μπορέσουμε να διερευνήσουμε περαιτέρω τις παραγωγικές δυνατότητες του μοντέλου μας, παραμετροποιούμε την υλοποίησή μας ως προς ένα σύνολο μεταβλητών που αφορούν μουσικά χαρακτηριστικά των παραγόμενων φράσεων και παρουσιάζονται στον Πίνακα 1 μαζί με τον αντίστοιχο συμβολισμό. Η διαδικασία

αυτή επιφέρει μάλιστα μια τροποποίηση στην εσωτερική δομή των δικτύων που απαρτίζουν το σύστημά μας ανάλογα με την επιθυμητή έξοδο, καθιστώντας το έτσι ιδιαίτερο ευέλικτο και ευπροσάρμοστο σε διαφορετικές πρακτικές σύνθεσης.

s	number of samples
l	latent dimension
t	number of tracks
r	bar resolution
p	number of pitches
m	number of measures
$o\ (= m \cdot r)$	number of total timesteps
b	beat resolution
i	lowest pitch

Table 1: Μεταβλητές Παραμετροποίησης

Διαδικασία Εκπαίδευσης

Η διαδικασία εκπαίδευσης του unconditional μοντέλου μας περιγράφεται μαθηματικά από την ακόλουθη minimax συνάρτηση τιμής:

$$\min_G \max_D V^*(G, D) = \mathbb{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}[D(G(\mathbf{z}))] + \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \quad (0.4.1)$$

Εύκολα μπορεί κανείς να διαπιστώσει ότι η V^* αποτελεί μια τροποποιημένη εκδοχή της τυπικής συνάρτησης V, η οποία παρουσιάστηκε στην ενότητα 0.2.2, καθώς περιλαμβάνει έναν επιπρόσθετο gradient penalty όρο, ο οποίος μέσω μιας κανονικοποίησης των υπολογιζόμενων παραγώγων εξασφαλίζει ταχύτερη σύγκλιση στην βέλτιστη κατάσταση και γενικότερη σταθεροποίηση της συνολικής διαδικασίας εκπαίδευσης [2, 20]. Όσον αφορά τον χρησιμοποιούμενο συμβολισμό, η κατανομή $p_{\hat{\mathbf{x}}}$ ορίζεται έμμεσα μέσω ομοιόμορφης δειγματοληψίας σε ευθείες γραμμές μεταξύ ζευγών σημείων που προέρχονται από την κατανομή των πραγματικών δεδομένων p_d και την κατανομή των παραγόμενων δειγμάτων p_g, αντίστοιχα.

Ακολουθώντας την σχετική βιβλιογραφία [2, 19, 21, 20], εφαρμόζουμε μια διαφορετική μέθοδο εκμάθησης, η οποία στηρίζεται σε διαδοχικές εναλλαγές μεταξύ k βημάτων βελτιστοποίησης του Discriminator D και ενός βήματος ενημέρωσης του Generator G. Με αυτό τον τρόπο εξασφαλίζεται ότι το διαχωριστικό δίκτυο διατηρείται αρκετά κοντά στην βέλτιστη λύση του, ενώ συγχρόνως το παραγωγικό προσαρμόζεται με επαρκώς αργό ρυθμό.

Προκειμένου να αποκτήσουμε μια πιο λεπτομερή εικόνα του μηχανισμού μάθησης, ενσωματώνουμε σε κάθε βήμα της διαδικασίας εκπαίδευσης μια επιπρόσθετη φάση επικύρωσης (*validation phase*), κατά την οποία εξετάζουμε την συμπεριφορά και την απόκριση των δομικών συνιστωσών του συστήματος σε δεδομένα εκτός του συνόλου εκπαίδευσης. Επιπλέον, εφαρμόζουμε τεχνικές *Early-Stopping* και *Checkpointing* για καλύτερο έλεγχο και ενδεχόμενο πρόωρο τερματισμό της εκπαίδευσης σε περίπτωση που η απόδοση του μοντέλου, η οποία υπολογίζεται βάσει μιας προκαθορισμένης μετρικής παρακολούθησης, φθίνει με την πάροδο του χρόνου, ξεπερνώντας ένα συγκεκριμένο κατώφλι.

0.4.2 Δεδομένα Εκπαίδευσης

Για την αναπαράσταση των δεδομένων εκπαίδευσης του unconditional συστήματός μας χρησιμοποιούμε το *multi-track pianoroll* format [2, 19]. Το τελικό μας dataset προκύπτει από την *LPD-5-cleansed* version του LPD [2] ύστερα από μια κατάλληλη διαδικασία προεπεξεργασίας, η οποία στοχεύει στην κατάτμηση των περιλαμβανομένων pianorolls σε μουσικές φράσεις συγκεκριμένης παραμετροποίησης:

1) Αρχικά εφαρμόζουμε μια διαδικασία υποδειγματοληψίας, έτσι ώστε να επιτευχθεί το επιθυμητό resolution στον χρονικό άξονα.

2) Στην συνέχεια, απορρίπτουμε νότες εκτός μιας συγκεκριμένης εμβέλειας προκειμένου να αποκτήσουμε το ζητούμενο εύρος τόνων.

3) Τέλος, συλλέγουμε με τυχαίο τρόπο από κάθε τραγούδι ένα μεταβλητό πλήθος υποψηφίων μουσικών φράσεων και επιλέγουμε μόνο εκείνες που εμπεριέχουν επαρκή αριθμό νοτών στα διάφορα tracks, σύμφωνα με ένα προκαθορισμένο κατώφλι.

0.4.3 Εργαλεία Αξιολόγησης

Objective Μετρικές

Όσον αφορά το κομμάτι της πειραματικής αξιολόγησης, υλοποιούμε από την αρχή τις 5 υπάρχουσες μετρικές βάσει της περιγραφικής τους ανάλυσης στο baseline paper [2]. Στην συνέχεια επεκτείνουμε το σύστημα αξιολόγησής μας με 3 επιπλέον προσθήκες ποσοτικών δεικτών, οι οποίες επικεντρώνονται σε τονικά χαρακτηριστικά και άλλα στοιχεία μουσικής υφής:

- **Used Pitches (UP):** μέσος αριθμός νοτών που χρησιμοποιούνται ανά μέτρο, συμπεριλαμβανομένων όλων των οκτάβων στην προκαθορισμένη εμβέλεια
- **Scale Ratio (SR):** ποσοστό νοτών στην δοθείσα μουσική κλίμακα[8] (%)
- **Polyphonic Rate (PR):** ποσοστό πολυφωνικών χρονικών βημάτων[9] (%)

User Study

Όσον αφορά το κομμάτι της υποκειμενικής αξιολόγησης, διεξάγουμε μια ποιοτική μελέτη στην μορφή ακουστικού πειράματος, η οποία μπορεί να διαχωριστεί σε δύο μέρη ανάλογα με το αντίστοιχο task. Το τμήμα που αφορά την Αυτόματη Παραγωγή Μουσικής χωρίς συνθήκες και περιορισμούς (Unconditional Generation) στοχεύει σε μια εμπεριστατωμένη ακουστική σύγκριση του μοντέλου μας με το MuseGAN. Η δομή του ερωτηματολογίου στηρίζεται σε ζεύγη ηχητικών δειγμάτων, από τα οποία ο χρήστης καλείται να επιλέξει εκείνο που προτιμά όσον αφορά 3 μουσικά κριτήρια:

- **Musical Naturalness:** Θα μπορούσε το εξεταζόμενο μουσικό απόσπασμα να έχει δημιουργηθεί από άνθρωπο;

[8] Για την υλοποίηση της SR μετρικής χρησιμοποιούμε την κλίμακα Ντο ματζόρε, καθώς η συντριπτική πλειοψηφία των κομματιών στο Lakh Pianoroll Dataset είναι στην συγκεκριμένη τονικότητα [22]. Επομένως, η SR υποδεικνύει στην ουσία το ποσοστό των φυσικών φθόγγων (χωρίς κάποιο σημείο αλλοίωσης).

[9] Ένα χρονικό βήμα καλείται πολυφωνικό εάν ο αριθμός νοτών που εκτελούνται ταυτόχρονα την συγκεκριμένη στιγμή υπερβαίνει ένα συγκεκριμένο κατώφλι (συνήθως λαμβάνει την τιμή 2).

- **Harmonic Consistency:** Οι ήχοι που παράγονται από τα διάφορα μουσικά όργανα είναι σε συμφωνία μεταξύ τους; Το προκύπτον αποτέλεσμα είναι ακουστικά ευχάριστο;
- **Musical Coherence:** Οι διάφορες μουσικές φράσεις που απαρτίζουν το εξεταζόμενο κομμάτι είναι μεταξύ τους συνδεδεμένες με κάποιο τρόπο; Υπάρχει συνοχή μουσικών ιδεών;

Τα ακουστικά δείγματα που απαρτίζουν κάθε test case επιλέγονται με τυχαίο τρόπο ανάμεσα σε 200 μουσικές φράσεις που έχουν παραχθεί με κάθε μοντέλο και παρουσιάζονται στον αξιολογητή με τυχαία σειρά. Οι συμμετέχοντες της έρευνάς μας είναι συνολικά 40 άτομα, κάθε ένα εκ των οποίων αξιολογεί 2 διαφορετικά ζεύγη ηχητικών αποσπασμάτων. Ορισμένα δημογραφικά στοιχεία παρουσιάζονται με παραστατικό τρόπο στην εικόνα 0.4.2.

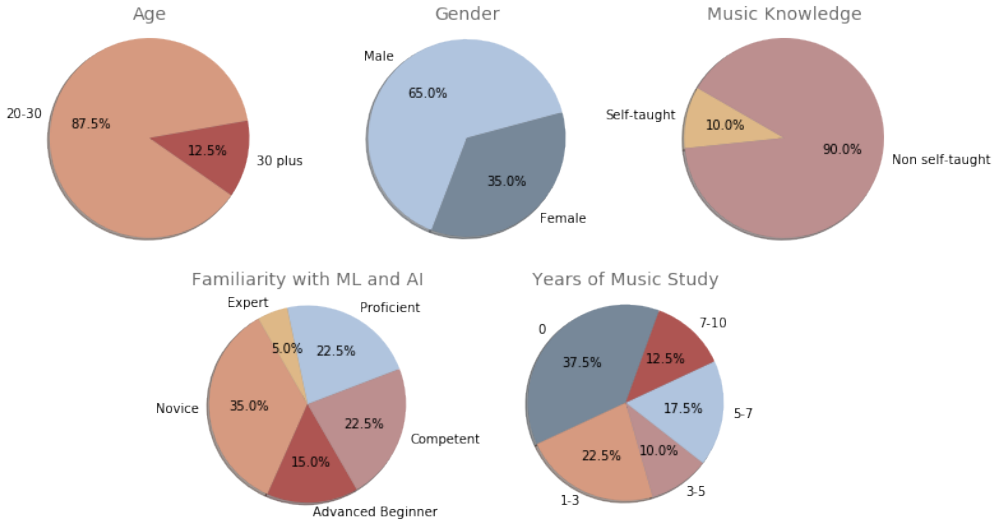


Figure 0.4.2: Δημογραφικά στοιχεία της ποιοτικής μελέτης

0.4.4 Πειράματα και Αποτελέσματα

		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Generation parameters	Number of Pitches	84	72	72	72	72	72	72	72	72	72
	Beat Resolution	24	4	8	12	16	4	8	12	16	4
	Number of Bars	4	4	4	4	4	4	4	4	4	4
	Lowest Pitch	24	24	24	24	24	24	24	24	24	24
	Samples per song	8	8	8	8	8	8	8	8	8	8
	Latent Dimension	128	128	128	128	128	128	128	128	128	256
Training parameters	Number of Steps	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
	Batch Size	16	16	16	16	16	16	16	16	16	4
	Number of Phrases	4	4	4	4	4	4	4	4	4	4
	Steps per G update	6	6	6	6	6	11	11	11	11	6
	Steps per Evaluation	50	50	50	50	50	50	50	50	50	50
	Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	Betas	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)

Table 2: Παραμετροποιήσεις Πειραμάτων

Προκειμένου να εξετάσουμε την αποδοτικότητα του μοντέλου μας ως προς την δημιουργία πολυφωνικής μουσικής και να διερευνήσουμε διάφορες πτυχές της παραγωγικής διαδικασίας,

διεξάγουμε ένα σύνολο πειραμάτων χρησιμοποιώντας διαφορετικές παραμετροποιήσεις της υλοποίησής μας (C_1-C_{10}), οι οποίες παρουσιάζονται αναλυτικά στον Πίνακα 2.

Σύγκριση διαφορετικών πειραματικών παραμετροποιήσεων

Η σύγκριση μεταξύ των διαφορετικών παραμετροποιήσεων του Πίνακα 2 (εκτός της C_1) πραγματοποιείται στο πλαίσιο της objective αξιολόγησης. Ειδικότερα, για κάθε configuration παράγουμε, χρησιμοποιώντας τον αντίστοιχο Generator του τελευταίου βήματος εκπαίδευσης, 20000 μέτρα οργανωμένα ανά τετράδες σε μουσικές φράσεις, στις οποίες εφαρμόζουμε στην συνέχεια τις μουσικές μετρικές που εξετάστηκαν ενδελεχώς προηγουμένως. Ο Πίνακας 3 συνοψίζει τα προκύπτοντα αποτελέσματα με την μορφή μέσων όρων. Τιμές πλησιέστερα στα αντίστοιχα στατιστικά γνωρίσματα της κατανομής των πραγματικών δεδομένων θεωρούνται καλύτερες, εκτός της inter-track μετρικής TD για την οποία οι μικρότερες τιμές είναι και οι ζητούμενες.

	EB (%)					**UPC**				**QN** (%)				**UP**			
	B	D	G	P	S	B	G	P	S	B	G	P	S	B	G	P	S
Ground-truth	1.6	1.1	4.1	5.1	3.2	2.48	4.16	4.2	4.57	91.7	85.3	89.7	89.7	2.72	5.8	5.9	6.8
C_2	0.3	0.0	0.9	1.9	2.1	2.89	4.4	4.88	5.14	59.0	58.2	57.2	60.8	3.14	5.96	6.58	7.61
C_3	0.4	0.0	0.9	0.7	0.7	3.12	5.18	5.33	5.14	49.0	52.2	56.5	64.6	3.4	7.57	7.73	**7.05**
C_4	0.0	2.1	0.6	1.2	0.9	3.04	**4.17**	**4.39**	5.47	50.9	59.7	65.9	70.3	3.39	**5.71**	**6.54**	7.75
C_5	0.0	**0.8**	1.6	1.0	**2.5**	3.09	4.05	4.58	4.14	**63.1**	**72.9**	**72.4**	**74.3**	3.32	5.9	6.6	5.97
C_6	0.5	0.1	1.8	0.8	0.7	**2.47**	4.9	5.07	5.4	54.3	48.9	52.9	50.6	**2.67**	6.67	7.24	8.22
C_7	0.1	0.1	1.6	0.2	0.4	2.75	4.36	4.87	5.49	56.2	64.9	59.1	57.2	3.15	6.06	6.8	7.72
C_8	**1.9**	0.1	**4.3**	**2.8**	0.4	2.64	5.81	6.08	5.09	**63.1**	56.7	60.1	64.3	2.86	8.19	8.44	7.78
C_9	0.0	0.2	1.5	0.0	0.2	3.06	3.92	5.41	5.48	62.9	54.5	55.0	45.3	3.4	5.62	7.3	9.24
C_{10}	0.2	0.1	0.0	0.0	0.4	2.69	5.09	5.16	**4.52**	57.2	69.5	66.8	60.8	2.99	7.59	7.3	7.12

	TD						**SR** (%)				**PR** (%)					**DP** (%)
	B-G	B-S	B-P	G-S	G-P	S-P	B	G	P	S	B	D	G	P	S	D
Ground-truth	0.7	0.73	0.7	0.7	0.67	0.66	75.7	74.6	73.9	72.6	1.2	15.2	57.3	60.8	64.1	83.1
C_2	0.86	0.91	0.9	0.98	0.99	0.97	79.0	82.1	78.7	75.0	2.6	21.7	**49.7**	**53.7**	**58.7**	79.6
C_3	0.57	0.53	0.56	0.6	0.62	0.59	80.4	76.3	**72.7**	70.0	0.6	6.8	47.5	47.7	55.0	92.3
C_4	0.37	0.38	0.36	0.39	0.39	0.38	70.6	82.5	81.9	78.4	0.1	4.6	28.5	37.9	50.2	88.4
C_5	**0.26**	**0.27**	**0.28**	**0.27**	**0.25**	**0.27**	82.5	73.8	77.5	77.1	0.2	2.9	42.9	47.0	55.4	53.1
C_6	0.9	0.96	0.92	1.08	1.03	1.09	80.5	77.6	79.0	78.7	**1.4**	10.8	42.6	43.5	57.3	**83.5**
C_7	0.5	0.56	0.51	0.61	0.56	0.6	**75.7**	**75.1**	78.5	75.5	0.7	5.7	35.0	34.8	48.9	96.0
C_8	0.42	0.37	0.42	0.45	0.49	0.46	70.6	71.9	66.3	67.7	0.1	4.4	39.5	40.0	52.6	92.6
C_9	0.27	0.31	0.31	0.34	0.34	0.38	76.3	82.3	76.1	82.8	0.3	3.5	23.4	33.6	50.0	58.7
C_{10}	0.96	0.89	0.95	0.93	1.04	1.05	84.4	69.4	78.2	**71.2**	1.9	**14.2**	66.9	51.7	56.8	81.6

Table 3: Objective μετρικές για διαφορετικές πειραματικές παραμετροποιήσεις

Κατ' αρχάς, παρατηρούμε ότι δεν υπάρχει κάποια συγκεκριμένη παραμετροποίηση ικανή να βελτιώσει ταυτόχρονα όλες τις objective μετρικές. Πιο συγκεκριμένα, διαπιστώνουμε ότι η αύξηση του beat resolution (C_5) οδηγεί σε ισχυρότερες αρμονικές σχέσεις μεταξύ των tracks (TD) και υψηλότερα ποσοστά "qualified" νοτών (QN), αλλά συγχρόνως επιδρά αρνητικά στα ρυθμικά χαρακτηριστικά των παραγόμενων δειγμάτων (DP κοντά στο 50%). Από την άλλη πλευρά, η χρήση υψηλότερων τιμών για την υπερπαράμετρο k, η οποία αναπαριστά το πλήθος βημάτων εκπαίδευσης ανά ενημέρωση του Generator, φαίνεται να συμβάλλει στην βελτίωση της επιθυμητής πυκνότητας νοτών (EB) αλλά και άλλων τονικών χαρακτηριστικών, όπως το SR. Τέλος, ο διπλασιασμός της λανθάνουσας διάστασης που αξιοποιείται από το παραγωγικό δίκτυο (C_{10}) επηρεάζει θετικά τον δείκτη PR για τα Drums, καθώς επίσης και τους δείκτες UPC και SR για τα Strings, τα οποία αποτελούν και το πιο προβληματικό μουσικό όργανο στην όλη

διαδικασία υπό την έννοια ότι συνήθως ενσωματώνει αρχετό θόρυβο.

Σύγκριση με Baseline

Η σύγκριση του unconditional μοντέλου μας με το MuseGAN πραγματοποιείται ως προς τους δύο βασικούς άξονες αξιολόγησης:

Objective Σύγκριση

Όσον αφορά το κομμάτι της ποσοτικής αποτίμησης, επιλέγουμε 2 διαφορετικές πειραματικές παραμετροποιήσεις του συστήματός μας για την objective αντιπαράθεση με τα 4 multi-track μοντέλα[10] που περιλαμβάνονται στην αρχιτεκτονική του MuseGAN. Η πρώτη είναι η C_1, η οποία αντιστοιχεί στο configuration της υλοποίησης της baseline αρχιτεκτονικής και η δεύτερη είναι η C_2, η οποία θεωρείται default στο πειραματικό πλαίσιο του συστήματός μας. Όπως και προηγουμένως, για κάθε μοντέλο παράγουμε 20000 μέτρα οργανωμένα ανά τετράδες σε μουσικές φράσεις, στις οποίες εφαρμόζουμε στην συνέχεια τις μουσικές μετρικές.

		EB (%)					**UPC**				**QN** (%)				**DP** (%)
		B	D	G	P	S	B	G	P	S	B	G	P	S	D
training data	*baseline*	8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
	ours	1.6	1.1	4.1	5.1	3.2	2.48	4.16	4.2	4.57	91.7	85.3	89.7	89.7	83.1
Baseline	jamming	6.59	2.33	18.3	22.6	6.10	1.53	3.69	4.13	4.09	71.5	56.6	62.2	63.1	93.2
	composer	0.01	28.9	1.34	0.02	0.01	2.51	4.20	4.89	5.19	49.5	47.4	49.9	52.5	75.3
	hybrid	2.14	29.7	11.7	17.8	6.04	2.35	4.76	5.45	5.24	44.6	43.2	45.5	52.0	71.3
	ablated	92.4	100	12.5	0.68	0.00	1.00	2.88	2.32	4.72	0.00	22.8	31.1	26.2	0.0
Ours	C_1	0.0	**0.7**	0.4	1.3	1.2	3.63	4.67	**4.64**	5.29	55.6	**75.8**	**74.1**	**75.9**	59.5
	C_2	**0.3**	0.0	**0.9**	**1.9**	**2.1**	**2.89**	**4.4**	4.88	**5.14**	**59.0**	58.2	57.2	60.8	**79.6**

Table 4: Intra-track Αξιολόγηση

Ο Πίνακας 4 συνοψίζει τα προκύπτοντα αποτελέσματα της intra-track αξιολόγησης με την μορφή μέσων όρων για τους κοινούς ποσοτικούς δείκτες μεταξύ των εξεταζόμενων frameworks. Τιμές πλησιέστερα στα αντίστοιχα μουσικά γνωρίσματα της κατανομής των πραγματικών δεδομένων θεωρούνται καλύτερες. Ωστόσο, λόγω της στατιστικής απόκλισης μεταξύ των δύο συνόλων εκπαίδευσης, η οποία πιθανώς οφείλεται στην τυχαία διαδικασία διαλογής μουσικών φράσεων από τα pianorolls του LPD-5-cleansed, η ακριβής σύγκριση των 4 baseline μοντέλων με τις 2 παραμετροποιήσεις της δικής μας αρχιτεκτονικής δεν είναι εφικτή. Παρόλα αυτά, στην περίπτωση των QN και DP μετρικών όπου η εν λόγω διαφορά είναι αμελητέα, παρατηρούμε ότι το σύστημά μας παρουσιάζει σημαντικά καλύτερη επίδοση από όλα τα baseline (χρωματιστά κελιά).

		TD					
		B-G	B-S	B-P	G-S	G-P	S-P
Baseline	jamming	1.56	1.60	1.54	1.05	0.99	1.05
	composer	1.37	1.36	1.30	0.95	0.98	0.91
	hybrid	1.34	1.35	1.32	0.85	0.85	0.83
Ours	C_1	0.2	0.22	0.2	0.21	0.2	0.21
	C_2	0.86	0.91	0.9	0.98	0.99	0.97

Table 5: Inter-track Αξιολόγηση

[10] Το μοντέλο ablated αντιστοιχεί σε μια παραλλαγή του Composer, η οποία δεν περιλαμβάνει Batch Normalization.

Ο Πίνακας 5 συνοψίζει τα προκύπτοντα αποτελέσματα της inter-track αξιολόγησης με την μορφή μέσων όρων. Σε αυτή την περίπτωση, μικρότερες τιμές θεωρούνται καλύτερες. Εύκολα μπορεί κανείς να διαπιστώσει ότι η παραμετροποίηση C_1 υπερβαίνει σημαντικά όλα τα υπόλοιπα μοντέλα σε ό,τι αφορά την αρμονικότητα των παραγόμενων δειγμάτων (TD κοντά στο 0.2 για όλα τα ζεύγη οργάνων), γεγονός που έρχεται σε συμφωνία με τα πορίσματα των προηγούμενων πειραμάτων σχετικά με την χρήση υψηλού beat resolution. Επιπλέον, παρατηρούμε ότι και η επίδοση της παραμετροποίησης C_1 είναι άξια αναφοράς, ειδικότερα στην περίπτωση αρμονικών συσχετίσεων μεταξύ ενός μελωδικού οργάνου (Bass) και ενός οργάνου συνοδείας (Piano, Guitar, Strings).

Subjective Σύγκριση

Όσον αφορά το κομμάτι της ακουστικής σύγκρισης των 2 εξεταζόμενων frameworks αυτόματης παραγωγής μουσικής, επιλέγουμε δείγματα από την παραμετροποίηση C_2 για την δική μας αρχιτεκτονική και αντίστοιχα από το Composer μοντέλο για το baseline. Τα προκύπτοντα αποτελέσματα αναπαρίστανται γραφικά στην εικόνα 0.4.3. Παρατηρούμε ότι το σύστημά μας παρουσιάζει σημαντικά καλύτερη επίδοση από το MuseGAN αναφορικά με όλα τα εξεταζόμενα μουσικά κριτήρια. Το γεγονός αυτό υποδεικνύει ότι η υλοποίησή μας, η οποία δίνει έμφαση σε ρυθμικά χαρακτηριστικά, πράγματι συμβάλλει στην φυσικότητα και την συνοχή των παραγόμενων δειγμάτων. Επιπλέον, διαπιστώνουμε ότι η shared/private δομή του Generator αλλά και του Discriminator επιδρά θετικά σε διάφορα τονικά στοιχεία, τα οποία καθορίζουν την συνολική αρμονικότητα μιας μουσικής σύνθεσης.

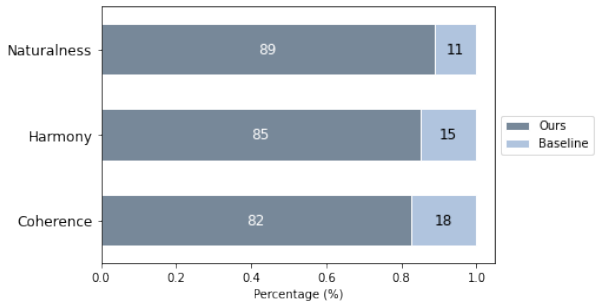


Figure 0.4.3: Subjective Σύγκριση

0.5 Conditional Generation

Σε συνέχεια της ερευνητικής μας μελέτης στον τομέα της Αυτόματης Σύνθεσης Μουσικής, επεκτείνουμε το μοντέλο μας σε ένα συνεργατικό πλαίσιο ανθρώπου-μηχανής προς την κατεύθυνση της αυτόματης παραγωγής μουσικής συνοδείας. Πιο συγκεκριμένα, δεδομένου ενός από τα εμπλεκόμενα tracks (προερχόμενου από ανθρώπινη μουσική σύνθεση), το σύστημά μας παράγει αυτόματα τα υπόλοιπα 4, θεωρώντας τα ως την ρυθμική και αρμονική του συνοδεία.

0.5.1 Μοντέλο

Αρχιτεκτονική Συστήματος

Σε γενικότερο πλαίσιο, το conditional μοντέλο μας διατηρεί την υποδομή συνελικτικού μηχανισμού Παραγωγικού Ανταγωνιστικού Δικτύου από το unconditional task, το οποίο εξετάστηκε ενδελεχώς στην προηγούμενη ενότητα. Ωστόσο, η ενσωμάτωση συνθηκών στην παραγωγική διαδικασία επιφέρει αναπόφευκτα τροποποιήσεις των υπαρχόντων δομικών μονάδων αλλά και προσθήκες νέων. Πιο συγκεκριμένα:

- **Generator:** Όπως φαίνεται και στο σχήμα της εικόνας 0.5.1a, ο Conditional Generator

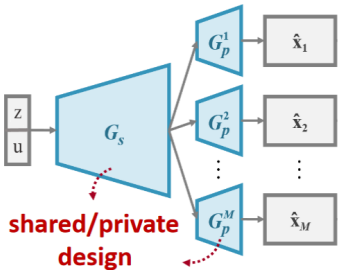
ακολουθεί την shared/private δομή του Unconditional Παραγωγικού Δικτύου, καθώς απαρτίζεται από ένα κοινό τμήμα G_s και 4 ιδιωτικά υποδίκτυα G_p, όσα δηλαδή και τα όργανα συνοδείας. Ωστόσο, το shared δίκτυο τροποποιείται κατάλληλα, έτσι ώστε να λαμβάνει 2 εισόδους: ένα τυχαίο διάνυσμα θορύβου $\mathbf{z}$ που προέρχεται από μια prior κατανομή $p_{\mathbf{z}}$ και το embedding $\mathbf{u}$ του conditional track στο χώρο του θορύβου.

- **"Global" Discriminator:** Προκειμένου να αποκτήσουμε ένα γενικό κριτή, ο οποίος αξιολογεί το κατά πόσον η παραγόμενη συνοδεία αρμόζει μουσικά στο δοθέν conditional track, ενσωματώνουμε στην αρχιτεκτονική του conditional μοντέλου μας τον υπάρχοντα Discriminator και τον ονομάζουμε "Global". Όπως φαίνεται και στο σχήμα 0.5.1b, η δομή του παραμένει αναλλοίωτη, καθώς αποτελείται από 5 private υποδίκτυα D_p, κάθε ένα εκ των οποίων αντιστοιχεί και σε διαφορετικό όργανο (συμπεριλαμβανομένου και του conditional), ακολουθούμενα από ένα shared δίκτυο D_s.

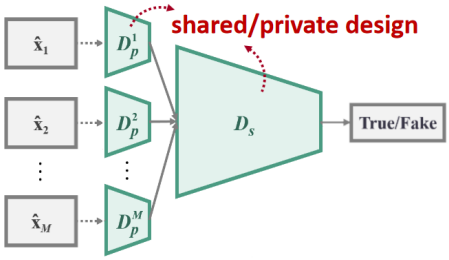
- **"Local" Discriminator:** Επεκτείνουμε την αρχική μας υλοποίηση, ενσωματώνοντας στο σύστημά μας και ένα δεύτερο Discriminator, τον οποίο ονομάζουμε "Local". Όπως υποδεικνύει και η ονομασία, αυτό το επιπρόσθετο διαχωριστικό δίκτυο αξιολογεί μόνο την παραγόμενη συνοδεία ως ανεξάρτητη μουσική σύνθεση. Δομικά, ακολουθεί την shared/private σχεδίαση του "Global", με μόνη διαφορά ότι σε αυτή την περίπτωση τα ιδιωτικά υποδίκτυα D_p είναι 4, όσο δηλαδή και τα accompaniment tracks.

- **Encoder:** Εκτός των τυπικών συνιστωσών ενός Παραγωγικού Ανταγωνιστικού Δικτύου, το conditional σύστημά μας περιλαμβάνει επίσης και έναν Κωδικοποιητή, ο οποίος είναι υπεύθυνος για την δημιουργία των embeddings των conditional tracks στο χώρο του θορύβου.

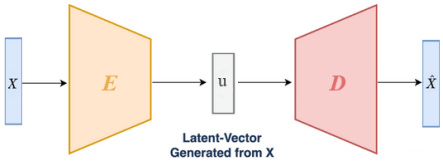
- **Decoder:** Υλοποιούμε και τον αντίστοιχο Αποκωδικοποιητή, ο οποίος επιτελεί την αντίστροφη λειτουργία, ανακατασκευάζοντας την αρχική είσοδο από την λανθάνουσα αναπαράσταση.



(a) Generator



(b) Discriminator



(c) Encoder και Decoder

Figure 0.5.1: Συνιστώσες του conditional μοντέλου

Υλοποίηση

Όπως και προηγουμένως, όλες οι δομικές συνιστώσες του conditional μοντέλου μας είναι υλοποιημένες ως Βαθιά Συνελικτικά Δίκτυα. Τα μεν διαχωριστικά, καθώς επίσης και ο Κωδικοποιητής, αποτελούνται από διαδοχικά συνελικτικά επίπεδα, έτσι ώστε να επιτυγχάνεται η αναγκαία διαστατική μείωση. Τα δε παραγωγικά, αλλά και ο Αποκωδικοποιητής, αποτελούνται από *transposed* συνελικτικά επίπεδα έτσι ώστε να επιτυγχάνεται η κατάλληλη διαστατική αύξηση. Επιπλέον, ακολουθούμε και σε αυτή την περίπτωση την τεχνική παραμετροποίησης ως προς το σύνολο μεταβλητών που αναγράφονται στον Πίνακα 1.

Διαδικασία Εκπαίδευσης

Ο μηχανισμός μάθησης του conditional μοντέλου μας στηρίζεται στις επιμέρους εκπαιδευτικές διαδικασίες των δομικών του συνιστωσών. Πιο συγκεκριμένα:

- **"Global" Discriminator:** Ο "Global" Discriminator μαθαίνει να διαχωρίζει τα αυθεντικά από τα παραγόμενα δείγματα (conditional + accompaniment), χρησιμοποιώντας μια κατάλληλα επιλεγμένη συνάρτηση κόστους, η οποία ποσοτικοποιεί τις εσφαλμένες προβλέψεις του και για *θετικά* αλλά και *αρνητικά* παραδείγματα εκπαίδευσης.
- **"Local" Discriminator:** Ο "Local" Discriminator ακολουθεί την ίδια μέθοδο, με μόνη διαφορά ότι σε κάθε περίπτωση εξετάζει μόνο τα μέρη της συνοδείας.
- **Generator:** Ο Generator μαθαίνει να δημιουργεί νέες μουσικές συνοδείες βάσει του feedback που λαμβάνει από το εμπλεκόμενο σχήμα διαχωριστικών δικτύων. Ειδικότερα, στην περίπτωση 2 Discriminators, το loss του Generator υπολογίζεται ως η μέση τιμή των αντίστοιχων προβλέψεων, εναλλακτικά λαμβάνεται υπόψιν μόνο η έξοδος του ενός.
- **Encoder:** Αναπτύσσουμε 2 διαφορετικές μεθόδους όσον αφορά την διαδικασία εκπαίδευσης του Κωδικοποιητή:
  - *1-phase*: Σε αυτή την περίπτωση ο Κωδικοποιητής εκπαιδεύεται μαζί με το Παραγωγικό Ανταγωνιστικό Δίκτυο, ακολουθώντας την πρακτική του Generator, καθώς τα δύο αυτά δίκτυα συνεισφέρουν από κοινού στην παραγωγή νέων υποψήφιων δειγμάτων.
  - *2-phase*: Όπως υποδεικνύει και η ονομασία, σε αυτή την περίπτωση η διαδικασία μάθησης του conditional μοντέλου μας διαιρείται σε δύο μέρη. Αρχικά, ο Κωδικοποιητής εκπαιδεύεται μαζί με τον αντίστοιχο Αποκωδικοποιητή ως ένα ενιαίο Autoencoder σύστημα, χρησιμοποιώντας το MSE loss ανάμεσα στα αρχικά και τα ανακατασκευασμένα conditional tracks και την Kullback–Leibler απόκλιση, η οποία αναπαριστά την στατιστική απόσταση ανάμεσα στην Τυπική Κανονική Κατανομή $\mathcal{N}(0, 1)$ και την κατανομή που μοντελοποιεί τον λανθάνοντα χώρο των παραγόμενων embeddings. Στην συνέχεια πραγματοποιείται η εκπαίδευση του GAN, κατά την διάρκεια της οποίας ο Encoder διατηρείται αμετάβλητος.

Όπως και στο unconditional task, η συνολική διαδικασία εκπαίδευσης στηρίζεται σε διαδοχικές εναλλαγές μεταξύ k βημάτων βελτιστοποίησης των εμπλεκόμενων Discriminators και ενός βήματος ενημέρωσης του Generator (ή και του Encoder στο 1-phase mode) [2, 19, 21, 20].

0.5.2 Πειράματα και Αποτελέσματα

Όσον αφορά το κομμάτι της πειραματικής μας μελέτης, εστιάζουμε σε 8 διαφορετικά μοντέλα αυτόματης παραγωγής μουσικών συνοδειών, τα οποία παρουσιάζονται αναλυτικά στον πίνακα της εικόνας 0.5.2 μαζί με τον αντίστοιχο συμβολισμό. Όπως μπορούμε να δούμε, τα εξεταζόμενα conditional μοντέλα διαφοροποιούνται ως προς τις δομικές συνιστώσες που απαρτίζουν το σύστημα (Global Discriminator / Global και Local Discriminators), τον αλγόριθμο εκπαίδευσης του Κωδικοποιητή (1-phase mode / 2-phase mode - AutoEncoder) και το είδος του conditional οργάνου, δηλαδή εκείνου που αποτελεί την βάση της διαδικασίας σύνθεσης (Piano / Guitar).

		AutoEncoder	Local Discriminator
Piano	P_{00}	-	-
	P_{01}	-	✓
	P_{10}	✓	-
	P_{11}	✓	✓
Guitar	G_{00}	-	-
	G_{01}	-	✓
	G_{10}	✓	-
	G_{11}	✓	✓

Figure 0.5.2: Conditional Μοντέλα

Objective Αξιολόγηση

Στο πλαίσιο της objective αξιολόγησης, παράγουμε με κάθε μοντέλο 20000 μέτρα οργανωμένα ανά τετράδες σε μουσικές φράσεις, στις οποίες εφαρμόζουμε στην συνέχεια τις μουσικές μας μετρικές και υπολογίζουμε τους αντίστοιχους μέσους όρους.

		EB (%)					**UPC**				**QN** (%)				**DP** (%)
		B	D	G	P	S	B	G	P	S	B	G	P	S	D
training data	*baseline*	8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
	ours	1.6	1.0	5.0	5.6	3.7	2.47	4.09	4.19	4.5	91.6	85.6	90.0	89.7	82.9
Ours	P_{00}	0.6	0.0	2.2	-	2.4	2.71	3.93	-	**4.33**	51.4	56.5	-	58.9	**86.1**
	P_{01}	0.2	0.0	1.8	-	1.5	2.57	**4.09**	-	4.76	**58.2**	56.1	-	**61.7**	86.3
	P_{10}	17.4	**0.2**	**3.0**	-	**4.4**	1.68	3.9	-	4.3	50.7	49.2	-	55.1	87.0
	P_{11}	**1.6**	0.0	0.7	-	0.9	**2.56**	4.19	-	5.16	54.8	**56.6**	-	51.0	86.2
Baseline	jamming	4.60	3.47	13.3	-	3.44	2.05	3.79	-	4.23	73.9	58.8	-	62.3	91.6
	composer	0.65	20.7	1.97	-	1.49	2.51	4.57	-	5.10	53.5	48.4	-	59.0	84.5
	hybrid	2.09	4.53	10.3	-	4.05	2.86	4.43	-	4.32	43.3	55.6	-	67.1	71.8

Table 6: Intra-track Αξιολόγηση για Piano

Ο Πίνακας 6 συνοψίζει τα αποτελέσματα των 4 intra-track μετρικών που είναι κοινές ανάμεσα στο conditional σύστημά μας και το MuseGAN για την περίπτωση του Piano. Όπως και προηγουμένως, τιμές πλησιέστερα στα αντίστοιχα μουσικά γνωρίσματα της κατανομής των πραγματικών δεδομένων θεωρούνται καλύτερες. Παρατηρούμε ότι η μεμονωμένη εφαρμογή της εκπαίδευσης 2 φάσεων (P_{10}) επιδρά θετικά σε ορισμένα μουσικά χαρακτηριστικά, όπως το EB, το DP και το UPC, αλλά φαίνεται να επηρεάζει αρκετά την μορφή του Bass track, ελαττώνοντας σημαντικά την συνολική συμβολή του στην σύνθεση (EB κοντά στο 18%). Από την άλλη πλευρά, η προσθήκη δεύτερου Discriminator, ανεξαρτήτως μεθόδου εκπαίδευσης, ευεργετεί σχεδόν όλους τους ποσοτικούς δείκτες, επιβεβαιώνοντας ότι το επιπρόσθετο feedback που παρέχει ως προς τις παραγόμενες συνοδείες είναι πράγματι χρήσιμο. Για λόγους πληρότητας, παραθέτουμε επίσης και τα αντίστοιχα αποτελέσματα για τα 3 multi-track μοντέλα του MuseGAN. Ωστόσο και πάλι λόγω της στατιστικής απόκλισης μεταξύ των συνόλων εκπαίδευσης, η απόλυτη σύγκριση δεν είναι εφικτή. Παρ' όλα αυτά, βλέπουμε ότι οι τιμές είναι στην ίδια τάξη μεγέθους, γεγονός το

οποίο υποδεικνύει ότι και η δική μας υλοποίηση για τις μετρικές παρέχει ουσιαστική ερμηνεία της παραγόμενης μουσικής.

		TD					
		B-G	B-S	B-P	G-S	G-P	S-P
Ours	P_{00}	0.82	0.83	0.88	0.87	0.95	**0.94**
	P_{01}	0.79	0.81	0.85	**0.85**	**0.94**	**0.94**
	P_{10}	**0.74**	**0.73**	**0.81**	0.94	1.02	1.01
	P_{11}	0.83	0.92	0.97	0.99	1.12	1.17
Baseline	jamming	1.51	1.53	1.50	1.04	0.95	1.00
	composer	1.41	1.36	1.40	0.96	1.01	0.95
	hybrid	1.39	1.36	1.38	0.96	0.94	0.95

Table 7: Inter-track Αξιολόγηση για Piano

Ο Πίνακας 7 συνοψίζει τα αποτελέσματα της inter-track μετρικής TD για την περίπτωση του Piano (μικρότερες τιμές θεωρούνται καλύτερες). Παρατηρούμε ότι το μοντέλο P_{10} παρουσιάζει την καλύτερη επίδοση συγκριτικά με όλα τα conditional μοντέλα (bold τιμές) αλλά και όλες τις baseline αρχιτεκτονικές (χρωματιστά κελιά) όσον αφορά την αρμονικότητα μεταξύ ενός μελωδικού οργάνου (Bass) και ενός οργάνου συνοδείας (Piano, Guitar, Strings). Από την άλλη πλευρά, ισχυρότερες αρμονικές σχέσεις μεταξύ οργάνων συνοδείας εντοπίζονται στην περίπτωση του μοντέλου P_{01}.

		UP				**SR** (%)				**PR** (%)				
		B	G	P	S	B	G	P	S	B	D	G	P	S
training data		2.71	5.68	5.85	6.71	75.9	74.4	74.1	72.8	1.1	15.2	55.7	61.8	62.3
Ours	P_{00}	2.94	5.79	-	**6.28**	81.7	**75.8**	-	77.1	**1.2**	13.3	40.6	-	44.2
	P_{01}	2.94	**5.77**	-	7.17	**77.1**	76.3	-	75.6	1.5	**15.2**	**48.7**	-	**59.9**
	P_{10}	1.74	5.05	-	6.07	82.2	80.6	-	79.0	0.2	10.1	22.2	-	30.2
	P_{11}	**2.84**	5.43	-	7.3	80.7	77.6	-	**72.3**	1.9	9.7	38.2	-	56.3

Table 8: Επιπρόσθετη Intra-track Αξιολόγηση για Piano

Ο Πίνακας 8 συνοψίζει τα αποτελέσματα των 3 επιπρόσθετων intra-track μετρικών μόνο για τα 4 conditional μοντέλα που χρησιμοποιούν το Piano ως συνθήκη κατά την παραγωγική διαδικασία. Εύκολα μπορεί κανείς να διαπιστώσει ότι για κάθε variant οι προκύπτουσες τιμές προσεγγίζουν αρκετά τις πραγματικές. Ειδικότερα στην περίπτωση του PR, αξιοσημείωτη είναι η επίδοση του P_{01}, βάσει της οποίας συμπεραίνουμε ότι η προσθήκη του Local Discriminator πράγματι συμβάλλει στην επίτευξη των κατάλληλων επιπέδων πολυφωνίας ανά track.

Τέλος, ο Πίνακας 9 συνοψίζει τα αποτελέσματα όλων των objective μετρικών για τα 4 μοντέλα που χρησιμοποιούν την Κιθάρα ως conditional όργανο. Παρατηρούμε ότι το μοντέλο G_{01} επιφέρει ισχυρότερες αρμονικές σχέσεις μεταξύ των εμπλεκόμενων tracks (TD), ενώ το επιπρόσθετο feedback που παρέχει ο δεύτερος Discriminator συνεισφέρει επίσης στα ρυθμικά χαρακτηριστικά των παραγόμενων συνοδειών (DP) αλλά και σε άλλα στοιχεία υφής, όπως το PR. Από την άλλη πλευρά, η εφαρμογή του αλγορίθμου εκπαίδευσης 2 φάσεων και για τους 2 συνδυασμούς διαχωριστικών δικτύων επιδρά θετικά στην πυκνότητα φθογγοσήμων (EB), καθώς επίσης και σε χρονικά αλλά και τονικά χαρακτηριστικά των παραγόμενων δειγμάτων, όπως ποσοτικοποιούνται από τους δείκτες QN, UP, UPC και SR, ειδικότερα για τα όργανα που παίζουν κατά κύριο λόγο συγχορδίες.

	EB (%)					**UPC**				**QN** (%)				**UP**			
	B	D	G	P	S	B	G	P	S	B	G	P	S	B	G	P	S
training data	1.8	0.9	4.3	5.2	3.6	2.47	4.21	4.14	4.49	91.8	87.5	91.6	90.5	2.7	5.85	5.84	6.75
G_{00}	0.8	0.0	-	2.1	1.8	**2.51**	-	5.04	**4.59**	**62.5**	-	49.3	**60.3**	**2.77**	-	7.31	6.91
G_{01}	0.0	0.0	-	3.1	0.0	3.05	-	4.31	5.28	57.6	-	52.4	59.6	3.36	-	6.18	7.69
G_{10}	**1.6**	0.0	-	1.8	**3.5**	2.35	-	**4.28**	4.01	50.2	-	**59.5**	58.6	2.59	-	**6.13**	5.88
G_{11}	0.4	**0.2**	-	**3.3**	0.6	2.32	-	4.62	4.66	55.6	-	47.8	57.9	2.46	-	6.4	**6.68**

	TD						**SR** (%)				**PR** (%)					**DP** (%)
	B-G	B-S	B-P	G-S	G-P	S-P	B	G	P	S	B	D	G	P	S	D
training data	0.71	0.72	0.7	0.69	0.66	0.66	75.4	73.5	73.4	73.1	0.8	15.5	59.7	61.0	62.6	85.0
G_{00}	**0.83**	0.85	0.9	0.96	1.01	0.98	84.7	-	80.9	**77.0**	1.1	10.9	-	53.9	53.4	87.1
G_{01}	0.87	0.87	**0.83**	**0.93**	**0.92**	**0.86**	86.7	-	83.6	83.9	2.8	**14.9**	-	**55.3**	**60.8**	**86.0**
G_{10}	0.84	**0.84**	0.84	**0.93**	0.95	0.89	82.0	-	79.8	85.4	**0.7**	6.0	-	37.5	44.0	91.7
G_{11}	0.89	0.87	0.88	1.06	1.09	0.97	**78.0**	-	**76.9**	80.5	**0.9**	9.7	-	42.1	54.4	83.7

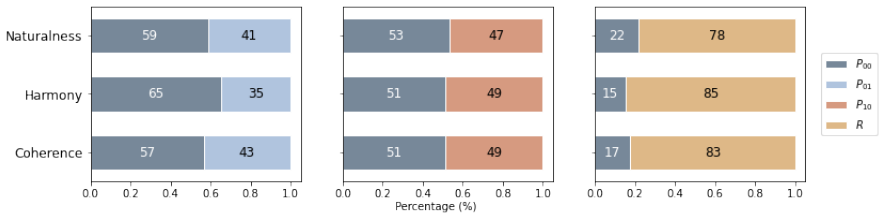
Table 9: Objective Αξιολόγηση για Guitar

Subjective Αξιολόγηση

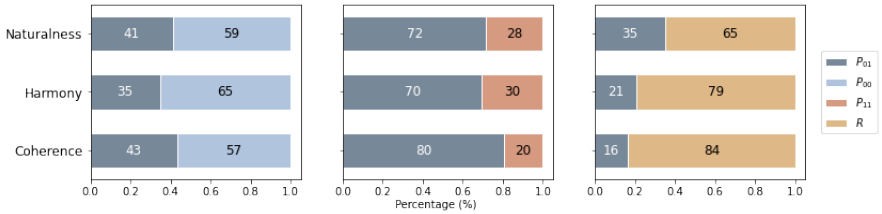
Το τμήμα της ποιοτικής μας μελέτης που αφορά την Αυτόματη Παραγωγή Μουσικής Συνοδείας (Conditional Generation) στοχεύει σε μια εμπεριστατωμένη ακουστική σύγκριση ανάμεσα στα μοντέλα του πίνακα 0.5.2. Κάθε test case του ερωτηματολογίου αποτελείται από 3 ηχητικά δείγματα, το πρώτο εκ των οποίων είναι το conditional track και τα υπόλοιπα 2 αντιστοιχούν σε πιθανές συνοδείες του, προερχόμενες είτε από διαφορετικά τεχνητά μοντέλα είτε και από την κατανομή των πραγματικών δεδομένων. Ο χρήστης καλείται να επιλέξει την συνοδεία που προτιμά για το εκάστοτε conditional track αναφορικά με 3 μουσικά κριτήρια: *Musical Naturalness*, *Harmonic Consistency*, *Musical Coherence*. Οι συμμετέχοντες της έρευνάς μας είναι συνολικά 40 άτομα, καθένα εκ των οποίων αξιολογεί 18 ακουστικά groups, επιτυγχάνοντας με αυτό τον τρόπο περίπου 45 συγκρίσεις για κάθε ζεύγος μοντέλων (συνολικά 16). Όλα τα conditional tracks και κατ' επέκταση οι υποψήφιες συνοδείες τους επιλέγονται με τυχαίο τρόπο ανάμεσα σε 32 ηχητικά δείγματα και παρουσιάζονται στον χρήστη με τυχαία σειρά.

Τα προκύπτοντα αποτελέσματα για την περίπτωση κατά την οποία το conditional όργανο είναι το Piano παρουσιάζονται γραφικά στα διαγράμματα της εικόνας 0.5.3. Κάθε bar-plot αναπαριστά τις προτιμήσεις των χρηστών ανάμεσα στα συγκρινόμενα μοντέλα με την μορφή ποσοστών. Παρατηρούμε ότι στην περίπτωση σύγκρισης με τις αυθεντικές μουσικές εκδοχές, η πλειοψηφία των τεχνητών δειγμάτων διακρίνονται εύκολα. Το υψηλότερο ποσοστό έναντι ανθρώπινης σύνθεσης επιτυγχάνεται από το μοντέλο P_{01} για την μουσική φυσικότητα (35%), γεγονός το οποίο υποδεικνύει ότι η προσθήκη του Local Discriminator πράγματι ευεργετεί την παραγωγική διαδικασία.

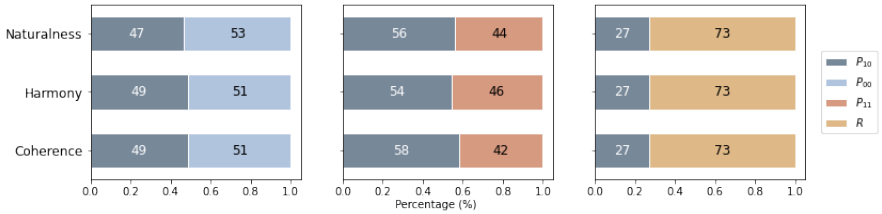
Όσον αφορά την σύγκριση ανάμεσα στα διάφορα frameworks που έχουμε αναπτύξει, διαπιστώνουμε ότι δείγματα του μοντέλου P_{01} είναι σημαντικά πιο προτιμητέα σε σχέση με το P_{11}, υποδηλώνοντας ότι η κατάλληλη μέθοδος εκπαίδευσης για την αρχιτεκτονική των 2 Discriminators είναι η 1-phase, όπως προέκυψε και κατά την objective αξιολόγηση. Επιπλέον, παρατηρούμε ότι το P_{10} υπερβαίνει την επίδοση του P_{11} σε όλα τα εξεταζόμενα μουσικά κριτήρια, αποτέλεσμα το οποίο συνεπάγεται ότι η αρμόζουσα δομική σχεδίαση για τον αλγόριθμο εκπαίδευσης 2 φάσεων περιλαμβάνει μόνο τον Global Discriminator, όπως αντίστοιχα υπέδειξε και η ανάλυση των μετρικών. Τέλος, διαπιστώνουμε ότι υπάρχει μια μικρή προτίμηση του P_{00} έναντι των P_{10} και



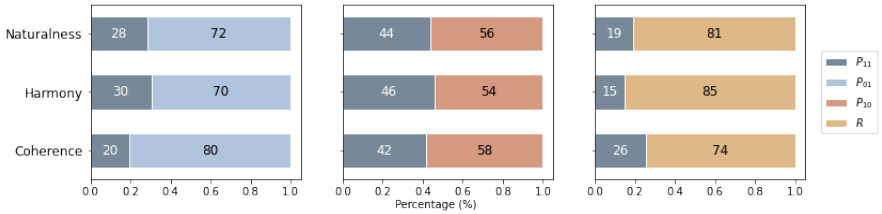
(a) P_{00}



(b) P_{01}



(c) P_{10}

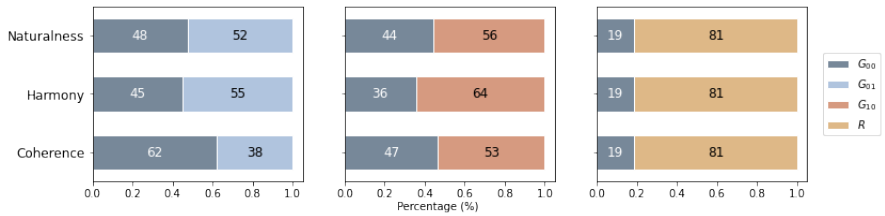


(d) P_{11}

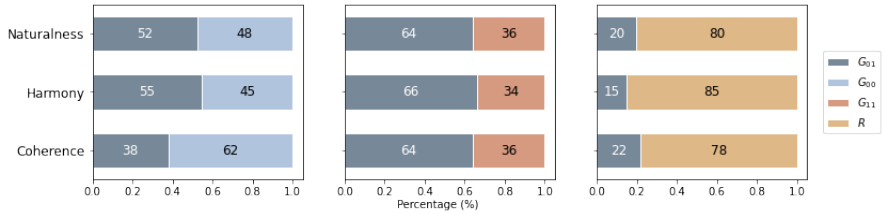
Figure 0.5.3: Subjective Αξιολόγηση για Piano

P_{01}, η οποία υποδηλώνει ότι και η πρωταρχική μας υλοποίηση παρέχει δυνατότητα παραγωγής ποιοτικών συνοδειών.

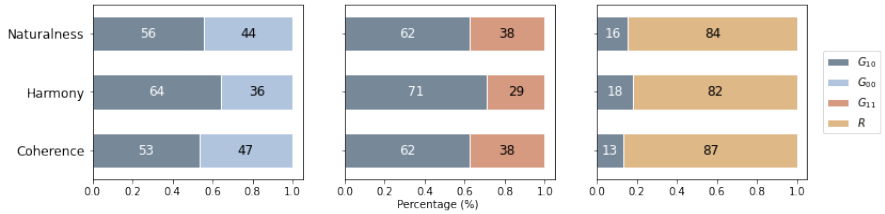
Τα προκύπτοντα αποτελέσματα για την περίπτωση κατά την οποία το conditional όργανο είναι η Κιθάρα παρουσιάζονται γραφικά στα διαγράμματα της εικόνας 0.5.4. Όπως και προηγουμένως, τα περισσότερα τεχνητά δείγματα διακρίνονται εύκολα από τα αντίστοιχα αυθεντικά ως προς όλες τις εξεταζόμενες πτυχές. Όλα τα ποσοστά προτίμησης κυμαίνονται στο εύρος 13-20%, υποδηλώνοντας ότι πιθανώς η Κιθάρα παρέχει λιγότερη πληροφορία στο σύστημα ως παραγωγική συνθήκη συγκριτικά με το Piano, καθώς συνήθως παίζει συγχορδίες ενώ το μέρος του Πιάνου περιλαμβάνει και κάποια μελωδικά στοιχεία. Όσον αφορά τα διάφορα frameworks που έχουμε αναπτύξει, τα πορίσματα που προκύπτουν είναι όμοια με την περίπτωση του Piano. Μια αξιοσημείωτη διαφορά έγκειται κατά την σύγκριση του G_{01} με το G_{00}, όπου παρατηρούμε ότι η χρήση μόνο του Global Discriminator φαίνεται να συμβάλλει αρκετά στην συνοχή των παραγόμενων συνοδειών (ποσοστό προτίμησης 62%).



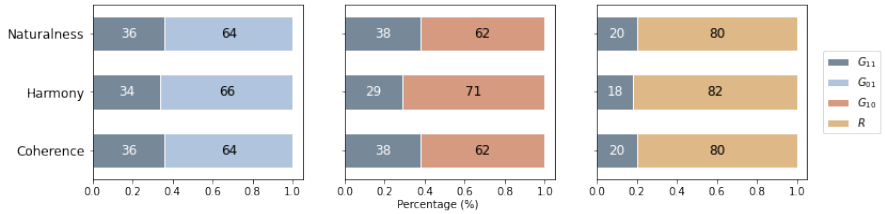
(a) G_{00}



(b) G_{01}



(c) G_{10}



(d) G_{11}

Figure 0.5.4: Subjective Αξιολόγηση για Guitar

0.6 Σύνοψη και Μελλοντικές Επεκτάσεις

Συμπερασματικά, η παρούσα Διπλωματική Εργασία επιχειρεί να μελετήσει και να διερευνήσει διεξοδικά το πρόβλημα δημιουργίας νέου μουσικού περιεχομένου με αυτόνομο τρόπο από μια υπολογιστική σκοπιά, κάνοντας χρήση Τεχνητών Νευρωνικών Δικτύων και εφαρμόζοντας μεθόδους Μηχανικής Μάθησης. Συνολικά, η έρευνά μας μπορεί να διαχωριστεί σε 2 βασικά μέρη:

- Το πρώτο αφορά το task της *unconditional* παραγωγής πολυφωνικών μουσικών φράσεων σε συμβολική αναπαράσταση για 5 διαφορετικά μουσικά όργανα.
- Το δεύτερο μελετά την κατεύθυνση της αυτόματης παραγωγής μουσικής συνοδείας (*conditional* generation) σε ένα συνεργατικό πλαίσιο ανθρώπου-μηχανής.

Μέσα από τα πειράματα που διεξάγουμε χρησιμοποιώντας ποικίλες παραλλαγές του μοντέλου μας και βασιζόμενοι σε διαφορετικές μεθόδους αξιολόγησης, καταλήγουμε σε ενδιαφέροντα συμπεράσματα σχετικά με την επίδραση των διαφόρων τροποποιήσεων στην ποιότητα και την

μουσικότητα των παραγόμενων δειγμάτων. Σε γενικές γραμμές, το μοντέλο μας επιδεικνύει ορισμένες επιθυμητές ιδιότητες, ανοίγοντας έτσι τον δρόμο για περαιτέρω βελτιώσεις και μελλοντικές επεκτάσεις, όπως για παράδειγμα η δημιουργία ολόκληρου τραγουδιού σε ανθρώπινο επίπεδο σύνθεσης ή ο εμπλουτισμός των συνθηκών με διαφορετικά modalities (π.χ. βίντεο, κείμενο) στην περίπτωση του δεύτερου task.

Chapter 1

Introduction

1.1 Problem Definition

By nearly every measure, interest in the area of the so-called **Artificial Intelligence** has exploded over the past decades, drawing a flurry of research activity across the globe. As a whole, this wide-ranging branch of Computer Science involves the development of smart machines capable of performing tasks that typically require human cognitive skills. The availability of massive data, the efficient and affordable computing power and also latest advances in technical domains have made AI systems a growing part of our everyday life, with applications ranging from recommendation engines in online platforms and chat-bots for customer support to self-driving cars.

Recently, the AI research has been expanded in the field of generative modeling, enabling the creation of unbelievably realistic pictures as well as artificially produced news articles. These particularly promising signs in the aforementioned domains that handle data modalities, such as text and image, have inspired researchers to further investigate the generation capabilities of computational systems towards other directions and deal with different types of processable information, such as audio and more specifically **music**, with the latter being the main focus of this thesis. Hence, the research problem that we aim to approach within the scope of Artificial Intelligence can be formally defined as follows:

Automatic Music Synthesis

The process of creating novel musical content in an autonomous manner, i.e. with minimum human intervention.

Music is generally perceived as a form of artistic expression of knowledge, experience, ideas and emotions, established on the arrangement of consonant sounds. To this end, exact interpretations vary considerably around the world, though it is an aspect of all human societies, as stated in [24]. Without consensus over the foundation and the substance of the music itself, the act of composing becomes undoubtedly more challenging. Even from the human perspective, the process of conceiving a piece of music is considered a superior mental task, which has not been decoded or explicitly analyzed yet. Therefore, the key for the automation of this functionality lies in the utilization of Machine Learning techniques. As opposed to handcrafted models, such as grammar-based or rule-based generation systems, the ML approach provides a agnostic learning framework that enables a computational machine to generalize from an arbitrary corpus of music and hence create novel content independently.

Figure 1.1.1: Artificial Artists [23]

Briot et al. [1] identify and tabulate the fundamental aspects of Automatic Music Synthesis problem as follows:

- ***Objective:*** Just as every task, either implemented by human or machine, is inherently related to an ultimate goal, so too the generation objective is the one that determines

and consequentially forms the process of composing musical pieces in an autonomous manner. It mainly refers to the type of musical content to be created, including monophony, polyphony, accompaniment, counterpoint, etc. and also the framework under which it will be used. For instance, the generated music samples might be intended for performance by human(s) (in the case of a musical score) or by machine (in case of an audio file).

- ***Representation:*** Generally, music can be represented in a computer, using various storage forms that employ different modalities, such as text, audio or other image-like symbolic formats. The selection of the proper representation encoding depends on the type of musical information that will be processed and the nature of the generation task that is implemented.
- ***Architecture:*** This term, from a computational scope, refers to the internal structure of the system designed to perform a specific music generation task. The type of the deep model that will be used for this purpose, is inextricably linked to the data format and hence the employed representation method.
- ***Strategy:*** The strategy applied for the automatic creation of novel musical content plays also a crucial role in the formulation of the examined research problem. It typically involves the implemented algorithm as well as other parameters that control to some extent the generation process.
- ***Mode:*** Last but not least, the generation mode mainly refers to other characteristics and features of the music synthesis framework, such as the variability and creativity degree of the model, as well as the interactivity with other systems or even human artists, directly or indirectly. Such functionalities are typically determined by the objective of the examined generation task.

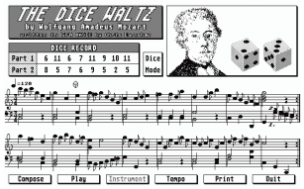


Figure 1.1.2: The Dice Waltz [25]

Despite the latest research towards the development of AI musicians, the idea of automatically generating music is older than the computer itself. It all started in 1787 when Mozart proposed a Dice Game for random sound selections, in order to combine them and finally form a musical piece. As described in [26], he used the dice to collect melody fragments for some minuets and composed nearly 272 tones manually. Later in the 50s, the first piece composed entirely by a computer, *The Illiac Suite*, was generated by a stochastic rule-based system [27]. The score of the piece was created by a computer and then transposed into traditional musical notation for performance by a string quartet. What Hiller and Isaacson had done in the Illiac Suite was to generate certain "raw materials" with the computer, modify them according to various functions and then select the best results via multiple rules [26]. This "generator-modifier-selector" paradigm was also later applied to MUSICOMP [28], one of the first computer systems for automated composition, written in the late 1950s and early 1960s by Hiller and Baker, which created Computer Cantata.

Another pioneering use of the computer in algorithmic synthesis is that of Iannis Xenakis, who created a program that would produce data for his "stochastic" compositions and is presented in great detail at his book *Formalized Music* [29] (1963). Xenakis

used the high-speed machine computations to calculate various probability theories that he applied in compositions like *Atrées* (1962) and *Morsima-Amorsima* (1962). The program would "deduce" a score from a "list of note densities and probabilistic weights supplied by the programmer, leaving specific decisions to a random number generator" [26].

Figure 1.1.3: Illiac Suite [30]

As in the previous example of the Illiac Suite, these scores were performed live on traditional instruments. There are more modern examples, as well, of algorithmic composition without the use of the computer. John Cage, for example, like Mozart, utilized randomness in many of his compositions, such as in *Reunion*, performed by playing chess on a photo-receptor equipped chessboard: "The players' moves trigger sounds, and thus the piece is different each time it is performed" [31]. Cage also delegated the compositional process to natural phenomena, as in his *Atlas Eclipticalis* (1961), which was composed by laying score paper on top of astronomical charts and placing notes simply where the stars occurred, again delegating the compositional process to indeterminacy [32].

However, all those mathematical and stochastic approaches could not successfully model the actual music composition rules that are customary and have been accumulated during human history. Therefore, the previously mentioned methods were useful only when people wanted to create really new and fresh styles, since the generated music tends to be unfamiliar and strange. Consequently, Artificial Intelligence and Deep Learning techniques have become the state-of-the-art approach to Automatic Music Generation, inducing already radical changes in the landscape of Music Industry, as demonstrated in Figure 1.1.4. However, according to most researchers the best is yet to come.

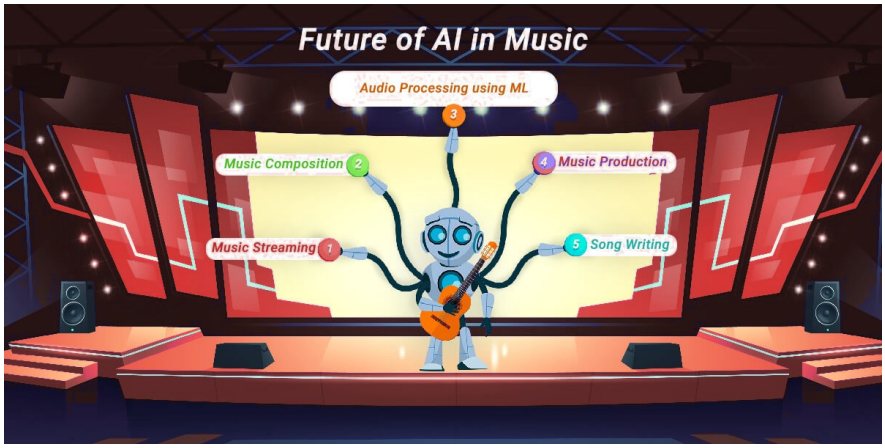


Figure 1.1.4: Applications of AI in Music Industry [33]

1.2 Challenges of the Task

Figure 1.2.1: Music and Memory [34]

Automatically composing realistic and aesthetically harmonic musical pieces is considered a particularly hard problem in the research field of generative modeling for many reasons, the most hampering being the inherent adversity in representing and processing musical content under the framework of a computational machine. In contrast to other modal forms characterized by a more specified structure, such as images, videos and text, music is intrinsically related to more abstract and not especially concrete concepts and senses, such as the emotion, that cannot be thoroughly defined or easily approached from a computational aspect. However, all those notional characteristics collectively enable our brains to make, store, and retrieve memories of music, even when we are not aware of doing so. As stated in [35], music is actually the last thing we forget. Thus, a simple song excerpt can be effortlessly memorized by a human, while at the same time incorporates too many variables for a computer.

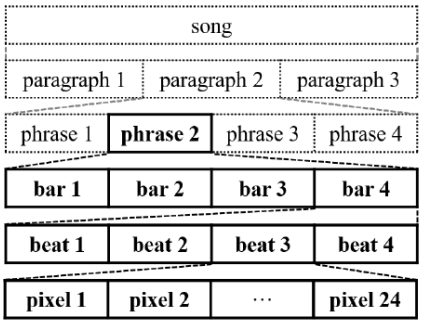
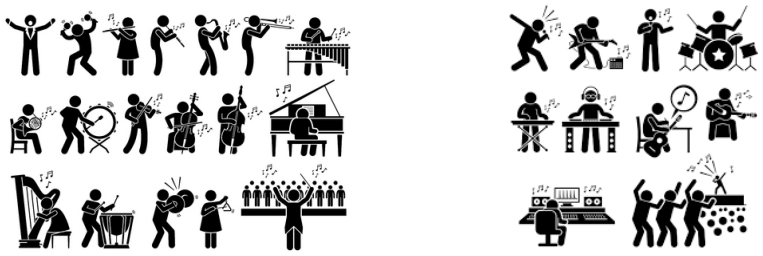


Figure 1.2.2: Hierarchical structure of a music piece [2]

Another major difficulty lies behind the intrinsically hierarchical arrangement of a musical piece. As demonstrated in Figure 1.2.2, a song is abstractly composed of higher-level building blocks, called paragraphs, which can be further subdivided into musical phrases. A phrase in music is defined as a substantial concrete musical thought that has a complete musical sense of its own and therefore is considered as one of the fundamental elements in the structure of a musical composition. Each phrase consists of smaller recurrent patterns, termed bars, which contain beats, formulated by a definite timestep number. As declared in [3, 4], the human brain focuses on such structural motifs, related to coherence, rhythm, tension and the emotion flow, while listening to music and thus a mechanism capable of capturing the aforementioned characteristics and also incorporating the self-reference, which occurs in multiple timescales, from patterns to phrases or even entire sections [5], is critical. However, it can be easily affirmed that the whole hierarchy of a musical piece is structured upon temporal units, as the various elements of musical perception are presented to the listener progressively in time. According to Dong et al. [2], *"music in an art of time"* and therefore modeling the variant temporal dependencies is essential in the context of Automatic Synthesis.

Furthermore, an additional key challenge arises from the fact that a musical piece is typically composed of multiple varying tracks. For instance, a modern orchestra combines instruments of different families, including bowed strings, brass, woodwinds and percussion, while the most common configuration in a rock band includes two guitars, a lead and a rhythm one, a bass, a drum set and possibly lead vocals, as graphically illustrated in Figure 1.2.3. Each individual track in an instrumental ensemble disposes its own musical properties and dynamics. However, all the different track components collectively unfold over time in an interdependent

manner. Various composition disciplines have emerged over the years in an attempt to model the interaction among different instruments. Such approaches are strongly influenced by the corresponding music genre or the historical period to which they are related. In the context of Automatic Music Synthesis, they formed the foundation of hand-crafted methods, which are mainly established on compositional rules. However, the major progress in the fields of Artificial Intelligence and especially Machine Learning have uncovered other more abstract and creative modeling approaches to the concept of multi-track interdependence, which are grounded on the human perception over the creation of musical pieces and vary depending on the respective system implementation.



(a) Classical Orchestra [36] (b) Rock Band [37]

Figure 1.2.3: Multiple Instruments

Another adversity that makes the examined research problem even more challenging emerges from the internal structure and arrangement of sounds in a polyphonic musical piece. As visually demonstrated in Figure 1.2.4, notes are typically presented into grouping formulations, such as *chords*, i.e. harmonic sets of multiple pitches/frequencies that are played simultaneously, *arpeggios*, which constitute a special type of "broken" chord where the tonal components are heard in a sequential form of ascending or descending order, or other *melodic motifs* and *harmonic patterns*. All these musical texture attributes, which inherently incorporate a notion of complexity, cannot be easily captured by a computational machine system. Former approaches, especially in the field of monophonic music generation, which by definition consists of a single unaccompanied melodic line and hence includes much simpler structural formulations, usually employ a chronological ordering of the various note events. However, as a matter of course, such implementations cannot be generalized in tasks of higher complexity, including polyphonic music generation. Therefore, a proper combination of data representation and processing method is required in order to effectively model the structural features of a polyphonic composition.

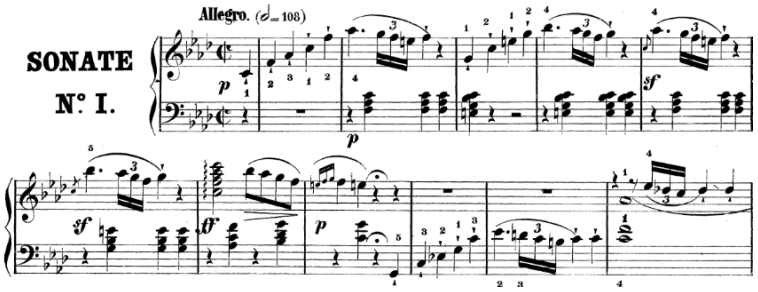


Figure 1.2.4: Beethoven's Piano Sonata in F minor [38]

Lastly, the assessment of music generation systems constitutes another crucial issue in the context of our research problem that uncovers a lot of challenges. Generally, the evaluation of artificial models is based on metrics that quantify their performance in terms of the objective of the implemented task. However, due to the nature of generative AI itself, it is rather hard to establish a standardized definition of concepts such as the performance improvement with respect to the quality of the produced results, that can be applicable to a huge variety of diverse studies in the field. To this end, the objective assessment practice, especially in the research area of music synthesis, still remains largely problematic. On the other hand, evaluation methods grounded on subjective criteria are typically more preferable in the field of generative modeling, since the ultimate judge of creative output is the human (listener or viewer). As Kang et al. [6] clarify, the assessment of music depends on complex and subjective understanding, which cannot be expressed with a simple combination of known rules, such as harmonics and counterpoint. Nevertheless, without consensus over the essence of creativity, the proper design of an experimental methodology that can lead to valid and reliable scientific evidence is often underestimated.

1.3 Thesis Outline & Contributions

This study attempts to investigate and offer further insights into the problem of Automatic Music Synthesis, one of the most challenging topics in the research field of generative modeling. We begin by providing a thorough consideration of the examined subject from a computational perspective and later proceed on a comprehensive analysis of the arising adversities and limitations. The rest of this thesis is organized in 7 chapters total (including the current one), as described below:

- In chapter 2, we provide a detailed overview of the theoretical and technical background required for the full comprehension of the employed methods and the design of the utilized computational modules. More specifically, we emphasize on some fundamental Machine Learning techniques and different types of Artificial Neural Networks, widely applied in the examined research area. We also include a brief analysis of the two ML frameworks that form the basis of our approach to the problem of Automatic Music Synthesis.

- The aim of chapter 3 is to investigate the various aspects of the examined problem, focusing on previous and related works in the field. In particular, we analyze different music representations and attempt to categorize the distinct generation tasks into which the general subject can be further divided, emphasizing on diverse architectures and design choices that can be made. We also list some commonly used datasets and engage on their usefulness. Finally, we present both objective and subjective evaluation methods and elaborate on their nuances and importance.

- Chapter 4 provides a complete overview of the baseline project, on which our proposed framework for automatic creation of novel musical pieces is established. We present the system architecture and the structural characteristics of the integrated modules, as well as the employed training dataset. We also display and discuss the results of the conducted experiments, under the scope of the applied evaluation methods.

- In chapter 5, we focus on the task of Unconditional Generation. More specifically, we develop a framework for the creation of multi-track polyphonic music samples from scratch. At first, we dive into the system architecture, the implementation of the various structural components, as well as the their respective training mechanism. We also elaborate on the employed form of data representation, the utilized dataset and the required preprocessing steps. Lastly, we display and discuss the results of the conducted experiments, under the scope of our proposed evaluation tools.

- In chapter 6, we focus on the task of Conditional Generation. In particular, we extent our previous model to a human-AI cooperative framework, capable of automatically producing accompaniments for user-defined tracks. In this case, we introduce some structural modifications in the system architecture and also emphasize on different training modes. Finally, we proceed in a comprehensive comparison among the resulting model variants, using both objective and subjective evaluation practices and present a thorough analysis of the produced results.

- Chapter 7 draws general conclusions regarding the research sections of the examined problem and also summarizes the experimental results and the contributions of this

thesis. In advance, it briefly discusses our thoughts on potential directions for future work.

At this point, we consider it particularly useful to briefly outline the contributions of our research study, which will be further discussed and thoroughly analyzed in following chapters of this thesis. Our contributions can be divided in two main sections, with respect to the implemented generation task. More specifically:

Unconditional Generation

- Based on the functional concept of MuseGAN, we design a framework for automatic generation of novel musical content in symbolic format from scratch, i.e without subjecting to any prior or supplementary information. The produced samples consist of 5 distinct polyphonic tracks: *Drums*, *Piano*, *Guitar*, *Bass* and *Strings*.
- We perform a customization of our implementation with respect to a group of parameters that define various generative configurations. This process induces an internal internal modulation in the architecture of the included modules depending on the current input arrangement. In this way, our proposed model becomes flexible and adaptable to different generation practices.
- We incorporate into our developed generative system proper auxiliary mechanisms for the monitoring of the the training process, which plays undoubtedly a crucial role in AI modeling. This closer inspection of the applied learning practice enables us to further investigate the behavior of the individual structural units and derive respective conclusions.
- We develop a novel implementation for the existing musical metrics, based on the descriptive analysis presented in the original paper. We further expand our employed objective evaluation system via the inclusion of 3 additional quantitative indicators that emphasize on tonal characteristics and texture attributes of the generated samples.
- We examine the effectiveness of our proposed model over the creation of aesthetic multitrack polyphonic musical pieces from scratch, by conducting a group of experiments with different generative configurations and applying our proposed objective metric system for the evaluation of the produced results.
- We conduct a qualitative study in the form of listening test across 40 subjects, in order to compare our proposed framework with the baseline project from an auditory perspective. We demonstrate that our developed music generation system significantly outperforms MuseGAN with respect to 3 musical criteria: *Musical Naturalness*, *Harmonic Consistency* and *Musical Coherence*.

Conditional Generation

- We extend our original model to a human-AI cooperative framework, by focusing on the task of Accompaniment Generation: given one track derived from the ground-truth distribution of human-composed music samples as conditional information, our proposed system automatically generates the 4 remaining tracks, considering them as the accompaniment parts of the conditional one in terms of rhythmic and harmonic support.

- We follow our customization practice from Unconditional Generation and parameterize the implementation of all structural units involved into our proposed accompaniment framework, including modules from the previously examined task that have been properly modified in order to adjust to the new generation practice and also the additional ones.
- We experiment over multiple variants of our conditional generative framework that mainly differ in terms of the structural components included in the system architecture, the utilized training algorithm and the type of conditional instrument.
- We evaluate the produced results using both objective (musical metrics) and subjective (user study) assessment methods. In this way, we prove that the proposed variations can lead to the creation of novel aesthetic accompaniments and actually contribute to the improvement of the generated musical quality. We also demonstrate that the outcomes derived from the objective evaluation are in agreement with the results of human assessment, indicating that our proposed implementation for the employed metrics provides a meaningful interpretation of the produced music from a computational perspective.

Chapter 2

Theoretical Background

This chapter provides a complete overview of the required theoretical background, concerning the problem of Automatic Music Synthesis. More specifically, in section 2.1 we present some fundamental Machine Learning techniques, while in section 2.2 we focus on different types of Artificial Neural Networks, widely used in the aforementioned research area. Lastly, sections 2.3 and 2.4 include a brief analysis of two fundamental ML frameworks, which form the basis of our approach to the field of Music Generation.

2.1 Machine Learning

The term **Machine Learning** was introduced in 1959 by Arthur Samuel, an IBM employee and pioneer in the field of Computer Gaming and Artificial Intelligence. According to Samuel, Machine Learning is defined as

a field of study that provides computers with the ability to "learn" without being explicitly programmed

In other words, Machine Learning is a branch of Artificial Intelligence (AI) and Computer Science, which focuses on developing algorithms that can access data in order to imitate the way that humans learn. For simple tasks assigned to computers, it is possible for a human to manually create the appropriate algorithm, including all the required steps that need to be executed by the machine for the solution of a specific problem. However, a variety of advanced applications have emerged from the constantly growing field of data science and evolving technology, especially in the fields of medicine, email filtering, speech and face recognition, computer vision and many more. The computational modeling of such tasks is overly complex for humans, making it difficult or even unfeasible to develop conventional algorithms to perform the needed processes. On account of this, it turns out to be much more effective to help the machine develop its own algorithm in order to solve such problems.

A constitutive key in the concept of Machine Learning arises from the sample data, also known as "training data", which the algorithms use to built mathematical models that make predictions or decisions in an autonomous way. Similar to how the human brain derives knowledge and understanding, Machine Learning relies on input in order to apprehend entities, domains and the connections between them and proceed to inferences based on the provided examples. This procedure can reveal trends and patterns within data that allow information businesses to augment or even replace human capabilities in terms of decision making and efficiency optimization. For this reason, the conformation of a dataset used by an ML algorithm is crucial, as it considerably influences the performance of the respective model. A small or particularly specialised dataset can render a model useless for real world applications, where generalization is a matter of great importance.

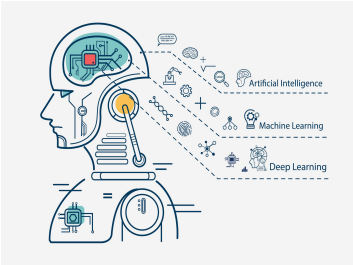


Figure 2.1.1: Artificial Intelligence [39]

From a computational scope, an ML system includes a set of learnable parameters. Their respective values are being updated during the training process in order to improve the

model's performance, measured by the corresponding output of the system when samples from the dataset are given as input. According to UC Berkeley, the learning framework[1] of an ML algorithm consists of roughly 3 components:

- **Decision Process**: It can be defined as the recipe of calculations or other computational steps that follows the ML algorithm in order to produce an estimate about the type of pattern prevailing on the input data, which can be either labeled or unlabeled.
- **Performance Index or Error Function**: It quantifies the efficiency of the model with respect to the examined task. As regards applications where the known samples are available, the formation of the aforementioned function cam be simple or even trivial (e.g. accuracy at a classification task), while the involved elusiveness and vagueness in various problems of different types (e.g. quality of generated musical piece) render this particular process complex and challenging.
- **Updating or Optimization Process**: It can be defined as the method that updates the values of the model parameters, based on the output of the cost function, in such a way that the discrepancy between the known samples and the model estimate is reduced. This procedure is repeated until a specific performance threshold is met, concerning the model fitness to the ground-truth examples.

The variety of different Machine Learning approaches can be categorized by the presence or absence of human influence on raw data, regarding the inclusion of a reward, the utilization of feedback or the existence of labels. There are three primary training practices, Supervised Learning, Unsupervised Learning and Reinforcement Learning, which will be discussed more explicitly in the following sections.

2.1.1 Supervised Learning: More Control, Less Bias

Supervised Learning (SL) includes a set of algorithms that leverage "labelled" training data in order to predict outcomes accurately. The term "labelled" data refers to input data already tagged with the correct output. More specifically, each sample is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). This structure of training examples provides the algorithm with the ability to analyze the corresponding dataset, capture correlations and associations among samples and exploit this kind of information in order to produce an inferred function which is able to predict future events.

The concept of Supervised Learning method is graphically depicted in Figure 2.1.2 through an example. We consider a dataset of different shape types, which includes square, circle and triangle. The aim is to train a model so that it identifies each shape. A prospective algorithm is described as follows:

- If the examined shape has 4 sides and all the sides are equal, then it will be labelled as a Square.
- If the examined shape has 3 sides, then it will be labelled as a Triangle.

[1] A visual Introduction to Machine Learning by R2D3 can be found in this website

- If the examined shape has no sides, using the evident definition, then it will be labelled as a Circle.

During training process, the parameters of the algorithm are updated until the model fits the training data appropriately. This is achieved by comparing the current output with the correct one, as the input data is fed into the system, in order to determine errors and modify the model parameters accordingly. As a result, when a new unseen shape instance is tracked during validation, the machine is qualified to predict the correct corresponding class label, based on the number of its sides.

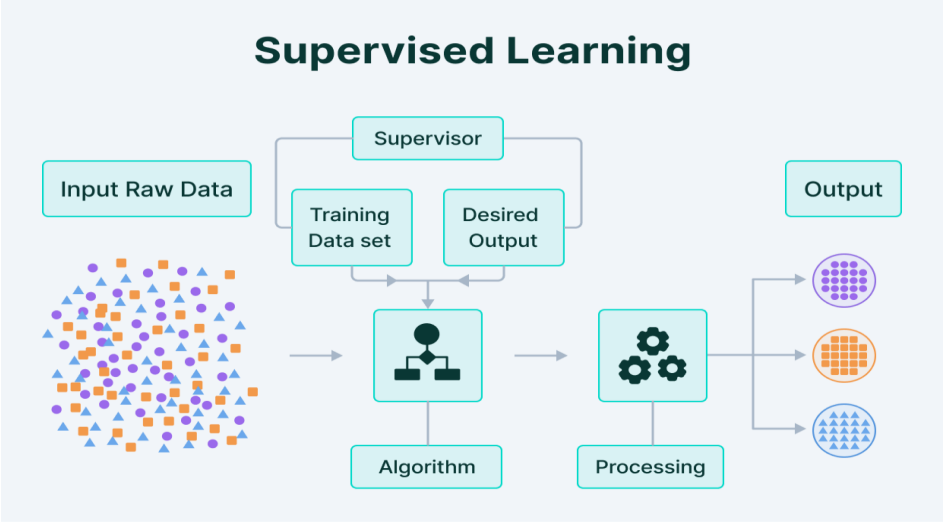


Figure 2.1.2: Supervised Learning Diagram [40]

It may be assumed that training data operate as the supervisor that tutors the machine to predict the output correctly, similar to the concept of a student learning under the supervision of the teacher. In effect, the model gains the ability of generalization from the training data to unseen situations in a "reasonable" way, following the archetypes of human learning process. This acquired attribute of computational systems is employed by various organizations and industrial companies for building applications that solve real-time problems at scale. Applications of that kind include *Text categorization*, *Face Detection*, *Signature recognition*, *Customer discovery*, *Spam detection*, *Weather forecasting*, *Predicting housing prices based on the prevailing market price*, *Stock price predictions*, etc.

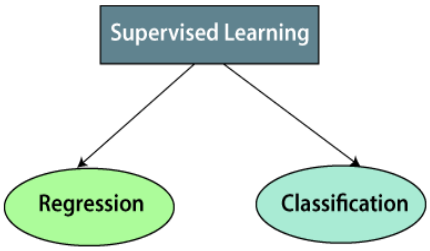


Figure 2.1.3: Categorization of Supervised Learning Problems [41]

Supervised Learning can be further divided into the following two groups of problems, as demonstrated in Fig. 2.1.3:

Regression

Regression is a technique for investigating the association between independent variables (features) and a dependent variable (outcome). From a visual scope, it generally involves finding and drawing a line of best fit through the given data points, as shown in Fig. 2.1.4. The distance between each point and the line is minimised in order to achieve the most suitable result.

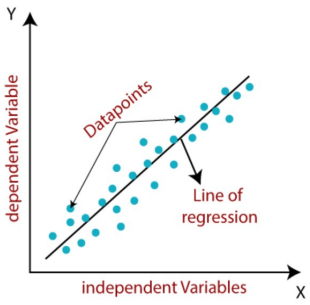


Figure 2.1.4: Regression [42]

Regression can be employed as a method for predictive modelling in the field of Supervised Machine Learning, concerning algorithms that predict continuous outcomes, that is, real-valued output variables, such as unique numbers, dollars, salary, weight or pressure, based on previous data observations. More specifically, regression analysis provides insights into the effect of a specific independent variable on the value of the dependent variable, when all other independent variables are held fixed.

There are multiple approaches for performing regression in the field of Machine Learning, incorporating different popular algorithms. Those distinct techniques may include different numbers of independent variables, process different types of data, or even assume a different relationship between the independent and dependent variables. Nevertheless, each variant has its own importance on diverse use case scenarios. Some of the most prevailing regression methods can be grouped into the categories graphically displayed in Figure 2.1.5.

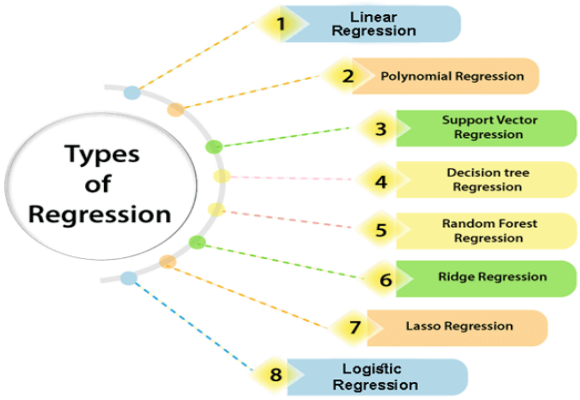


Figure 2.1.5: Types of Regression [43]

Classification

In the context of Machine Learning, Classification refers to the process of identifying the category that corresponds to a specific sample-observation. From a mathematical point of view, classification predictive modeling is the task of approximating the mapping function

(f) from input variables (X) to categorical output variables (y), that best fits the respective training dataset. As such, the training set should be sufficiently representative of the problem and contain many examples of each class label. Throughout this procedure, the model is able to recognize specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined. The main goal is the assignment of new unseen samples into the formatted categories. Classification methods can be applied on both structured or unstructured data.

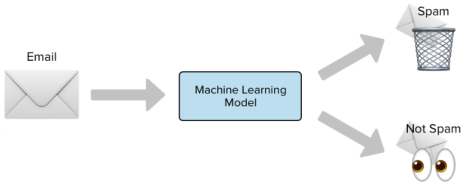


Figure 2.1.6: Spam Detection [44]

Spam Detection is a typical example of a supervised machine learning problem that leverages the method of Classification. A spam-detector algorithm must find a way to filter out spam mails, avoiding at the same time the flagging of authentic messages that users want to see in their inbox. This means that the machine learning model should be supplied with a set of examples of spam and ham (i.e. non-spam) messages and explore the relevant patterns that separate the two different categories, as presented in Fig. 2.1.6. Heart Disease Detection can be also identified as a classification problem. In this case, the model uses training data in order to identify the relation of the given input variables to the corresponding class. Once the classifier is trained accurately, it can be used to detect heart disease for a particular patient.

Both the aforementioned applications belong to the category of **Binary Classification**, since only two class labels are encountered (e.g. spam-non spam). On the other hand, **Multi-class Classification** includes tasks characterized by three or more distinct classes, such as Face Classification and Optical Character Recognition. An additional variant of the typical (multi-class) classification problem is established on the assignment of multiple labels to each data sample and it is called **Multi-label Classification**. For instance, in the task of Object Detection, the model predicts the presence of multiple semantic objects contained in every digital image-sample, by locating their position, indicating their scale and assigning multiple labels corresponding to different entity categories, such as humans, buildings, or cars.

The wide variety of classification tasks has induced the development of multiple different algorithms that can be used in order to solve the corresponding problems. The choice depends on the nature of the application and the form of available data. The most commonly used classification algorithms are demonstrated in Figure 2.1.7.

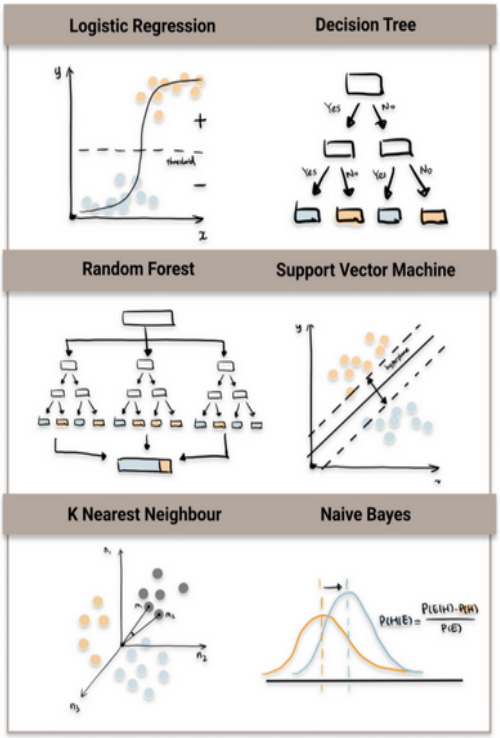


Figure 2.1.7: Classification Algorithms [45]

The statistical ability of an algorithm to adapt properly to new, previously unseen data, drawn from the same distribution as the training ones, either in the case of *Regression* or *Classification*, is measured by the so-called generalization error. This quantity is employed in the evaluation procedure of the model's performance in the context of a cross-validation formula.

Cross-validation, also known as Rotation Estimation or Out-of-Sample Testing, belongs to a group of model validation techniques for assessing the generalizability of the results produced by a statistical analysis to an independent data set. In particular, it is a resampling method that uses different portions of the dataset to test and train a model on different iterations. One round of cross-validation involves partitioning data into complementary subsets, performing the analysis on one subset, called the training set, and validating the analysis on the other subset, called the validation set or testing set. For the purpose of variability reduction, in most methods multiple rounds are performed using different partitions and the individual validation results are combined to provide an estimate of the model's predictive performance. In summary, cross-validation is used to derive an as much as possible accurate estimate of the model's fitness in prediction and flag various problems, such as overfitting.

Overfitting is a concept in data science that constitutes a common pitfall for deep learning algorithms. From a mathematical perspective, it can be defined as the production of an analysis that corresponds too closely or exactly to a particular set of data. In other words, it occurs when a model attempts to fit the training data entirely and results in memorizing the data patterns, the noise or any other random fluctuations and interpreting them as part of underlying data structure. The problem is that these notions cannot apply in the case of unseen data scenarios, affecting negatively the model's ability to generalize, as it fails to predict future observations reliably.

Overfitting essentially arises from the difference between the criterion used for selecting the model and the one used to assess the suitability of a model. For instance, a model might be selected by maximizing its performance on some set of training data, and yet its suitability might be determined by its ability to perform well on unseen data. Furthermore, the non-parametric and non-linear methods, used by these types of machine learning algorithms, can easily amplify the establishment of unrealistic models with lack of generalizability, due to the flexibility incorporated in the learning process of a target function.

Underfitting, the counterpart of overfitting, is another major problem that afflicts supervised machine learning algorithms. It occurs when a data model is unable to capture the relationship between features of a dataset and output variables accurately, generating a high error rate on both the training set and unseen instances. An underfitted model might have a simple structure that cannot establish the dominant trend within samples, due to short training time, lack of features, uncleaned training data containing noise and outliers or overwhelming regularization. Therefore, it results in problematic or erroneous outcomes on new data and cannot be leveraged for classification or prediction tasks. Since this behavior can be identified during training procedure, underfitted models are usually easier to track than overfitted ones.

In both scenarios, the model generalizes poorly to unseen data, which is an important factor concerning real-world applications. If we define *bias* as the quantity that measures the

difference between the model's prediction and the target value and *variance* as an indicator of the inconsistency of different predictions over varied datasets, then we can affirm that an overfitted model exhibits high variance and low bias. This is due to the fact that it can represent the training data accurately but lacks at the same time the ability to generalize at different testing sets. On the other hand, unlike overfitting, underfitted models experience high bias and low variance, due to their simplified structure that overlooks regularities in data and fails to approximate the underlying function. This demonstrates the bias-variance trade-off, which represents the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set. On account of this, the goal is to track down the "sweet spot" between underfitting and overfitting. An illustration of the forenamed observations is presented in Figure 2.1.8.

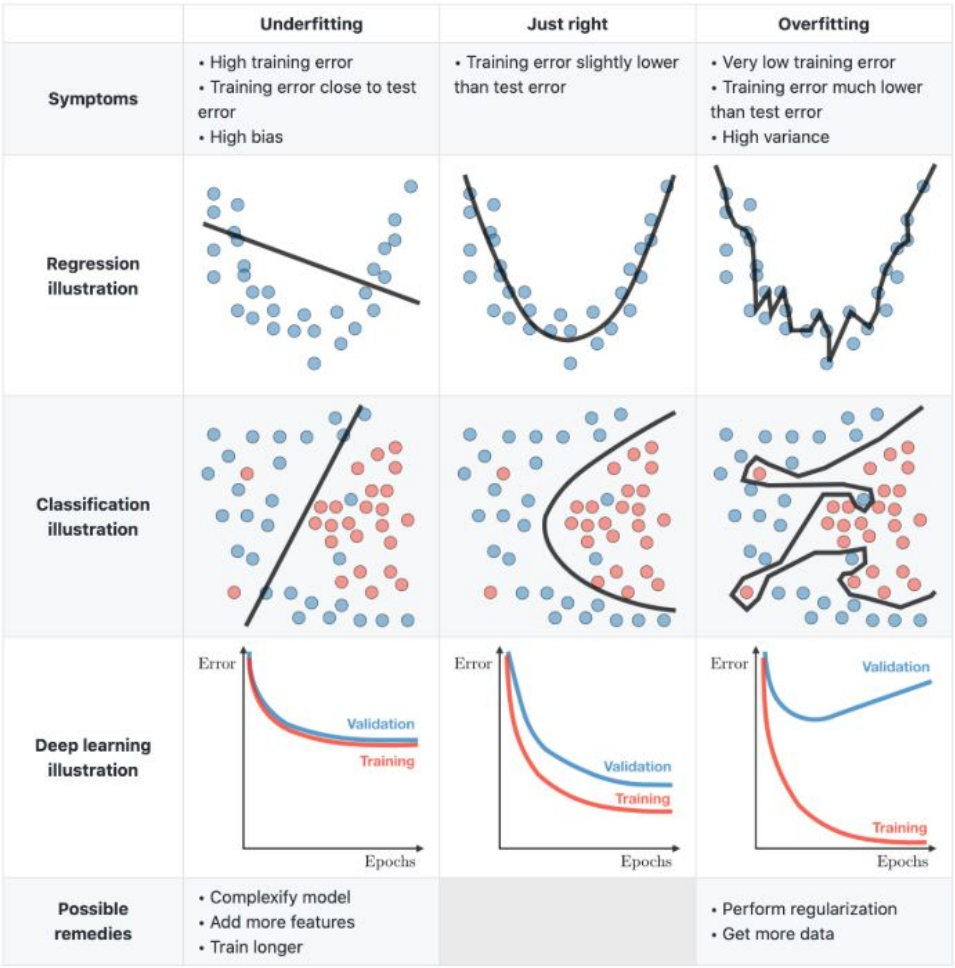


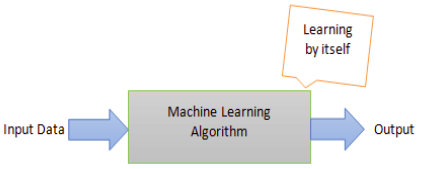
Figure 2.1.8: Overfitting and Underfitting Overview [46]

2.1.2 Unsupervised Learning: Speed and Scale

Unsupervised Learning (UL) encompasses a group of machine learning algorithms that analyze and cluster unlabeled or unclassified data, namely samples which are not given label,

and thus do not correspond to any predefined target output. As the name suggests, this procedure is accomplished without any human surveillance or superior guidance, following the archetypes of the learning mode of human brain. The principal idea is that through the exposure to large volumes of varying data, the machine discovers hidden patterns and insights, identifies correlations and relationships among samples, detects similarities and differences in information and leverages all these features in order to build a compact internal representation of the underlying data structure.

In the context of Unsupervised Learning, algorithms are left to their own devices to determine disparate or interesting aspects about the input features, by exploring autonomously the given dataset. In this way, without any prior knowledge, unsupervised methods are capable of inferring a function that describes the intrinsic data distribution. This notion of self-organization renders such kind of systems the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation and image recognition.



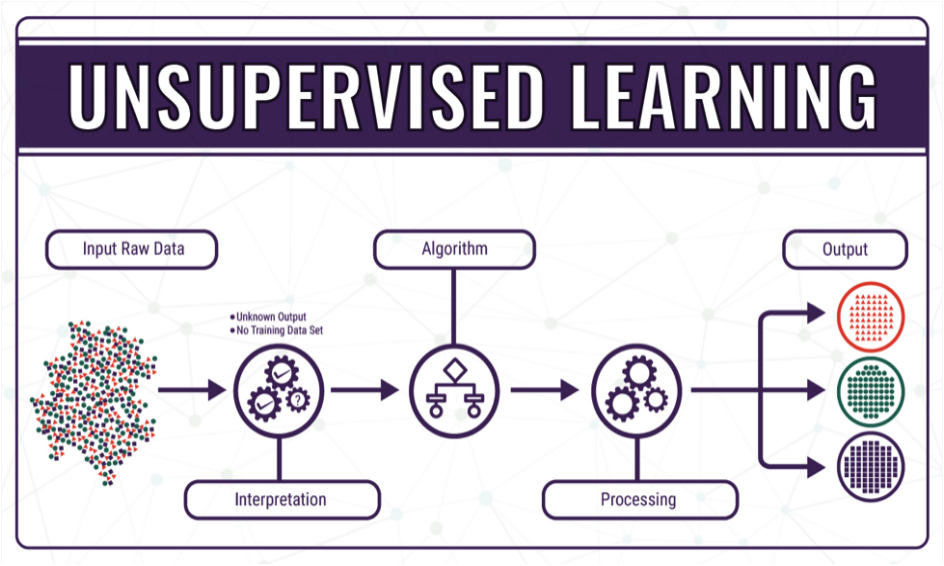


Figure 2.1.9: Unsupervised Learning Diagram [47]

The overall mechanism of Unsupervised Learning systems is pictorially demonstrated in the diagram of Figure 2.1.9. Let us again consider a dataset containing different types of geometrical shapes, such as squares, triangles and circles, which are not accompanied with extra information in the form of a tag. Let us also assume that the aforementioned set of data samples comprises the input of a model that utilizes the method of Unsupervised Learning. The task of the corresponding algorithm is to identify the input instances, by performing a clustering of the geometrical shapes into groups, based only on similarities and associations among them. To this effect, a stage of data interpretation is required in order to extract hidden patterns and determine features, which contribute to the formation of categories at the processing step. Contrary to the Supervised Learning concept, where the labels can impose a strategy for the problem solution, in this case the machine is programmed to learn by itself,

since an equivalent set of generally applicable instructions for the grouping of the input data cannot be defined.

One of the most important benefits of unsupervised learning techniques concerns the structure of the utilized dataset. Unlabeled data do not require any kind of human intervention or annotation, which is a significantly time-consuming procedure; as such, they constitute the most common type of dataset regarding the majority of real-world applications. Furthermore, such methods can be used for modeling more complex tasks compared to supervised learning, since they have the ability to detect and reveal hidden patterns and intrinsic features of the underlying data distribution, that can facilitate the approach of compound problems and contribute to the categorization part.

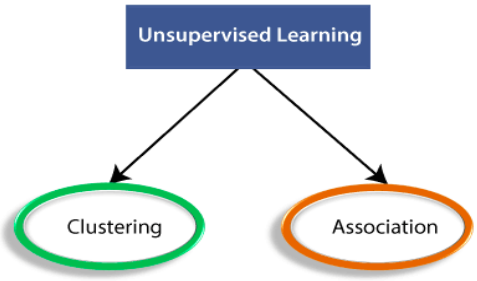


Figure 2.1.10: Categorization of Unsupervised Learning Problems [48]

Unsupervised Learning can be broadly categorized into two classes of problems, which are presented in Figure 2.1.10 and will be more extensively discussed further down.

Clustering

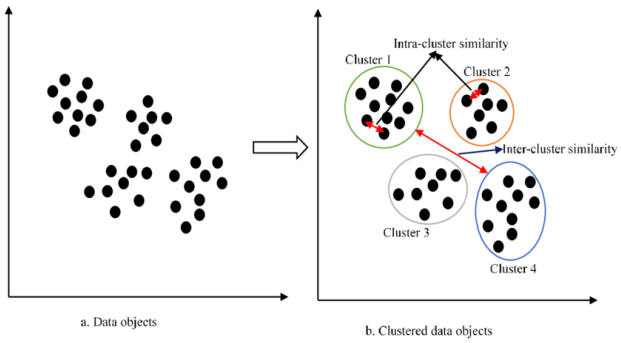


Figure 2.1.11: Clustering [49]

Clustering can be considered an important concept when it comes to Unsupervised Learning. This method involves organizing unlabelled data into groups, called clusters, by discovering patterns or detecting the inherent structure that may exist in the collection of input instances. More specifically, Cluster analysis can be defined as the task of grouping objects into clusters in such a way that samples in the same group are more similar, in some sense, to each other than to those in other groups. Therefore, a cluster is a collection of objects which are "similar" between them and "dissimilar" to the objects belonging to other clusters. From a computational aspect, Cluster analysis can be formulated as an iterative process of knowledge discovery or interactive multi-objective optimization, that tracks down commonalities among data points and performs a categorization, based on the presence and absence of those attributes.

Clustering is the main task of Exploratory Data Analysis and a common method for Statistical Data Analysis, used in many fields, including Pattern Recognition, Image Analysis,

Information Retrieval, Bioinformatics, Data Compression and Computer Graphics. It can be performed by a variety of algorithms that approach the constitution of clusters as well as the procedure of their formation in a different manner. Popular notions of clusters include groups with small distances between members, dense areas of the data space, intervals or particular statistical distributions. The selection of the proper clustering algorithm in conjunction with the setting of multiple parameters, such as the distance function to be used, a density threshold or the number of expected clusters to be formed, depend on the structure of the input object collection, the data format and the nature of the application that will exploit the produced results. A visual overview of the most prevailing clustering algorithms employed in Data Science and Mining is presented in Figure 2.1.12.

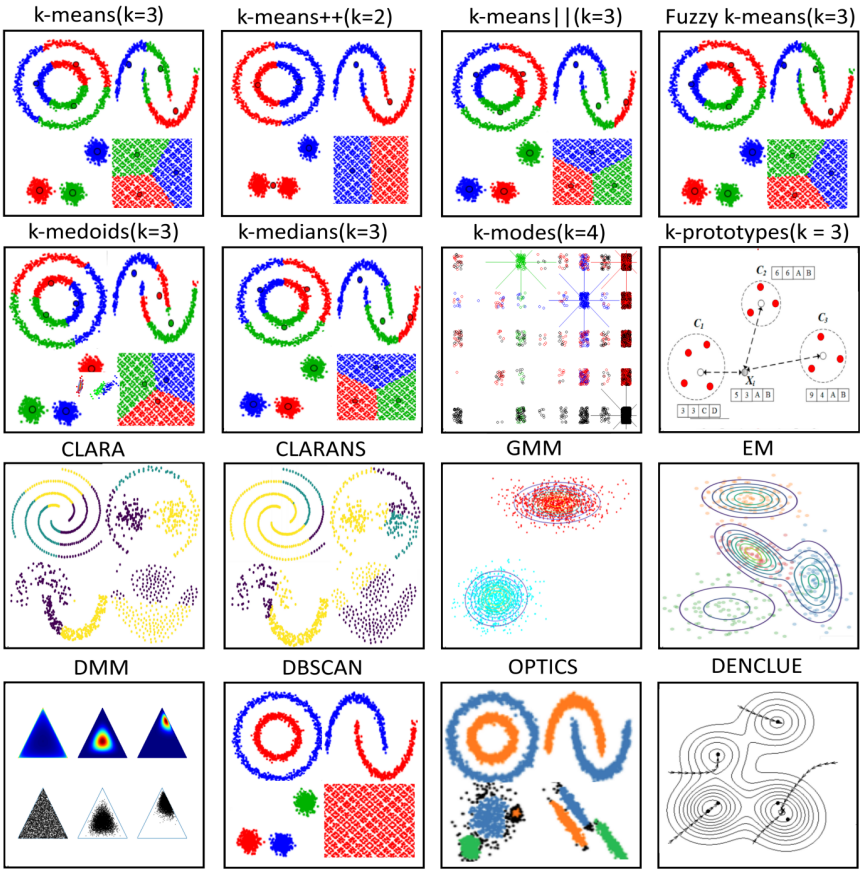


Figure 2.1.12: Clustering Algorithms [50]

Association

Association is a rule-based unsupervised method for discovering interest relations between variables, hidden in large datasets. These correlations are usually represented in the form of rules or frequent item-sets. It is a descriptive, not predictive, formula that detects new and engaging insights between different objects in a set, frequent patterns in transaction data or any sort of relational database. Association rules are employed in multiple application areas, such as Web Usage Mining, Intrusion Detection, Continuous Production and Bioinformatics.

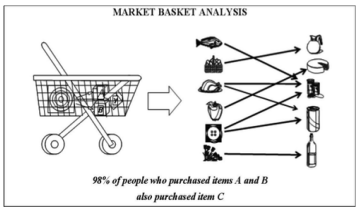


Figure 2.1.13: Market Basket Analysis [51]

In addition to the aforementioned problems and tasks, this technique is most frequently utilized under the framework of Market Basket Analysis, as shown in Figure 2.1.13. In this case, association rules allow for discovering regularities between products in large-scale transaction data, recorded by point-of-sale (POS) systems in supermarkets. For instance, the rule $\{onions, potatoes\} \implies \{burger\}$ formatted from the sales data of a specific store, indicates that if a customer buys onions and potatoes together, they are also likely enough to buy hamburger meat. Such kind of information comprises the basis for decisions and actions, concerning marketing activities, such as promotional pricing or product placements. Understanding consumption habits of customers enables businesses to develop better cross-selling strategies and recommendation engines.

In general, association rules are formulated by identifying frequent if-then patterns among data and using a particular criterion under Support, Confidence and Lift in order to determine the most significant relations. The aforementioned quantities can be defined as follows:

- **Support** expresses the frequency of an item appearance in the given dataset.
- **Confidence** indicates the number of times the if-then statements are found to be verified.
- **Lift** shows the number of times the if-then statements are expected to be verified and is introduced in order to compare the actual and the expected Confidence.

2.1.3 Reinforcement Learning: Rewards Outcomes

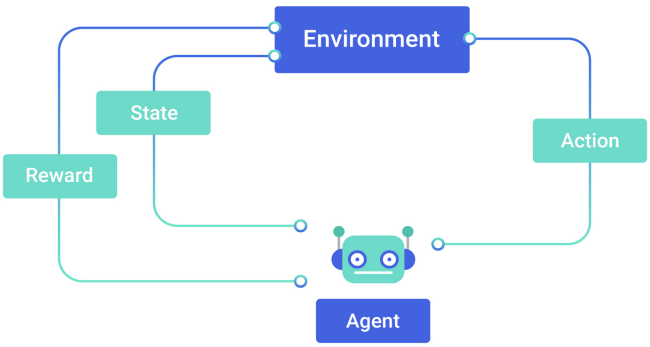


Figure 2.1.14: Reinforcement Learning Diagram [52]

Reinforcement Learning (RL) constitutes a feedback-based Machine Learning technique that enables an intelligent agent to learn how to act inside an interactive environment by trial and error, using feedback from its own experiences, in order to maximize the notion of cumulative reward. Given a set of prescribed rules for accomplishing a distinct goal, this method allows machines and software programs to automatically determine the ideal behavior that maximizes their performance, through a procedure of seeking positive rewards, which

can be received when a beneficial toward the ultimate goal action is performed and avoiding punishments, which can be received in the opposite situation.

As demonstrated in Figure 2.1.14, the fundamental elements that describe a Reinforcement Learning problem are the following:

- **Agent**: An entity that has the ability to explore the environment and operate upon it, in order to perform a specific task.
- **Environment**: The physical or virtual world that surrounds the agent.
- **State**: It describes the current situation and is modified by the actions of the agent.
- **Action**: It is the move made by the agent which alters the status of the environment.
- **Reward**: The evaluation of an action, which can be either positive or negative (it is returned to the agent from the environment in the form of feedback).
- **Policy**: It is the strategy that the agent applies in order to determine the next action, based on the current state. In other words, it constitutes a mechanism for mapping states to actions.
- **Value**: Expected reward that the agent would receive by taking an action in a particular state.

Both Supervised and Reinforcement Learning methods utilize a form of mapping between the input and the corresponding output. However, in the case of Supervised Learning, the feedback provided to the agent is a set of parameter reformulations that rectify its course to the global goal, since the structure of labeled input-output data pairs impose the explicit correction of all sub-optimal actions. On the other hand, Reinforcement Learning uses rewards and punishments as signal indicators of good and bad behavior respectively, which are both acceptable elements of the training process, since the main focus is on finding a balance between exploration of uncharted territory and exploitation of the current knowledge. This is called **Exploration vs Exploitation trade-off** (Figure 2.1.15) and represents the dilemma that an agent faces at every step of the algorithmic process, between exploring new states and maximizing its overall reward at the same time. In other words, the procedure of establishing an optimal policy with respect to a particular task may involve some short-term sacrifices, which nevertheless enable the agent to collect adequate amount of information and as a consequence make the best overall decision in the future.

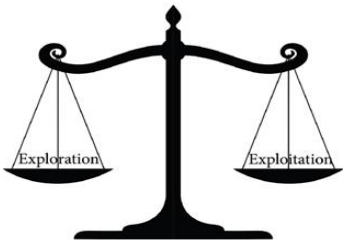


Figure 2.1.15: Exploration vs Exploitation trade-off [53]

As regards the comparison between Unsupervised and Reinforcement Learning, it can be affirmed that those two methods differ in the matter of the objective of the whole procedure. As explained before, the goal of Unsupervised Learning techniques is to detect similarities and differences and discover hidden patterns among unstructured data, while in the case of Reinforcement Learning the principal aim is to track down the most appropriate action model that maximizes the total cumulative reward in the context of performing a specific task.

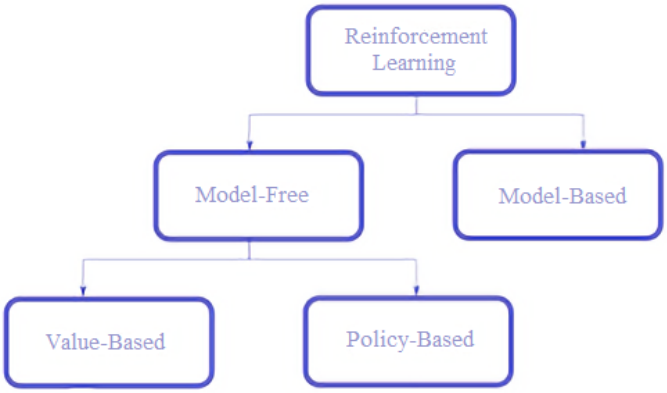


Figure 2.1.16: Reinforcement Learning Approaches

The various approaches concerning the implementation of a Reinforcement Learning system in Machine Learning can be broadly divided, in accordance with the diagram of Figure 2.1.16, into the following categories:

Model-based methods

In the framework of model-based RL approaches, the agent is enabled to construct a functional representation of its environment. By performing actions and observing the outcomes that include the next state and the immediate reward, the agent is capable of learning the aforementioned virtual model and hence formulating the optimal behaviour in an indirect manner, overcoming at the same time the issue of lack of prior knowledge. Since the model representation depends on the corresponding environment, which can vary among different problems, a generic algorithm cannot be established for this kind of methods.

Model-free methods

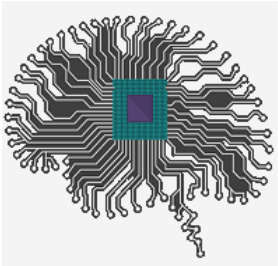
In the case of model-free RL systems, the agent does not take into account the environment's response to local actions and only concerns itself with determining which action to perform given a specific state. More precisely, it does not consider predictions derived from environmental information, that may refer to the expected next reward or the full distribution of next states and next rewards, laying emphasis on the procedure of learning instead of planning. Since it can be extremely difficult to construct a sufficiently accurate representation of the environment, as required by the model-based strategy, model-free methods have been proven significantly useful for a wide variety of problems, surpassing approaches that incorporate a larger degree of complexity.

From a computational aspect, every policy employed by a Reinforcement Learning algorithm, can be described by two functions: the **State Value (V-value)**, which maps each state to the corresponding expected reward, considering actions performed by the agent in accordance with the given policy from the input state onward and the **State-Action Value (Q-value)**, which maps a state-action pair to the corresponding expected reward, considering all previous pairs from the input state and beyond, according to the given policy. Therefore, discovering the optimal policy and finding the optimal V-value or Q-value are equivalent procedures,

as regards the task to be resolved. Under this concept, model-free methods can be further partitioned as follows:

- **Value-based**: Value-based algorithms aim at specifying the optimal State-Action Value or the optimal State Value. In this way, the optimal policy is indirectly established, since it can be derived from the aforementioned functions.
- **Policy-based**: Policy-based algorithms track down directly the optimal policy, by building an explicit representation of it during learning.

2.2 Artificial Neural Networks



The term **Artificial Neural Network (ANN)**, or simply **Neural Network (NN)**, refers to a family of Machine Learning and Artificial Intelligence algorithms, established on biological studies concerning the structure and the respective functionalities of the human brain and nervous system. As the name suggests, an Artificial Neural Network can be defined as a computational system that models the human brain. The concept of Artificial Neural Networks was first introduced in 1943, when two mathematicians, Warren McCulloch and Walter Pitts, built a circuitry system intended to approximate simple biological operations of the human brain. This notion of emulating intellectual processes in a computational form and enable machines to understand and learn things in order to perform tasks in a human-like manner has flourished over the years, rendering ANNs the fundamental tool for Deep Learning algorithms.

From a constructional perspective, an Artificial Neural Network comprises a collection of interconnected units or nodes, called *artificial neurons*, which are organized in multiple layers. Similar to the mechanism of synapses in the biological brain, each connection, also called *edge*, in the equivalent artificial system can transmit a signal to other neurons. As demonstrated in Figure 2.2.2, an artificial neuron receives signals from other neurons, processes them and broadcasts the result of this procedure at neurons connected to it. The aforementioned signals are real numbers and the output of each neuron is computed by some function of the sum of its inputs, mimicking the indeterminate behaviour of biological neurons, which are enabled and disabled irregularly when a particular operation is performed. Neurons and edges are typically characterized by a weight value that adjusts as learning proceeds. This quantity increases or decreases the strength of the signal at a connection, representing the significance degree of the corresponding input information. Neurons may also have a threshold that determines whether the respective aggregate signal is to be transmitted.

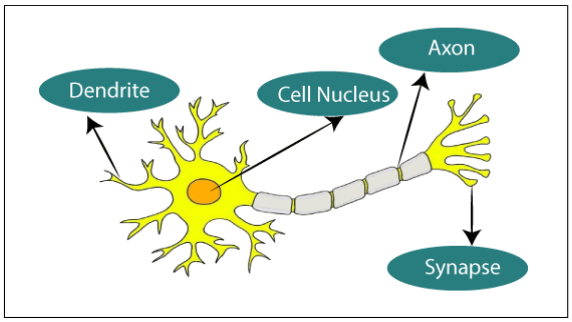


Figure 2.2.1: Biological Neuron [54]

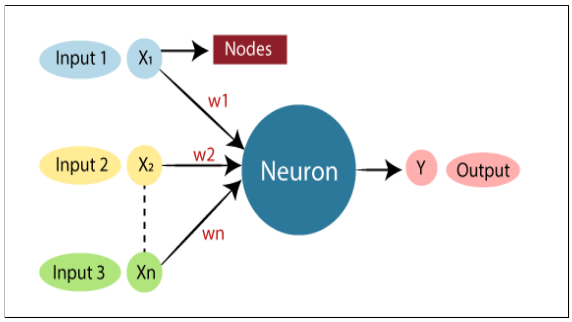


Figure 2.2.2: Artificial Neuron [54]

In comparison with the biological model, depicted in Figure 2.2.1, it may be stated that the correspondence between the two structures can be summarized as follows:

- Dendrites ⇔ Inputs
- Cell Nucleus ⇔ Neuron

- Axon ⇔ Output
- Synapse ⇔ Interconnections

As mentioned before, artificial nodes are usually aggregated into layers. The neurons contained in each tier can be considered as parallel processors operating on the same input information. Different layers may perform varying transformations on their respective inputs. The architecture of a multilayer neural network is graphically presented in Figure 2.2.3.

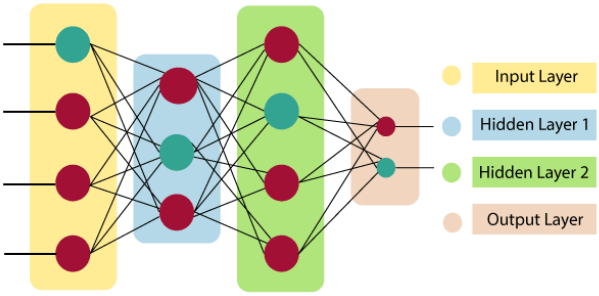


Figure 2.2.3: Multilayer Neural Network [54]

- **Input Layer**: As the name suggests, it is the first layer, which receives the input data in several different formats.
- **Hidden Layer**: Hidden layers represent intermediate layers of the network, where the included artificial neurons receive a set of weighted inputs and produce an output through an activation function. These node tiers usually perform processing calculations in order to extract hidden features or to detect patterns among input data. Hidden layers constitute a typical part of nearly any neural network structure, since they essentially simulate the operations of the human brain.
- **Output Layer**: As the name suggests, it is the final layer, which produces the ultimate result of the whole procedure.

There are many different types of Artificial Neural Networks. Although they all share the same objective of modeling the human brain activity, they can differ in multiple aspects, including the degree of complexity, the structure of artificial neurons and the connections between them (e.g. node density, network depth, activation filters, etc.), the data flow or the use cases. The most prevailing categories of Artificial Neural Networks, especially in the research area of Automatic Music Synthesis which constitutes the principal subject of this thesis, will be examined in greater detail at the subsequent sections.

2.2.1 Perceptron

The **perceptron** constitutes the fundamental building block of a typical Artificial Neural Network, since it comprises only a single neuron, as displayed in Figure 2.2.4. It was initially implemented in the mid of 20^{th} century by Mr. Frank Rosenblatt, as a machine intended to perform specific computations in order to detect properties of the input data distribution or business intelligence.

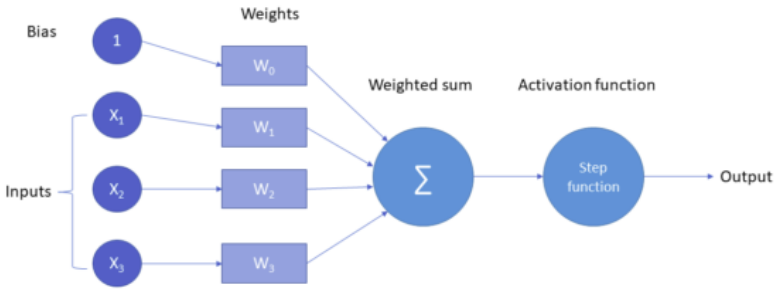


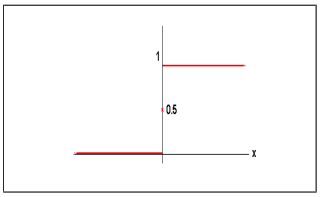
Figure 2.2.4: Perceptron [55]

As demonstrated in Figure 2.2.4, the perceptron unit consists of the following main components:

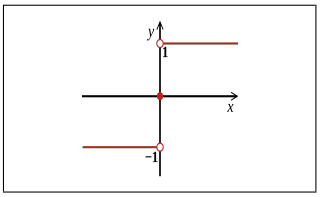
- **Input Nodes or Input Layer**: This module feeds the initial data into the perceptron system for further processing. Each input node corresponds to a real numerical value, representing some feature of the input.

- **Weights & Bias**: The weight parameter represents the influence degree of the associated input neuron to the computational procedure of determining the system's output. In order to mathematically model this correlation, weights are linearly combined with the respective input values. Bias is an adjustable, numerical term added to the perceptron's equation formula, that allows the shifting of the activation function along the x axis and can be considered as the decision threshold.

- **Activation function**: It is usually a non-linear partially differentiable function that maps the linearly aggregated form of input attributes to a specific interval or a defined set of values, determining in this way the output of the respective node. Activation functions introduce a notion of non-linearity in the processing mode of neural networks, as the initial module of artificial neurons performs just a linear transformation of the input. This additional feature is particularly crucial, since it enables machines to deal with more complex tasks, that cannot be essentially approached through a linear method. As regards the case of perceptron systems, the most widespread activation function is the Step function, which is illustrated in Figure 2.2.5a. A common alternative is the Sign function, which is depicted in Figure 2.2.5b.



(a) Unit Step Function



(b) Sign Function

Using mathematical notation, the perceptron mechanism can be described as follows:

Let $\mathbf{x} \in \mathbb{R}^n$ be a real-valued vector which represents the input features and $\mathbf{w} \in \mathbb{R}^n$ the internal real-valued vector of the system's weights.

1. The dot product of the aforementioned vectors is computed:

$$\mathbf{w^T} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

   where n is the number of input attributes.
2. The bias term b is added to the produced value:

$$\mathbf{w^T} \cdot \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b$$

3. The result is applied to the activation function, generating the perceptron's output (*we utilize Step Function for the formulation*):

$$y = f(\mathbf{x}) = \begin{cases} 1, & \mathbf{w^T} \cdot \mathbf{x} + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

Evidently, the value of the perceptron's output $f(\mathbf{x})$ can be considered as an indicator of the class in which the input $\mathbf{x}$ is assigned, establishing the framework of a typical binary classification task. More specifically, in order to classify $\mathbf{x}$ as a "positive" instance, the respective weighted compound of the input attributes must produce a value greater than $-b$, leading in this way the perceptron's neuron over the threshold value. Topologically, bias determines, in essence, the position of the decision boundary, which partitions the underlying vector space into two distinct groups. Furthermore, due to the fact that the classification rule is based on a linear combination of the input features and the corresponding weights, the perceptron unit can be employed as a type of linear classifier.

The main objective of perceptron's training process is to learn the threshold function

$$\boxed{f(\mathbf{x}) = \mathbf{w^T} \cdot \mathbf{x} + b}$$

which by definition maps the real-valued input vector $\mathbf{x}$ to a single binary output value $f(\mathbf{x})$, that designates the class of the respective data instance. Spatially, the aforementioned function represents the decision surface, which partitions the generalized data space into two regions in accordance with the class distribution, as demonstrated in Figure 2.2.6 for the 2D case. Therefore, the training algorithm of the perceptron model aims at computing the weight vector $\mathbf{w}$ and the bias parameter b, so that all samples included in the training set are correctly classified by the corresponding predictor function f. On account of this fact, a labeled dataset is required, rendering the Perceptron Rule one of the principal

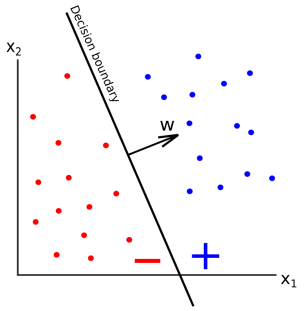


Figure 2.2.6: Illustration of Decision Boundary [56]

algorithms within the framework of Supervised Learning, which has been explicitly presented in Section 2.1.1.

The Perceptron training rule is presented below. In practice, it optimizes the values of $\mathbf{w}$ and b with respect to a cost function $L(d, f(\mathbf{x}))$, which typically corresponds to the minimization of misclassifications in the training set.

Algorithm 1: Perceptron Training Rule

Definitions:

- r denotes the **learning rate** of the perceptron. It is a hyperparameter which determines the step size at each iteration of the algorithm, while moving toward a minimum of a loss function. It is usually between 0 and 1, with larger values implying more volatile weight updates.
- $y = f(\mathbf{z})$ represents the perceptron's **output** for an input vector $\mathbf{z}$.
- $D = \{(\mathbf{x_1}, d_1), \ldots, (\mathbf{x_s}, d_s)\}$ is the labeled **training set** of s data samples, where $\mathbf{x_j}$ symbolizes the n-dimensional input vector and d_j the corresponding desired output value.
- $x_{j,i}$ indicates the i-th feature value of the j-th training input vector.
- w_i signifies the i-th value in the $n-$dimensional weight vector, to be multiplied by the value of the i-th input feature.
- $w_i(t)$ shows the i-th value in the weight vector at iteration t
- The weight vector $\mathbf{w}$ is augmented with the bias parameter b at position 0, hence $w_0 \Leftrightarrow b$. As a result, the input vector is equivalently augmented by setting $x_{j,0} = 1$, since bias does not correspond to any input feature.
- The employed loss function is the so-called `0-1 Loss`, which returns 1 when the target and output are not equal and 0 otherwise.

Input:

- Learning Rate r
- Training Set D
- Maximum Number of Iterations T

Initialization:

- Initialize weight vector $\mathbf{w}(0) = \mathbf{0}$ (or to a small random value)
- $t \leftarrow 0$

repeat
  for each *training example* $(\mathbf{x_j}, d_j)$ *in dataset* D **do**
    `Calculate the actual output.`

$$y_j(t) = f(\mathbf{w^T}(t) \cdot \mathbf{x_j}) = f(w_0(t) + w_1(t)x_{j,1} + \cdots + w_n(t)x_{j,n})$$

    `Update the weights.`

$$w_i(t+1) = w_i(t) + r \cdot (d_j - y_j(t)) \cdot x_{j,i}, \quad 0 \leq i \leq n$$

    `Update iteration parameter.` $t \leftarrow t + 1$
  endfor
until *stopping criterion is satisfied* ***or*** *maximum iteration number is reached*

As it can be pointed out, in the case of a false prediction the algorithm adjusts the weight values by a proportion of the input vector that has been misclassified. Otherwise, the weight vector remains unmodified. The training procedure is terminated when the classification error is less than a user-specified threshold or a predefined number of iterations has been completed. If the utilized training set is linearly separable, i.e. the respective two classes can be distinguished by a hyperplane, then the convergence of the perceptron's training algorithm is guaranteed. In the inverse situation, no "approximate" solution will be gradually approached under the standard process, but instead, learning will completely fail, since perceptron is a linear classifier, as mentioned before.

2.2.2 Multilayer Perceptron

As the name suggests, a **Multilayer Perceptron (MLP)** is composed of multiple perceptron units, organized in numerous layers. Concretely, a typical MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer, which are visually demonstrated in Figure 2.2.7. Generally, all possible associations between perceptron building blocks of consecutive layers are present and therefore every feature of the respective input vector affects all attributes of the corresponding output vector, rendering MLP a typical example of fully-connected networks. By construction, such computational systems fall under the class of feedforward Artificial Neural Networks, since the connections between the nodes, contained in the model structure, do not form cycles or loops and consequently the information flows only in a forward manner through the successive node layers.

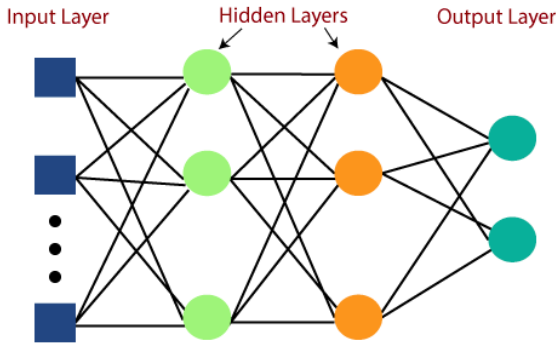


Figure 2.2.7: Multilayer Perceptron [57]

The Multilayer Perceptron was developed in order to tackle the inadequacy of simple perceptron systems in modeling non-linear data and solving more complex problems. Its hierarchical structure, where each layer is feeding the next one with its individual computational results and the respective internal representations of the input data, enables the extraction of features at different scales or resolutions and the eventual combination of them into high-order characteristics. This mechanism renders the Multilayer Perceptron the most suitable approach for more composite tasks and at the same time the most commonly used type of Artificial Neural Network.

Under this framework, Multilayer Perceptron can employ arbitrary activation functions, in order to be disposed to perform either regression or classification. Figure 2.2.8 displays some

of the most prominent non-linearities that are integrated in the MLP mechanism in an effort to emulate the effect of action potentials in the biological neurons of the human brain. The two historically common activation functions are *Sigmoid* and *Tanh*, which are graphically depicted in the first row of Figure 2.2.8. As it might be seen, they are both continuously differentiable functions that perform normalization by squashing the respective input values. Their most distinguishable characteristic is their finite range, which makes them suitable for classification tasks.

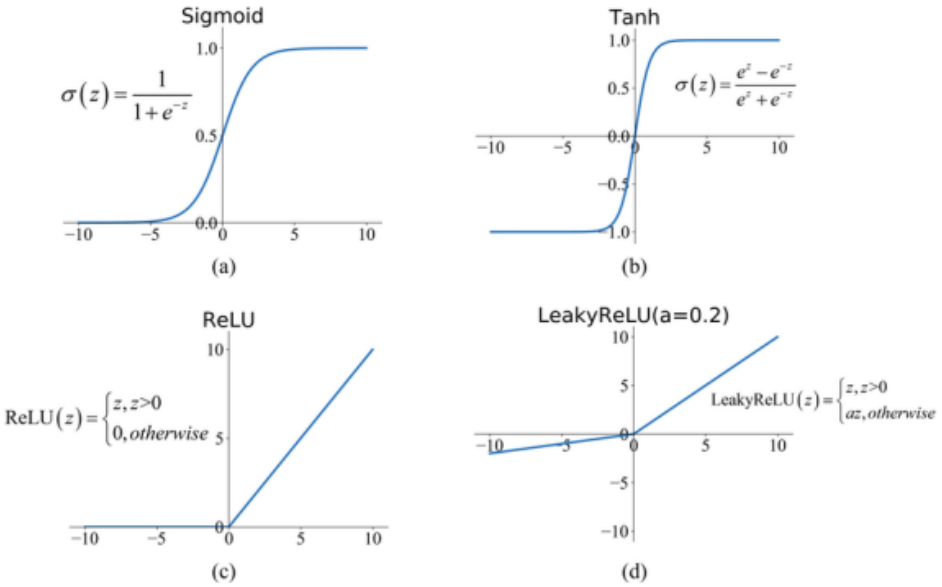


Figure 2.2.8: Activation functions [58]

However, the most frequently utilized activation function in recent studies concerning the developments of deep learning techniques, is the *Rectifier Linear Unit* and its variants, illustrated at the second row of Figure 2.2.8. ReLU is a partially differentiable function, characterized by infinite range of values, that enables Multilayer Perceptron models to overcome numerical problems related to sigmoids. Due to its simple formula, the overall computational requirements of the training procedure, including time and resources, can be significantly reduced, compared to the aforementioned alternatives. *Leaky ReLU* is a ReLU-based type of activation function that introduces a small slope for negative values, which is predetermined before training. It is widely applied in tasks suffering from sparse gradients.

It can be easily affirmed that the training algorithm of Perceptron cannot be directly applied in the case of an MLP network, since the rectification quantity of the weight values corresponding to each node should be calculated with respect to the overall model's output, in terms of minimizing a specified cost function. Therefore, another, more generalized, supervised learning method, called **Backpropagation**, has been established, as regards the training process of feedforward neural networks, including Multilayer Perceptrons. The backpropagation algorithm was initially introduced in the 1960s, but its importance wasn't fully appreciated until a famous paper [59] by David Rumelhart, Geoffrey Hinton and Ronald Williams was published almost 30 years later, in 1986. This survey highlights the prevalence of backpropagation over earlier learning approaches, concerning various tasks, which

had previously been considered insoluble. Nowadays, the backpropagation algorithm can be characterized as the workhorse of learning in the field of Artificial Neural Networks.

The principle of the backpropagation technique is grounded on the utilization of a cost function, which quantifies the difference between the system's output and a known expected value. The main goal of the algorithm is to iteratively adjust the weights and biases throughout the neural network's structure, based on the currently calculated error, so that the respective cost gradually decreases towards its minimum point. This can be achieved by applying a typical gradient descent procedure, which employs small repeated steps in the opposite direction of the gradient of the examined function, i.e. the direction of steepest descent, in order to detect its local minimum. To this end, the gradients of the aforementioned loss function with respect to the parameters, corresponding to the various layers of the network, are computed through the *Chain Rule of Calculus*[2] and applied during the update process. Therefore, the function that combines the internal weightings and the input signals in each neuron of the multilayer network should be differentiable and more specifically have a bounded derivative, as the activation functions presented in Figure 2.2.8.

The selection of the appropriate loss index is indissolubly associated with the form of the activation function used in the output layer of the neural network, as well as the nature of the problem, which the model attempts to solve. The configuration of the output layer, in essence, defines the framework of the examined task, while a suitable cost function constitutes a computational tool that has the ability to effectively quantify the error calculated in the aforementioned framework. For instance, in case of a binary classification problem the most frequently applied combination of loss function and output layer's non-linearity is sigmoid activation unit along with Cross-Entropy loss, which is measured as number between 0 and 1 representing the difference between the predicted probability distribution and the ground-truth (0 corresponds to the perfect classifier).

The Backpropagation algorithm is presented below in the form of pseudocode. In practice, it manages to properly modify the system's internal state, based on the error calculated at each iterative step, in two distinct stages of execution:

- **Forward Pass or Forward Propagation**: This stage involves the calculation and the storage of all intermediate variables of the system successively from the input layer to the output layer, based on the current values of weights and biases. As mentioned before, during this procedure, each neuron performs two operations, the computation of the weighted sum and the processing of the produced result through an activation function, which determines the behaviour of this particular model unit.
- **Backward Pass or Backward Propagation**: At this stage, the error between the actual output of the system and the expected one is distributed inside the network. This is accomplished by traversing the network in reverse order, from the output to the input layer, and concurrently calculating the gradient of the loss function with respect to the various model parameters, using the *Chain Rule*. During this procedure, all intermediate variables, in the form of partial derivatives, that may be required for further calculations are being stored.

[2]The chain rule is a mathematical formula that expresses the derivative of a composition of differentiable functions in the form of a gradient chain, representing the dependencies among the individual functions.

Algorithm 2: Backpropagation Training Algorithm

Notation:

- $D = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)\}$ is the labeled **training set** of N data samples, where $\mathbf{x_j}$ symbolizes the input vector and y_j the corresponding desired output value.
- w_{ij}^k denotes the weight between node j in layer l_k and the node i in layer l_{k-1}.
- b_i^k denotes the bias parameter for node i in layer l_k.
- a_i^k symbolizes the activation of node i in layer l_k, i.e. the weighted sum of corresponding input values plus bias.
- o_i^k symbolizes the output of node i in layer l_k.
- r_k indicates the number of nodes in layer l_k.
- δ_j^k denotes the error term which corresponds at node j in layer l_k and represents the partial derivative $\frac{\partial C}{\partial a_j^k}$ in the chain rule formula.
- g denotes the activation function for the nodes of the hidden layers.
- γ indicates the learning rate.
- $C(\mathbf{y}, \hat{\mathbf{y}})$ symbolizes the cost function, which defines the error between the ground truth value $\mathbf{y}$ and the calculated output $\hat{\mathbf{y}}$ for all input-output pairs $(\mathbf{x_i}, y_i) \in D$.

Preliminaries:

- To simplify the mathematical formula, the bias b_i^k for node i in layer l_k will be incorporated into the weights as w_{0i}^k with a fixed output of $o_0^{k-1} = 1$ for node 0 in layer l_{k-1}.
- The activation a_i^k and the corresponding output o_i^k of node i in layer l_k can be calculated as follows:

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ji}^k o_j^{k-1} \overset{w_{0i}^k = b_i^k}{=} \sum_{j=0}^{r_{k-1}} w_{ji}^k o_j^{k-1} \tag{2.2.1}$$

$$o_i^k = g(a_i^k) \tag{2.2.2}$$

- The partial derivative of the activation with respect to the weight values can be derived from the following equation:

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k} \left(\sum_{l=0}^{r_{k-1}} w_{lj}^k o_l^{k-1} \right) = o_i^{k-1} \tag{2.2.3}$$

- The error terms δ_j^k can be computed by the backpropagation formula:

$$\delta_j^k = \frac{\partial C}{\partial a_j^k} = \sum_{l=1}^{r_{k+1}} \frac{\partial C}{\partial a_l^{k+1}} \frac{\partial a_l^{k+1}}{\partial a_j^k} = \sum_{l=1}^{r_{k+1}} \delta_l^{k+1} \frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{2.2.4}$$

$$a_l^{k+1} = \sum_{j=1}^{r_k} w_{jl}^{k+1} g'(a_j^k) \implies \frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1} g'(a_j^k) \tag{2.2.5}$$

$$(2.2.4) \overset{(2.2.5)}{\implies} \delta_j^k = g'(a_j^k) \sum_{l=1}^{r_{k+1}} w_{jl}^{k+1} \delta_l^{k+1} \tag{2.2.6}$$

Initialization: Initialize all weights and biases at zero or any other small random value.

1 **repeat**

2 `Forward Phase.`

3 **for each** *training example* $(\mathbf{x_d}, y_d)$ **do**

- Calculate a_j^k and o_j^k using equations (2.2.1) and (2.2.2) respectively for each node j in layer l_k, by proceeding from the input to the output layer.
- Compute the final output $\hat{y}_i$.
- Store all the intermediate results.

4 `Backward Phase.`

5 **for each** *training example* $(\mathbf{x_d}, y_d)$ **do**

- Calculate $\frac{\partial C}{\partial w_{ij}^k}$ for each weight w_{ij}^k, by proceeding from the output to the input layer, according to the chain rule:

$$\frac{\partial C}{\partial w_{ij}^k} = \frac{\partial C}{\partial a_j^k}\frac{\partial a_j^k}{\partial w_{ij}^k} \overset{(2.2.3)}{=} \delta_j^k o_i^{k-1}$$

  The error terms δ_j^k can be recursively computed by the equation (2.2.6).
- Store all the intermediate results.

6 `Total Gradient.` Combine all the individual partial derivatives of the Cost function, that have been calculated during the backward pass for each input-output pair $(\mathbf{x_d}, y_d)$, in order to compute the total gradient $\nabla_{\mathbf{W}} C(\mathbf{W})$, which corresponds to the entire training dataset D.

7 `Weight Update.`

$$\mathbf{W} = \mathbf{W} - \gamma \nabla_{\mathbf{W}} C(\mathbf{W}) \tag{2.2.7}$$

8 **until** *stopping criterion is satisfied* ***or*** *maximum iteration number is reached*

There is a wide variety of different optimization algorithms establishing on the concept of the Gradient Descent. One of the most popular variants of the typical procedure is its stochastic approximation called Stochastic Gradient Descent or SGD for short [60], which updates all the model parameters for each training example, instead of computing the gradient of the employed cost function for the whole dataset, as presented in the algorithm above. Due to the integrated randomness, SGD is noisier than the typical Gradient Descent and also requires larger number of iterations for convergence. The most commonly applied optimization method with several deep learning applications, especially in the field of Computer Vision [61, 62, 63], is an extension of SGD named Adam. It was initially introduced by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper [64]. Adam computes individual adaptive learning rates for each weight parameter in the network based on moving average estimations of the first and second moments of the respective gradients. It is considered suitable for problems with non-stationary objectives, large datasets, multiple parameters and sparse derivatives and is usually suggested as the default optimization method [65].

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) constitute a class of Artificial Neural Networks in the field of Deep Learning, designed for processing and analyzing data with grid-like topology or structured information in the form of generalized arrays. Their ability to extract spatial features and capture topological patterns in the input data has made them the most prevailing architecture for various tasks and applications, such as Image and Video Recognition [66], Recommendation Systems [67], Natural Language Processing [68], Brain-Computer Interfaces [69] and Financial Time Series [70].

The concept of Convolutional Networks was inspired by the biological mechanism of human vision and therefore, such kind of systems mostly deal with input data in the image-format. A digital image can be defined as a computational representation of visual information. As demonstrated in Figure 2.2.9, it contains a series of pixels arranged in a grid-like fashion, whose values denote the brightness level or the color. According to the biological archetypes, when the human vision system perceives a physical image, the individual cortical neurons respond to stimuli only in a restricted region of the visual scope, known as the receptive field. The biological neurons are connected with each other in such a way that their corresponding receptive fields partially overlap and consequently cover the entire optical area. In order to emulate this procedure, each neuron in the artificial implementation of the visual network processes the information that appertains to its own receptive field, which can be demarcated through the utilization of specified structures, called filters. These kernel grids, which are shared-weight among the neurons of each network layer, slide along input features and provide translation-equivariant responses, known as feature maps. In this way, CNNs manage to assemble hierarchical patterns of increasing complexity and detect various significant aspects and characteristics of the input image-like data.

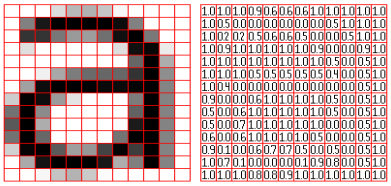


Figure 2.2.9: Image representation as a grid of pixels [71]

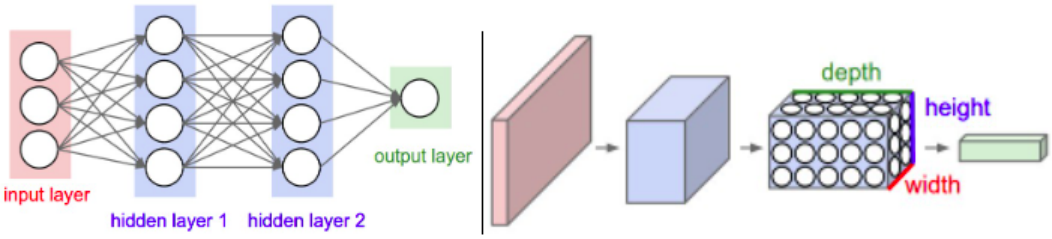


Figure 2.2.10: CNN architecture in comparison with a typical MLP model [7]

The principal objective of Convolutional Neural Networks is to model a single differentiable function that efficiently maps the input to a corresponding output, by incorporating an automated feature extraction process. Similar to the case of typical feedforward systems, each node in the network topology receives a specific part of the input attributes, performs a generalized dot product and optionally applies a non-linearity afterwards. However, in order to embed and encode the topological properties of the grid-like input into the architecture of the system, the artificial nodes of each layer in a Convolutional Neural Network are arranged in

three dimensions, *width*, *height* and *depth*, introducing the notion of computational volumes, as illustrated in Figure 2.2.10. The main types of layers generally involved in the structure of a CNN will be presented in detail at the following subsections.

Convolutional Layer

The convolutional layer constitutes the fundamental building block of a CNN model, as it performs the major computational processes regarding the system's mechanism. As the name suggests, this layer applies grid-shaped feature detector filters to the input image, through a mathematical operation, called *Convolution*. This procedure is graphically displayed in Figure 2.2.11 for the case of 2-dimensional matrices. As it might be seen, the value of each output point can be calculated as the Frobenius product between the kernel and an equally-sized slice of the input data. In essence, the utilized filter consists of learnable weight parameters and abstractly represents an attribute, which the model aims to detect. It is usually spatially smaller than the input image, as regards the width and height dimensions, while the connectivity along the depth axis is always equal to the depth of the input volume.

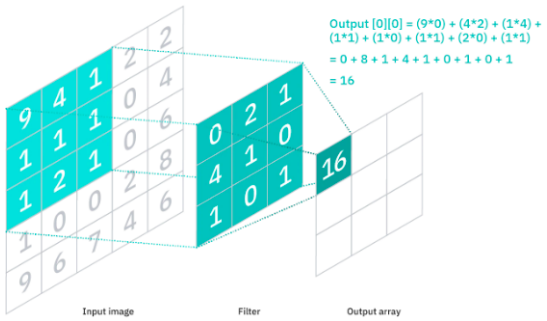


Figure 2.2.11: Illustration of the Convolution Mechanism [9]

As mentioned before, each convolutional neuron processes only the data pixels that correspond to its receptive field, i.e. a restricted portion of the input image, which is determined by the size of the aforementioned filter. This fact implies that the produced output array is not directly mapped to each input value, as occurs in the case of standard feedforward neural networks. Therefore, convolutional (and pooling) layers are commonly referred to as "partially connected" layers.

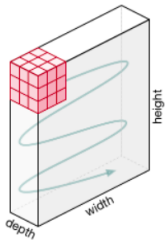


Figure 2.2.12: Kernel sliding [72]

Under this framework of local connectivity, in order to cover the entire data area, the kernel unit slides along the input image, according to Figure 2.2.12. In this way, the output representation of every neuron's receptive region is produced through the filter's convolutional response and the respective activation map is created. During this process, the weights included in the applied feature detector remain fixed, introducing the concept of "parameter sharing" among the nodes of the convolutional layer. Their values are properly adjusted during the training procedure, using the BackPropagation algorithm, which has been explicitly presented in section 2.2.2. However, there are three hyperparameters involved in the convolutional operation, that crucially affect the volume size of the output, but need to be set before the training process of the CNN model. These include the following:

- **Number of filters:** Multiple filters can be employed in a convolutional layer, in order to extract different characteristics from their respective common input. For instance, Figure 2.2.13 demonstrates 5 distinct nodes, associated with 5 different filters, that share the same receptive field and thus process data contained in the same region of the input volume.

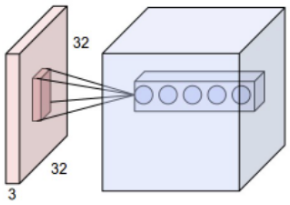


Figure 2.2.13: Convolutional layer with multiple filters [7]

The distinct activation maps, that will be created in correspondence with the aforementioned kernels, can be stacked along the depth dimension, determining in this way the output volume. Therefore, the number of utilized filters affects the depth of the produced output.

- **Stride:** This non-negative quantity represents the shift of the sliding filter window over the input matrix and in essence, determines how densely or sparsely the convolution is applied. As shown in Figure 2.2.14, if the stride value is set to 1, then the kernel moves across 1 pixel at a time, resulting in heavily overlapping receptive fields between the input columns and accordingly to large output volumes.

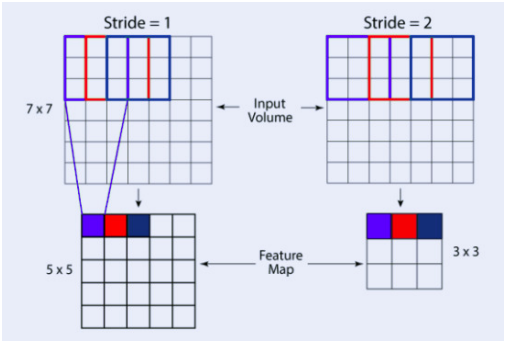


Figure 2.2.14: Stride in CNN [73]

On the other hand, a higher stride value S implies that the kernel is translated S units at a time per output and consequently skips some features, along the width or height dimension as well, before being applied again. In this case, the smaller overlap of the receptive fields leads to spatially smaller feature maps.

- **Padding:** Padding refers to the practice of surrounding a matrix with layers of zeroes or another specified small value, in order to preserve features that exist at the border of the original matrix and control the size of the output feature map.

There are three padding categories:

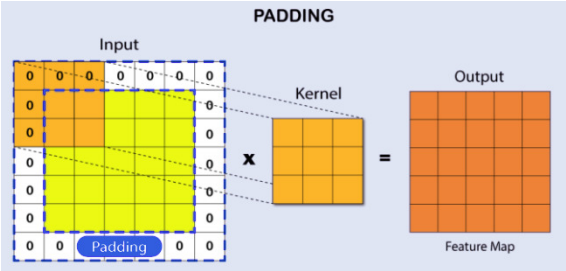


Figure 2.2.15: Padding in CNN [73]

– *Valid Padding*: This is also known as *no padding*. In this case, the respective convolutional layer does not pad at all, hence the output size shrinks, depending on the input dimensions and the applied kernel.

– *Same Padding*: This type of padding ensures that the output matrix has the same size as the input one.

– *Full Padding*: It can be defined as the maximum padding that is able to increase the size of the output feature map.

Transposed Convolutional Layer

Transposed convolutional layer constitutes a special type of the standard convolutional layer, which has been thoroughly examined in the previous section. As the name suggests, it performs a regular convolution operation but reverts its spatial transformation, in terms of generating an output feature map with greater dimensions that the input one. This upsampling process is accomplished by properly modifying the input grid-like vector. Similar to the typical version, a transposed convolutional layer is defined by the same hyperparameters, including stride and padding. However, in this case, these values correspond to the process of applying a standard convolution to the output in order to produce a feature map with the same dimensions as the given input. The computational steps involved in the implementation of a transposed convolution are graphically illustrated in the diagram of Figure 2.2.16.

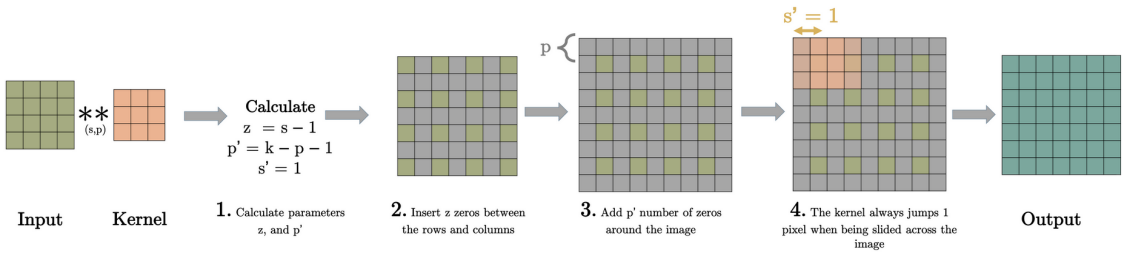


Figure 2.2.16: Transposed Convolution [74]

A complete comparison between the two examined convolutional operations can be summarized in the Table 2.1, where k denotes the kernel size, p symbolizes padding, s represents stride and i indicates the input size.

Conv Type	Operation	Zero Insertions	Padding	Stride	Output Size
Standard	Downsampling	0	p	s	$(i + 2p - k)/s + 1$
Transposed	Upsampling	$s - 1$	$k - p - 1$	1	$(i - 1) \cdot s + k - 2p$

Table 2.1: Comparative summary of the two convolution types (adapted from [74])

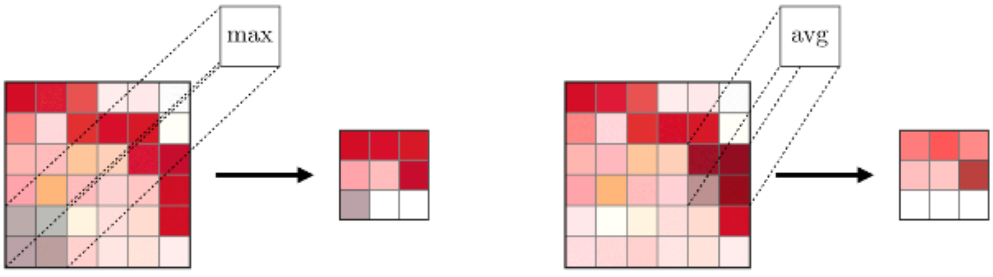
Pooling Layer

The pooling layer is responsible for progressively reducing the spatial size of the convolved feature representation, by performing a downsampling process. Similar to the convolutional layer, it applies a sliding filter across the entire input, which is able to combine multiple data attributes, corresponding to neuron clusters from the previous layer, into a single value, using a non-linear operation. However, the kernel used in this case, does not contain any weight parameters, since it only aggregates the values within its receptive field.

Under the aforementioned framework, the pooling layer manages to derive a summary statistic of the nearby outputs and extract dominant features, which are rotation- and position-invariant, thus without affecting the efficiency of the training procedure. At the same time, this dimensionality reduction induces a decrease of the computational power required to process the data, since a smaller amount of essential parameters is involved. Pooling layers are usually inserted in-between successive Convolutional layers in the CNN architecture, enabling in this way the neurons contained in the subsequent convolutional layer to have a larger receptive field and be capable of discovering higher-scale patterns, without changing the size of their corresponding filters.

Two dominant types of pooling layers can be distinguished:

- **Max pooling:** As the name suggests, max pooling returns the maximum value from the portion of the image covered by the kernel, retaining in this way the most prominent features of the activation map.
- **Average pooling:** This pooling operation averages the values of the image slice, which corresponds to the kernel view. In this way it highlights the average presence of features in the examined activation map.



(a) Max pooling (b) Average pooling

Figure 2.2.17: Types of pooling layes [75]

Normalization Layer

Normalization can be defined as a procedure that standardizes a set of values corresponding to numeric variables into a typical scale, without deforming existing contrasts or correlations in the input range. In the context of Deep Learning, a Normalization layer performs this operation, by shifting and scaling the input features into a distribution centered around 0 with standard deviation equal to 1, typically via the computation of statistical data attributes, such as mean and variance, during runtime. The inclusion of Normalization layers into the architecture of deep models is essential, since this mechanism stabilizes the gradient descent step during the training process and also ensures a faster convergence for a given learning rate. Two prevailing normalization methods can be distinguished:

- **Batch Normalization (BN)**
  Batch Normalization aims to decrease the internal covariance shift and hence accelerate the training of deep neural nets, via a standardization step that transforms each input included in the current mini-batch, based on its corresponding mean value and standard deviation. This adjustment benefits the gradient flow through the network, as it restricts the distribution of the input data to any particular layer and reduces in this way the dependence of the produced gradients on the scale of the various parameters, enabling at the same time the use of higher learning rates without the risk of divergence. From a mathematical perspective, a Batch Normalization layer applies the following formula for a mini-batch $\mathcal{B}$ and the learnable parameters γ and β:

$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \textit{mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \textit{mini-batch variance}$$

$$\hat{x_i} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \textit{normalize}$$

$$y_i = \gamma \cdot \hat{x_i} + \beta \qquad \textit{shift and scale}$$

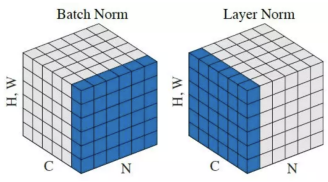


Figure 2.2.18: Normalization methods [76]

- **Layer Normalization (LN)**
  Inspired by the results of Batch Normalization, Hinton et al. [77] introduced a novel method called Layer Normalization, which standardizes the activations along the feature direction instead of the mini-batch axis, as graphically demonstrated in Figure 2.2.18. Unlike the previous case, Layer Normalization directly estimates the statistics

from the summed inputs to the neurons within a hidden layer. In this way, it does not impose any new dependencies between training cases or additional constraints on the size of the mini-batch. From a mathematical perspective, the layer normalization transforms the input features in accordance with the previous formula, but using the following statistics instead (H denotes the layer width):

$$\mu_l = \frac{1}{H}\sum_{i=1}^{H}\alpha_i^l \qquad \textit{layer mean}$$

$$\sigma_l^2 = \frac{1}{H}\sum_{i=1}^{H}(\alpha_i^l - \mu_l)^2 \qquad \textit{layer variance}$$

Non-Linearity Layer

As mentioned before, Convolution can be algebraically implemented as the *Frobenius* product between two equally sized matrices, one representing the input features arranged in a grid-like manner and the other a filter composed of learnable weight parameters. It can be easily affirmed that this operation inherently generalizes the weighted sum computed by the perceptron neurons of standard MLP systems, which have been examined in detail at section 2.2.2. Thus, regarding the case of CNN models, the convolutional operator represents the linear processing module of the artificial neurons, as it performs a linear transformation of the input in a multi-dimensional space.

However, images or topologically structured information in general, comprise several non-linear, irregular and particularly complex characteristics, that cannot be explicitly captured through a linear method. This limitation can be tackled by emulating the mechanism of typical Multilayer Perceptrons, which imposes the incorporation of non-linearities into the system's architecture. To this end, non-linearity layers are often placed between consecutive convolutional or pooling layers, in order to apply an activation function to the corresponding feature maps. Some of the most frequently used activation functions in the framework of Convolutional Neural Networks are graphically displayed in Figure 2.2.8 of section 2.2.2.

Fully-Connected Layer

In contrast to the concept of partial connectivity, which characterizes the convolutional and pooling layers in the architecture of a CNN, each neuron in a fully-connected layer is associated with all the nodes in the preceding and the succeeding layers, as occurs in regular MLP systems (section 2.2.2). Therefore, its respective activations can be computed in the standard mode, by performing a matrix multiplication followed by a bias effect.

For the purpose of compatibility with the grid-like structure of the produced feature maps, the operation of a Fully-Connected layer involves a flattening procedure. As demonstrated in Figure 2.2.19, the entire matrix of the activation map is transformed into a single column, which is then fed to the fully-connected module of the neural network for further processing.

If present, FC layers are usually found towards the end of CNN architectures and can be used to optimize several objectives. For instance, they are able to perform tasks such as the

classification of input images into multiple categories, based on features, extracted through the previous convolutional and pooling layers and their respective filters.

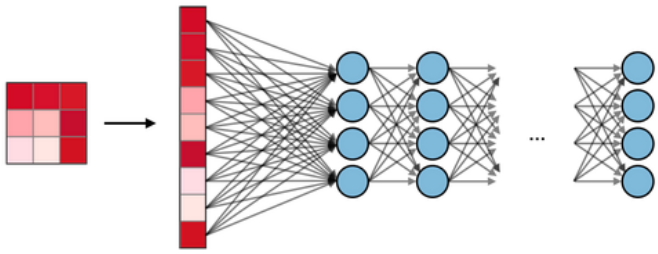


Figure 2.2.19: Fully-connected layer [75]

2.2.4 Recurrent Neural Networks

Recurrent Neural Networks, or **RNNs** for short, constitute a class of Artificial Neural Networks, designed to process sequential data and exhibit temporal dynamic behavior. The term "sequential data" refers to information structures that contain elements arranged in some kind of order. Examples include time series, DNA sequences in the field of biomedical informatics, sequences of user actions or the successive frames of a video. Based on the concept of an integrated type of "memory" in the form of an internal hidden state, RNN models are capable of capturing temporal or ordinal dependencies among sequential data points and hence are utilized in various related tasks, such as unsegmented, connected handwriting recognition [78], speech recognition [79, 80], language translation [81] and image captioning [82]. Furthermore, these systems are incorporated into many popular applications such as Siri, voice search and Google Translate.

As discussed in former sections of this analysis, traditional feedforward neural networks allow information to flow only in the forward direction, from the input nodes, through the hidden layers and finally to the output nodes. Within this framework, which is illustrated in Figure 2.2.7, the computational operations concerning the model's decisions in each step, are based only on the current input representation, without taking into account prior features and elements that might be associated with present information and therefore influence to some degree the future events. As a consequence, feedforward systems are not suitable for tasks that involve sequential data, due to the aforementioned inability of retaining prior knowledge or past-related attributes to forecast subsequent values. Recurrent Neural Networks are introduced to tackle this exact limitation, by incorporating such a mechanism into their structure.

RNNs are basically derived from feedforward neural networks, by applying the transformation demonstrated in Figure 2.2.20. In essence, the nodes contained in different layers of the feedforward system are compressed in order to form a single layer of the corresponding RNN model. The term "recurrent" arises from the fact that the exact same operations are performed for every element of the input sequence, with the time-dependent output being computed based on previous information. This process can be graphically depicted as an "unfolding" of the RNN layer across the time axis, in such a way that the current input instance $x(t)$ is mapped to its corresponding output value $y(t)$ through a hidden "memory"

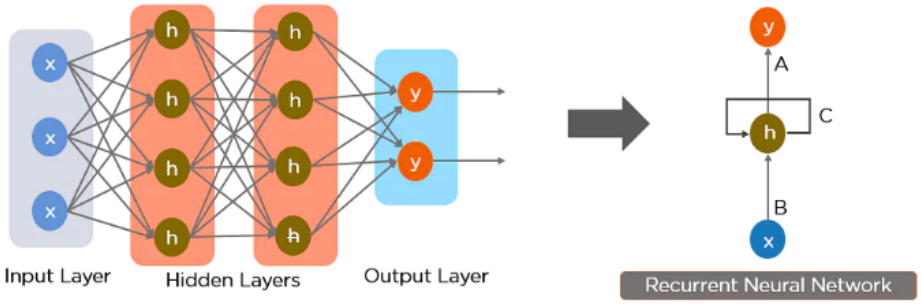


Figure 2.2.20: Conversion of Feedforward to Recurrent Neural Network [83]

mechanism $h(t)$ (Figure 2.2.21). To this effect, the connections between the nodes of the network form a directed or undirected computational graph along the temporal input sequence. All these interrelations are computationally modeled by distinct weight matrices.

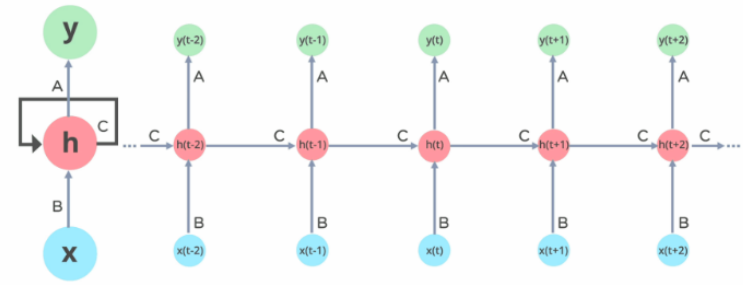


Figure 2.2.21: Recurrent Neural Network [83]

The training process of Recurrent Neural Networks is established on a variant of the conventional BackPropagation algorithm called **BackPropagation Through Time** (BPTT). This method retains the basic principles of the standard algorithm regarding the error calculations from the output to the input layer for the weight adjustments. Their main difference lies in the fact that BPTT sums the errors at each time-step, since the parameters are shared across each layer of the recurrent network. However, throughout this procedure, RNN models tend to experience two significant problems, concerning the size of the computed gradients, which represent the slope of the loss function along the error curve and are used in order to update the weight values. These issues can be described as follows:

- **Vanishing Gradient Problem**: During training procedure, the gradients, which are being calculated with respect to the various model parameters, are traversing the network in the backward direction, from the output to the input cell, using the *Chain Rule of Calculus*. If the effect of a layer on its subsequent one is small, then the value of the corresponding partial derivative will be respectively small, leading to a gradual shrinking of the multiplicative gradients, as the backpropagation through time occurs. On account of this, the aforementioned product can exponentially decrease to zero, resulting in insignificant values that do not affect the weight updating. As a consequence, the model is no longer capable of learning correlations between temporally distant events,

since the effect of earlier inputs cannot be captured and hence it is based only on its short-term memory.

- **Exploding Gradient Problem:** This issue refers to the opposite behaviour in comparison to the previously explained problem, since it arises from the accumulation of large error gradients during the backward phase of the training procedure. This significant increase in the norm of the gradient causes very large updates to the network weights, leading the model to an unstable state.

The exploding gradient problem can be easily controlled in practice, by applying the *Gradient Clipping* technique, which rescales those gradient values that surpass a predefined threshold, ensuring in this way that the gradient descent procedure will be performed in a reasonable framework even if the loss landscape of the model is irregular. On the other hand, the vanishing gradient limitation remains a matter of major concern regarding the RNN well-functioning. In order to tackle this problem, the following two specialized versions of Recurrent Neural Networks were created:

Long Short Term Memory (LSTM)

This RNN architecture was introduced by Sepp Hochreiter and Juergen Schmidhuber in 1997. Their survey [85] attempts to address the problem of capturing long-term dependencies among sequential data, which constitutes an outcome of the vanishing gradient limitation and pertains to the inability of RNN models to accurately predict the current state, when former attributes that influence the current forecast do not correspond to the recent past. The LSTM mechanism tackles this short-term memory issue, by augmenting the cells, included in the hidden layers of the recurrent network structure, with additional gates that filter out information that is irrelevant to the prediction of the current output. As shown in Figure 2.2.22, each LSTM cell receives three different states as input, the current data instance $x(t)$, the short-term memory from the previous cell $h(t-1)$ and lastly the long-term memory $c(t-1)$ and employs three distinct gates in order to regulate the information to be kept or discarded before passing on the long-term and short-term features to the next cell. These gates can be defined as follows:

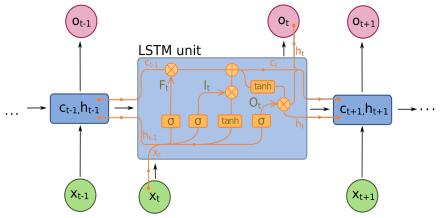


Figure 2.2.22: LSTM unit [84]

- **Input gate:** It processes the current input data as well as the short-term memory of the previous cell in the network topology, in order to determine the supplementary information, which should be added to the network's long-term memory (cell state).

- **Forget gate:** It multiplies the incoming long-term memory by a forger vector, composed of the current input and the short-term memory attribute, in order to filter out the unuseful elements.

- **Output gate:** It specifies the new hidden state (short-term memory), which will be passed on to the subsequent cell at the next-time step, by taking into account the current input, the previous short-term memory and the newly updated cell state,

computed by the forget gate. Moreover, as the name suggests, the output of the current time-step can be derived from the aforementioned result.

Gated Recurrent Unit (GRU)

This gating mechanism, which is graphically illustrated in Figure 2.2.23, was introduced in 2014 by Kyunghyun Cho et al. [86] in order to address the short-term memory problem of standard RNN models, as well. In comparison with the LSTM unit, GRU variant incorporates fewer parameters, as it lacks the cell state input and includes only the two following types of gates:

- **Update gate:** The update gate is responsible for determining the amount of previous information that will be transferred to the next cell, given the former hidden state $h(t-1)$ and the current input instance $x(t)$. Under this concept, the model can even decide to copy all the past information, eliminating in this way the risk of the vanishing gradient problem.

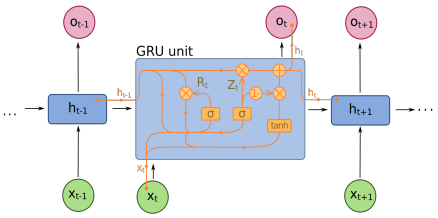


Figure 2.2.23: GRU unit [84]

- **Reset gate:** The reset gate is used for determining the amount of past information that should be ignored. In other words, it is in charge of deciding whether the hidden state of the previous cell is important or not. To this end, it applies an equivalent to the update gate formula with its corresponding weight matrices.

2.3 Generative Adversarial Networks

Generative Adversarial Networks, or **GANs** for short, constitute a class of Machine Learning frameworks, initially introduced by Ian Goodfellow and other researchers at the University of Montreal in 2014 [21] as a different approach in the concept of generative modeling. French Computer Scientist Yann LeCun has described GANs as the "the most interesting idea in the last 10 years in Machine Learning" [87], since the enlightening idea behind this algorithmic architecture represents a real conceptual breakthrough in the research field of Deep Learning. The applications of Generative Adversarial Networks have been rapidly expanded over the past decade into multiple domains and areas, including Art and Fashion [88, 89, 90], Astronomy [91, 92], Physics [93, 94], Video Games [95] and Audio Synthesis [96].

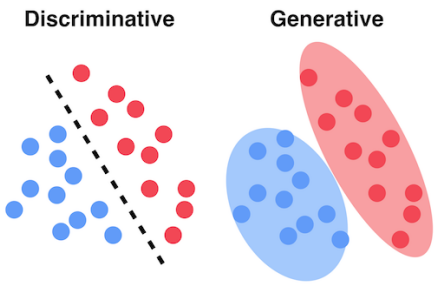


Figure 2.3.1: Generative vs Discriminative models [97]

The GAN mechanism is intrinsically interrelated with the concept of two contrastive approaches in the field of statistical classification, which will be thoroughly analyzed further down:

- **Discriminative models:** As visually presented in Figure 2.3.1, discriminative models distinguish decision boundaries among observed data. Based on features that characterize the input instances, systems of such kind predict the label or the class that corresponds to each data point. In this way, the input distribution is projected into distinct categories. In simple words, the aforementioned logistical models operate as classifiers that "discriminate" examples of input variables across different groups. More formally, using mathematical notation, given a set of data instances X and a set of labels Y, discriminative models capture the conditional probability $p(Y|X)$, by estimating a discriminative function $f : X \rightarrow Y$, and hence are also referred to as *Conditional models*.

- **Generative models:** On the other hand, generative models can be considered as a class of statistical algorithms that are capable of generating new content in the form of data instances, by capturing the underlying distribution of individual classes in the input dataset, as illustrated in Figure 2.3.1. From a mathematical perspective, this process can be considered equivalent to the problem of deriving a complex random variable from a specific probability distribution, that plausibly fits in the input space. However, due to the inherently composite nature of a random variable and the inability to explicitly express the aforementioned distribution, a simplified parametric estimation method is employed. As demonstrated in the diagram of Figure 2.3.2, this method is

based on the utilization of a model (usually neural network), that learns to approximate a transformation function from a simpler form of input random variable, such as white noise, to an output random variable that incorporates the desired properties in terms of the underlying distribution. More formally, given a set of data instances X and possibly a set of target labels Y, generative systems capture the the joint probability $p(X, Y)$, or just $p(X)$ if there are no labels.

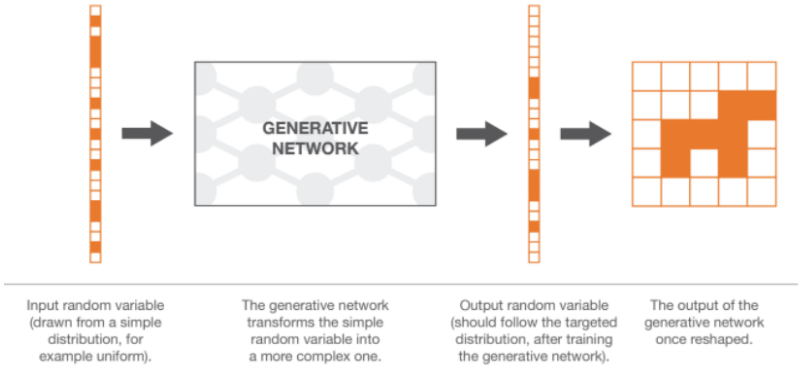


Figure 2.3.2: Generative modeling [98]

Generative Adversarial Networks provide a computational framework for the training procedure of generative models, based on a rivalry mechanism. More specifically, GAN systems replace the direct comparison between the generated and the ground-truth distribution, applied by other statistical techniques, with an "indirect" approach, that takes the form of a downstream discrimination task between real and generated samples, in order to force the aforementioned distributions to get as close as possible. To this end, as graphically displayed in the diagram of Figure 2.3.3, a typical GAN architecture, consists of a *Generator* network, which is trained to produce samples following a target distribution and a *Discriminator* network, which aims to identify the fake samples provided by the Generator. In more detail:

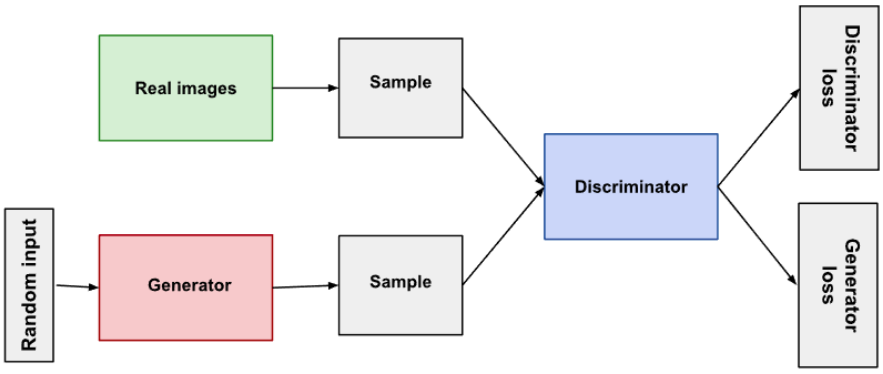


Figure 2.3.3: Overview of GAN structure [99]

2.3.1 Discriminator

The discriminator network learns to distinguish the real data from the fake samples created by the generator model, by evaluating them in terms of authenticity. In this way, it penalizes the generator for producing implausible results. It can be implemented as any type of network architecture, depending on the nature of the data to be classified.

In Figure 2.3.3, the two "Sample" boxes represent the distinct sources of training data fed into the discriminator. As it might be seen, its input consists of real instances, which are considered as *positive* examples during training, as well as fake samples, which are used as indicators of the *negative* data group. In either case, the discriminative model decides whether the current instance belongs to the ground-truth dataset or not.

The training procedure of the discriminator network is graphically depicted in the diagram of Figure 2.3.4. During this phase, the weights incorporated in the generator architecture remain unaffected, as its involvement in the process concerns only the construction of fake examples in order to be evaluated by the discriminator and hence a stable generator structure enables the detection of flaws or other characteristics that can contribute to the learning process of the discriminator. More specifically, at first, the input features are passed through the discriminator network, which operates as a classifier, trying to make successful predictions regarding the labels that correspond to the data instances (real-fake). The produced outcome is then utilized for the calculation of a properly selected cost value, also denoted as discrimination loss, which quantifies the misclassification errors. Based on this result, the weight parameters of the discriminative network are updated through the BackPropagation Algorithmic procedure, which has been explicitly discussed in section 2.2.2.

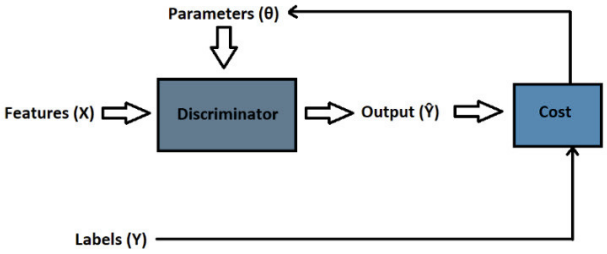


Figure 2.3.4: Block Diagram of Discriminator [100]

2.3.2 Generator

The generator network learns to create novel data instances, by incorporating the feedback from the discriminator. In particular, its training objective is to increase the discrimination error rate, by "fooling" the discriminator network into classifying its output candidates as real. In this way, it is able to indirectly discover underlying properties of the ground-truth data distribution.

More specifically, the input of the generator model is a fixed-length random vector, which can be considered as the seed of the generative process. This vector is typically sampled from a predefined latent space, which comprises compressed representations of high-level concepts regarding a specific data distribution (e.g. a multivariate normal distribution). During the forward pass, the generative network transforms this random noise vector into a meaningful form of output, which is interpreted as a new sample in the domain of interest. Typically, the space, from which the randomized input is sampled, has smaller dimensionality in comparison to the target one.

The training procedure of the generator is graphically depicted in the diagram of Figure 2.3.5. Similarly to the previously mentioned case, during this phase, the weight parameters included in the discriminative network are not updated, since its involvement in the process concerns only the evaluation of the fake samples, produced by the generator and this mechanism should be consistent with respect to the generator's learning process. As explained in former sections of this analysis, the main objective of a training algorithm is the minimization of a properly selected cost function, through the adjustment of the weight values contained in the network structure. However, as in might be seen, in the framework of a Generative Adversarial Network the generator is not directly associated with its respective loss, since it is penalized for producing data instances which are identified as fake by the discriminator. Therefore, this additional network should be included in the backpropagation process. To this end, in order to properly calculate the impact of generator weights on the corresponding output, the backpropagation flows back to the generator through the discriminator.

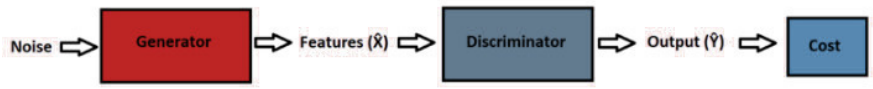


Figure 2.3.5: Block Diagram of Generator [100]

Thus, the training procedure of the generative model involves the following computational steps:

(1) At first, random noise is sampled from a specified distribution and passed through the generator neural network, in order to be transformed into realistic output examples.

(2) The generated output features are then fed into the discriminator, which classifies them as either "fake" or "real".

(3) Based on the difference between the actual output and the predictions of the discriminator concerning the labels of the examined samples, the corresponding loss is calculated.

(4) Lastly, the algorithm backpropagates through both the generator and the discriminator to obtain the gradients, which are used in order to update only the generator's weight parameters.

2.3.3 Overall Training

Generally, the overall training of a Generative Adversarial Network proceeds in alternating periods between the individual learning processes of the Generator and the Discriminator respectively, which have been presented in detail up above. In practice, it has been proven that the direct alteration is not effective and can even result in overfitting phenomena on finite datasets. To this end, Goodfellow et al. [21] introduced a novel learning practice based on consecutive interchanges between k steps of optimizing the Discriminator D and one step of optimizing the Generator G. In this way, D is being maintained near its optimal solution, so long as G adjusts slowly enough.

Figure 2.3.6 provides a graphical illustration of this learning approach. The black dotted line represents the ground-truth data distribution p_d, while the green solid one indicates the Gen-

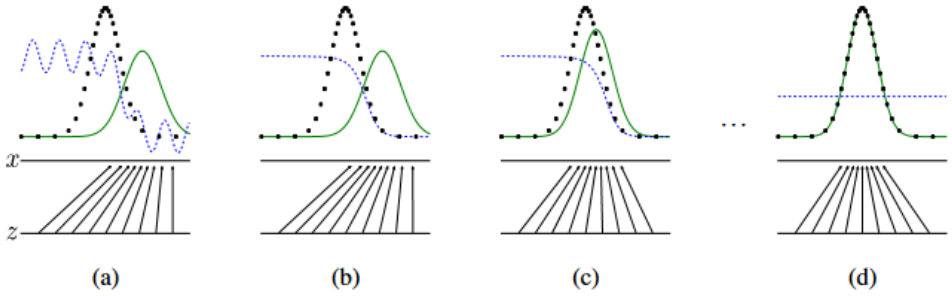


Figure 2.3.6: Illustration of GAN learning [21]

erator distribution p_g, which emerges from the produced fake candidates. The blue dashed line corresponds to the decision boundary formatted by the Discriminator's predictions. The lower horizontal line is the latent domain from which the random noise $\mathbf{z}$ is sampled. The upward arrows demonstrate how the mapping $\mathbf{x} = G(\mathbf{z})$ imposes the non-uniform distribution p_g on transformed samples.

(a) The depicted adversarial pair approximates a convergence point. The two data distributions p_d and p_g are similar, while the Discriminator D is considered a partially accurate classifier.

(b) For k consecutive steps D is trained to discriminate the real samples from the fake ones, all produced by the same stable Generator model and in this way converges to its optimal form D^*.

(c) After an update of G, the corresponding distribution p_g is shifted towards regions of the x-domain, which are more likely to be classified as real by D.

(d) After several training steps, the ground-truth and the output distributions ideally coincide ($p_d = p_g$). At this point, G and D cannot be further improved, since D is unable to differentiate between them, i.e. $D(\mathbf{x}) = \frac{1}{2}$. This is equivalent to randomly predicting the label that corresponds to each examined sample.

In practice, the convergence point which is depicted in step **(d)** of Figure 2.3.6 cannot be explicitly determined or approached, as it constitutes a fleeting, rather than stable, state. Therefore convergence under the framework of GAN systems still remains an open problem.

2.4 Autoencoder

An **Autoencoder** is defined as a type of feedforward Artificial Neural Network used to learn efficient data encodings for a given system configuration in an unsupervised manner. It was initially introduced in 1980s by Hinton and the PDP group [101] to address the problem of "backpropagation without a teacher", by using the input data as the supervisor of the learning process; that is, learning to reconstruct an input signal. More recently, autoencoder systems have led to numerous state-of-the-art results in various challenging tasks, especially in the field of Data Compression, by producing lower-dimensional representations of the original input data through a statistical redundancy elimination procedure. Therefore, such kind of models can tackle the so-called "curse of dimensionality" problem, which arises from datasets with multiple attributes and hence high-dimensional feature spaces and often leads to overfitting phenomena. To this end, autoencoders are applied in various domains and tasks, including facial recognition [102], feature detection [103], anomaly detection and word meaning extraction [104, 105].

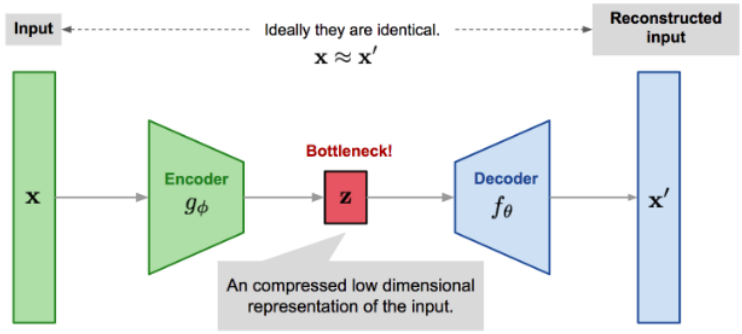


Figure 2.4.1: Autoencoder architecture [12]

As demonstrated in Figure 2.4.1, an autoencoder system consists of two main structural components:

- **Encoder:** As the name suggests, this module transforms the input data into a compressed form, known as *code*. More specifically, the initial vector is passed through a series of layers that successively perform dimensionality reduction, resulting in a network "bottleneck". This hidden layer typically comprises fewer neurons in comparison with the input and hence constraints the amount of information that can traverse the full network. In this way, it captures a low-order representation of the input that encodes the respective data features into a latent space. This term refers to an embedding of a set of items or other abstract entities within a generalized manifold, which incorporates the notion of similarity between objects in the form of topological proximity. Thus, the encoder network maps the input data points to their respective encodings in a consistent manner, retaining at the same time the crucial information attributes.
- **Decoder:** The decoder network operates as an interpreter of the produced code, as it "decompresses" this hidden representation into a vector in the original space. In this manner, it is able to reconstruct the initial input, based on the latent attributes of its corresponding encoded form. This module typically comprises a set of layer blocks that successively perform an upsampling process, in order to properly expand the di-

mensions of the bottleneck's output. In general, the decoder architecture mirrors the structure of the respective encoder model, as it usually consists of near-complement layers to the ones included in the encoder, but in reverse order. A near-complement layer can be defined as a layer that is used to undo, up to a certain limit, the operations performed by the original one. For instance, in the case of Convolutional Neural Networks, a transposed convolutional layer is considered as the near-complement of the convolutional layer.

It can be easily affirmed that the "bottleneck" constitutes the key attribute of the autoencoder mechanism, since without its presence the model could easily learn to flawlessly duplicate the input values to the output, by simply passing them through multiple flat layers of the same width. Instead, autoencoder systems attempt to approximately reconstruct the original input approximately, based on the produced code. Therefore, an as much as possible effective intermediate representation is crucial, as it can facilitate the full decompression process. To this end, such kind of models are trained and hence optimized under the framework of the typical BackPropagation algorithm, as presented in section 2.2.2, by minimizing the reconstruction error, which quantifies in a compact form the differences between the original input and the generated output. However, there are four hyperparameters involved in the learning procedure of an autoencoder system, that need to be set before training:

- **Code size:** It is the most important hyperparameter concerning the tuning of the autoencoder, as it defines the number of nodes contained in the "bottleneck" layer and hence the size of the respective latent representation. Apparently, smaller code size implies higher compression.
- **Number of layers:** Another significant hyperparameter regarding the autoencoder architecture is the depth of the encoder and the decoder components respectively. While a higher depth increases model complexity, a lower depth is faster to process.
- **Number of nodes per layer:** The number of nodes in each layer defines the corresponding weight values that need to be adjusted. Typically, this quantity decreases along the subsequent layers of the encoder and increases accordingly in the decoder, due to their symmetry in terms of structure.
- **Loss function:** The loss function used to model the reconstruction error during the training process of an autoencoder system depends heavily on the input data type. More specifically, in case of images, the most frequently employed cost function is the Mean Squared Error (MSE), which can be computed as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.4.1}$$

where n denotes the input and output size (number of elements) respectively, $\hat{y}_i$ represents the i-th scalar value in the output vector and y_i indicates the corresponding target value, i.e the input one. However, if the input values are within the range $[0, 1]$, as in the MNIST dataset [106], the Binary Cross Entropy loss can be also utilized. Following the aforementioned mathematical notation, it can be calculated using the underneath formula:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \cdot \log \hat{y}_i + (1-y_i) \cdot \log\left(1-\hat{y}_i\right)\right) \tag{2.4.2}$$

Several variants of Autoencoder systems have been proposed over the last few years, in order to address defective aspects of the typical mechanism and improve essential model properties, such as generalizability. Some of the most prevailing types of Autoencoders are graphically displayed in Figure 2.4.2:

(A) Classic autoencoder mechanism.

(B) In the context of Denoising Autoencoders (DAEs) [107, 108], the input is partially corrupted by inserting noise or masking some individual values of the corresponding vector (depicted in white) in a stochastic manner. The model is trained to recover the original input.

(C) The Sparse Autoencoder (SAE) [109, 110] explicitly penalizes the use of hidden node connections (inactivated nodes are indicated in white), in such a way that each layer is sensitized toward specific attributes of the input data. In this case, a reduction in the number of nodes at the network hidden layers is not required.

(D) The Variational Autoencoder (VAE) learns the underlying distribution of the latent space, which can be defined by a mean value μ and a standard deviation σ and then decodes samples of this distribution in order to recover the original input. It can be employed for generative modeling tasks [111, 112].

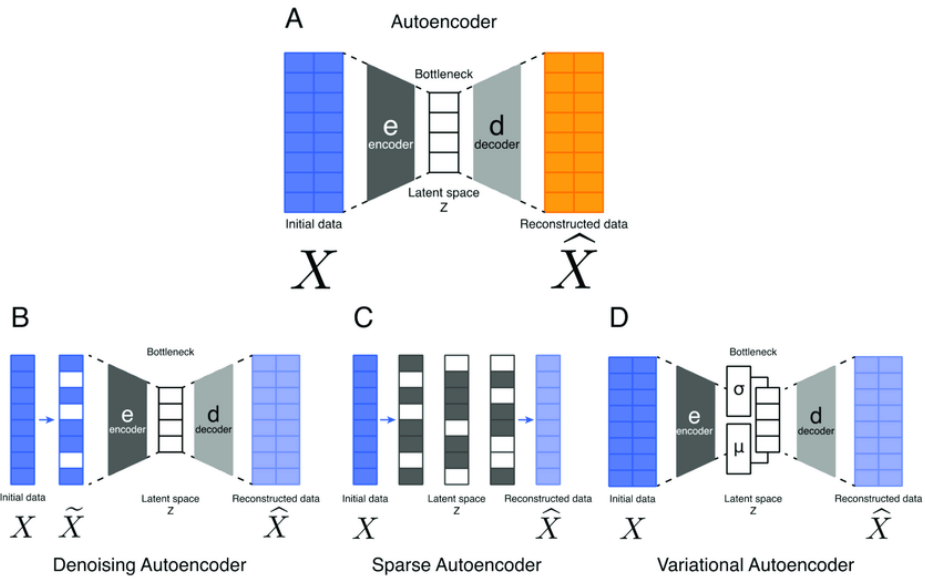


Figure 2.4.2: Different types of AE systems [113]

Chapter 3

Related Work

The aim of this chapter is to investigate the various aspects of the problem of Automatic Music Synthesis. In particular, section 3.1 briefly presents different music representations utilized under the operating framework of computers. Section 3.2 includes a general categorization of the distinct tasks into which the aforementioned research subject can be divided, as well as a detailed analysis of several techniques that can be applied for experimentation and different architectures and design choices that can be made. Section 3.3 lists some commonly used datasets in the examined research field and elaborates on their usefulness. Finally, in section 3.4, we present several metrics employed for the objective evaluation of the considered models, along with subjective assessment methods and emphasize on their nuances and importance.

3.1 Music Representations

As mentioned before, music can be represented in the framework of a computational machine using various storage formats that typically employ different data modalities. In the following subsections we provide a brief overview of the most frequently used representation forms in the research field of Automatic Music Synthesis.

3.1.1 MIDI

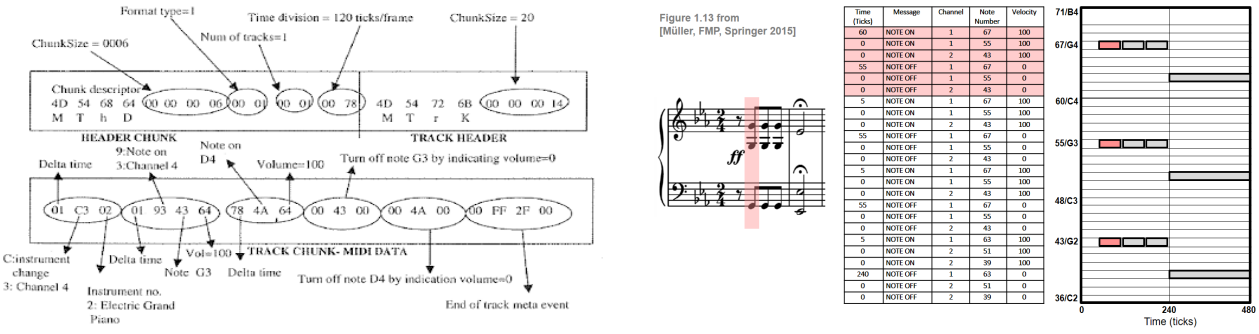


Figure 3.1.1: MIDI file format [114, 115]

As described in [86], MIDI (Musical Instrument Digital Interface) is a technical standard that represents a communication protocol applied in a wide variety of electronic musical instruments, computers and related audio devices for playing, editing, and recording music. As graphically illustrated in Figure 3.1.1, a MIDI file contains elements of 2 distinct types:

- *Header chunks:* A header chunk describes the file format and the number of track chunks.
- *Track chunks:* Each track chunk corresponds to one single header and includes the playable notes in the form of MIDI events. Each MIDI event is preceded by a delta-time, which represents the required number of ticks before its execution. This variable-length encoded value is predefined in the file header chunk. The MIDI event tokens are composed of 2 parts: the first 4 bits contain the actual command, while the rest the

respective MIDI channel. In total, there are 16 channels, 8 commands and 128 notes represented by a matching between pitches and unique numbers.

3.1.2 MusicXML

MusicXML is an XML-based file format for representing Western music notation in a symbolic fashion that is considered readable from both human and machine. Besides the typical information provided by the MIDI format, this encoding standard includes also a huge variety of additional music symbols, such as rest (pause), slur (symbol for legato performance), beam (connection among equally valued notes), key and time signatures, articulation marks (specify the length, volume and style of individual notes' attack) and organization tokens (e.g. repeat signs, da capo). It is usually employed for the storage of lead sheets. The term lead sheet stands for a basic form of musical notation that specifies only the essential elements of a song:

Figure 3.1.2: MusicXML [116]

- *melody*: typically represented by modern western music symbols.
- *lyrics*: inline text usually written below the notes.
- *harmony*: specified with chord signs above the staff.

3.1.3 Pianoroll

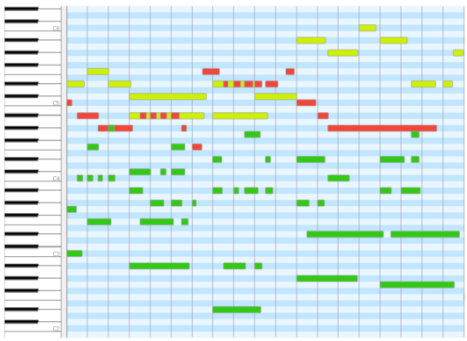


Figure 3.1.3: An example of MIDI file in a pianoroll view [117]

The pianoroll representation of a music piece is an image-like symbolic format inspired by the automatic piano. In particular, it constitutes a scoresheet-like matrix, where the horizontal axis X represents the increasing time and the vertical axis Y the pitch range. Notes are graphically displayed as bars in this grid, with the left edge of each bar indicating the "on" time of the corresponding pitch value and the bar length designating the respective note duration. This is where the name comes from, since the pattern of note bars, which is illustrated in Figure 3.1.3, resembles the holes in old paper player-piano rolls that force the musician to play only the hole-specified notes. In essence, pianoroll can be considered as a method that enables us to graphically display data included in the MIDI file format.

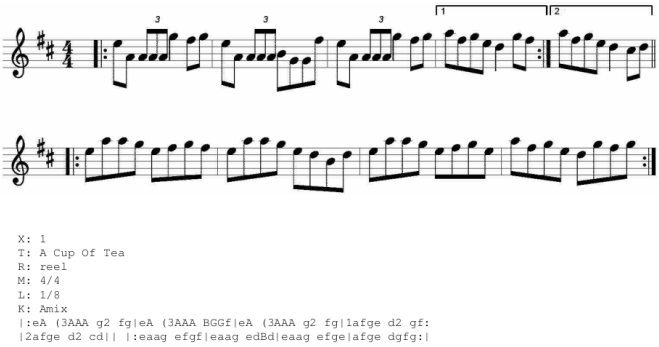
3.1.4 Text

ABC notation is a shorthand symbolic format for recording and storing music in plain text. It was originally developed by Chris Walshaw in the late 1980s for folk music and traditional fragments in Western Europe and was later extended to support representations of complete

classical music scores. The basic standard uses the letters a-g, A-G and z to represent the corresponding notes and rests, along with other signs for the note length, the music key, the accidentals (sharp, flat), etc. The first 6 lines of a music file in ABC notation constitute the header, which includes the following metadata:

- X: number of distinct tunes in the file
- T: title of the music song
- R: rhythm[1] of the song (hornpipe, jig, reel, waltz, polka, etc.)
- M: time signature
- L: default note length
- K: music key

The header is followed by the main text representing the melody, as shown in Figure 3.1.4.



```
X: 1
T: A Cup Of Tea
R: reel
M: 4/4
L: 1/8
K: Amix
|:eA (3AAA g2 fg|eA (3AAA BGGf|eA (3AAA g2 fg|lafge d2 gf:
|2afge d2 cd|| |:eaag efgf|eaag edBd|eaag efge|afge dgfg:|
```

Figure 3.1.4: ABC notation of the traditional song *A Cup of Tea* [1]

Another text-based file format for musical information storage is the Humdrum [118]. As graphically illustrated in Figure 3.1.5, each data stream corresponding to a polyphonic staff forms an individual column called *spine*. In the way, the musical time progresses by consecutive rows, since all elements of the same row occur simultaneously during performance. This spreadsheet-like grid can be augmented with additional columns representing other musical features, such as harmonic analysis labels in the form of scale degrees.

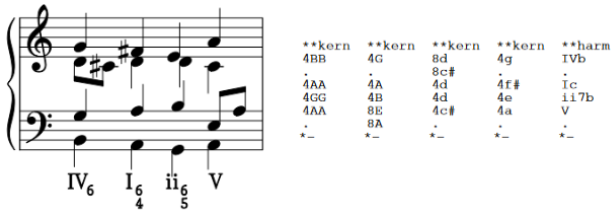


Figure 3.1.5: Example of humdrum data representation [119]

[1]This information is mostly used during playback.

3.1.5 Audio

The audio file format is an encoding standard for storing digital auditory information on a computer system. The sound data, which represent original musical pieces recorded by proper electronic devices, are stored in the form of raw bitstream usually embedded in a suitable container. The utilized bit layout can be compressed in order to reduce the size of the file. An example of the most frequently used audio file format is graphically illustrated in Figure 3.1.6.

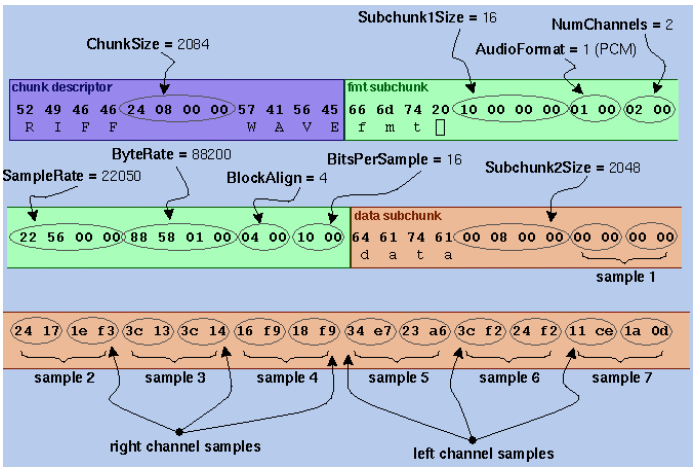


Figure 3.1.6: Waveform Audio File Format [120]

3.2 Tasks and Methods

It is hard to provide a complete taxonomy of the various types of Automatic Music Generation Systems and their respective tasks, since the research field is vast. Therefore, a more prevailing categorization is discussed in the following sections.

3.2.1 Generation from Scratch

As the name suggests, Generation from Scratch refers to a family of compositional methods that computationally approximate the process of creating novel musical content without subjecting to any formulated basis or prior source of musical information regarding the structure and the form of the generated pieces. Such approaches can be further categorized in accordance with the type of musical texture that is modeled under each generation framework. In music, the term "texture" encompasses various characteristics of a musical composition, ranging from melodic and harmonic materials to rhythmic attributes. All these features that collectively describe the overall quality of the different sounds are usually defined and correspondingly affected by multiple attributes, including the number and type of instruments, the tempo, the musical genre, the style and so forth. Based on this criterion, three main classes of generation systems can be distinguished:

Monophonic Music Generation

Monophony is the simplest form of musical texture, since it consists of a single unaccompanied melodic line, typically sung by a single singer or played by a single instrumentalist. The prominent feature of monophonic music, which is considered extremely useful from the perspective of the computational implementation, is the clear arrangement of the included notes in a temporal order, with only one sound event at each time slot. A typical example of a monophonic music score is presented in Figure 3.2.1.

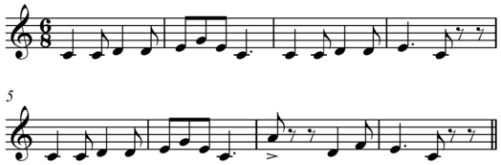


Figure 3.2.1: Monophonic music piece [121]

A complete chronology of monophonic music generation systems is thoroughly discussed in [122] and graphically depicted in the timeline diagram of Figure 3.2.2. As it might be seen, the initial approach towards the automation of monophonic synthesis was attempted by Bretan et al. [123] in 2016 via the introduction of a generative method based on unit selection. Their proposed system architecture comprises a deep autoencoder module that is applied to create a finite-size unit library, with the term unit corresponding to melodic phrases of variable bar length, and a generation module incorporating a Deep Structured Semantic Model (DSSM) with an LSTM in order to learn to predict the next unit for the given input. This system has been proven capable of producing novel monophonic melodies that do not conform to a provided harmonic context, but sometimes the lack of good units may lead to poor performance.

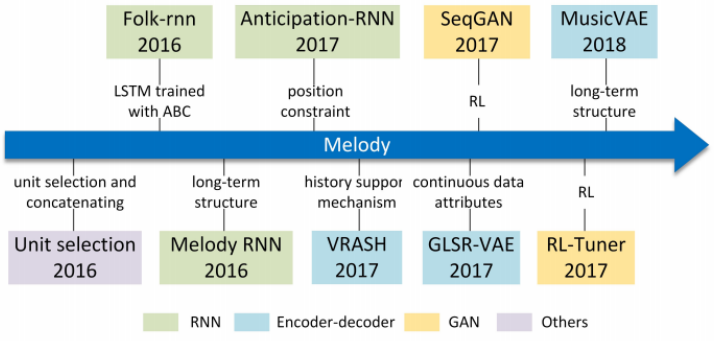


Figure 3.2.2: Chronology of monophonic music generation [122]

The subsequent works emphasize on the use of Recurrent Neural Networks for the automatic creation of novel monophonic musical content, since such systems are designed to process sequential data and capture temporal dependencies, as discussed in section 2.2.4 of chapter 2. Sturm et al. [124] developed generative LSTM models using approximately 23.000 monophonic music transcriptions stored in textual format. The selection of this data type is mainly based on the homogeneity it provides with respect to the stylistic conventions of the examined music genres (Celtic, Morris, etc.). They built a *char-rnn*, which is a character-based system modeling the joint probabilities for each textual character given the previous 50 characters and a *folk-rnn*, which is a token-based system modeling the joint probability of each token given all previous tokens in the current transcription. In this way, they were able to produce, at either a character or a token level, new transcriptions similar to the ones contained in the training material. Similarly, Hadjeres et al. [125] proposed a novel architecture called *Anticipation-RNN*, which enables the enforcement of user-defined positional constraints. The utilized their system for the generation of melodies in the style of the soprano parts of the J.S. Bach chorale harmonizations.

It can be easily observed that, besides RNNs, Variational AutoEncoders are also applied in the field of monophonic music synthesis. Roberts et al. [126] presented MusicVAE, a Recurrent Variational AutoEncoder for modeling monophonic sequences of musical notes with long-term structure. This architecture incorporates a novel hierarchical RNN as Decoder module, which initially segments the input sequence into non-overlapping parts and produces embeddings for each one of them. Afterwards, it utilizes the extracted latent representations in order to autoregressively generate each subsequence independently, addressing in this way the "posterior collapse" problem in recurrent VAEs, where the model tends to ignore the latent space.

Generative Adversarial Networks are also included within the recent approaches in monophonic music generation. Yu et al. [127] developed SeqGAN, a system that combines GANs with the Reinforcement Learning framework for the generation of music sequences composed of discrete tokens. In particular, the state represents the musical content generated so far, while the action corresponds to the selection of the next note to be produced. The Discriminator evaluates the complete sequence and the obtained reward signal is transmitted back to the intermediate state-action steps using Monte Carlo search. SeqGAN constitutes the first attempt towards the extension of GAN applications in problems dealing with discrete data.

It also exhibits excellent performance on generating sequences of other input modalities, such as poems and speech language.

Polyphonic Music Generation

Polyphony is a more complex type of musical texture as compared to monophony, which has been explicitly presented up above, since it typically involves multiple melodic lines of independent structure that are combined to flow and unfold in a coordinated manner, as graphically demonstrated in Figure 3.2.3. This formulation of musical entities implies dependencies along more than one axes, including sequential patterns across time and harmonic intervals occurring between notes that are played simultaneously. Therefore, automatically creating novel polyphonic music is undoubtedly a challenging problem in terms of computational implementation, since it requires a mechanism capable of capturing and modeling all the aforementioned time-related and harmonic features.

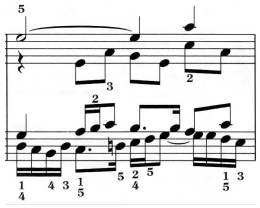


Figure 3.2.3: Polyphonic music piece [128]

A complete chronology of polyphonic music generation systems is extensively presented in [122] and graphically summarized in the timeline diagram of Figure 3.2.4.

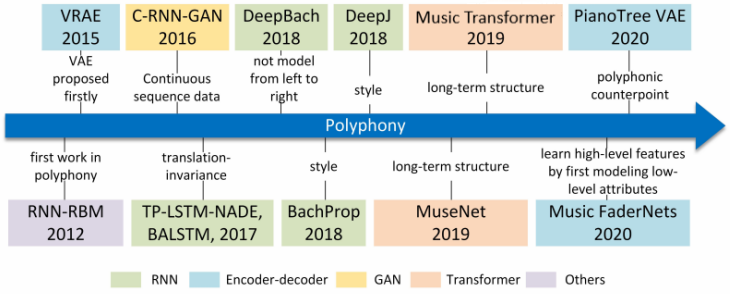


Figure 3.2.4: Chronology of polyphonic music generation [122]

The first comprehensive approach in the area of polyphony is the study of Boulanger-Lewandowski et al. [129]. They developed a probabilistic framework for the modeling of polyphonic sequences in the generalised pianoroll representation format with several applications in music transcription tasks. Their proposed system combines the structure of Restricted Boltzmann Machines (RBM) used to learn composite distributions over the so-called *simultaneities* at each timestep, i.e. the notes performed simultaneously formatting harmonic patterns, along with a typical Recurrent Neural Network capable of capturing the various temporal dependencies. They demonstrated that their innovative architecture outperforms popular methods in the field of Music Information Retrieval.

Classical music is probably the most characteristic example of polyphony and therefore constitutes a valuable source of training data for multiple studies in this research area. Hadjeres et al. [130] focused on the 4-part chorale harmonizations by Johann Sebastian Bach and introduced DeepBach, a dependency system for automatic generation of polyphonic music in the style of Bach. Their core architecture is established on a non-sequential approach of

music and consists of deep Recurrent Neural Networks that model the neighboring context of each note. The generation is performed via a pseudo-Gibbs sampling procedure, which also enables the enforcement of user-defined positional constraints, such as notes, rhythms or cadences, through an interactive graphical interface. The produced samples are quite convincing even under the assessment of professional musicians and to a large extent coherent without significant levels of plagiarism. Besides chorale-like pieces, DeepBach is also applicable in other polyphonic types.

In the context of RNN-based approaches towards polyphony we can also distinguish the study of Mao et al. [131]. They introduced DeepJ, an end-to-end generative framework for automatic creation of polyphonic music based on a predefined mixture of composer styles. Their proposed system leverages the Biaxial LSTM design [132], which models each note as a probability conditioned on the musical content of all the previous timesteps and also the notes within the current time step that have already been generated. They incorporated into this structure style and volume (pitch dynamics) embeddings in order to be able to learn and enforce specific compositional fashions in the generation process. The produced results indicated that DeepJ indeed provides control over the artistic style of the generated polyphonic music but cannot effectively address the problem of long-term structure.

In an attempt to tackle this limitation and handle long-range dependencies in polyphonic musical compositions, Huang et al. [5] proposed Music Transformer, a sequence model with a modified relative attention mechanism capable of capturing the self-reference in music that occurs on multiple timescales, including repeated motifs, phrases or even entire sections. Their developed system can be applied for generation of minute-long pieces in a token-based symbolic format and also be extended in an accompaniment generation framework via a sequence-to-sequence setup conditioned on melodies. The results derived from the evaluation demonstrated that their improved Transformer module can effectively create long musical sequences with coherent structure.

Deep unsupervised models such as Variational AutoEncoders are also applied in the research field of polyphony. One of the recent approaches is the work of Wang et al. [133]. They introduced PianoTree VAE, a novel model structured upon the VAE framework for learning semantically meaningful latent representations of polyphonic musical segments. The most intriguing characteristic of this system is that it employs a tree-structured musical syntax in order to reflect the hierarchy of the various musical entities within a composition. More specifically, the overall architecture can be regarded as a tree, where the nodes correspond to embeddings of concepts such as a score, an independent note event or a grouping formulation of simultaneous notes. The edges are modeled by recurrent networks that can either perform an encoding of the children into their parent or a decoding of a parent to its children. The conducted experiments indicated that PianoTree VAE can adequately capture the latent space of the musical data, resulting in decent reconstructions and therefore can be utilized for downstream generation of novel polyphonic content.

Multi-instrumental Music Generation

As the name suggests, **multi-track music** consists of several different instruments that collectively unfold over time in a cooperative manner. Each instrument performs its own musical part in the overall composition and is characterized by individual dynamics in terms

of harmonics and orchestration features. Multi-track music can be placed in the spectrum of polyphony, however in this case the interaction between the included tracks, which can be either monophonic or even polyphonic, forms an additional dimension of polyphonicity as compared to the typical texture establishment.

A complete chronology of multi-track music generation systems is extensively presented in [122] and graphically summarized in the timeline diagram of Figure 3.2.5.

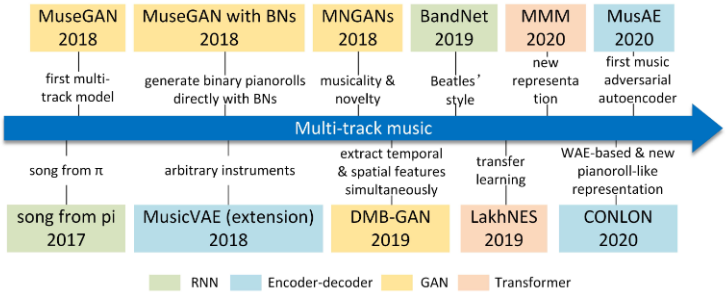


Figure 3.2.5: Chronology of multi-instrumental music generation [122]

The first comprehensive attempt for automatic creation of novel multi-track musical pieces was made by Chu et al. [134] in 2017. Inspired by the Song from π [135], a piano video on YouTube where the sequence of digits of the mathematical constant π is utilized for creation of music based on specified harmonic conversion rules, they introduced a framework for pop music generation that leverages the idea of gradually transforming randomness into acoustically pleasant sounds. To this end, they built a hierarchical Recurrent Neural Network conditioned on a specific music scale, where the bottom layers generate the melody, while the higher ones produce accompanying effects, such as chords and drums. They conducted several qualitative studies that demonstrated the validity of their proposed model and also expanded its applications towards neural dancing & karaoke and neural story singing.

Under the framework of GANs, the most popular model for multi-track polyphonic music generation is MuseGAN[2] [2], which has been proposed by Dong et al. in 2018 and laid the foundations for other subsequent related approaches in the field [19, 136]. The architecture of MuseGAN comprises three different model variants reflecting distinct human compositional practices. The overall implementation is structured upon a convolutional mechanism, which has been proven efficient at detecting local, translation-invariant patterns and hence music is represented in the pianoroll image-like symbolic format. The system is trained on a dataset containing over 100.000 bars of rock music and applied to generate samples of five tracks: bass, drums, guitar, piano and strings. The results highlighted the harmonic and rhythmic structure of the produced music, which nevertheless is still behind the level of human musicians.

In an effort to extend the Transformer architectures, that have recently presented particularly promising results in piano score generation [5], to a multi-instrumental setting, Donahue et al. [137] introduced LakhNES, a generative high-dimensional language model capable of capturing repeated patterns in long music sequences of multiple tracks. Their proposed methodology

[2]Since the MuseGAN model is the one that we base our experimental setup upon, we will present it in more detail in chapter 4.

is structured upon a pre-training mechanism in order to address the data availability problem in the multi-track domain. More-specifically, they initially trained their model using a large collection of heterogeneous music and then fine-tuned it in a smaller dataset containing four-instrument scores from an early video game sound synthesis chip called NES. The produced results indicated that this transfer learning approach improves the model performance from both quantitative and qualitative aspects.

In the context of modeling an efficient latent space for symbolic multi-track music, Valenti et al. [138] introduced MusAE, the first Music Adversarial Autoencoder. The major advantage of Adversarial Autoencoders compared to the standard architecture is their ability to impose a specific prior distribution on the latent variables via adversarial regularization in the form of an additional discriminative task [139]. MusAE leverages this mechanism for more controllable generation via the injection of high-level information concerning musical genre and style into the latent space. It can also be applied for reconstruction of musical phrases with high accuracy and creation of realistic interpolations between musical sequences, by properly modifying the latent attributes of the different tracks.

3.2.2 Music Arrangement

Music Arrangement constitutes a different approach towards the automatic creation of novel musical content established on methods for reconstructing and reconceptualizing musical compositions. This process typically involves alterations and modifications of the original pieces in terms of harmony, orchestration, melodic material or chord progression, based on specific reference information of any structure. Following the categorization that is extensively discussed in [117], Music Arrangement can refer to three distinct conditional generation tasks, which are schematically illustrated in Figure 3.2.6 for the piano case and will be thoroughly examined in the subsequent sections. As can be observed, Arrangement acts as a bridge between the three fundamental forms of music representation: the full score, the audio and the lead sheet, which specifies only the essential elements of a musical composition (melody, harmony, lyrics). The aforementioned modal formats are generally employed as reference conditions for each corresponding family of arrangement problems.

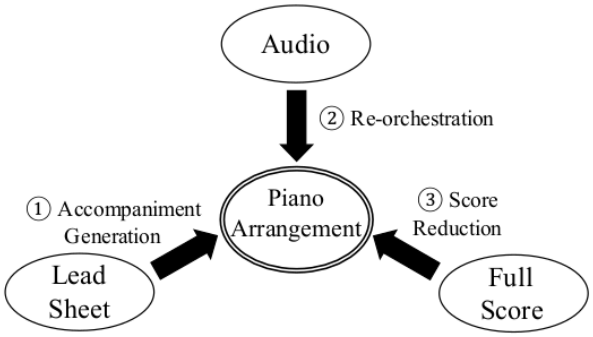


Figure 3.2.6: Illustration of the role of piano arrangement in the three forms of music composition [117]

Accompaniment Generation

In music the term *accompaniment* refers to the musical part that supports rhythmically and/or harmonically the main theme of a musical composition, which can be either a song or an instrumental piece. There is a wide variety of accompaniment schemes depending on the music genre and the overall configuration of included instruments or vocals. From the human perspective, the creation of accompaniment is typically established on a primary idea for a main melody that is gradually enriched with chords or other supplemental patterns and figures. Therefore, the automation of this process under the framework of a computational machine inevitably involves a source of prior information on which the accompaniment generation is conditioned. In the general case, this musical information is provided to the model in the form of a lead sheet, as depicted in Figure 3.2.6.

Throughout the years numerous studies have been published in an attempt to address the particularly challenging problem of Automatic Accompaniment Generation. It may be certain that the first comprehensive approach has been introduced by Simon et al. [140] in 2008. They proposed MySong, an interactive framework for automatic harmonization of vocal melodies designed to be intuitive to users who lack musical experience. At its core, MySong is structured upon a Hidden Markov Model[3] that learns the statistics of the chord transitions presented in the training database along with the association between the notes in the conditional melodic lines and the observed chord types. The interface of MySong application leverages the probabilistic data derived from this process in order to select the best fitting chord progression for novel melodies that are fed as input to the system from users by just singing into a microphone. Since there are more than one matching accompaniments for a given melody, MySong allows users to adjust and modify the proposed chords using parameters representing easy perceivable musical properties. The evaluation results demonstrated the quality of the produced accompaniments and also highlighted the usability of the system even by non-musicians.

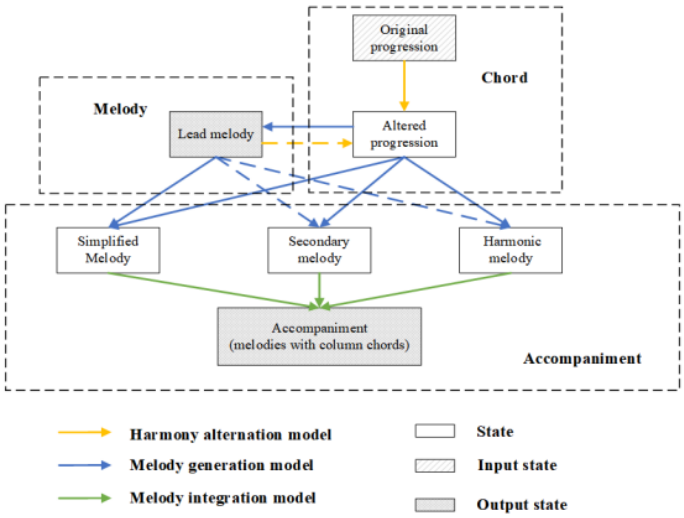


Figure 3.2.7: System diagram of the pop-song automation framework [141]

[3] HMMs constitute a class of graphical probabilistic models used to describe the relation and evolution of observable events depending on set of hidden unknown variables that cannot be directly inspected.

In an attempt to unify tasks such as the lead melody generation and the accompaniment arrangement that are mainly treated as separate, Wang et al. [141] developed a pop-song automation framework, which is graphically illustrated in Figure 3.2.7. As can be seen, their proposed system consists of three different models represented by the corresponding coloured arrows: the *harmony alternation model*, the *melody generation model* and the *melody integration model*. Initially, the harmony alteration model modifies properly the input chord progression with respect to a specified music style. The altered chord sequence is then fed into the melody generation model that produces the lead melody along with various accompaniment textures in the form of additional melodic lines or patterns via seasonal ARMA (AutoRegressive Moving Average) processes [142], which are tools for the statistical modeling of time series. Finally, the melody integration model combines the produced melodies into the final accompaniment scheme. The experimental results demonstrated that the generated melodies are characterized by desirable properties, such as musicality and overall structure.

In the context of online accompaniment generation, Jiang et al. [143] introduced RL-Duet, a novel model that supports interactive real-time generation of musical content in a human-machine duet setup. Since this framework inevitably requires the computer's response to human input, the examined problem is formulated upon a Reinforcement Learning basis. More specifically, the generation agent learns a policy for performing actions, i.e. producing musical notes, based on the previously generated context, which can be regarded as the state of the algorithm and includes the long-term temporal structure, as well as the inter-part harmonization. However, the key feature of this method lies in the employed reward system, which isn't established on hand-crafted compositional rules but instead is derived from the training data. The experiments have shown that RL-Duet is able to produce diverse machine counterparts of high quality, harmony and coherence.

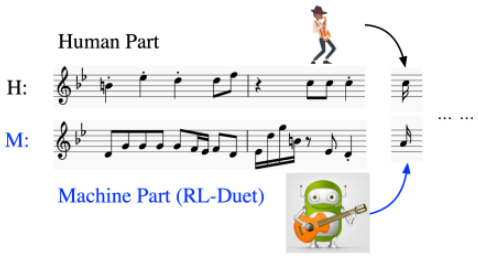


Figure 3.2.8: RL-Duet [143]

Another pop music accompaniment generation framework named PopMAG has been introduced by Ren et al. in [144]. This approach aims at addressing the challenges of modeling the harmonic structure and capturing the long-term dependencies in accompaniments consisting of multiple distinct tracks. For this purpose the authors developed a novel symbolic representation format called Multi-track MIDI. MuMIDI encodes multi-track MIDI events into a single sequence and also integrates different note attributes, such as pitch, velocity and duration, into one step, as opposed to the standard formulation, in order to confine the overall sequence length. The system architecture is established on an enhanced sequence-to-sequence model that employs a transformer-based structure for both encoder and decoder components and operates in an autoregressive manner in order to predict the accompaniment tokens. The experimental analysis demonstrated that PopMAG outperforms, both objectively and subjectively, other state-of-the-art models.

Transcription and Reorchestration

In music the term *transcription* refers to the practice of recording auditory pieces into a written form of symbolic music notation. As discussed in [145], this process from the human

perspective is intrinsically interrelated with higher mental abilities of the brain, such as the perception of the various sounds, the identification of the included instruments, the estimation of musical attributes (pitch, rhythm, onset, offset, etc.) and the analysis of expressive timing and dynamics. To this end, the design of algorithms that implement the autonomous conversion of acoustic signals into music sheets is considered one of the most challenging tasks in the field. It is worth mentioning that the majority of transcription methods typically involve a re-orchestration of the input music audio. More specifically, music parts of the original work are assigned or restructured in order to be performed by different instruments than the ones forming the initial piece. In this way, the musical ideas incorporated into the original composition are preserved in the produced score, but expressed under a diverse musical arrangement.

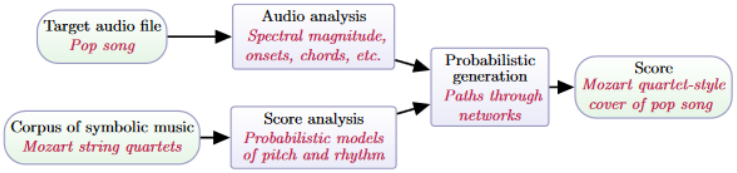


Figure 3.2.9: Overview of Song2Quartet [146]

One of the most popular approaches in Automatic Music Transcription is the work of Percival et al. [146]. They introduced Song2Quartet, a generative system that produces string quartet cover versions of popular songs. An overview of the proposed methodology is graphically illustrated in the diagram of Figure 3.2.9. As can be seen, Song2Quartet includes an audio analysis module, which is responsible for detecting recognizable musical features of the target pop song, such as themes, rhythms and chord voicings, via time-frequency spectral processing and a score analysis module, which captures characteristics of the string quartet style from a symbolic corpus of classical music. These two modules are combined under a probabilistic formulation established on the framework of dynamic programming, which results in the construction of a statistical network of possible musical notes for each timestep. Consequently, the musical score of the cover version is generated by detecting the optimal path through this network. The produced results confirm the effectiveness of Song2Quartet over the creation of pieces that follow the conventions of classical string quartet music, retaining at the same time the prevailing features of the target song.

Score Reduction

In music, the term *reduction* refers to an arrangement of an existing score or composition in general that involves modifications of the structural information in such a way that the overall musical complexity is reduced. For instance, the number of the included parts may be altered or the rhythmical attributes may be simplified. It also encompasses cases where musical pieces that are originally intended for multiple instruments are re-arranged to be performed by smaller musical ensembles or even a single instrument.

Piano reduction from ensemble scores has been traditionally one of the most widely investigated fields of Music Arrangement, forming the basis for more advanced techniques and approaches in the research area. Nakamura and Sagayama [147] formulated piano reduction as an optimization problem of consistency between the original and the produced piano

score under constraints on the degree of performance difficulty. More specifically, in order to be able to measure quantitatively the performance difficulty they initially applied a stochastic HMM-based model established on piano fingering principles for each hand. This model provides a statistical interpretation of the naturalness characterizing each fingering, by assuming that every output probability depends on the performed pitches only through their geometrical positions on the keyboard, which is represented as a two-dimensional lattice and graphically depicted in Figure 3.2.10. Their proposed piano reduction algorithm is based on a combination of the two-hand fingering models along with a stochastic method that involves probabilistic measures of how notes in ensemble music scores are likely to be edited. The experimental results confirmed the effectiveness of their system over the creation of piano reductions with controllable performance difficulty in terms of note and chord density, tempo and rhythm. They also discussed a possible extension to other forms of music arrangement, by replacing the fingering model with an equivalent model of the target instrument and adapting properly the editing probabilities.

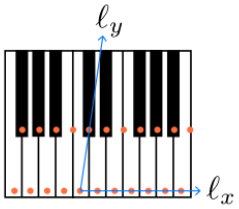


Figure 3.2.10: Representation of position on the piano keyboard with a two-dimensional lattice [147]

Another approach towards the computational implementation of automatic piano reduction under the framework of controllable performance difficulty is presented in [148]. Nakamura and Yoshii leveraged the ideas described in [147] and introduced a statistical modeling method established on the concept of iterative optimization. Following similar strategy, the developed quantitative measures of the playability level with respect to the performance error rate, using statistical generative models for piano scores. For the probabilistic description of the musical fidelity degree, they integrated a prior piano-score module along with one representing how ensemble scores are likely to be edited. The overall system produces the reduced scores via an iterative inference procedure. The conducted experiments demonstrated that this iterative optimization approach improves the controllability over the performance difficulty and the corresponding musical fidelity to a large extent, as compared to the method utilized in [147]. Moreover, the results of both subjective and objective evaluation indicated that the incorporation of the sequential dependence of pitches and the fingering motion (Figure 3.2.11) in the piano-score model has a beneficial impact on the quality and naturalness of the produced scores, especially in high-difficulty cases.

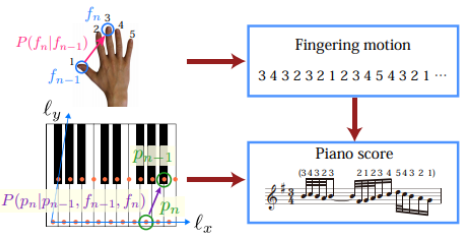


Figure 3.2.11: Piano-score model incorporating fingering motion [148]

3.2.3 Music Style Transfer

The term *style transfer* has been originally introduced in the field of Artificial Intelligence by Gatys et al. [149] in an attempt to computationally process and handle the artistic content of natural images. More specifically, this technique refers to capturing explicit features of an image in the context of stylistic information and applying them to a different image.

The particularly promising results of this method in computer vision tasks have inspired the interest of the research community for experimentation in the music domain. Equivalently, music style transfer is defined as a practice for automatic generation of novel human-like music established on the disentanglement and the reattachment of the musical content and the musical style of different pieces. More specifically, in the general case, a style encoding is initially separated from the latent representation of a song and then inserted into a suitable generation framework that retains the necessary information about the target musical content. As discussed in [150], the musical style isn't a well-defined concept from a scientific point of view, as it encompasses multiple levels of attributes, such as the historic period, the composer, the performance characteristics and the involved emotions, the music genre or other texture elements. According to the analysis in [122], a more prevailing categorization of the various musical style transfer methods that employ different interpretations of the term is the following:

- **Score Style Transfer:** In the context of music scores, the style generally refers to explicit compositional characteristics intrinsically interrelated with the music genre, such as the scale type, the tonal motifs, the chord progressions [151] or the rhythmic attributes [152]. There is also a class of techniques for creation of novel stylistic formats in the symbolic domain based on fusion processes [153].
- **Audio Style Transfer:** In the context of auditory representations, the music style involves sound features, such as timbre, i.e. the tone quality of notes performed by different instruments [154] and audio texture, which refers to the overall temporal homogeneity of the acoustic events [155, 156].
- **Singing Style Transfer:** Singing is a process that integrates music content along with textural information in a particularly expressive fashion. Singing style transfer includes Singing Voice Conversion (SVC) methods, where the singer's timbre is properly altered with respect to a specified target without affecting the linguistic context [157] and also Speech-to-Sing (STS) approaches, which are established on the conversion of speech into singing voice [158].
- **Composition Style Transfer:** As discussed in [150], composition style transfer involves modifying attributes of a musical piece in a meaningful way, retaining at the same time identifiable characteristics, such as melodic patterns and underlying harmonic elements [159, 160].

3.2.4 Music Completion/Inpainting

Another form of automatic creation of novel musical content under fitting constraints is the so-called Music Completion or Inpainting. This generation practice refers to filling the missing or lost information in a piece of music. In contrast to previous approaches in the field that implement the generation process in a sequential manner, this method embraces the iterative and non-serial standards of human music creation and leverages both past and future musical context in order to produce intermediate segments and phrases. In this way, the generation procedure is extended to a more interactive and collaborative framework between human and machine, allowing users to adjust specific parts of the composition according to their personal preferences or subjective criteria. According to the analysis in [122], the various

musical inpainting methods can be categorized as follows:

- **Score Inpainting:** As described in [161], within the framework of score generation the inpainting process is formulated as the modeling task of creating a musical fragment $\mathcal{C}_i$, typically composed of a small number of bars, which can connect a past musical context $\mathcal{C}_p$ and a future musical context $\mathcal{C}_f$ in a musically meaningful and consistent manner (Figure 3.2.12). The developed score inpainting systems are generally based on an interactive music generation perspective, enabling users to edit the produced samples according to their personal ideas and acquire novel machine-generated suggestions [161, 162, 163].

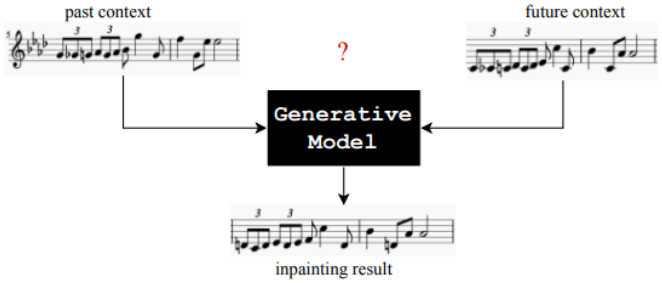


Figure 3.2.12: Musical Score Inpainting [161]

- **Audio Inpainting:** Audio inpainting refers to the restoration of lost or missing sound information via processing techniques, such as interpolation, extrapolation and signal reconstruction, applied in the waveform that corresponds to the examined auditory content or other sound representation formats, as graphically illustrated in Figure 3.2.13. There are systems designed to accurately recover musical and instrumental samples of short damage in the range of 10ms [164], as well as models that attempt to tackle the challenging problem of long corruptions [165]. In this case, the complete restoration is considered unrealistic and therefore the inpainting algorithms typically introduce new auditory information semantically compatible with the surrounding musical context.

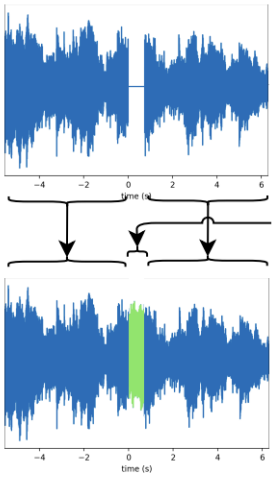


Figure 3.2.13: Audio Inpainting [165]

3.3 Datasets

In the computer music community and especially the field of generating modeling, the dataset, i.e the collection of data instances used to enable a computational machine to produce novel musical content in an autonomous manner, plays undoubtedly a crucial role in the formulation of the whole learning mechanism. The trade-off between the number of included samples and the degree of consistency among them constitutes a major issue when developing deep learning algorithms. On the one hand, if the utilized dataset is pretty heterogeneous, a good generative model should be able to distinguish different subcategories and hence generalize well. On the other hand, if there only subtle differences between the contained classes, then it is significant to examine if the so-called "average" model can lead to musically interesting results.

The selection of a suitable dataset is also closely related to the implemented generation task and more specifically the form of music representation under the framework of a computer, which, as mentioned before, may include various data modalities. In the context of our analysis, we follow the work of Ji, Luo and Yang [122], which attempts to categorize the various datasets that are commonly used in different studies and approaches in the research area of Automatic Music Generation from the perspective of the employed music storage format. A summary of the aforementioned is graphically presented in Table 3.1 along with supplementary information at the end of this section.

3.3.1 MIDI

One of the most popular datasets in the field of symbolic music synthesis [129, 163] is structured upon the compositions of J.S.Bach. The so-called JSB Chorus [166] is an entire corpus of 402 four-part harmonized chorales for soprano, alto, tenor and bass, which can be directly obtained via the Python package of the Music21 [167] toolkit used for analyzing, searching and converting music in symbolic format. However, this dataset is significantly small and also lacks expressive information. In an attempt to tackle this limitation, Ferreira et al. [168] introduced VGMIDI, a novel dataset consisting of 823 MIDI piano pieces derived from video game soundtracks. The duration of the included samples ranges from 26 seconds to 3 minutes. A small proportion of the contained pieces (around 95 MIDI files) are annotated by 30 human subjects according to Circumplex (valence-arousal) model of emotion and hence are accompanied by a sentiment label.

One of the largest symbolic music corpora including 176.581 unique MIDI files is the Lakh MIDI Dataset or LMD for short, which has been created by Colin Raffel [16]. This dataset provides unlimited polyphonic and expressive attributes and contains various genres, instruments and time periods. It incorporates the following:

- *LMD-full:* The full collection of 176.581 MIDI files, without duplicates. Each file is named according to its corresponding MD5 checksum.
- *LMD-matched:* A subset of 45.129 files that have been matched with entries in the Million Song Dataset (MSD) [17].
- *LMD-aligned:* All files from the LMD-matched that are aligned to the 7digital preview

MP3s in the MSD.

The Projective Orchestral Database (POD) emerged from the study of Crestel et al. [169] over the musical correlation between piano scores and their respective orchestral arrangements. The authors developed a novel method for automatic alignment between piano and orchestral versions, an example of which is graphically illustrated in Figure 3.3.1 and also introduced the corresponding task of automatic orchestration of piano scores. This process resulted in a novel symbolic database containing 392 MIDI files.

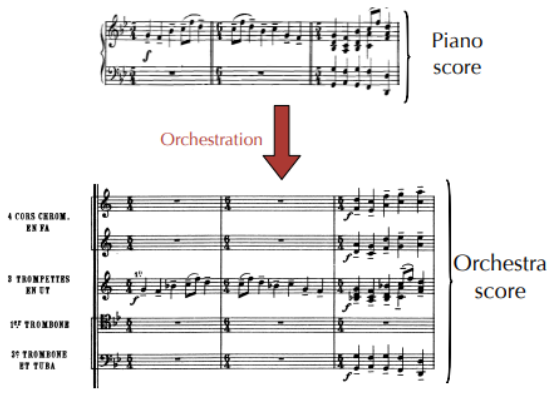


Figure 3.3.1: Projective orchestration of the first 3 bars of Modest Mussorgsky's piano piece *Pictures at an Exhibition* by Maurice Ravel [169]

In the research area of performance generation over polyphonic music the e-Piano Competition Dataset [170] can be distinguished, as it constitutes the largest public collection of solo performances by professional pianists in MIDI file format, including compositions of Chopin and Liszt, as well as some Mozart sonatas too. This dataset provides enhanced control of timing and performance dynamics, along with high-quality expressive attributes, but is does not contain the corresponding music scores of the included pieces.

Following similar concept, ByteDance has recently released GiantMIDI-Piano [171], a significantly large collection of classical pieces for piano in MIDI file format, including 10.854 music works of 2.784 different composers. The dataset has been formed via the utilization of an open-source piano transcription system of high resolution [172], capable of converting audio files in MIDI format. The produced MIDI files incorporate performance information, such as pedal events and dynamics elements.

3.3.2 MusicXML

Theorytab Dataset (TTD) [173] is the largest publicly accessible collection of lead sheet fragments stored in XML format, containing around 16K unique musical pieces. This corpus has been collaboratively created by users of the online music forum *TheoryTab* via uploading snippets of popular songs and voluntarily annotating them with structural (e.g. Intro, Verse, Chorus) and also genre labels. The aforementioned forum is hosted by Hooktheory, a company that builds music software and provides interactive learning material in order to help musicians to gain further insights into the process of synthesis.

Yeh et al. [174] leveraged TheoryTab and collected a new set of samples, called Hooktheory Lead Sheet Dataset or HLSD for short [175]. This corpus comprises 11.329 lead sheets only in 4/4 time signature. Each segment consists of high-quality human-transcribed melodies along with their corresponding chord progressions, which are denoted by two different types of symbols: literals (e.g. Gmaj7) and scale degrees with respect to the given key (e.g. VI_7), including also inversions if they are applicable and other chord extensions.

Another lead sheet dataset with applications in chord progression generation tasks [176] is derived from the Wikifonia database, a public repository of samples in MusicXML format that terminated its service in 2013. However, a subset of the provided data that contains 5.533 western lead sheets of varying genres, such as rock, pop, country, jazz, folk, R&B, children, etc., has been retained.

An additional source of processable music scores that are transcribed voluntarily by the community users and can be exported in MusicXML format is the web platform MuseScore. Jeong et al. [177] used this online database in order to collect the scores that correspond to the recordings of Yamaha e-piano junior competitions, resulting in a total of 26 pieces by 16 composers.

3.3.3 Pianoroll

Only a few datasets are stored in the pianoroll format, since it is particularly easy to convert pianorolls into MIDI files and vice versa through the open-source Python package Pypianoroll that has been introduced by Dong, Hsiao and Yang [178] for handling such symbolic representations. One of them is the Lakh Pianoroll Dataset or LPD for short, a large collection of 174.154 multi-track pianorolls derived from the Lakh MIDI Dataset [16] and utilized during the training process of the MuseGAN system [2]. More specifically, the authors employed the LMD-matched version, which is presented up above, and applied specific preprocessing operations in order to acquire pianorolls of the desirable configuration in terms of music attributes.

3.3.4 Text

The Nottingham Music Database (NMD) [179] is a collection of 1200 British and American folk songs stored in a special text format. Using a specific QBasic program called NMD2ABC and some Perl scripts, a significantly large proportion of the included pieces have been translated into ABC notation. The created dataset has been recently edited by Seymour Shlien, who corrected a few problems in terms of restoring missing beats. Since the songs consist of simple melodic lines accompanied by chord progressions, the start-up company Jukedeck performed a cleaning processing of the ABC version of the database by decoupling the melody and the chord part for each file [180]. Another free online dataset focusing on traditional music is the ABC tune book of Henrik Norbeck [181], which consists of more than 2.800 music scores and lyrics of Irish and Swedish songs in ABC format.

As regards the Humdrum data format, it is worth mentioning the KernScores online library, which has been developed in order to organize and store music scores derived from various sound sources following humdrum notation and contains more than 7 million notes in 108.703

files. This website additionally provides direct conversion into MIDI format and PDF files of music scores in case of copyright-free scanning. Cherla et al. [182] employed for the evaluation of their proposed model the Essen Folksong Collection from KernScores library, which consists of melodies and chorales from 7 distinct traditions.

3.3.5 Audio

The NSynth dataset is a large-scale and high-quality corpus consisting of 305.979 4-second monophonic audio snippets of musical tones played by 1.006 different instruments. It has been introduced by Engel et al. [183] for the training of their proposed audio synthesis model, which is based on the idea of music factorization into individual note entities. Each musical note corresponds to a unique pitch, timbre (tone color) and envelope (ADSR). Samples for every included instrument have been generated with sampling rate of 16kHz using all pitches (21-108) and 5 distinct velocities (25, 50, 75, 100, 127) of the standard MIDI piano. Each snippet is accompanied by 3 additional metadata attributes, derived from human assessment and heuristic algorithms:

- *Source:* The sound generation practice for the musical tone (acoustic, electronic, synthetic).
- *Family:* The class of the employed instrument.
- *Qualities:* Sonic qualities of the note, depending upon its corresponding waveform.

One of the largest publicly accessible datasets in auditory format with several applications in the field of Music Information Retrieval (MIR) is the Free Music Archive or FMA for short [184]. This corpus contains 917GB of audio files arranged in a hierarchical taxonomy of 161 genres, including 106.574 pieces from 16.341 artists and 14.854 albums. Each sample is accompanied by precomputed features, along with track- and user-level metadata, tags and artist biographies. The following subsets are also available, as they extend the utilization of FMA in cases of low computational resources:

- *Full*: the complete dataset as presented above.
- *Large*: the full dataset with audio trimmed to 30-second clips extracted from the middle of the tracks.
- *Medium*: the collection of 25.000 30-second track clips annotated with a single genre label and sampled in accordance with the completeness of their metadata.
- *Small*: the collection of the top 1.000 30-second clips from the 8 most popular genres of the medium set.

Another public but also significantly smaller dataset (around an order of magnitude as compared to NSynth) is the Minst, which can be considered equivalent to the MNIST (primary dataset in Computer Vision) in the field of audio processing. This corpus incorporates 4 disparate solo instrument collections (*University of Iowa-MIS*, *Philharmonia*, *RWC*, *Goodsounds*), resulting in a total of 50.912 notes from 12 different instruments.

The most popular collection of audio recordings in the field of signing voice research with applications to Speech-to-Singing (STS) tasks is the so-called NUS Sung and Spoken Lyrics

Corpus (NUS-48E corpus) [185]. This dataset contains 115 minutes of singing data and 54 minutes of speech data corresponding to 48 (20 unique) English songs performed by 12 subjects (6 males and 6 females) in various sound and accent types. It also provides a complete set of transcriptions and duration annotations at the phone level.

3.3.6 Multimodality

Another family of music datasets employ multi-modal information derived from different sources, such as scores, lyrics, audio files, etc. in order to provide a more complete and general representation of the included data. One of them is the MAESTRO dataset [186], which consists of 1.282 real virtuosic piano performances stored in MIDI and audio formats. The contained musical pieces (approximately 430) are derived from the 9-year International Piano e-Competition and are mainly of classical genre, including composers from the 17^{th} to early 20^{th} century. Each pair of audio and MIDI data is annotated with additional information, such as the composer, the title and the year of performance. The achieved alignment degree between the two modalities is ~ 3 ms.

In the research field of expressive drum modeling the lack of proper training corpora in terms of sufficient data of varying genres has led to the creation of a novel multi-modal dataset called GMD (Groove MIDI Dataset) [187]. This corpus consists of 22.000 bars of tempo-aligned expressive drumming performed by 10 musicians (5 professionals and 5 amateurs) in the presence of metronome. The utilization of this device enables the quantization of played notes to the nearest time division, yielding in this way a musical score, but also enforces a consistent tempo that limits the drummer's musical expression. The sound data were recorded in 1.150 MIDI files via Roland TD-11 electronic drum kit, which has been proven capable of capturing high-quality performance features. Each pair of samples is also annotated with relevant metadata, such as music genre, tempo and anonymized drummer identifiers.

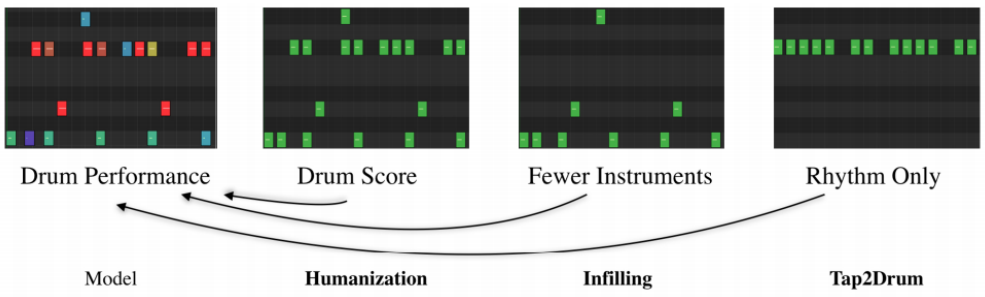


Figure 3.3.2: Learning to groove through inverse sequence transformations for drumming [187]

The multi-modal corpora are particularly useful in the context of Musical Arrangement, as they provide alternative representations of the same musical content. According to Wang et al. [117] a proper arrangement dataset should be characterized by the following properties:

- *Style-consistency:* The re-conceptualization of the original piece should be style-consistent.

- *Time alignment:* The arrangement should be time-aligned with the original music version (audio, lead sheet, full score), in order to serve as a supervisor of the learning algorithm.
- *Sufficient annotations:* The dataset should provide additional structured information, such as key, beat and chord labels, in order to ensure better control of the generation process.

Despite several promising generative models, the lack of corpora satisfying the aforementioned requirements becomes one of the main bottlenecks in the research area. To this end, Wang et al. [117] proposed a novel dataset called POP909. As the name suggests, it is structured upon 909 popular songs composed by 462 artists, spanning around 60 years from the earliest in 1950s to the latest around 2010. In particular, multiple versions of piano arrangements created by professional musicians are included for each song and stored in two aligned modalities: MIDI format and original audio. Furthermore, each song is accompanied by manually annotated tempo curves and machine-extracted key and chord labels using MIR algorithms. Aside from the arrangement task, POP909 is considered as a high-quality source for structural and cross-modal music generation.

In an attempt to overcome problems related to automatic alignment methods, Foscarin et al. [188] introduced ASAP (abbreviation of Aligned Scores and Performances), a novel dataset comprising 222 digital musical scores, stored in MusicXML and MIDI format, aligned with 1.068 performances, recorded as MIDI and partially audio files with approximately 3ms precision, of Western classical piano music from 15 composers. Each pair of score and performance samples is annotated with metadata, including the composer and the title of the piece, along with supplementary information, such as the exact positions of all beats, downbeats, time and key signature changes, as demonstrated in Figure 3.3.3. These annotations are automatically produced by a new workflow that combines score analysis and alignment algorithms, aiming at a radical reduce of the time required for manual processing. ASAP is the largest known corpus providing fine-grained alignment between text, MIDI and audio data and hence constitutes a valuable source for a wide variety of MIR tasks.

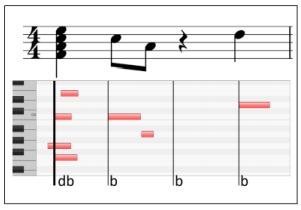
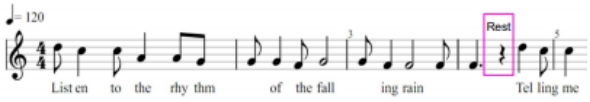


Figure 3.3.3: Beat and downbeat annotations produced by ASAP workflow [188]



Lyrics	List	en	to	the	rhy	thm	of	the	fall	ing	rain	Tel	ing	me
Note	D5	C5	C5	A4	A4	G4	G4	F4	G4	F4	F4	D5	C5	C5
Duration	0.5	1	0.5	1	0.5	1	1	0.5	2.5	1	4	1	0.5	1
Rest	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 3.3.4: An example of alignment between lyrics and melody [189]

Automatically generating melodic lines from lyrics is considered one of the most challenging problems in the research field of AI music, as it requires the detection of underlying latent

associations between the two data representations. In order to tackle the limited availability of paired lyrics-melody datasets, Yu and Canales [189] created a novel corpus comprising 12.197 MIDI songs that are provided as aligned pairs of melody and corresponding lyrics. This collection incorporates samples derived from different music sources, such as the LMD-full (7.998 files) and the reddit MIDI dataset (4.199 files).

Existing models in the field of Music Information Retrieval typically focus on representation learning methods for 2 distinct data modalities [190]. In an attempt to extend the research towards multiple data types, Zeng et al. [191] introduced the Music Ternary Modalities Dataset (MTM). As illustrated in Figure 3.3.5, this corpus includes 3 different aligned data modalities: sheet music, lyrics and music audio in the form of spectrograms. For each song the respective representations are extracted by specialized pretrained models.

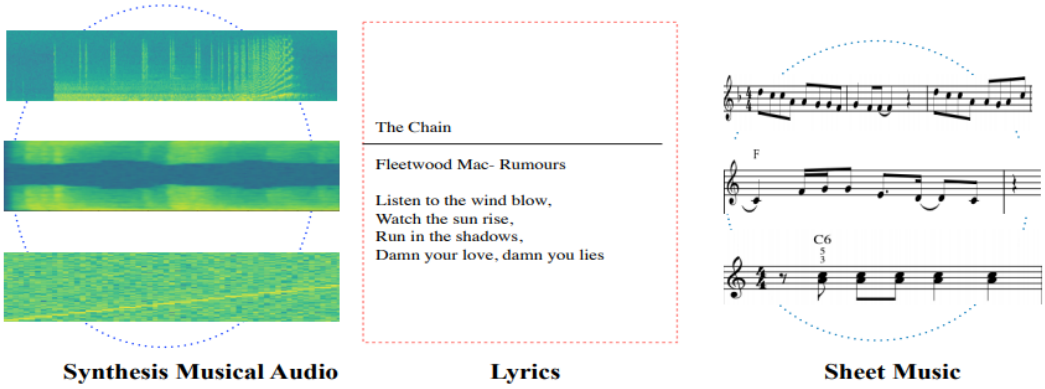


Figure 3.3.5: Examples of 3 data modalities in MTM Dataset [191]

Format	Name	Modality			Applicable Task	Size
		Score	Performance	Audio		
MIDI	JSB Chorus	✓			Polyphonic	402 Bach chorales
	VGMIDI	✓	✓		Polyphonic with sentiment	823 piano soundtracks
	LMD	✓	✓		Multi-instrumental	176.581 MIDI files
	POD	✓			Orchestral	392 pairs of MIDI files
	e-Piano CD	✓	✓		Polyphonic & Performance	∼1400 MIDI files
	BitMIDI	✓			Polyphonic	113.244 MIDI files
	Archives	✓			Polyphonic	Classical music
	Internet	✓			Polyphonic & Style	130.000 pieces of 8 genres
	ADL Piano MIDI	✓	✓		Polyphonic	11.086 piano MIDIs
	GiantMIDI-Piano	✓	✓		Polyphonic	10.854 MIDI files
MusicXML	TheoryTab	✓			Polyphonic	16K lead sheets
	Hooktheory	✓			Polyphonic	11.329 lead sheets
	Wikifonia	✓			Polyphonic	2.252 lead sheets
	MuseScore	✓	✓		Performance	Yamaha e-Competitions
Pianoroll	LMD	✓	✓		Multi-instrumental	size of LMD
Text	NMD	✓			Monophonic	1.000 folk songs
	Norbeck's book	✓			Monophonic	2.800 Irish and Swiss songs
	FolkDB	✓			Monophonic	Unknown
	KernScores	✓			Polyphonic	108.703 files
Audio	NSynth			✓	Music audio	305.979 notes
	FMA			✓	Music audio	106.574 tracks
	Minist			✓	Music audio	50.912 notes
	GTZAN			✓	Music audio	1.000 audios of 30s
	SOL			✓	Music audio	120.000 sounds
	NUS			✓	Sing Voice	48 English songs
Multimodal	MusicNet	✓		✓	Fusion	330 recordings
	MAESTRO	✓	✓	✓	Fusion	172h of piano performances
	NES	✓		✓	Multi-instrumental	1.000 pieces
	Piano-MIDI	✓	✓	✓	Polyphonic & Performance	332 classical piano pieces
	Groove-MIDI	✓	✓	✓	Drum	1.150 files of 13.6h
	POP909	✓	✓	✓	Polyphonic	909 songs
	ASAP	✓	✓	✓	Fusion	222 scores
	Lyrics-Melody	✓			Fusion	13.937-note sequences
	MTM	✓		✓	Fusion	Unknown

Table 3.1: A summary of existing datasets (adapted from [122])

3.4 Evaluation

As discussed in [192], the assessment of generative systems, especially in the field of music creative modeling, has been proven particularly challenging. The most prevailing categorization of evaluation strategies is based on the perspective under which a specific problem's objective is approached and concepts such as creativity and efficiency are computationally interpreted.

On the one hand, the generally preferable assessment practice in the research area of generative modeling involves subjective methods, which mostly rely on human feedback over the quality of the produced musical content, since human is considered the ultimate judge of creative output, either as listener or even viewer. However, without a clear definition and consensus on the essence of human inventiveness and in view of the challenges arising from the proper design and conduction of experiments that can lead to valid, reliable and replicable results, subjective evaluation remains largely problematic.

On the other hand, methods for objective evaluation of generative systems are generally desirable, as they provide an interpretation of the model's performance with respect to a specific tack, which is conceptually closer to the operating framework of a machine. Nevertheless, despite the benefit of easy implementation, it is rather hard to approach music with certain rigorous metrics that usually lack of musical relevance in terms of musical rule systems or heuristics and also establish standardized definitions of improvement and quality applicable to different models and generation cases.

To sum up, there is no unified criterion for the results of music generation systems. According to the analysis in [122], a summary of the existing music evaluation methods from both objective and subjective aspects is conducted and explicitly presented in the following subsections.

3.4.1 Objective Evaluation

The term *objective evaluation* refers to the quantitative consideration of music generation models and their produced content, which is typically established on the utilization of different metrics, closely related to the implemented task. Such indices only reflect the ability of the model to process data, but cannot actually represent the generation efficiency, especially in case of music that involves a highly innovative form of artistic expression. Yang et al. [192] split the various objective evaluation methods, applied in recent studies in the research field of Music Synthesis, into the following categories:

Probabilistic measures and metrics without musical domain knowledge

Evaluation metrics established on probabilistic measures and statistical properties are widely used in various tasks included in the area of Image Processing and Computer Vision, exhibiting consistent behaviour with respect to the model performance [193]. Therefore, they are increasingly integrated into music-related tasks as well [194]. However, since this metric family is basically derived from a different domain, it does not involve music relevance, but instead focuses on forms of statistical divergence between the generated and the original samples or other similar characteristics.

One of the most frequently applied statistical indices is Likelihood, which represents the joint probability of the observed data viewed as a function of the chosen model's parameters [195]. In the context of music score inpainting via convolutional mechanisms, Huang et al. [196] developed a frame-wise evaluation algorithm based on the computation of the negative log-likelihood for each sample in an autoregressive manner. Similarly, the training objective of the recurrent model proposed by Johnson [197] for polyphonic music composition and prediction is the maximization of the log-likelihood for each note sequence, which has been proven a meaningful quantitative indicator of performance. However, Theis et al. [193] state that good performance with respect to particular criterion is not necessarily observed in the context of a different standard and report examples of bad samples with significantly high likelihoods.

Metrics	**Definition**
Loss (L)	Loss represents the cost associated to the model performance with respect to a specific task. Computationally, it provides a quantitative estimation of the inconsistency degree between the predicted results and the ground-truth and therefore is typically employed as the optimization objective. However, loss reduction indicates that the model can understand the problem numerically, but doesn't necessarily imply improvement of the generated musical quality, while on the other hand a model with non-converging loss cannot produce particularly fulfilling music pieces.
Perplexity (PPL)	Perplexity measures the predictive ability of a probability model. It is computed as an exponentiation of the respective distribution entropy. Low perplexity on the test set indicates that the model is suitable for unknown data, that is, the model can generalize well in terms of producing novel musical content.
BLEU score	The BLEU score is derived from the field of Natural Language Processing and is used to measure the similarity between the validation set and the generated samples.
Inception Score (IS)	IS is an algorithm used to assess the quality of produced samples. In particular, generated examples are fed to a pretrained Inception classifier and the respective score is computed as the mean KL divergence between the conditional output class probabilities and the marginal distribution of the same. IS penalizes models whose examples aren't easily classified into a single class, as well as models whose examples collectively belong to only a few of the possible classes.
Frechet Inception Distance (FID)	FID is an evaluation metric mostly applied in GAN systems. It is established on the 2-Wasserstein (or Frechet) distance between multivariate Gaussian distributions, fit to features extracted from a pretrained Inception classifier. This metric is correlated to perceptual quality and diversity on synthetic distributions.
Number of Statistically Different Bins (NDB)	NDB quantifies the diversity among the generated samples. In particular, the training examples are clustered into 50 Voronoi cells by k-means algorithm and the generated ones are assigned to the nearest cell. The respective index represents the number of cells where the number of included training samples is significantly different from the number of assigned generated examples by a two-sample Binomial test.

Table 3.2: Objective evaluation metrics without musical domain knowledge (partially adapted from [122])

Other more efficient quantitative indices, mostly derived from the field of mathematical optimization, are presented in Table 3.2 along with a brief description. All these metrics are

widely used in the area of music generation with several applications in multiple formats [198, 199] and they have actually become the de-facto standard for measuring the performance of such systems [200].

There is also a variety of task-specialized metrics, including reconstruction accuracy in VAE models [201], chord prediction accuracy [176], style likelihood [151] and style classification accuracy [202]. In the field of performance modeling the objective evaluation mainly involves the Mean Square Error (MSE) and the correlation between the characteristics of human and machine performance accordingly. Furthermore, in the context of drum generation, Gillick et al. [112] proposed, among other metrics, the Mean Absolute (MAE) and the Mean Squared Error of the onset, as well as the Kullback-Leibler divergence (KL) between the distributions of the generated onset and drum velocities.

Nevertheless, it should be mentioned that the ultimate goal of generative systems is to automatically create novel musical content, not to make predictions. Therefore, all the aforementioned probabilistic metrics can only be exploited as references and not as decisive measures in terms of generated musical quality.

Metrics using general musical domain knowledge

In order to address the multi-faceted nature of music generation systems and acquire means for their assessment based on human perception, a variety of musically-oriented metrics have been proposed. This quantitative indicator family integrates musical domain knowledge and enables detailed evaluation with respect to statistical measures of specific musical qualities, typically in the form of comparison between the descriptive statistics of the authentic musical content and the ones corresponding to the artificially produced. However, researchers can develop different musically motivated metrics according to the implemented generation tasks, providing at each case a well formulated definition that emphasizes on the respective association between the metric value and the musical characteristic that represents.

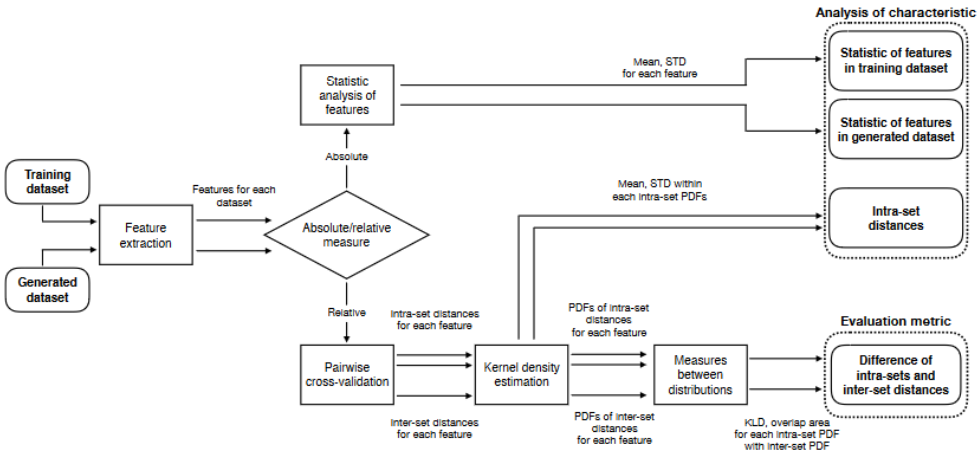


Figure 3.4.1: General workflow of musical evaluation strategy [192]

Musical metrics are widely applied in the field of score generation, since musical scores can explicitly model various features of the included notes. In this context, Yang et al. [192] developed an objective evaluation strategy based on a set of musically informed metrics and

features that has been utilized in a wide variety of symbolic music generation models [143]. The workflow of their proposed method is graphically illustrated in the diagram of Figure 3.4.1. As can be seen, this process involves *absolute metrics*, aiming to provide insights into properties and characteristics of the collected data, as well as *relative metrics* used for the comparison among different groups of samples. In particular, the input of the evaluation system consists of a training and a generated dataset, with the first representing the target space. Custom-designed features from the musical knowledge domain (mostly pitch- and rhythm-based) are extracted from both datasets and utilized for the computation of absolute and relative measurements, resulting in a group of inter-set and intra-set distances, along with other similarity measures between distributions, such as Kullback-Leibler Divergence (KLD) and Overlapping Area (OA). The evaluation framework has been released as an open-source toolbox, including the demonstrated evaluation and analysis methods along with visualization tools.

As regards other musical metrics mainly applied in an autonomous evaluation fashion, Ji et al. [122] distinguish 4 major categories: *pitch-related*, *rhythm-related*, *chord/harmony-related* and *style transfer-related*. Some of the most prevailing and commonly used examples for each class are briefly presented in Table 3.3, along with a small definition.

Types	**Metrics**	
	Name	**Definition**
Pith-related	Used Pitch Classes	Number of used pitch classes per bar (from 0 to 12).
	Tone Span	Number of half-tone steps between the lowest and the highest tone in a sample.
	Consecutive Pitch Repetitions	For a specified length l, CPR measures the frequency of occurrences of l consecutive pitch repetitions.
	Pitch Variations	PV measures how many distinct pitches are played within a sequence.
Rhythm-related	Qualified Rhythm	QR measures the frequency of note durations within valid beat ratios.
	Rhythm Variations	RV measures how many distinct note durations are contained within a sequence.
	Off-beat Recovery	Given an offset d, OR measures how frequently the model can recover back onto the beat after being forced to be off for d timesteps.
Chord/Harmony-related	Chord Tonal Distance	CTD is the average value of tonal distance computed for every pair of adjacent chords in a given sequence.
	Tonal Distance	Harmonicity between a pair of tracks.
	Chord Coverage	The number of chord labels with non-zero counts in the sequence histogram.
Style Transfer	Style Fit	Cosine similarity (cs) between output and reference style profiles.
	Content preservation	Correlation between chroma representations of source and generated segments.

Table 3.3: Objective evaluation metrics with musical domain knowledge (adapted from [122])

The use cases of the aforementioned metrics that describe 4 fundamental attributes of a musical composition vary significantly, depending on the implemented task and the ontology of the model itself. For instance, Chuan et al. [203] focus on tonal characteristics, such as the

pitch tension levels and the frequencies of melodic intervals, during the evaluation of their proposed predictive deep network that models polyphonic music under the combination of both CNN and LSTM modules. In the context of singing voice synthesis, Sturm et al. [194] conduct a statistical analysis of their developed deep autoregressive mechanism on different application scenarios, emphasizing on musical features related to pitch, timing and timbre. Similarly, Dong et al. [2] assess their GAN-based generative framework for polyphonic music of multiple instruments in terms of tonal characteristics, rhythmic patterns, as well as inter-track harmonic distances observed in produced samples.

On the hand, Sabathé et al. [204] introduce a novel objective evaluation metric for their VAE model, which is computed as the Mahalanobis distance between signature vectors composed of high-level symbolic music descriptors of the generated and real musical pieces accordingly. The aforementioned signature vectors are explicitly presented in Table 3.4 down below.

Signature vectors	**Definition**
Number of notes	Number of notes in the piece divided by the length of the piece.
Occupation rate	The ratio between the number of non-null values in the pianoroll representation and the length of the piece.
Polyphonic rate	The number of time steps where two or more notes were played simultaneously, divided by the total number of notes in the piece.
Pitch range descriptors	The maximum, minimum, mean and standard deviation of the non-null pitches in the piece. All values were divided by 127 in order to force these descriptors to be bounded between 0 and 1.
Pitch interval range	An interval is a difference in pitch between two consecutive notes. All intervals were scaled between 0 and 1 (i.e.,divided by 127) and the maximum, minimum, mean and standard deviation were computed.
Duration range	The duration is the number of time steps during which a note is held. As before, the maximum, minimum, mean and standard deviation of all durations in the piece were computed (no scaling was performed).

Table 3.4: Signature vectors (adapted from [122])

Lastly, in the field of music performance the majority of objective evaluation metrics focus on the interpretation characteristics of the performed musical pieces, such as velocity and timing. Examples of such quantitative measures are Mean Velocity (MV), Variation of Velocity (VV), Mean Duration (MD) and Variation of Duration (VD) [205]. However, since performance is a form of live expression of the musical ideas captured within a musical score, metrics applied in score generation can be also employed for evaluation of performance.

Task/model specific metrics

As the various approaches in the filed of Automatic Music Synthesis differ to a large extent on multiple aspects of the generation mechanism, a group of evaluation metrics are particularly designed for specific models or implemented tasks. These methods are based on different theories or algorithms in order to assess musical properties and according to [122] can be categorized as follows:

Structure

Wang et al. [206] developed the Variable Markov Oracle or VMO for short, a novel method for guided music synthesis and improvisation based on the detection of inherent data clusters in an audio signal along with their respective sequential time relation. As regards the evaluation part, this method is mainly applied for visualization of the identified repeated patterns in the examined music samples. At its core, VMO is structured upon a combination of a suffix tree algorithm called Factor Oracle (FO) [207], which is used for retrieval of repeated sub-strings in a symbolic sequence, and its continuous extension named Audio Oracle (AO) [208], which introduces a threshold θ representing the degree of similarity between features in the continuous time series domain. Therefore, the utilization of VMO requires the conversion of music signals from time-domain waveforms to appropriate representations, such as chromagrams. Under different values of θ, the algorithm constructs diverse symbol sequences and suffix structures from the input signals in terms of containing variable amount of original information patters. The optimal threshold is indicated by the Information Rate (IR), which captures the self-similarity, an almost integrant musical property especially in pop songs where rhythmic patterns and melodies are often repeated on a short time scale. Intuitively, higher IRs occur when repetition and variation are in balance.

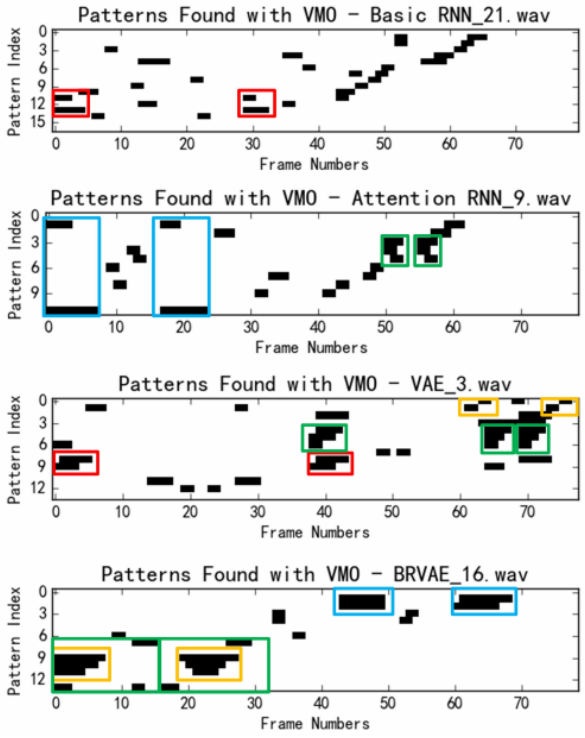


Figure 3.4.2: Patterns discovered in each sample by VMO (colored boxes) [206]

Chen et al. [209] utilized VMO in order to investigate the self-similarity in music structure by comparing the IRs of samples generated by different models. Since the graphical illustrations of the examined musical sequences in pianoroll-like format allow for easier visual inspection of repeated patterns, they employed the corresponding plots for the detection of such structural motifs, as demonstrated in Figure 3.4.2.

Originality

In an attempt to computationally quantify the variation degree of the generated music from the original corpus and examine the creativity of their proposed model, Hadjeres et al. [130] applied a novel plagiarism analysis method, based on the creation of histograms representing the length of the longest note subsequence in produced samples that can be detected identically in the training dataset. If the corresponding histogram peak is short, then the generated music is considered innovative, since it doesn't incorporate directly transferred segments of significant length. Chu et al. [134] evaluated the creativity of their proposed generative framework, using the same methodology as [130]. They also recorded additional information, such as the number of repeats for each generated melody segment, which is considered a supplementary index of the model's ability to generate pieces of varied melodic properties. Hakimi et al. [210] focused on jazz solos and assessed their originality with respect to a set of source samples, by measuring the proportion of n-grams that appear in both generated and ground-truth solos accordingly.

Style

In the field of Music Style Transfer, the most prevailing evaluation practices are typically established on the training of specific style classifiers responsible for judging whether the generated music style meets the expected profile. In other words, high performance of the classifier implies that the generated pieces exhibit the desired style characteristics. A case in point is the Minimum Distance Classifier (MDC) employed by Jin et al. [211] in order to evaluate the generated music in terms of classical style. Following similar concept, Brunner et al. [202] built a binary classifier that outputs a probability distribution over 2 style domains ranging across different genres. In case of singing style transfer, the most commonly applied evaluation approaches involve specialized metrics, such as Log Spectral Distance (LSD) representing phoneme clarity, Singer Identity (SI), for which a classifier has to be trained to model the probabilistic association between musical segments and target singers and Raw Chroma Accuracy (RAC), typically used for melody transfer assessment [212].

3.4.2 Subjective Evaluation

As extensively discussed in the previous section, a wide variety of objective metrics have been proposed in an attempt to provide a consistent quantitative interpretation of the produced musical content and a comprehensive measure of the respective model performance. Standard probabilistic and statistical indicators are generalizable and applicable to significantly varying approaches but lack of music relevance, while the variability and diversity of metrics that take into account musical domain knowledge leads to comparability issues and even biased evaluation. However, despite the advantages and drawbacks related to each metric family, in all cases there is still a gap between the quantitative consideration of music quality and human judgement. Subjective evaluation aims to bridge this gap.

The term *subjective evaluation* refers to assessment practices that involve human feedback on the generated musical content. The human perception over a musical composition is mainly based on the unconscious identification of salient themes, structural elements and features that cannot be explicitly defined under the framework of a computational machine and therefore subjective evaluation is considered the most persuasive post-hoc method. However, due

to the involved subjectivity, the results derived from this process cannot form any absolute measurement of quality, but only uncover relative differences or improvements. In the context of comparability, the main challenges arise from the lack of a standard experimental methodology, along with the absence of a general reference.

As described in [122], the most prevailing subjective evaluation approaches either follow the concept of a listening test or include expert analysis based on compositional theory. Both categories will be thoroughly presented in the following subsections.

Listening Test

Listening test is undoubtedly the most commonly employed evaluation method in the research field of music generation, as it provides the ability for comprehensive assessment or even comparison among varying models from an auditory perspective, regardless of the music generation level. It can be successfully applied in both score and audio generation tasks. The main difference is that the first case requires some additional preprocessing steps in order for performance characteristics to be rendered and the corresponding audio files to be synthesized. According to [213], a properly designed listening test should meet the following requirements:

- Sufficient number of listening subjects with adequate diversity in terms of demographics in order to offer statistically significant results.
- Uniform distribution of subjects' music knowledge level, including both music amateurs who lack relative background and experts in the field of music composition.
- Controlled environment with specific acoustic characteristics and equipment for the conduction of the experiments.
- Identical instructions and stimuli given to every subject involved in the procedure.

As can be affirmed, the design of a listening experiment that can lead to valid and reliable results uncovers a lot of challenges. Controlling all the relevant variables ranging the selection of samples, the listening environment, the recruitment of qualified participants to the formulation of the examined questions, has been proven particularly hard. As stated in [192], the majority of contemporary subjective studies address different evaluation criteria and follow diverse methodologies regarding the structure of the questionnaire and the nature of the experiment, reporting in this way varying results that cannot be explicitly compared or represent a scientific benchmark.

The simplest form of listening experiment is the Turing test, originally introduced as concept by Alan Turing in 1950 [214]. Turing's "imitation game" investigates the ability of a computational machine to exhibit intelligent behaviour close to the human level. In the context of Automatic Music Synthesis, the Turing test targets the compositional origin of the examined musical pieces. More specifically, the subjects judge whether the music samples are generated by computer or created by human. This strategy provides a qualitative measure of the model's generation efficiency in terms of specific musical properties, such as musical naturalness and therefore has been applied in several studies in the field [196, 130, 215].

Haque et al. [216] conducted a side-by-side evaluation experiment for acoustic comparison be-

tween auditory samples produced by their sequence-to-sequence model and the corresponding ground-truth, based on a quantitative approach of binary selection. In particular, the human evaluators were asked to provide for each listening pair a score ranging from `-1` (generated audio is worse than the original) to `+1` (generated audio is better that ground-truth). Bretan et. al [123] evaluated subjectively their proposed generative framework using a forced-choice ranking method. In particular, each test case consists of 4 8-bar sequences produced by 4 different models with a shared 4-measure seed. The participants of the study rank the candidates in terms of *transition naturalness* and *style consistency* between the first and second 4 bars, *naturalness* and *likeability* of the generated segments (last 4 measures) and the *overall likeability*.

Another category of listening tests focuses on the subjective assessment of a model's ability to create music with a specific target style. For instance, Mao et al. [131] performed a subjective style analysis established as a classification task among 3 different music genres. More specifically, the participants were asked to categorize music samples generated by DeepJ as baroque, classical or romantic. Similarly, Zhao et al. [217] study the correlation of music generated by their proposed model with particular emotions by conducting a listening experiment in which subjects identify the emotion class of each music sample.

There is also a group of listening tests that require musical knowledge of advanced level and relevant background for the evaluation of the produced music. In this case, the scoring criteria are not subjective questions, but professional music evaluation metrics and therefore only music experts and experienced composers are recruited to participate. For instance, Wei et al. [218] conducted a listening experiment on professional musicians and drum performers in order to collect feedback in the from of detailed comments over the structural compatibility between the generated drum patterns and the corresponding melodic tracks, as well as the stability and variability of the generation result.

Visual Analysis

In the context of visual analysis, no auditory perception of the produced results is considered. Human raters evaluate the quality of generated music subjectively only through visual inspection of proper music representation formats, including music score, pianoroll, waveform, spectrogram, etc. According to [122], visual analysis methods can be categorized as follows:

- **Score Analysis**: As the name suggests, score analysis is typically based on the musical information derived from a music score. Depending on the implemented generation task, it emphasizes on different characteristics of interest, such as pitch changes, rhythmic patterns, structural motifs, transition between bars, etc. The evaluation criteria mainly rely on music theory principles and therefore score analysis is usually conducted by experts in the field. However, different judges may express different opinions on the same score, introducing a notion of subjectivity in the process. To this end, score analysis is regarded as a practice for subjective evaluation. Dong et al. [2] perform a qualitative analysis of the produced pianorolls in terms of the overall musicality, as well as the individual features of each involved instrument. Pati et al. [161] follow similar methodology in order to evaluate and compare scores inpainted by different models.
- **Waveform/Spectrogram Analysis**: The subjective evaluation of auditory samples

in the context of visual analysis is typically based on the elaborate inspection of the corresponding signal waveforms[4] or other representations in the time-frequency domain, such as spectrograms[5] and rainbowgrams[6]. Engel et al. [183] employ the latter format in order to compare audio samples produced by interpolation in the latent space with the originals. Also common is the utilization of mel-spectrograms[7] and F_0 contour[8] maps, as implemented in [219] and [220] respectively.

[4] https://en.wikipedia.org/wiki/Waveform

[5] Spectrogram is a visual representation of a signal's spectrum across time.

[6] Rainbowgrams can be regarded as CQT spectrograms with magnitude indicated by line intensity and frequency by color.

[7] Mel-spectrograms are spectrograms where the frequencies are converted in the psychophysical mel scale.

[8] F_0 denotes the fundamental frequency at which vocal chords vibrate in music sounds. It is perceived by the ear as pitch. An F_0 contour represents the F_0 oscillations over time in the course of a utterance.

Chapter 4

Baseline Project: MuseGAN

This chapter aims at providing a complete overview of the baseline project on which our proposed framework for the problem of Automatic Music Synthesis is established. In particular, section 4.1 introduces the main characteristics of the MuseGAN project, as well as the various challenges that it attempts to tackle. Section 4.2 focuses on the architecture of the system and the structural attributes of the integrated temporal and multi-track modules, while section 4.3 includes a detailed analysis of the utilized training dataset. Lastly, section 4.4 presents the employed evaluation methods as well as the results produced by the conducted experiments.

4.1 Overview & Challenges

MuseGAN, which is the abbreviation of **Mu**lti-track **se**quential **G**enerative **A**dversarial **N**etwork, constitutes, as the name suggests, a novel framework for symbolic multi-track music generation, based on the mechanism of Generative Adversarial Networks, which have been thoroughly presented in section 2.3 of chapter 2. This project[1] was initially introduced by Dong et al. [2] at the *Association for the Advancement of Artificial Intelligence* (AAAI) Conference in 2018 and has laid the foundation for various generative systems that were developed within the research area of music synthesis. Therefore, it can be considered as a landmark among the state-of-the-art approaches to the examined research problem.

More specifically, MuseGAN is a GAN-based generative model, able to automatically produce polyphonic musical sequences for multiple tracks and particularly for Piano, Guitar, Bass, Strings and Drums, as shown in Figure 4.1.1. Under this framework, both the Generator and the Discriminator system components are implemented as deep Convolutional Neural Networks. An abstract and simplified diagram of this system configuration is graphically illustrated in Figure 4.1.2. As it might be seen, the Generator network receives a random noise vector that follows the Gaussian distribution as input and by performing successive upsampling convolutional operations, produces fake samples in the target space. Conversely, the Discriminator network acts in reverse convolutional mode in order to evaluate as real or fake data instances derived from both distributions. According to the detailed analysis in section 2.2.3 of chapter 2, CNNs are designed to process data with grid-like topology or structured information in the form of generalized arrays and hence the musical samples have to be represented in an image-like symbolic format.

Figure 4.1.1: Musical tracks in MuseGAN [13]

The concept of the MuseGAN mechanism is derived from the research fields of Computer Vision and Image Processing. However, the task of composing realistic and aesthetically harmonic musical pieces in an automated manner can be considered particularly challenging, due to the inherent difficulty of modeling music under the framework of neural networks, in contrast to other modalities, such as images, videos and text, which are characterized by a more specified structure.

[1] The entire implementation code, the utilized dataset, as well as some rendered audio samples are available at MuseGAN's website

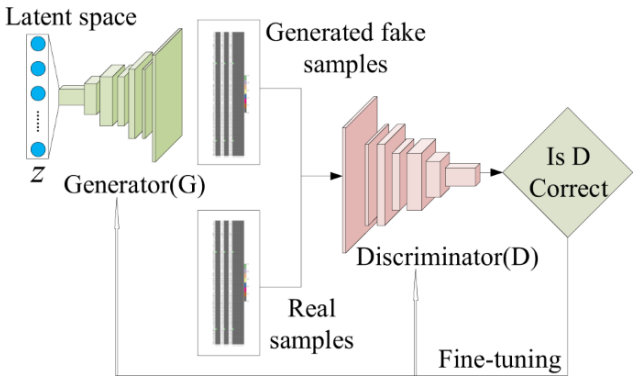


Figure 4.1.2: GAN implemented with CNNs [221]

First and foremost, this is owed to the intrinsically hierarchical arrangement of a musical piece. As demonstrated in Figure 4.1.3a, a song is abstractly composed of higher-level building blocks, called paragraphs, which can be further subdivided into musical phrases. A phrase in music is defined as a substantial concrete musical thought that has a complete musical sense of its own and therefore is considered as one of the fundamental elements in the structure of a musical composition. Each phrase consists of smaller recurrent patterns, termed bars, which contain beats, formulated by a definite timestep number. As Herremans and Chew [3] report, the human brain focuses on such structural motifs, related to coherence, rhythm, tension and the emotion flow, while listening to music and thus the incorporation of a mechanism capable of capturing the aforementioned characteristics is critical in the context of Automatic Music Synthesis. However, it can be easily affirmed that the whole hierarchy of a musical piece is structured upon temporal units, as the various objects of musical perception are presented to the listener progressively in time. To this end, MuseGAN system includes a temporal model in order to generate samples, composed of few bars, that are associated in a coherent manner.

Secondly, a musical piece is typically composed of multiple varying tracks. For instance, a modern orchestra combines instruments of different families, including bowed strings, brass, woodwinds and percussion, while the most common configuration in a rock band includes two guitars, a lead and a rhythm one, a bass, a drum set and possibly lead vocals. Each individual track in an instrumental ensemble disposes its own musical properties and dynamics. However, all the different track components collectively unfold over time in an interdependent manner, as illustrated in Figure 4.1.3b. Various composition disciplines have emerged over the years in an attempt to model the interaction among different instruments. Such approaches are strongly influenced by the corresponding music genre or the historical period they are related and they formed the foundation of rule-based methods in the context of Music Synthesis with the use of Artificial Neural Networks and Machine Learning frameworks. However, MuseGAN incorporates a more abstract and creative modeling approach to the concept of multi-track interdependence. In particular, it employs three different GAN-based models, whose mechanism is established on the human perception over the creation of musical pieces.

Lastly, notes in a polyphonic musical piece are typically presented into grouping formulations, such as *chords*, i.e. harmonic sets of multiple pitches/frequencies that are played

simultaneously, *arpeggios*, which constitute a special type of "broken" chord where the tonal components are heard in a sequential form of ascending or descending order, or other *melodic motifs* and *harmonic patterns*, as visually demonstrated in Figure 4.1.3c. All these musical texture attributes, which inherently incorporate a notion of complexity, cannot be easily captured by a computational machine system and therefore a suitable combination of data representation and processing is required in order to effectively model the structural features of a polyphonic composition. Former approaches, especially in the field of monophonic music generation, which by definition consists of a single unaccompanied melodic line and hence includes much simpler structural formulations, usually employ a chronological ordering of the various note events. However, as a matter of course, such kind of implementation cannot be generalized in tasks of higher complexity, including polyphonic music generation. To this end, under the framework of MuseGAN project, the musical samples are represented in an image-like symbolic format with the bar being considered as the basic compositional unit, so that composite patterns and grouping structures are perceivable as a whole. On account of this fact, both the generative and the discriminative sub-models are implemented as Convolutional Neural Networks, which, as thoroughly explained in section 2.2.3 of chapter 2, are specialized at detecting local, translation-covariant features through successive convolutional operations.

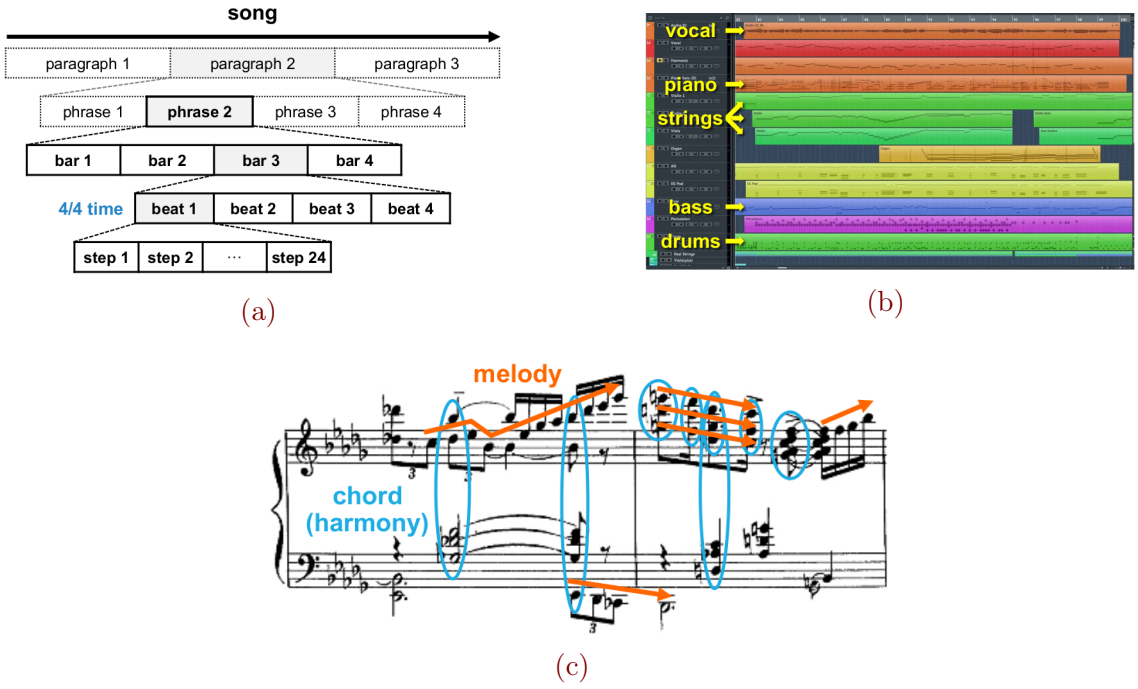


Figure 4.1.3: Challenges of Automatic Music Generation [13]

4.2 Architecture

4.2.1 Generative Adversarial Networks

As mentioned before, the core architecture of the MuseGAN system is established on the framework of Generative Adversarial Networks. According to the respective analysis in section 2.3 of chapter 2, a GAN model is typically composed of the following two individual modules:

- **Generator:** The Generator network G creates novel data instances, by mapping a random noise vector $\mathbf{z}$ sampled from a prior distribution $p_{\mathbf{z}}$ to the target data space. Thus, $G = G(\mathbf{z}; \theta_g)$ can be considered as a differentiable function computationally implemented by a an artificial neural network with parameters θ_g, which transforms the input distribution $p_{\mathbf{z}}$ to an output distribution p_g.
- **Discriminator:** The Discriminator network D evaluates the input data instances $\mathbf{x}$ in terms of authenticity, by predicting the label of their respective origin class. Therefore, $D = D(\mathbf{x}; \theta_d)$ can also be considered as a differentiable function computationally implemented by an artificial neural network with parameters θ_d, which maps the input data $\mathbf{x}$ to a single scalar value. Essentially, this output represents the probability that $\mathbf{x}$ is derived from the real data distribution p_d rather than the model distribution p_g.

These two structural components are involved into an adversarial learning procedure, which is graphically displayed in Figure 4.2.1. More specifically, the Discriminator is trained to distinguish the ground-truth samples from the fake ones, while the Generator aims at "fooling" its opponent, by counterfeiting the real data distribution as best as possible. This training framework can be mathematically formulated by employing the concept of the minimax decision rule. The term "minimax" refers to an optimization strategy usually applied in the field of game theory for minimizing the potential loss corresponding to the worst-case scenario, i.e. the maximum loss. This cost is related to the decisions of the first player during the game, assuming that their opponent responds in an optimal manner. In this context, the GAN mechanism can be intuitively modeled as a two-player turn-based game, where the alternating actions of the two opponents, the Generator and the Discriminator, involve the update of their respective weight parameters. Following the aforementioned notation, the adversarial game can be described by the minimax value function $V(G, D)$:

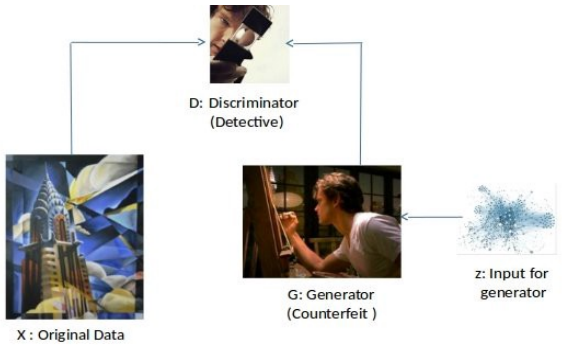


Figure 4.2.1: Illustration of GAN mechanism [222]

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_d}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{4.2.1}$$

As it can be easily seen, the first term of formula 4.2.1 represents the log probability of

predicting that the real samples are actually genuine, while the second one corresponds to the log probability of classifying the fake samples, produced by G, as unauthentic. Thus, the Discriminator aims at maximizing both the aforementioned quantities, in order to learn to assign correct labels to both kinds of training examples and effectively distinguish them. On the other hand, the objective of the Generator is to minimize the second term, so that D cannot identify the counterfeit data instances successfully.

As Gulrajani et al. report in [20], if the Discriminator is trained before each parameter update of the Generator, then the minimization of the value function 4.2.1 is equivalent to the minimization of the Jensen-Shannon divergence between the distributions p_d and p_g, which are defined up above. However, according to [21], this method in practice may not provide sufficient gradients for the learning procedure of G, resulting in vanishing gradients and mode collapse phenomena. This is owed to the fact that the quantity $\log\left(1 - D(G(\mathbf{z}))\right)$ saturates, since at the early stages of training the Generator produces fake data which are clearly diverging from the ground-truth distribution and hence are easily distinguishable by the Discriminator. Therefore, Goodfellow et al. [21] advocate that, in order to to circumvent this difficulty, the Generator should be instead trained to maximize the term $\mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[\log\left(D(\tilde{\mathbf{x}})\right)]$, where $\tilde{\mathbf{x}} = G(\mathbf{z})$ implicitly defines the model distribution p_g. This objective results in the same fixed point concerning the dynamics of G and D, but provides much stronger gradients early in learning.

However, even this modified loss function can misbehave in the presence of a good Discriminator [223]. To this end, Arjovsky et al. [224] introduce the WGAN model, which employs the Wasserstein distance (alternatively termed as the Earth Movers distance), in order to stabilize the training procedure and provide meaningful learning curves, particularly useful for debugging and hyperparameter searching. Gulrajani et al. [20] improve this framework, by enforcing a Lipschitz constraint in the form of an additional gradient penalty term at the minimax objective function, which is consequently transformed as follows:

$$\min_G \max_D V^*(G, D) = \mathbb{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}}[D(G(\mathbf{z}))] + \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \quad (4.2.2)$$

where $p_{\hat{\mathbf{x}}}$ is implicitly defined by uniform sampling along straight lines between pairs of points derived from the data distribution p_d and the generator distribution p_g accordingly. This modification is found to ensure faster convergence to better optima and also require less hyperparameter tuning. On account of this, the training process of the MuseGAN system is based on the aforementioned formula 4.2.2.

4.2.2 Modeling Multitrack Interdependency

According to human experience, two prevailing approaches concerning the composition of musical pieces can be distinguished:

- **Jamming mode:** The term "jam" refers to a relatively informal musical event, process or activity, where a group of musicians, typically including various instrumentalists, improvise music without extensive preparation or predefined arrangements. In this way, novel musical content can be created in a cooperative manner.

- **Composer mode:** This compositional technique involves the presence of a principal coordinator, referred to as *composer*, who arranges the various instruments based on harmonic and orchestration principles. In this way, structured musical pieces can be produced, as the individual musical parts in the overall composition are designed and organized in a consonant and coherent manner.

Based on the aforementioned music creation modes, Dong et al. [2] propose three different models, which are established on the GAN framework in order to capture multi-track interdependency and will be thoroughly examined in the following sections of this analysis.

Jamming Model

As illustrated in Figure 4.2.2, the Jamming Model comprises multiple GAN modules that operate independently in order to generate multi-track music. In particular, each individual Generator G_i produces music samples which correspond to a specific track included in the overall composition from a private input random vector $\mathbf{z_i}$ and receives feedback in the form of backpropagated supervisory signals from its respective Discriminator D_i. The possible values of the index i range from 1 to M, where M denotes the number of tracks. Therefore, a musical piece consisting of M tracks requires, under the framework of the Jamming Model, M Generators and M Discriminators accordingly.

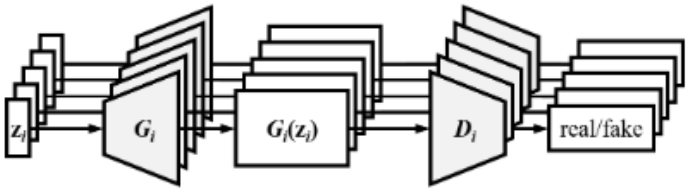


Figure 4.2.2: Jamming Model [2]

Composer Model

As demonstrated in Figure 4.2.3, the Composer Model consists of a single pair of adversarial networks, regardless of the value of M, which denotes the number of the employed music tracks. More specifically, the Generator G receives a shared random vector $\mathbf{z}$ as input, which can be considered as the intention of the composer and produces a multi-channel music sample, where each channel represents one of the included tracks. The single Discriminator D evaluates the musical segments in terms of authenticity, by examining the corresponding M tracks collectively.

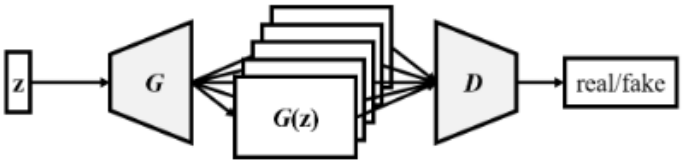


Figure 4.2.3: Composer Model [2]

Hybrid Model

As the name suggests, the Hybrid Model constitutes a combined implementation of both the aforementioned systems, that merges the notion of unconfined improvisation in the jamming context along with the musical discipline imposed by the composer's arrangement. As graphically displayed in Figure 4.2.4, the hybrid architecture comprises multiple Generator modules G_i, each one corresponding to a specific track included in the overall composition and thus to the actions of one musician/instrumentalist, but only one Discriminator network D. The input of each Generator consists of a private *intra-track* random vector $\mathbf{z_i}$, which can be considered track-specialized, as well as a shared *inter-track* random vector $\mathbf{z}$, which coordinates to some extent the generation process performed by the various musicians G_i. The Discriminator D evaluates the musical segments in terms of authenticity, by examining the corresponding M tracks collectively. Therefore, under the framework of the Hybrid Model, a musical piece of M tracks requires M Generators and one Discriminator.

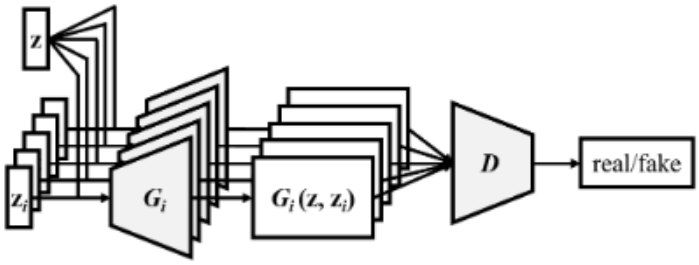


Figure 4.2.4: Hybrid Model [2]

It can be easily affirmed that the core mechanism of the Hybrid Model enables flexible variations of the track-specific generation procedure, either in terms of network structure (e.g. different number of layers, filter size, etc. among the various Generators) or regarding the form of the private input $\mathbf{z_i}$, retaining at the same time the desired overall inter-track interdependency.

4.2.3 Modeling Temporal Structure

All the GAN-based multi-track models, which have been presented in the previous section, are capable of generating polyphonic musical pieces for multiple tracks with duration up to one bar. The "bar" or else "measure" is a segment of time corresponding to a specific number of beats, in which every beat is represented by a particular note value. As mentioned before, this small musical container constitutes the basic building block of a musical composition, since the boundaries between consecutive bars are usually the spots where harmonic changes occur. To this end, in order to be able to generate music samples with longer duration, such as a musical phrase and ensure at the same time coherence and consistency among the produced bars, a mechanism that captures and handles temporal dependencies is crucial. Therefore, Dong et al. [2] design two distinct temporal models corresponding to different use cases, which will be thoroughly examined further down.

Generation from Scratch

Based on the concept of TGAN, a temporal GAN-based model proposed by Saito et al. [14] in 2017 for learning semantic representations of unlabeled videos and generating image sequences, this method aims at producing fixed-length musical phrases, by integrating bar progression at the Generator's workflow in the form of an augmented dimension. More specifically, in this case the Generator module consists of two sub-networks, the *temporal structure generator* G_{temp} and the *bar generator* G_{bar}, which are graphically illustrated in the diagram of Figure 4.2.5. As can be seen, G_{temp} maps the input random vector $\mathbf{z}$ to a sequence of latent variables $\vec{\mathbf{z}} = \{\vec{\mathbf{z}}^{(t)}\}_{t=1}^{T}$, where $T > 0$ denotes the total number of bars to be generated. It can be easily affirmed that each one of these latent components corresponds to a specific bar included in the musical segment which will be eventually composed and incorporates to some extent information about temporal characteristics. The bar generator G_{bar} transforms, as the name indicates, the resulting vector $\vec{\mathbf{z}}$ into a musical phrase in a sequential manner (i.e. bar by bar). Using formal notation, the overall function of the Generator under this temporal framework can be formulated as follows:

$$G(\mathbf{z}) = \left\{ G_{bar}\left(G_{temp}(\mathbf{z})^{(t)}\right) \right\}_{t=1}^{T} \tag{4.2.3}$$

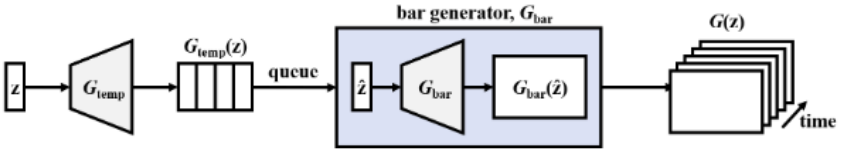


Figure 4.2.5: Generation from Scratch [2]

Track-conditional Generation

This method can be considered as an extension of the MuseGAN model to a human-AI cooperative framework, as it can be applied in music accompaniment generation or in other tasks, where the involved generative procedure is conditioned to some kind of prior information. In particular, it aims to capture the underlying temporal structure of a specific human-composed track, which is assumed to be given as input to the model in the form of a bar sequence $\vec{\mathbf{y}}$, in order to generate the remaining tracks. As demonstrated in the diagram of Figure 4.2.6, the conditional Generator G^o produces the consecutive bars of the formatted accompaniment segment in a sequential manner, by receiving two distinct inputs, the conditional track $\vec{\mathbf{y}}^{(t)}$ and a time-dependent random noise vector $\vec{\mathbf{z}}^{(t)}$, where t signifies the index of the current bar. However, since the conditional bar sequence is usually represented in a high-dimensional space, an additional encoder network E is included in the system architecture. This module maps $\vec{\mathbf{y}}^{(t)}$ to a low-dimensional embedding in the space of $\vec{\mathbf{z}}^{(t)}$, by extracting inter-track features that can be useful for the generation of the other musical parts, as suggested in former related approaches [15]. Using formal notation, the overall function of the conditional Generator can be formulated as follows:

$$G^o(\vec{\mathbf{z}}, \vec{\mathbf{y}}) = \left\{ G^o_{bar}\left(\vec{\mathbf{z}}^{(t)}, E(\vec{\mathbf{y}}^{(t)})\right) \right\}_{t=1}^{T} \tag{4.2.4}$$

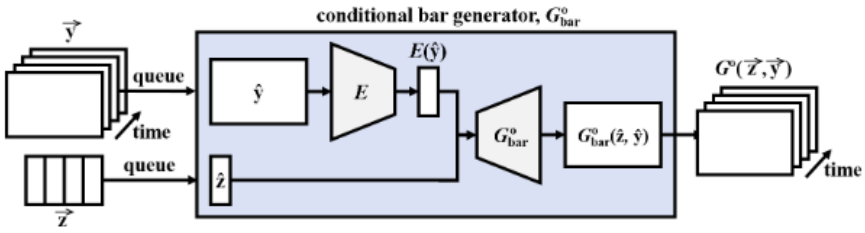


Figure 4.2.6: Track-conditional Generation [2]

4.2.4 MuseGAN

MuseGAN constitutes an integration and augmentation of the multi-track and temporal models, which have been elaborately presented at the previous sections. As illustrated in Figure 4.2.7, the input of the system consists of four different parts:

- an inter-track time-independent random vector $\mathbf{z}$
- an inter-track time-dependent random vector $\mathbf{z}_t$
- M intra-track time-independent random vectors $\mathbf{z}_i$
- M intra-track time-dependent random vectors $\mathbf{z}_{i,t}$

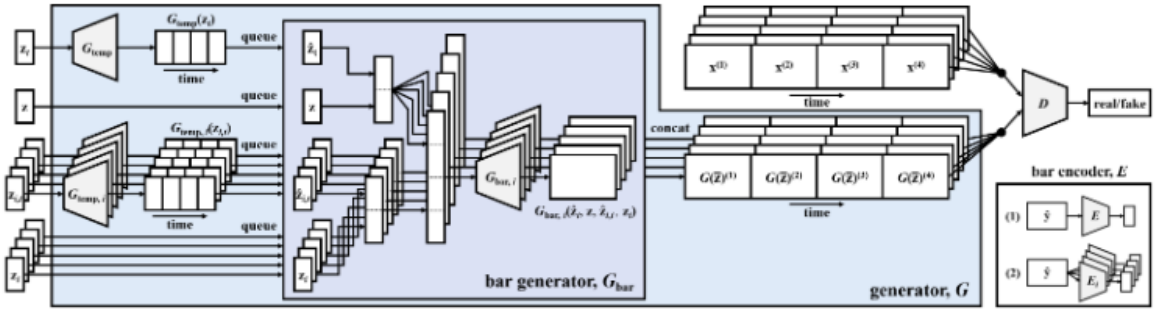


Figure 4.2.7: MuseGAN system diagram [13]

The *shared* temporal structure Generator G_{temp}, as well as the corresponding *private* temporal structure Generator $G_{temp,i}$ for each track i, where the index i ranges from 1 to M, map the time-dependent input random vectors $\mathbf{z}_t$ and $\mathbf{z}_{i,t}$ to sequences of latent variables that contain inter-track and intra-track temporal information respectively. The resulting output series are concatenated with the time-independent random vectors $\mathbf{z}$ and $\mathbf{z}_i$ and then fed into the bar Generator G_{bar}, which produces musical phrases in a sequential manner. Using formal notation, the overall generation procedure can be formulated as follows:

$$G(\bar{\mathbf{z}}) = \left\{ G_{bar,i}\left(\mathbf{z}, G_{temp}(\mathbf{z}_t)^{(t)}, \mathbf{z}_i, G_{temp,i}(\mathbf{z}_{i,t})^{(t)}\right) \right\}_{i,t=1}^{M,T} \tag{4.2.5}$$

where $\bar{\mathbf{z}} = (\mathbf{z}, \mathbf{z}_t, \mathbf{z}_i, \mathbf{z}_{i,t})$ for each $i \in \{1, \ldots, M\}$ and $t \in \{1, \ldots, T\}$. As regards the case of the track-conditional generation, the aforementioned mechanism is slightly modified by the inclusion of the additional encoder module that extracts useful inter-track features from the user-provided musical part.

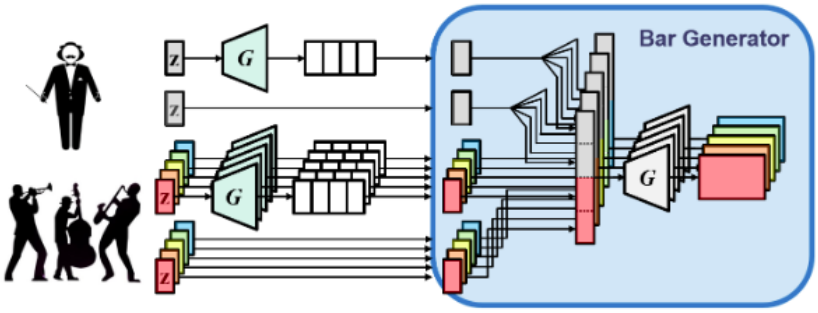


Figure 4.2.8: Bar Generator in MuseGAN [13]

As graphically displayed in Figure 4.2.8, the inter-track input components intuitively represent features that do not depend on a specific track, but instead are related to the general configuration of the musical composition. Such characteristics include, among others, the chord progression or the musical style. Therefore, the aforementioned random vectors can be considered associated to the composer generative mode. On the other hand, following an equivalent reasoning, the remaining intra-track input components represent track-dependent features, such as melody and groove, and hence are inherently related to the jamming mode.

4.2.5 Implementation Details

Figure 4.2.9 demonstrates the network architectures for the aforementioned structural components of the MuseGAN system. As it can be seen, all the involved modules are implemented as deep Convolutional Neural Networks, which have been thoroughly presented in section 2.2.3 of chapter 2. In particular, the values included at each one of the depicted tables represent orderly the following elements:

- *Convolutional Layers*: number of filters, kernel size, stride, batch normalization (BN) and activation function.
- *Fully-connected Layers*: number of hidden nodes and activation function.

Both employed Generators successively augment the dimensions of the input vector through transposed convolutional operations, which are initially applied along the time axis and afterwards along the pitch axis. On the other hand, the Discriminator module displays the opposite behaviour in terms of successively compressing the spatial dimensions of the corresponding input vector, first along the pitch axis and then along the time one, through the utilization of typical convolutional layers. Following a similar implementation, the Encoder

network mirrors to some extent the architecture of the bar Generator, in order to produce the latent embeddings.

Input: $\mathbf{z} \in \Re^{32}$					
reshaped to (1)×32 channels					
transconv	1024	2	2	BN	ReLU
transconv	K_{temp}	3	1	BN	ReLU
Output: $G_{\text{temp}}(\mathbf{z}) \in \Re^{32 \times K_{\text{temp}}}$ (K_{temp}-track latent vector)					

(a) Temporal Generator G_{temp}

Input: $\widetilde{\mathbf{x}} \in \Re^{4 \times 96 \times 84 \times 5}$ (real/fake piano-rolls of 5 tracks)				
reshaped to (4, 96, 84) × 5 channels				
conv	128	$2 \times 1 \times 1$	(1, 1, 1)	LReLU
conv	128	$3 \times 1 \times 1$	(1, 1, 1)	LReLU
conv	128	$1 \times 1 \times 12$	(1, 1, 12)	LReLU
conv	128	$1 \times 1 \times 7$	(1, 1, 7)	LReLU
conv	128	$1 \times 2 \times 1$	(1, 2, 1)	LReLU
conv	128	$1 \times 2 \times 1$	(1, 2, 1)	LReLU
conv	256	$1 \times 4 \times 1$	(1, 2, 1)	LReLU
conv	512	$1 \times 3 \times 1$	(1, 2, 1)	LReLU
fully-connected	1024			LReLU
fully-connected	1			
Output: $D(\widetilde{\mathbf{x}}) \in \Re$				

(b) Discriminator D

Input: $\mathbf{z} \in \Re^{128}$					
reshaped to (1, 1)×128 channels					
transconv	1024	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	256	2×1	(2, 1)	BN	ReLU
transconv	128	3×1	(3, 1)	BN	ReLU
transconv	64	1×7	(1, 7)	BN	ReLU
transconv	K_{bar}	1×12	(1, 12)	BN	tanh
Output: $G_{\text{bar}}(\mathbf{z}) \in \Re^{96 \times 84 \times K_{\text{bar}}}$ (K_{bar}-track piano-roll)					

(c) Bar Generator G_{bar}

Input: $\mathbf{y} \in \Re^{96 \times 84}$ (piano-rolls of the given track)					
conv	16	1×12	(1, 12)	BN	LReLU
conv	16	1×7	(1, 7)	BN	LReLU
conv	16	3×1	(3, 1)	BN	LReLU
conv	16	2×1	(2, 1)	BN	LReLU
conv	16	2×1	(2, 1)	BN	LReLU
conv	16	2×1	(2, 1)	BN	LReLU
Output: $E(\mathbf{y}) \in \Re^{16}$					

(d) Encoder E

Figure 4.2.9: Network architectures for the structural components of MuseGAN system [2]

4.3 Data

4.3.1 Data Representation

As mentioned before, CNNs are designed to process data with grid-like topology or structured information in the form of generalized fixed-size arrays. To this end, the musical samples, which are processed under the framework of the MuseGAN system are represented in the *pianoroll* format. According to the extended analysis in section 3.1.3 of chapter 3, the pianoroll format can be defined as a binary-valued scoresheet-like matrix representing the presence of notes over different timesteps. Figure 4.3.1 demonstrates a pianoroll representation of a 4-bar musical fragment. As can be seen, the horizontal axis represents time in a symbolic formulation that discards tempo information resulting in equally sized time fragments for each beat, while the vertical axis represents notes ordered from the low-pitched to the high-pitched ones. A colored pixel (pixel with value 1) indicates that a specific pitch is played at the current timestep.

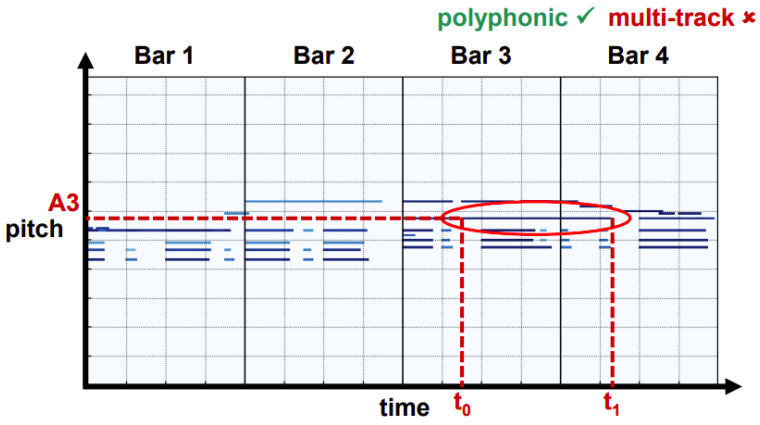


Figure 4.3.1: Pianoroll format with symbolic timing [13]

However, the aforementioned method can represent polyphonic musical pieces corresponding only to one single track. To that end, in order to tackle this limitation and model music composed of multiple tracks, Dong et al. [2] introduce the *multi-track pianoroll* representation format. As graphically displayed in Figure 4.3.2, a multi-track pianoroll is defined as a set of piano-rolls corresponding to different musical tracks.

Using formal notation, an M-track piano-roll of one bar is represented as a tensor

$$\mathbf{x} \in \{0,1\}^{R \times S \times M} \tag{4.3.1}$$

where R denotes the number of timesteps included in a bar and S symbolizes the total number of pitch candidates. Consequently, an M-track piano-roll of T bars is represented as a sequence of tensors

$$\vec{\mathbf{x}} = \{\vec{\mathbf{x}}^{(t)}\}_{t=1}^{T} \tag{4.3.2}$$

where $\vec{\mathbf{x}}^{(t)} \in \{0,1\}^{R \times S \times M}$ indicates the multi-track pianoroll of bar t.

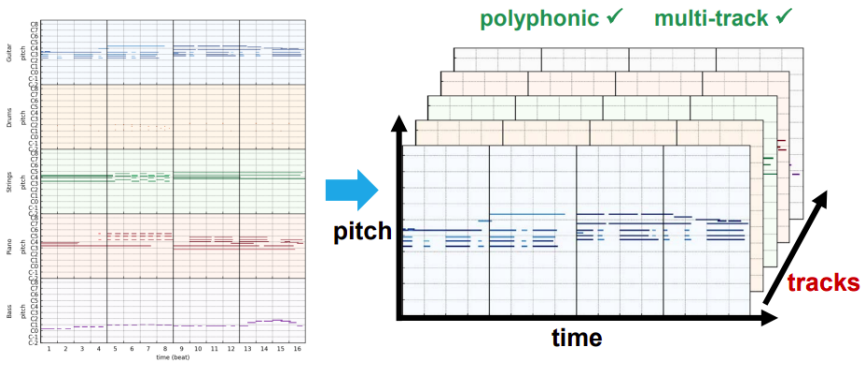


Figure 4.3.2: Multi-track pianoroll format [13]

4.3.2 Dataset

The piano-roll dataset employed under the framework of the MuseGAN system is derived from the Lakh MIDI Dataset (LMD) [16], which constitutes one of the largest symbolic music corpora, including 176.581 unique MIDI files created by Colin Raffel. As discussed in section 3.1.1 of chapter 3, this dataset incorporates unlimited, polyphonic, inconsistent expressive characteristics and encompasses various music genres, instruments and time periods. However, most of the included MIDI files are quite noisy, since they are mainly scraped from the web or user-generated. To this end, Dong et al. [2] utilize a subset of the LMD, known as *LMD-matched*, which, as the name suggests, comprises 45.129 files matched and aligned with the corresponding entries in the Million Song Dataset (MSD) [17]. The resulting set of training examples after the conversion of the aforementioned MIDI files into multi-track piano-rolls is called **Lakh Pianoroll Dataset** or **LPD** for short and can be found on the project's website.

4.3.3 Data Preprocessing

All the steps involved in the data preprocessing procedure, which is applied in order to construct the final set of training examples, are graphically illustrated in the diagram of Figure 4.3.3.

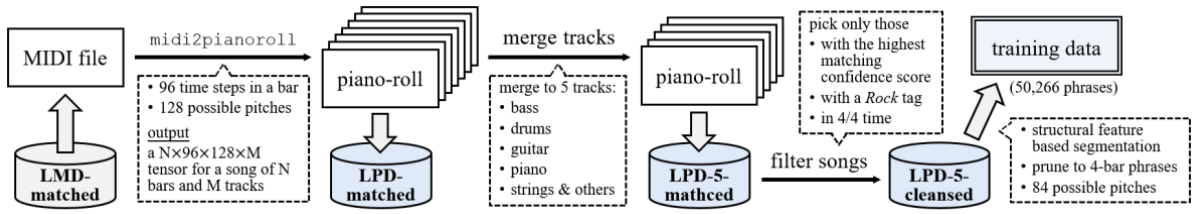


Figure 4.3.3: Illustration of the dataset preparation and data preprocessing procedure [2]

As can be seen, at first, the MIDI files contained in the matched version of the LMD are converted into multi-track piano-rolls, using the python module *pretty_midi* [225], which has been proposed by Raffel and Ellis in 2014 for creating, manipulating and analyzing such kind of music storing format. For each bar, the height of the score-like piano-roll matrix is

set to 128, covering from C-1 to G-9, while the width dimension, which as mentioned before represents the time resolution, is set to 96, in order to model common temporal patterns, such as triplets and 16th notes. During this transformation process from MIDI files to multi-track pianorolls, an extra minimal-length (i.e. of one timestep) rest is added between two consecutive notes of the same pitch, in order to be distinguished from one single long note of the same corresponding duration, while notes shorter than two timesteps are dropped. The aforementioned pause enforcing method is applied to all tracks except drums, where only the onsets are encoded.

Naturally, some of the involved tracks tend to be sparse in terms of containing only a few notes in the entire musical piece. Therefore, Dong et al. in order to tackle this inherent data imbalance issue, which impedes the learning process of the system, employ a merging technique that integrates tracks corresponding to similar instruments. In particular, each multi-track piano-roll included in the matched version of the produced LPD dataset is compressed into five distinct track categories: bass, drums, guitar, piano and strings. Instruments out of this list are considered as part of the strings except those in the Percussive, Sound Effects and Synth Effects families. After this step, the *LPD-5-matched* set of music samples is created, consisting of 30.887 5-track piano-rolls.

Subsequently, in order to ensure a degree of homogeneity and uniformity among the various musical pieces, which are included into the *LPD-5-matched* dataset, a filtering process is performed, based on the metadata provided in the LMD and MSD respectively. More specifically, only the pianorolls which are in 4/4 time, correspond to a "rock" tag and present higher confidence score in matching with any entry in the MSD, are retained. The resulting set of music samples is called *LPD-5-cleaned* and contains 21.425 multi-track pianorolls.

Finally, in order to collect musically meaningful phrases for the training of the embedded temporal model, the pianorolls included in the *LPD-5-cleansed* version are segmented according to a state-of-the-art algorithm, which is called *structural features* and has been proposed by Serra et al. in 2012 [226]. Under the framework of the MuseGAN system, a phrase is considered as a 4-bar musical segment and therefore longer segments are pruned into the proper size. Furthermore, notes below C-1 or above C-8 are discarded, since they are particularly uncommon in the majority of the examined musical compositions, resulting in 84 possible values as regards the range of the pitch axis. In this way, 50.266 musical phrases with the aforementioned dimensions are acquired as the final set of training data for the MuseGAN system.

Consequently, as graphically demonstrated in Figure 4.3.4, the size of the target output tensor, which represents the artificial piano-roll of a musical segment is

$$\boxed{4 \times 96 \times 84 \times 5}$$

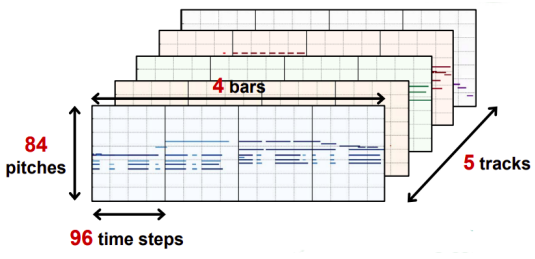


Figure 4.3.4: Data configuration [13]

4.4 Evaluation & Results

4.4.1 Objective Evaluation

As thoroughly discussed in section 3.4.1 of chapter 3, objective evaluation in the research field of Automatic Music Synthesis refers to a quantitative consideration of the examined generative systems and their produced musical pieces. This process is typically based on statistical criteria and hence employs a set of evaluation metrics in an attempt to model the generation efficiency and capture properties of the highly intricate form of musical expression. Under this framework, Dong et al. [2] propose one inter-track and four intra-track musical metrics, that can be computed for both the real and the generated samples:

- **Empty Bars (EB):** ratio of empty bars included in the examined track (in %)
- **Used Pitch Classes (UPC):** mean number of pitch classes[2] used per bar (from 0 to 12)
- **Qualified Notes (QN):** ratio of "qualified" notes[3] (in %)
- **Drum Pattern (DP):** ratio of notes in beat patterns of 4/4 rhythm[4] (in %)
- **Tonal Distance (TD):** measures the harmonicity between a pair of musical tracks[5]

		empty bars (EB; %)					used pitch classes (UPC)				qualified notes (QN; %)				DP (%)
		B	D	G	P	S	B	G	P	S	B	G	P	S	D
training data		8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
from scratch	jamming	6.59	2.33	18.3	22.6	6.10	1.53	3.69	4.13	4.09	71.5	56.6	62.2	63.1	93.2
	composer	0.01	28.9	1.34	0.02	0.01	2.51	4.20	4.89	5.19	49.5	47.4	49.9	52.5	75.3
	hybrid	2.14	29.7	11.7	17.8	6.04	2.35	4.76	5.45	5.24	44.6	43.2	45.5	52.0	71.3
	ablated	92.4	100	12.5	0.68	0.00	1.00	2.88	2.32	4.72	0.00	22.8	31.1	26.2	0.0
track-conditional	jamming	4.60	3.47	13.3	—	3.44	2.05	3.79	—	4.23	73.9	58.8	—	62.3	91.6
	composer	0.65	20.7	1.97	—	1.49	2.51	4.57	—	5.10	53.5	48.4	—	59.0	84.5
	hybrid	2.09	4.53	10.3	—	4.05	2.86	4.43	—	4.32	43.3	55.6	—	67.1	71.8

Figure 4.4.1: Intra-track evaluation [2]
(**B**: Bass, **D**: Drums, **G**: Guitar, **P**: Piano, **S**: Strings)

Figures 4.4.1 and 4.4.2 demonstrate the computational results derived from the objective evaluation process of the MuseGAN system. In particular, 20.000 bars generated by each one

[2] A pitch class can be defined as the group of all pitches that are related by octave and enharmonic equivalence. In music theory, 12 distinct pitch classes can be distinguished, formulating a circular note space called the *chromatic circle*.

[3] A *qualified* note can be considered as a note with duration greater than 3 timesteps (i.e. a 32^{th} note). The number of qualified notes included into a musical piece indicates to some extent the musical property of fragmentation, which, as the name suggests, refers to the use of fragments or the partitioning of a musical idea into segments. To this end, the QN metric demonstrates if the examined music samples are overly fragmented, with higher values denoting lower fragmentation of the produced pieces.

[4] As elaborated in the previous section, the dataset used under the framework of the MuseGAN system comprises pianorolls only in 4/4 time, which correspond to *Rock* songs. DP measures the notes presented at the downbeats of 4/4 rhythm in accordance with the employed time resolution.

[5] This evaluation metric is inspired by the work of Harte et al. [18], who proposed a novel model for the Equal Tempered Pitch Class Space, which maps 12-bin chroma vectors to the interior space of a 6-D polytope, where the vertices represent the pitch classes. In this way, close harmonic relations such as fifths and thirds appear as small Euclidean distances.

of the proposed models, which have been thoroughly presented in section 4.2, are evaluated in terms of the aforementioned musical metrics. As regards the track-conditional scenario, the generation of the four accompaniment instrumental parts is conditioned on the Piano track. Furthermore, under the temporal framework of generation from scratch, an *ablated* version of the composer model, which does not incorporate batch normalization (BN) layers, is also included in the process for further comparison. The intra-track values produced from this model variant can be considered as reference, due to its minimal learning ability.

According to the elaborate analysis concerning the concept of Generative Adversarial Networks in section 2.3 of chapter 2, the distributions of real and fake data samples and accordingly their respective statistics are forced to get as close as possible through an adversarial training procedure. Therefore, as regards the intra-track evaluation part, which is presented in the table of Figure 4.4.1, values approximating the ones included in the first row, which correspond to the metrics as measured in the training set, are considered better with respect to the efficiency of the model. Consequently, the best performance is achieved by the jamming model, possibly due to the fact that each generator module involved in the jamming structure is designed to emphasize on its own respective track and hence improving musical attributes related the intra-track objective metrics. Aside from the ablated model, the resulting **DP** values indicate that drums manage to capture underlying rhythmic patterns from the training data, despite the relatively high **EB** in the composer and the hybrid model. From **UPC** and **QN**, it can also be observed that all models tend to use more pitch classes and produce fairly less qualified notes in comparison with the set of ground-truth samples. The authors ascribe this form of noise which is introduced into the generation process to the binarization method applied in order to transform the continuous-valued output of the Generator modules to a binary-valued pianoroll.

		tonal distance (**TD**)					
		B-G	B-S	B-P	G-S	G-P	S-P
train.		1.57	1.58	1.51	1.10	1.02	1.04
train. (shuffled)		1.59	1.59	1.56	1.14	1.12	1.13
from scratch	jam.	1.56	1.60	1.54	1.05	0.99	1.05
	comp.	1.37	1.36	**1.30**	0.95	0.98	0.91
	hybrid	**1.34**	**1.35**	1.32	**0.85**	**0.85**	**0.83**
track-conditional	jam.	1.51	1.53	1.50	1.04	0.95	1.00
	comp.	1.41	**1.36**	1.40	**0.96**	1.01	**0.95**
	hybrid	**1.39**	**1.36**	**1.38**	**0.96**	**0.94**	**0.95**

Figure 4.4.2: Inter-track evaluation [2]
(**B**: Bass, **D**: Drums, **G**: Guitar, **P**: Piano, **S**: Strings)

As regards the inter-track evaluation part, which is presented in the table of Figure 4.4.2, larger **TD** values imply weaker harmonic relations between the examined pairs of tracks and hence the smaller ones correspond to the intended model behaviour. As can be seen, this is achieved by the composer as well as the hybrid model. This result suggests that the aforementioned architectures are more suitable for the task of multi-track polyphonic generation compared to the jamming model in terms of cross-track harmonic relations, since they both involve the presence of a principal coordinator, who arranges to some extent the various instruments, based on harmonic and orchestration principles.

4.4.2 Subjective Evaluation

As elaborated in section 3.4.2 of chapter 3, subjective evaluation constitutes an indispensable assessment practice in the research area of music generation systems, since it attempts to bridge the gap between the quantitative evaluation of the music quality, which emerges from the use of objective metrics, and the human judgement.

			H	R	MS	C	OR
from scratch	non-pro	jam.	2.83	3.29	2.88	2.84	2.88
		comp.	3.12	**3.36**	2.95	3.13	3.12
		hybrid	**3.15**	3.33	**3.09**	**3.30**	**3.16**
	pro	jam.	2.31	3.05	2.48	2.49	2.42
		comp.	2.66	3.13	2.68	2.63	2.73
		hybrid	**2.92**	**3.25**	**2.81**	**3.00**	**2.93**
track-conditional	non-pro	jam.	**2.89**	**3.44**	2.97	**3.01**	**3.06**
		comp.	2.70	3.29	**2.98**	2.97	2.86
		hybrid	2.78	3.34	2.93	2.98	3.01
	pro	jam.	2.44	**3.32**	2.67	2.72	2.69
		comp.	2.35	3.21	2.59	2.67	2.62
		hybrid	**2.49**	3.29	**2.71**	**2.73**	**2.70**

Figure 4.4.3: User Study [2]
(**H**: Harmonious, **R**: Rhythmic, **MS**: Musically Structured, **C**: Coherent, **OR**: Overall Rating)

In this context, Dong et al. [2] conduct a user study in the form of a listening test, which, as mentioned in section 3.4.2 of chapter 3, is considered the most fundamental, common and at the same time convincing evaluation method, providing human feedback in a comprehensive manner. The participants of the survey are 144 subjects, who have been recruited mostly from the Internet via the social circles of the authors. Through a simple questionnaire probing their musical background, the users are divided into two groups, the "pro" (44) and the "non-pro" (100). Each subject listens to 9 audio clips presented in random order, where each one of them consists of 3 four-bar phrases generated by one of the proposed models and quantized by 16^{th} notes. Then the user rates the aforementioned clips using a 5-point Likert scale (1 denotes the minimum and 5 the maximum), in terms of whether they

1) have pleasant harmony

2) have unified rhythm

3) have clear musical structure

4) are coherent

5) the overall rating

The produced results are demonstrated at the table of Figure 4.4.3. As can be seen, all participants, irrespective of their musical knowledge level or experience, show a preference for the hybrid model as regards the temporal framework of Generation from Scratch. In the case of Track-conditional Generation, the music experts favor the hybrid model, while the non-experts slant towards the jamming model. Furthermore, it can also be observed that the hybrid as well as the composer models receive higher scores for the criterion *Harmonious* under the Generation from Scratch framework in comparison with the ones associated to the jamming model. This inference is in accordance with the results of the objective evaluation,

as it indicates that the aforementioned model structures, which incorporate a coordinated track arrangement mechanism, are capable of handling multi-track interdependency in a more efficient manner.

Chapter 5

Unconditional Generation

This chapter aims at providing a complete overview of our proposed GAN-based framework for the task of Unconditional Generation. In particular, section 5.1 introduces the main characteristics of the generation problem that we attempt to tackle. Section 5.2 includes a detailed description of the system architecture, the implementation of the various structural components, as well as their respective training mechanism. Section 5.3 elaborates on the utilized form of data representation, the dataset and the required preprocessing steps. Section 5.4 focuses on the employed experimental protocol, while section 5.5 engages on a thorough analysis of the results produced using the aforementioned setup. Lastly, section 5.6 presents our user study related to this task and discusses its subjective findings.

5.1 Task Description

As thoroughly discussed in chapter 3, the problem of Automatic Music Synthesis involves a huge variety of different techniques, methods and architectures that aim to emulate the numerous variants of the human compositional practice and also model the diverse aspects and multifarious attributes that characterize musical pieces. Since it is particularly hard to fully investigate the undoubtedly vast research field of AI music, our initial approach to the aforementioned problem in the context of this thesis focuses on the task of **Unconditional Generation** of multi-track polyphonic musical samples.

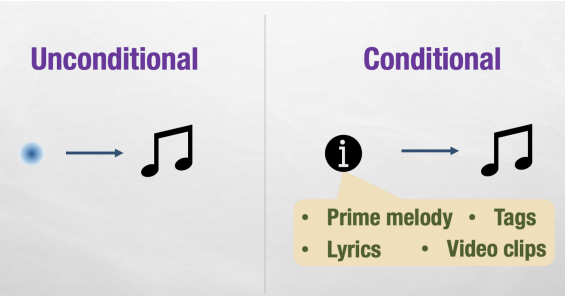


Figure 5.1.1: Unconditional vs Conditional Generation [227]

According to the specialized analysis in section 3.2.1 of chapter 3, *polyphony* is defined as a type of musical texture which consists of two or more distinct melodic lines that are combined to flow and unfold simultaneously in a coordinated manner, introducing in this way complex patterns along the time axis, as well as harmonic dependencies between rhythmically concurrent notes. As a matter of course, the automatic generation of polyphonic music is notably challenging, especially coupled with the use of multiple different musical instruments or tracks. The term "unconditional" suggests that the generation procedure does not include any prior knowledge or supplementary information from the human user, contrary to the "conditional" case, which typically relies on additional human-provided data in the form of lyrics, lead sheet, the respective chord progression or a primary melodic line for instance. As graphically demonstrated in Figure 5.1.1, the unconditional mechanism is typically based on the transformation of a random input into a meaningful form of musical expression.

5.2 Model

Our proposed model for automatically generating polyphonic musical segments of multiple tracks from scratch, i.e. without being subjected to conditional information of any kind, is grounded on the MuseGAN system, which constitutes one of the most prevailing state-of-the-art approaches to the examined research problem, as elaborately discussed in chapter 4. The specific details as well as the respective alterations we performed on the baseline project will be meticulously presented in the following subsections.

5.2.1 Architecture

Following [2], we design a framework for polyphonic music generation in symbolic format including 5 distinct tracks (*Bass, Guitar, Strings, Piano, Drums*), based on a Convolutional GAN mechanism. The core architecture of our proposed system is inspired by a later work of Dong and Yang [19], which introduces the incorporation of binary neurons into the structure of the MuseGAN model for directly generating binary-valued pianorolls and studies various binarization methods that can be employed. As graphically displayed in the diagram of Figure 5.2.1, two fundamental modules are included:

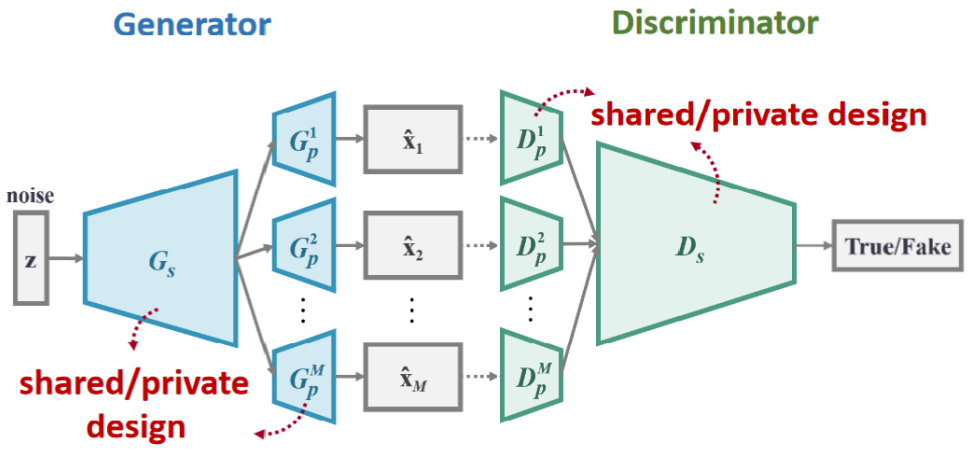


Figure 5.2.1: Architecture diagram of our proposed model

- **Generator**
  The Generator component, as thoroughly discussed in previous chapters of this thesis, produces novel data instances, by mapping an input random noise vector $\mathbf{z}$, sampled from a prior distribution, to the output target data space of musical representations, via consecutive upsampling convolutional operations. As shown in Figure 5.2.1, it consists of a *shared* network G_s, followed by M *private* subnetworks G_p^i $(i = 1, \dots M)$, each one corresponding to a specific track included in the musical composition. The *shared Generator* G_s initially produces a common high-level and more abstract form of the output musical segment, which intuitively represents a general musical idea that is jointly shared among the various tracks. On account of this fact, G_s can be considered as a composer that coordinates and arranges the various instruments based on harmonic and orchestration principles. Consequently, each *private Generator* G_p^i transforms this

abstract representation into the final piano-roll output for the corresponding track, according to its own musical properties, such as textural elements, melodic patterns, rhythmical motifs etc. Thus, these track-specialized modules can be regarded as distinct musicians improvising over the individual characteristics of each instrument in the context of jamming mode. It can be easily observed that this structure differs from our baseline project [2] regarding all the proposed multitrack interdependency models, which have been extensively presented in section 4.2.2 of chapter 4.

- **Discriminator**
  The Discriminator component evaluates the input data instances $\mathbf{x}$ in terms of authenticity, by predicting the label of their respective origin class. This process is performed via successive convolutional operations. As demonstrated in Figure 5.2.1, from a structural perspective the Discriminator module mirrors the Generator's design. More specifically, it consists of M private subnetworks D_p^i $(i = 1, \ldots, M)$, with each one corresponding to a specific track included in the musical composition, followed by a shared network D_s. At first, each *private Discriminator* D_p^i extracts low-level features and detailed attributes from the corresponding track of the input pianoroll. Their produced outputs are then concatenated and fed into the *shared Discriminator* D_s, which formulates a common, high-level abstraction of the final music representation. Similar to the Generator case, the analogy between the aforementioned structural units and jamming-composer modes accordingly is evident. The main difference between this network mechanism and the reference system lies in the incorporation of track-focused discriminative modules, since MuseGAN employs only one shared Discriminator that evaluates all the contained musical tracks collectively.

As can be seen, our system provides a more compact and consistent mechanism for the unconditional generation task, especially with respect to the input, since it requires only one random noise vector as opposed to MuseGAN, which employs 4 different kinds of inputs, each one representing distinct musical dependencies. It can also be regarded as a structural variation of the hybrid model that incorporates the shared-private design for both Generator and Discriminator modules. This further justifies our architecture since the hybrid model itself merges the two compositional practices from an implementation perspective and according to the respective analysis in section 4.4 of chapter 4, it outperforms the other multi-track interdependency models in terms of inter-track and subjective evaluation, while achieving adequate scores as regards the intra-track objective metrics.

5.2.2 Implementation

As previously noted, all the Generator and Discriminator modules involved in our proposed system are designed as deep Convolutional Neural Networks [2, 19] and implemented using the open source ML framework PyTorch, which is formulated on the Python programming language and the Torch library and constitutes one of the most preferred platforms for deep learning research. We employ as reference the code[1] for the ISMIR tutorial on Music Generation with GANs presented in 2019.

[1] https://github.com/salu133445/ismir2019tutorial

According to the relevant analysis in sections 4.2.5 and 4.3.3 of chapter 4, the original project is designed to handle data of only a specific configuration regarding time-related attributes, such as the number of produced bars and the beat resolution, as well as tonal characteristics like the number of used pitches. In order to tackle this limitation and further investigate the generative capabilities of our proposed system, we extend the baseline implementation by performing a customization process with respect to a group of parameters, that define various generative configurations and are presented in Table 5.1 with their respective notation. This modification allows us to experiment over multiple aspects of the generative procedure and compare their effect on the produced result and its corresponding musical properties. Furthermore, it entails a modulation on the internal structure of our Generator and Discriminator components in accordance to the input and the desired output configuration, resulting in a particularly flexible and adaptive mechanism.

s	number of samples
l	latent dimension
t	number of tracks
r	bar resolution
p	number of pitches
m	number of measures
$o\ (= m \cdot r)$	number of total timesteps
b	beat resolution
i	lowest pitch

Table 5.1: Parameter Notation

Input: $\mathbf{z} \in \mathbb{R}^{s \times l}$ (reshape to $s \times l \times 1 \times 1 \times 1$)					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
transconv	$2 \cdot l$	$m \times 1 \times 1$	$(m, 1, 1)$	Batch	ReLU
transconv	l	$1 \times r/2 \times 1$	$(1, 1, 1)$	Batch	ReLU
transconv	$l/2$	$1 \times 1 \times p/4$	$(1, 1, 1)$	Batch	ReLU
transconv	$l/4$	$1 \times 1 \times (p/4 + 1)$	$(1, 1, 1)$	Batch	ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times l/4 \times m \times r/2 \times p/2}$					

Table 5.2: Shared Generator G_s

Input: $\mathbf{x} \in \mathbb{R}^{s \times l/4 \times m \times r/2 \times p/2}$					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
transconv	$l/8$	$1 \times (r/2 + 1) \times 1$	$(1, 1, 1)$	Batch	ReLU
transconv	1	$1 \times 1 \times (p/2 + 1)$	$(1, 1, 1)$	Batch	ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times m \times r \times p}$ (stack along track axis for final vector)					

Table 5.3: Private Generator G_p

Tables 5.2 and 5.3 demonstrate our parameterized network architectures for the involved Generator modules. As can be seen, both these structural units successively augment the dimensions of the input vector via transposed convolutional operations, which are initially applied along the time axis and afterwards along the pitch axis. Following the reference

implementation of [2], a Batch Normalization layer (BN) is added before each non-linear activation function.

Accordingly, Tables 5.4 and 5.5 display the customized network configuration for the Discriminator components of our proposed music generation framework. It can be easily observed that the Discriminator modules act in reverse mode, compared to the Generator ones, in terms of gradually compressing the spatial dimensions of the corresponding input vector, first along the pitch axis and then along the time one, via the utilization of typical convolutional layers. In this case, the Layer Normalization[2] practice is applied before the non-linearity, as it does not depend on the employed batch size and can be considered more feature-oriented.

Input: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times o \times p}$ (reshape to $s \times 1 \times m \times r \times p$)					
`Layer Type`	`Number of Filters`	`Kernel Size`	`Stride`	`Normalization`	`Activation`
conv	$l/8$	$1 \times 1 \times (p/2+1)$	$(1,1,1)$	Layer	Leaky ReLU
conv	$l/8$	$1 \times (r/2+1) \times 1$	$(1,1,1)$	Layer	Leaky ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times l/8 \times m \times r/2 \times p/2}$ (stack along track axis for next layer)					

Table 5.4: Private Discriminator D_p

Input: $\mathbf{x} \in \mathbb{R}^{s \times tl/8 \times m \times r/2 \times p/2}$					
`Layer Type`	`Number of Filters`	`Kernel Size`	`Stride`	`Normalization`	`Activation`
conv	$l/2$	$1 \times 1 \times (p/4+1)$	$(1,1,1)$	Layer	Leaky ReLU
conv	$l/2$	$1 \times 1 \times p/4$	$(1,1,1)$	Layer	Leaky ReLU
conv	l	$1 \times r/2 \times 1$	$(1,r/2,1)$	Layer	Leaky ReLU
conv	l	$(r/2+1) \times 1 \times 1$	$(1,1,1)$	Layer	Leaky ReLU
conv	$2 \cdot l$	$r/2 \times 1 \times 1$	$(1,1,1)$	Layer	Leaky ReLU
dense	1	(reshape to $s \times 2 \cdot l$ before)			
Output: $\mathbf{y} \in \mathbb{R}$					

Table 5.5: Shared Discriminator D_s

It should be pointed out that our parameterization, from a mathematical perspective, is formulated on multiples and mostly submultiples of the number 2, which constitutes to some extent the basis of the note value[3] system in music theory. Since the utilized training music samples are only in 4/4 time signature, this modified network architecture enables our proposed system to emphasize on rhythmical attributes and in this way capture temporal patterns and motifs presented in the examined musical segments, which can be further employed for the generation of novel data instances.

5.2.3 Training Process

According to the elaborated and detailed analysis on the framework of Generative Adversarial Networks presented in section 2.3 of chapter 2, the GAN learning process can be modeled as a two-player turn-based game, where the alternating actions of the two opponents, the Generator and the Discriminator, involve the update of their respective weight parameters

[2]The implementation code is derived from https://github.com/pytorch/pytorch/issues/1959.

[3]In music notation, a note value indicates the relative duration of a note, based on the characteristics of its representation form.

via the typical BackPropagation Algorithm. Following [2], we employ the modified version of the minimax objective function for the training of our proposed system, that was initially introduced by Gulrajani et al. in [20] and can be mathematically described as follows:

$$\min_{G}\max_{D} V^{*}(G,D) = \mathbb{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] + \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2] \quad (5.2.1)$$

In the context of the utilized notation, p_d represents the ground-truth distribution of the real music samples, $p_{\mathbf{z}}$ indicates the prior distribution from which the input random noise vector $\mathbf{z}$ is sampled and $p_{\hat{\mathbf{x}}}$ is implicitly defined by uniform sampling along straight lines between pairs of points derived from the data distribution p_d and the generator distribution p_g, accordingly. As can be easily observed, the value function V^*, besides the typical probability expressions, includes an additional gradient penalty term, which is found to ensure faster convergence to better optima and stabilize the overall training process through a regularization of the computed gradient magnitude [20].

As regards the consecutive interchanges between the individual learning procedures of the Generator and the Discriminator modules respectively, we follow the relevant research literature [20, 2, 19] and embrace the training strategy which is thoroughly described and graphically illustrated in section 2.3.3 of chapter 2. More specifically, we introduce into our implementation a condition whereby the Generator is updated after every k optimization steps of the Discriminator and further experiment over different values of the hyperparameter k. Furthermore, we employ a batch-learning policy for the training of our proposed system, under which the incorporated model parameters are updated at each step using data derived only from the current mini-batch. Algorithm 3 summarizes all the aforementioned training details and features in a pseudocode format.

In order to gain insights of our learning process, we incorporate an additional *validation phase*. As the name suggests, this phase enables us to observe and also evaluate the behaviour of our proposed GAN over unseen data instances, which are excluded from the set of training music examples. From a computational aspect, it involves the estimation of the Discriminator and Generator losses as performance indicators of the current training step, using samples derived from the so-called *validation set* and the assessment of both real and generated piano-rolls, produced by the current model state, via our employed metric system.

In advance, we further exploit this additional training feature by including an auxiliary Early-Stopping mechanism into our proposed generative framework. In the field of Machine Learning, *Early-Stopping* can be defined as an optimization technique, which is applied in order to prevent overfitting phenomena without compromising model accuracy and efficiency. This procedure mostly relies on the monitoring of a specified performance measure, which is associated with the model's generalization error. If this quantity starts to degrade, indicating the model begins to learn the statistical noise that is inherently integrated into the training dataset and eventually surpasses a predefined limit, a respective condition triggers the termination of the training process. Since validation strategy constitutes the most frequently used and at the same time effective approach regarding the implementation of this method, we employ as our monitoring metric the mean sum of the Discriminator and the Generator losses, as computed during validation phase at each training step.

Finally, we also apply a *checkpointing* technique for the purpose of saving various versions of our model components at successive stages of the training process, as they can be afterwards used for the generation of novel music pieces in an inference context.

Algorithm 3: Mini-batch Training Algorithm for GANs with Gradient Penalty

In **Input:**
- Gradient penalty coefficient λ
- Number of Discriminator iterations per Generator iteration k
- Batch size m
- Adam[4] hyperparameters α, β_1, β_2
- Number of total training steps N

for *number of training iterations* **do**

`Update Discriminator` D_w `by ascending its stochastic gradient.`

for k *steps* **do**

for i=1, ..., m **do**

`Sample`: real data $\boldsymbol{x} = (\boldsymbol{x}_t, \boldsymbol{x}_a) \sim p_d$

latent variable $\boldsymbol{z} \sim p_{\boldsymbol{z}}$

random number $\epsilon \sim U[0, 1]$

$\bar{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$

$\hat{\boldsymbol{x}} \leftarrow \epsilon \cdot \boldsymbol{x} + (1 - \epsilon) \cdot \bar{\boldsymbol{x}}$

$L^{(i)} \leftarrow D_w(\bar{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$

endfor

$$w \leftarrow Adam\left(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$$

endfor

`Update Generator` G_θ `by descending its stochastic gradient.`

Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p_{\boldsymbol{z}}$

$$\theta \leftarrow Adam\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} [-D_w(G_\theta(\boldsymbol{z}^i))], \theta, \alpha, \beta_1, \beta_2\right)$$

endfor

[4] Adam is an optimization algorithm that can be used instead of the typical stochastic gradient descent procedure for the update of the network weights in an iterative framework. This method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the respective gradients. For more details, please refer to section 2.2.2 of chapter 2.

5.3 Data

5.3.1 Data Representation

Following [2, 19], we employ the *multi-track pianoroll* format for the representation of the examined music samples under the framework of our proposed convolutional GAN-based system. According to the detailed analysis in section 4.3.1 of chapter 4, a multi-track pianoroll is defined as a set of piano-rolls, each one corresponding to a specific musical instrument included in the overall composition. As graphically demonstrated in Figure 5.3.1, the vertical axis of the score-like pianoroll matrix represents pitches in ascending order, while the horizontal dimension indicates time in a symbolic format that discards the tempo information. The contained binary values designate the presence (1) or absence (0) of notes over different timesteps.

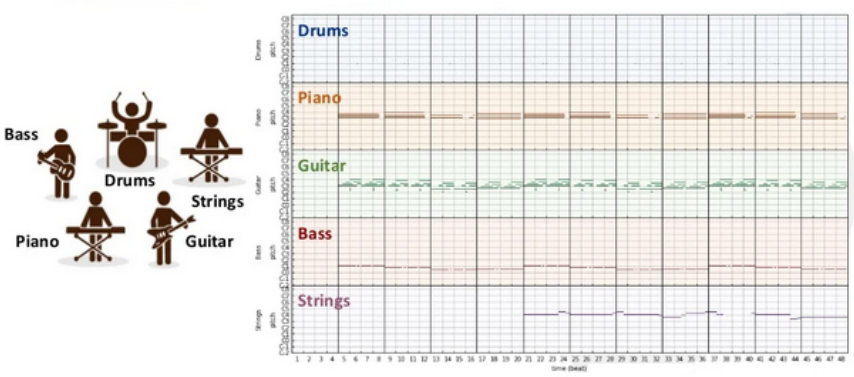


Figure 5.3.1: Multi-track pianoroll format [228]

5.3.2 Dataset

The pianoroll dataset used for the training process of our model is derived from the Lakh MIDI Dataset (LMD) [16] after the preprocessing procedure, which is thoroughly described in section 4.3.3 of chapter 4 and graphically illustrated in Figure 4.3.3. More specifically, we employ the *LPD-5-cleansed* version[5], which contains only those pianorolls with the higher matching confidence score to MSD entries [17], a "Rock" tag and 4/4 time signature.

5.3.3 Data Preprocessing

As mentioned before, the final set of training examples, which will be processed under the framework of our proposed model, requires the segmentation of the pianorolls included in the cleansed dataset into musical phrases of proper format, regarding the temporal and the tonal arrangement. To this end, we develop a routine responsible for the data preparation, based on the open source Python library Pypianoroll. Following our customization practice regarding the internal architecture of both the Generator and Discriminator modules, which is elaborately presented in the previous section, we further parameterize our implementation

[5] https://salu133445.github.io/lakh-pianoroll-dataset/dataset.html

with respect to a group of user-defined attributes that determine the data configuration. In particular:

1) At first, a downsampling process is applied. In this way, the temporal resolution of the input pianorolls is set to the proper size, as specified by the parameters *beat resolution* and *measure resolution* respectively.

2) Afterwards, the target pitch range is acquired (vertical dimension), in accordance with the input parameters *lowest pitch* and *number of pitches*. Notes outside this scope are discarded.

3) Finally, a variable number of candidate samples is collected from each song included in the multi-track pianoroll dataset, based on a randomized rule. The size of each sample is defined by the parameter *number of measures*, which equivalently indicates the length of a musical phrase. Only samples that contain an adequate amount of notes among the various tracks, as specified by a fixed threshold, are retained.

Eventually, after these data preprocessing steps, the resulting set of musical examples comprises approximately 15600 phrases, derived from 7323 distinct rock songs. The exact number varies for each particular configuration, mainly due to the integrated randomness concerning the selection of candidates, as well as the downsampling process, in accordance with the specified resolution. Furthermore, we perform a splitting of our final dataset into a training and a validation set accordingly. As the name suggests, the training set is used for the iterative updates of the model weights, while the validation one is deployed during our introduced validation phase for the evaluation of the system performance. The utilized split ratios are graphically demonstrated in the pie chart of Figure 5.3.2. The overall training duration is less than 3 hours with a GeForce RTX 2080 Ti GPU.

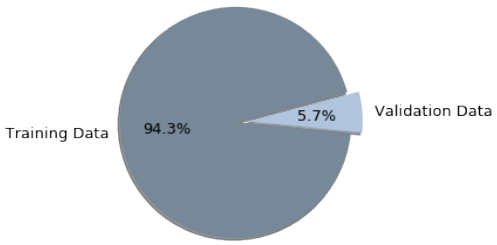


Figure 5.3.2: Dataset split ratios

5.4 Experimental Protocol

5.4.1 Experimental Setup

In order to evaluate our proposed model and and thoroughly examine its effectiveness over the creation of aesthetic multi-track polyphonic musical pieces from scratch, we conduct a group of experiments that enable us to investigate various aspects of the generative process. The corresponding details as well as the produced results will be extensively presented in the following sections of this chapter. Before proceeding further, we consider it useful to define the experimental configurations that will be employed on the proximate analysis. These configurations are summarized in Table 5.6. In more detail:

- C_1 simulates the data configuration utilized under the framework of MuseGAN.
- C_2-C_5 examine 4 distinct values of the parameter *beat resolution* under $k = 6$ (number of steps per Generator update).
- C_6-C_9 examine 4 distinct values of the parameter *beat resolution* under $k = 11$ (number of steps per Generator update).
- C_{10} investigates doubling the latent dimension along with a 4 times smaller batch size.

		C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
Generation parameters	Number of Pitches	84	72	72	72	72	72	72	72	72	72
	Beat Resolution	24	4	8	12	16	4	8	12	16	4
	Number of Bars	4	4	4	4	4	4	4	4	4	4
	Lowest Pitch	24	24	24	24	24	24	24	24	24	24
	Samples per song	8	8	8	8	8	8	8	8	8	8
	Latent Dimension	128	128	128	128	128	128	128	128	128	256
Training parameters	Number of Steps	10000	10000	10000	10000	10000	10000	10000	10000	10000	10000
	Batch Size	16	16	16	16	16	16	16	16	16	4
	Number of Phrases	4	4	4	4	4	4	4	4	4	4
	Steps per G update	6	6	6	6	6	11	11	11	11	6
	Steps per Evaluation	50	50	50	50	50	50	50	50	50	50
	Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
	Betas	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)	(0.5, 0.9)

Table 5.6: Experiment Configurations

5.4.2 Objective Metrics

As pointed out in section 4.4.1 of chapter 4, Dong et al. [2] utilize one inter-track and four intra-track musical metrics for the objective evaluation of the MuseGAN, that can be computed for both the real and the generated samples:

- **Empty Bars (EB):** ratio of empty bars included in the examined track (in %)
- **Qualified Notes (QN):** ratio of "qualified" notes (in %)
- **Drum Pattern (DP):** ratio of notes in beat patterns of 4/4 rhythm (in %)
- **Tonal Distance (TD):** measures the harmonicity between a pair of musical tracks
- **Used Pitch Classes (UPC):** mean number of pitch classes used per bar (from 0 to 12)

In the context of our approach, we initially re-implement from scratch all the aforementioned musical metrics according to their descriptive analysis, as it is presented in the original paper [2]. For this purpose, we employ the Python library NumPy, which constitutes a fundamental package for scientific computing based on structured matrices. Therefore, we transform the multi-dimensional real-valued torch tensors produced by our Generator module, as well as the real samples, into binary arrays via a *Thresholding* operation. As the name suggests, *thresholding* is the process of setting to zero all values that are lower than a predefined threshold and mapping the rest to one. We also apply this technique for the visualization of these real-valued vectors in the form of multitrack pianorolls.

We further expand our employed objective evaluation system, by introducing three additional musical metrics[6] that emphasize on tonal characteristics and texture elements:

- **Used Pitches (UP):** mean number of unique pitches used per bar, including all octaves in the predefined range
- **Scale Ratio (SR):** ratio of notes in the given music scale[7] (in %)
- **Polyphonic Rate (PR):** ratio of polyphonic timesteps[8] (in %)

It is worth mentioning that these supplementary quantitative indices are also considered intra-track and can be calculated for both real and fake samples, produced by our proposed generative system.

[6] We base our implementation on an existing code version of the baseline project.

[7] As stated in LMD Statistics, a significantly large number of MIDI files contained in the Lakh MIDI Dataset and consequently in our Lakh Pianoroll Dataset (LPD) are in *C major* scale, since this musical key constitutes the most frequently used and easily applied choice for many automatic MIDI transcription software packages. Therefore, we use the *C major* scale for the implementation of our SR metric, which accordingly indicates the percentage of physical tones without accidentals, such as sharp and flat, in the corresponding pianoroll.

[8] A timestep is considered polyphonic if the number of pitches being simultaneously played at this specific temporal slot exceeds a specified threshold. Typically, this threshold value is set to 2.

5.5 Results

5.5.1 Analysis of Training Process

As discussed in section 2.3 of chapter 2, the core mechanism of GAN systems in the field of generative modeling is fundamentally based on the adversarial learning game between the two opponents, the Generator and the Discriminator. However, the ideal training method still remains an open problem, since it is particularly hard to explicitly identify the convergence state from a computational perspective. Thus, a thorough examination of the training process of our proposed model regarding the task of Unconditional Generation is considered essential.

To this end, in order to gain insights into the learning procedure and elaborately inspect the behavior of the individual system components, we employ the experiment configuration C_2, which is presented in Table 5.6 and constitutes the default case under our music generation framework. Other model variants corresponding to different generative configurations in the context of our customized implementation lead to similar results and hence are not included at this section of the experiment analysis.

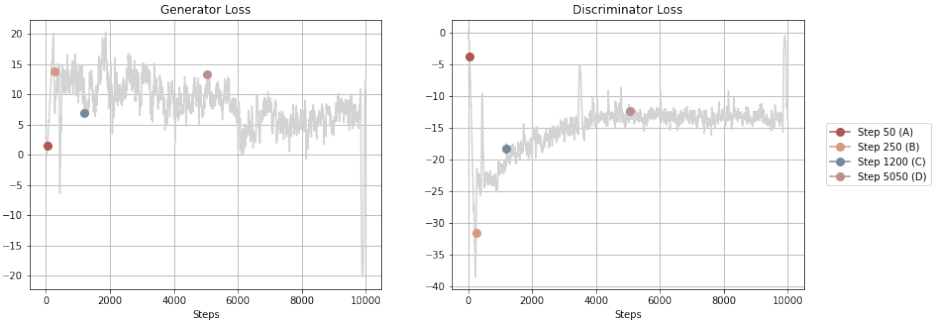
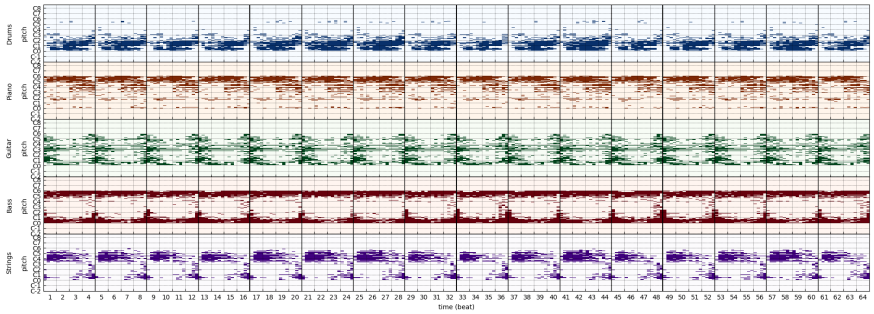


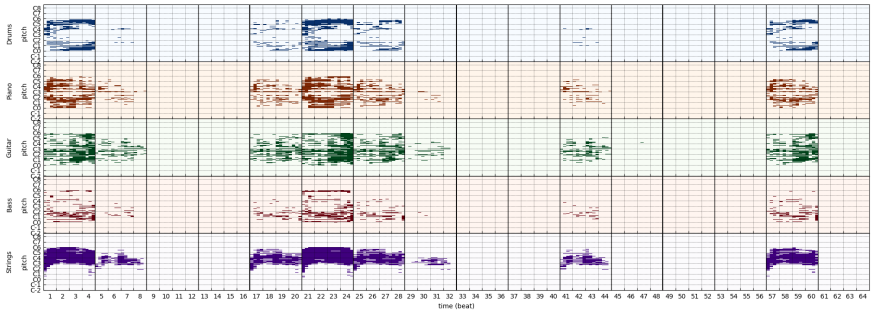
Figure 5.5.1: Training Loss of Generator and Discriminator Modules

Figure 5.5.1 demonstrates the training losses of the Generator G and the Discriminator D accordingly, formulated as functions of training steps. More specifically, the gray curves indicate the normalized loss values, which are smoothed via a moving average operation (over N steps). It can be observed that the Discriminator loss initially follows an increasing trend, as the Generator gradually uncovers underlying properties of the target data distribution and hence produces more plausible music samples that are not easily distinguishable, and approximately after point D it saturates. On the other hand, the Generator loss slowly decreases in a more irregular fashion, as it progressively learns patterns and data features that can "fool" the Discriminator in terms of inducing a gradual reduce of its classification accuracy.

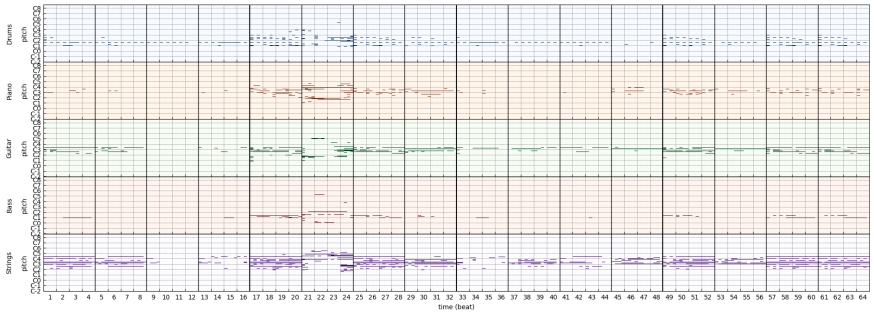
Figure 5.5.2 displays the generated piano-rolls that correspond to the four points marked in Figure 5.5.1 and are produced during the validation phase, which is integrated at each step of our training process, as mentioned in section 5.2.3. This illustration enables us to observe the evolution of the created samples during training and examine the learning procedure from a visual perspective. For further inspection, we also present in Table 5.7 the values of our employed objective musical metrics, computed at the aforementioned training steps for the respective piano-rolls. It is worth underlining the following observations:



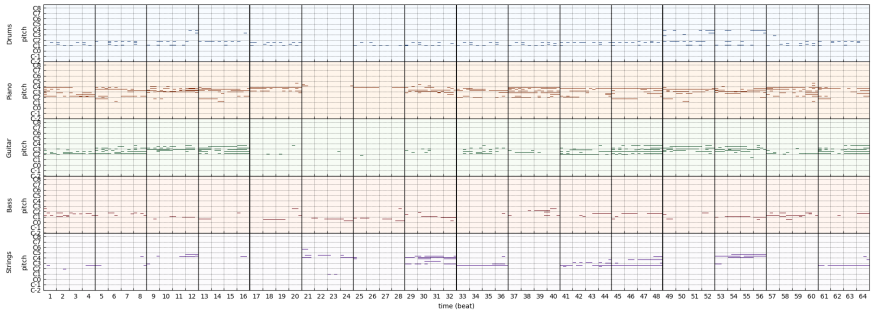
(a) Step 0 (point A)



(b) Step 250 (point B)



(c) Step 1200 (point C)



(d) Step 5050 (point D)

Figure 5.5.2: Evolution of the generated piano-rolls as a function of update steps

	EB (%)					**UPC**				**QN** (%)				**UP**			
	B	D	G	P	S	B	G	P	S	B	G	P	S	B	G	P	S
Point A	0.0	0.0	0.0	0.0	0.0	12.0	12.0	12.0	12.0	51.3	45.9	49.6	47.8	55.69	62.25	51.75	57.0
Point B	50.0	56.2	43.8	50.0	50.0	4.94	5.94	5.56	6.0	37.7	61.5	54.8	71.1	14.62	26.69	21.19	21.94
Point C	31.2	0.0	6.2	12.5	0.0	1.75	4.5	4.38	6.06	53.1	53.5	31.7	48.3	1.94	5.44	5.94	9.88
Point D	0.0	6.2	0.0	0.0	18.8	3.31	4.56	5.62	2.31	49.3	35.8	49.3	58.7	3.62	5.94	7.69	2.62

	TD						**SR** (%)				**PR** (%)					**DP** (%)
	B-G	B-S	B-P	G-S	G-P	S-P	B	G	P	S	B	D	G	P	S	D
Point A	0.7	0.64	0.52	0.95	0.94	0.85	58.5	62.8	58.1	63.1	100.0	94.9	100.0	100.0	100.0	53.5
Point B	0.63	0.52	0.59	0.44	0.57	0.45	52.0	60.6	66.2	60.2	37.7	52.4	61.5	54.8	71.1	52.8
Point C	0.76	0.97	0.66	0.96	0.9	1.1	87.1	90.7	81.4	93.1	2.7	15.6	26.2	15.2	73.0	87.0
Point D	0.96	0.66	1.01	0.87	1.07	0.93	59.1	95.1	61.9	71.6	0.8	2.3	31.2	46.9	9.8	82.2

Table 5.7: Metric values at marked points of training process

- At **point A**, i.e. the initial step of the training process, where the Generator is characterized by complete ignorance regarding the target data distribution and its properties, the output samples are quite random and involve a significant amount of noise. From a quantitative aspect, this results in zero EB rates for all the comprised tracks, suggesting that there aren't any empty bars in the corresponding pianoroll and also the inclusion of all twelve pitch classes in multiple octaves, as indicated by UPC and UP values. Furthermore, QN and PR point out that the produced music is extremely polyphonic and approximately half of the contained notes are "qualified". Lastly, since the examined key is C major, 7 out of the 12 pitch classes are considered in scale, resulting in SR close to 60% and hence relatively small TD values, implying moderately strong harmonic relations among tracks.

- At **point B**, which corresponds to the training step 250, a denoising trend can be clearly observed, since the corresponding pianoroll contains significantly less notes in comparison with the previous point and moreover half of the produced bars are empty, as indicated by the respective EB values. This fact implies that the model begins to discover the note density of the target data distribution. From UPC and UP accordingly, we can conclude that, at this stage of the training, the Generator also begins to grasp the proper pitch range of each track. In particular, it can be easily seen that clusters of notes are gathering between specific boundaries, resulting, in this way, at even smaller TD values and consequently stronger harmonic relations. The rest of the objective metrics display similar behavior with point A.

- As the training progresses, the Generator gradually detects rhythmical motifs and other latent texture elements of the target distribution, such as the duration of the contained notes. As can be observed, the pianoroll of **point C** (step 1200) includes longer notes especially at the Bass track, which tends to become monophonic according to the standards of Rock music. This qualitative inference aligns with the increase of the corresponding QN rate and the decrease of the PR metric respectively. The UP and UPC values indicate an improvement of the pitch ranges for each individual track, while the increase of SR ratios implies that the majority of them are in scale. Furthermore, the significantly high value of DP percentage suggests that the Drum track captures to a large extent the 4/4 rhythm, via the inclusion of notes in 8- or 16-beat patterns.

- Lastly, it can be easily observed that the pianoroll of **point D**, which corresponds to the training step 5050, approximately follows the desired ground-truth distribution of music segments, as it is characterized by essential musical properties. In comparison with the previous training point, the respective `EB`, `UP` and `UPC` values indicate a further denoising of Drum and Strings, which consequently entails an improvement of the harmonic interrelations between Strings and the other tracks, as suggested by the smaller `TD` values. Additionally, `PR` shows that the Bass track plays a single melodic line composed of the lowest pitches, while the rhythmic pattern of the Drum track is much more evident.

5.5.2 Model for Inference

As mentioned in previous sections of this chapter, two distinct losses are involved in the training process of our proposed GAN-based generative system:

- *Discriminator loss:* As the name suggests, the Discriminator loss represents the cost arising from incorrect predictions over the origin class of the examined data samples. In practice, it comprises two individual losses, one quantifying the misclassification errors concerning data instances derived from the ground-truth distribution and the other indicating the cost related to the predictions over unauthentic candidates originated from the Generator distribution.
- *Generator loss:* The Generator loss quantifies the feedback from the Discriminator regarding its classification predictions over the fake samples. More specifically, it represents the cost arising from the successful identification of its produced data instances as fake by its opponent.

During our experiments, we employ the mean sum of the aforementioned loss values, computed at the validation phase of each training step, as the monitoring metric of our auxiliary Early-Stopping mechanism. We also attempt to calibrate this sum via the introduction of proper weights corresponding to the Generator and the Discriminator modules accordingly.

We observed that the learning procedure becomes smoother and more regular for larger Discriminator loss weights, since the Early-Stopping system requires a significant number of steps to trigger the training termination. However, in this case, the generated music samples are quite noisy and substantially differ from the ones included in the ground-truth distribution. This is probably owed to the fact that the Generator loss, which is indissolubly associated with the generation performance and by extension determines the quality of the produced musical pieces, is disregarded. On the other hand, larger Generator loss weights induce an early termination of the training process, which may prevent the Generator from discovering underlying patterns and features of the target data distribution.

Therefore, we can conclude that the utilization of a combination of the two distinct losses in the form of sum as the monitoring index, leads to a training imbalance between the two individual components of our GAN system, which negatively affects their respective performance. To this end, we employ for our Inference process the Generator model of the last training step and not the one indicated by the aforementioned quantity, as it has been experimentally proven [2] that this Generator version is capable of producing music of

sufficient quality.

Nevertheless, Heusel et al. [229] introduced in 2017 a novel monitoring metric for the training of GAN systems in the research field of Computer Vision, known as *Fréchet Inception Distance* or *FID* for short. From a mathematical perspective, it is defined as the Wasserstein distance between two multivariate Gaussian distributions, which represent the real and fake images accordingly. The features modeled by each distribution are extracted from the inception layer of a deep CNN, called *Inception-v3* [230], which is usually employed for various Computer Vision tasks. However, the inclusion of this metric into our research problem in the area of Music Generation requires an equivalent network trained to extract musical features from representations of symbolic format, such as the multi-track piano-roll. Thus, we consider the aforementioned implementation as a potential direction for future work.

5.5.3 Qualitative Inspection

Figure 5.5.3 illustrates the multi-track pianoroll of 4 musical phrases generated from scratch by our proposed GAN-based framework during the process of Inference. The employed Generator network corresponds to the experiment configuration C_2, as presented in Table 5.6. It is worth pointing out the following qualitative observations:

- The included tracks are generally playing in the same music scale, preserving an approximate pitch range from C2 to C4.
- Chord-like intervals can be detected at multiple timesteps, especially in Guitar, Piano and Strings, which are the mainly polyphonic tracks and hence tend to play the accompaniment parts.
- The Bass track is principally monophonic, playing a single melodic line often composed of the lowest pitches.
- The Drum track follows an evident rhythmic motif, which mainly comprises notes in 8- or 16-beat patterns.
- The pitches of the melodic tracks (all expect Drums) sometimes overlap. This fact indicates nice harmonic relations among the included instruments, that can contribute to an acoustically pleasant result.

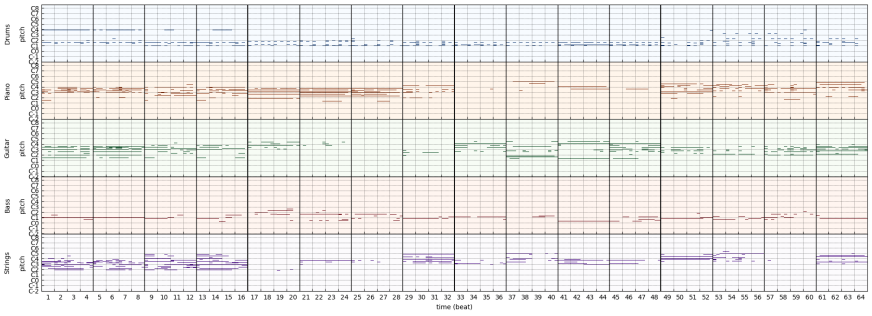


Figure 5.5.3: Pianoroll generated using our proposed model

5.5.4 Experimentation over Generative Configurations

Our customized model implementation, which is thoroughly discussed in section 5.2.2, enables us to experiment over different generative configurations with respect to the included parameters. In order to examine the resulting variants of our framework and compare their performance on the task of Unconditional Generation, we evaluate them in terms of our proposed objective metrics, as presented in detail at section 5.4. More specifically, for each experimental configuration included in Table 5.6 (all except C_1) we generate, using the respective Generator of the last training step, 20000 bars organised in 4-measure musical phrases, on which we afterwards perform our objective assessment.

Table 5.8 provides an overview of the produced results in the form of mean values. In order to gain further insights into the musical properties of the ground-truth distribution and acquire a reference point, we also apply our objective metrics on the employed set of training music samples. The respective statistics are displayed in the first row of Table 5.8. Since the main goal of the Generator is to approximate as best as possible the quality of real music pieces, values closer to the training ones correspond to better performance. However, following our baseline project, in case of the inter-track evaluation part represented by `TD` metric, smaller values are considered better regarding the efficiency of the corresponding model.

At first glance, we can easily observe the absence of a particular parameter configuration capable of improving all the employed objective musical metrics. This fact highlights the need for inclusion of human feedback into our assessment practice, since the quantitative indicators cannot establish a unified criterion for the designation of the best model in terms of generation efficiency and produced musical quality.

	EB (%)					**UPC**				**QN** (%)				**UP**			
	B	D	G	P	S	B	G	P	S	B	G	P	S	B	G	P	S
Ground-truth	1.6	1.1	4.1	5.1	3.2	2.48	4.16	4.2	4.57	91.7	85.3	89.7	89.7	2.72	5.8	5.9	6.8
C_2	0.3	0.0	0.9	1.9	2.1	2.89	4.4	4.88	5.14	59.0	58.2	57.2	60.8	3.14	5.96	6.58	7.61
C_3	0.4	0.0	0.9	0.7	0.7	3.12	5.18	5.33	5.14	49.0	52.2	56.5	64.6	3.4	7.57	7.73	**7.05**
C_4	0.0	2.1	0.6	1.2	0.9	3.04	**4.17**	**4.39**	5.47	50.9	59.7	65.9	70.3	3.39	**5.71**	**6.54**	7.75
C_5	0.0	**0.8**	1.6	1.0	**2.5**	3.09	4.05	4.58	4.14	**63.1**	**72.9**	**72.4**	**74.3**	3.32	5.9	6.6	5.97
C_6	0.5	0.1	1.8	0.8	0.7	**2.47**	4.9	5.07	5.4	54.3	48.9	52.9	50.6	**2.67**	6.67	7.24	8.22
C_7	0.1	0.1	1.6	0.2	0.4	2.75	4.36	4.87	5.49	56.2	64.9	59.1	57.2	3.15	6.06	6.8	7.72
C_8	**1.9**	0.1	**4.3**	**2.8**	0.4	2.64	5.81	6.08	5.09	**63.1**	56.7	60.1	64.3	2.86	8.19	8.44	7.78
C_9	0.0	0.2	1.5	0.0	0.2	3.06	3.92	5.41	5.48	62.9	54.5	55.0	45.3	3.4	5.62	7.3	9.24
C_{10}	0.2	0.1	0.0	0.0	0.4	2.69	5.09	5.16	**4.52**	57.2	69.5	66.8	60.8	2.99	7.59	7.3	7.12

	TD						**SR** (%)				**PR** (%)					**DP** (%)
	B-G	B-S	B-P	G-S	G-P	S-P	B	G	P	S	B	D	G	P	S	D
Ground-truth	0.7	0.73	0.7	0.7	0.67	0.66	75.7	74.6	73.9	72.6	1.2	15.2	57.3	60.8	64.1	83.1
C_2	0.86	0.91	0.9	0.98	0.99	0.97	79.0	82.1	78.7	75.0	2.6	21.7	**49.7**	**53.7**	**58.7**	79.6
C_3	0.57	0.53	0.56	0.6	0.62	0.59	80.4	76.3	**72.7**	70.0	0.6	6.8	47.5	47.7	55.0	92.3
C_4	0.37	0.38	0.36	0.39	0.39	0.38	70.6	82.5	81.9	78.4	0.1	4.6	28.5	37.9	50.2	88.4
C_5	**0.26**	**0.27**	**0.28**	**0.27**	**0.25**	**0.27**	82.5	73.8	77.5	77.1	0.2	2.9	42.9	47.0	55.4	53.1
C_6	0.9	0.96	0.92	1.08	1.03	1.09	80.5	77.6	79.0	78.7	**1.4**	10.8	42.6	43.5	57.3	**83.5**
C_7	0.5	0.56	0.51	0.61	0.56	0.6	**75.7**	**75.1**	78.5	75.5	0.7	5.7	35.0	34.8	48.9	96.0
C_8	0.42	0.37	0.42	0.45	0.49	0.46	70.6	71.9	66.3	67.7	0.1	4.4	39.5	40.0	52.6	92.6
C_9	0.27	0.31	0.31	0.34	0.34	0.38	76.3	82.3	76.1	82.8	0.3	3.5	23.4	33.6	50.0	58.7
C_{10}	0.96	0.89	0.95	0.93	1.04	1.05	84.4	69.4	78.2	**71.2**	1.9	**14.2**	66.9	51.7	56.8	81.6

Table 5.8: Inference objective metrics for different experiment configurations

A closer inspection indicates that the increase of the utilized beat resolution (configuration

C_5) results in significantly strong harmonic relations among the tracks (TD) and also a large proportion of "qualified" notes (QN), probably due to the fact that notes of longer duration are more easily generated when a larger number of timesteps is employed for each beat. However, it seems to negatively affect the rhythmic attributes of the produced musical pieces (DP), as only half of the contained notes are in 8- or 16-beat patterns.

On the other hand, the selection of higher values for the hyperparameter k, which denotes the number of training steps per Generator update, has a positive impact on the note density (EB) and also other tonal characteristics of the generated samples, such as the SR. As stated in [21], the inclusion of more steps for the individual training process of the Discriminator ensures that it is fine-tuned near its optimal solution, while the Generator remains fixed. In this way, the Generator is implicitly forced to uncover more detailed features of the latent target distribution in order to fool its opponent, thereby producing candidates that approximate human-composed music to a larger extent.

Lastly, doubling of the latent dimension (configuration C_{10}) induces a slight improvement of UPC and SR metrics for the Strings track, which is the most problematic of the included instruments, as it usually incorporates a significant amount of noise. Furthermore, it benefits the Drum track in terms of the desired polyphonic rate (PR).

5.5.5 Objective Comparison with Baseline

In order to examine if our proposed modifications and additional extensions constitute an actual novelty in the research field of Automatic Music Synthesis and more specifically the task of Unconditional Generation, we proceed in a comprehensive comparison between our music generation system and the baseline project, MuseGAN, which is extensively presented in chapter 4, under the employed objective metrics.

In particular, we select two different experimental configurations, corresponding to two distinct variants of our proposed framework, to be compared with the four multitrack interdependency models included in MuseGAN and thoroughly examined in section 4.2.2 of chapter 4. The first one is configuration C_1, which is equivalent to the original implementation in terms of parameter values and the other is configuration C_2, which is considered as default in the context of our conducted experiments. Both the aforementioned configurations are explicitly presented in Table 5.6. In order to ensure a fair comparison between the two music generation approaches, we follow the inference practice of our baseline project. In particular, for each model variant involved in the evaluation process, we generate, using the respective Generator of the last training step, 20000 bars organised in 4-measure musical phrases. Afterwards we apply our objective metrics on the produced musical segments and calculate the respective mean values.

Table 5.9 summarizes the results of the intra-track evaluation for all the examined models, in terms of those objective musical metrics which are shared between the two implementations. As mentioned previously, in this case values closer to the ones representing musical properties of the ground-truth distribution correspond to the desirable generative behavior and hence are considered better. However, it can be easily observed that there is a substantial divergence between the training data statistics, as measured under the two examined frameworks. This difference may result from the randomized rule that determines the collection of training

		EB (%)					UPC				QN (%)				DP (%)
		B	D	G	P	S	B	G	P	S	B	G	P	S	D
training data	*baseline*	8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
	ours	1.6	1.1	4.1	5.1	3.2	2.48	4.16	4.2	4.57	91.7	85.3	89.7	89.7	83.1
Baseline	jamming	6.59	2.33	18.3	22.6	6.10	1.53	3.69	4.13	4.09	71.5	56.6	62.2	63.1	93.2
	composer	0.01	28.9	1.34	0.02	0.01	2.51	4.20	4.89	5.19	49.5	47.4	49.9	52.5	75.3
	hybrid	2.14	29.7	11.7	17.8	6.04	2.35	4.76	5.45	5.24	44.6	43.2	45.5	52.0	71.3
	ablated	92.4	100	12.5	0.68	0.00	1.00	2.88	2.32	4.72	0.00	22.8	31.1	26.2	0.0
Ours	C_1	0.0	**0.7**	0.4	1.3	1.2	3.63	4.67	**4.64**	5.29	55.6	**75.8**	**74.1**	**75.9**	59.5
	C_2	**0.3**	0.0	**0.9**	**1.9**	**2.1**	**2.89**	**4.4**	4.88	**5.14**	**59.0**	58.2	57.2	60.8	**79.6**

Table 5.9: Results of intra-track evaluation of the baseline models, as well as our proposed framework.

samples from the songs included in the employed pianoroll dataset, as explained in section 5.3.3. Thus, the two generative systems cannot be exactly compared in the context of intra-track evaluation metrics. Furthermore, the re-training of the original model using our set of data instances is not feasible, mainly due to the diverse implementations. Nevertheless, as it might be seen, both of our model variants accomplish to significantly approximate the statistics of the real distribution (bold values denote greater proximity). Moreover, as regards the QN and DP metrics, where the aforementioned divergence is negligible, we remark that our model outperforms all the baseline variations to a large degree (coloured cells).

		TD					
		B-G	B-S	B-P	G-S	G-P	S-P
Baseline	jamming	1.56	1.60	1.54	1.05	0.99	1.05
	composer	1.37	1.36	1.30	0.95	0.98	0.91
	hybrid	1.34	1.35	1.32	0.85	0.85	0.83
Ours	C_1	0.2	0.22	0.2	0.21	0.2	0.21
	C_2	0.86	0.91	0.9	0.98	0.99	0.97

Table 5.10: Results of inter-track evaluation of the baseline models, as well as our proposed framework

The results of the inter-track evaluation, which is represented by TD metric, are demonstrated in Table 5.10 for all models involved in the comparative analysis. As referred to earlier, in this case smaller values correspond to smaller Euclidean distances in the interior space of a 6D polytope and hence imply stronger harmonic relations between the examined pairs of tracks. It can be easily affirmed that our model with configuration C_1 notably surpasses all the baseline multi-track interdependency architectures, by generating extremely harmonic music segments, as indicated by the corresponding TD, which is around 0.2 for all the included pairs. This observation comes in agreement with the outcomes of the experiments discussed in the previous section, which indicate that the increase of the utilized beat resolution induces an improvement of the harmonicity in the produced music samples. Furthermore, it should be pointed out that the performance of the model with configuration C_2, even though weaker than C_1, is also considerable, especially regarding the harmonic distance between a melody-like track, such as the Bass and a chord-like track, such as the Piano, Guitar and Strings.

Finally, the supplementary Table 5.11 displays the results corresponding to the three additional intra-track metrics, that we have incorporated into our objective evaluation system, only for the two examined variants of our proposed framework. Bold values denote greater

		UP				**SR** (%)				**PR** (%)				
		B	G	P	S	B	G	P	S	B	D	G	P	S
training data		2.72	5.8	5.9	6.8	75.7	74.6	73.9	72.6	1.2	15.2	57.3	60.8	64.1
Ours	C_1	4.33	6.86	**6.56**	8.01	85.2	**81.0**	84.0	**70.9**	**0.8**	1.9	33.2	31.3	36.9
	C_2	**3.14**	**5.96**	6.58	**7.61**	**79.0**	82.1	**78.7**	75.0	2.6	**21.7**	**49.7**	**53.7**	**58.7**

Table 5.11: Additional results on the evaluation of our proposed framework

proximity to the ground-truth distribution. As can be seen, the generation efficiency of both experiment configurations is confirmed also in the context of tonal characteristics, as quantified by UP and SR metrics and texture elements, such as PR.

5.6 User Study

5.6.1 Experimental Setup

As discussed before, the objective metrics cannot precisely reflect the human perception over a piece of music, since they are just quantitative indicators of musical properties. Thus, human evaluation is considered essential. To this end, following [2], we conduct a user study in the form of a listening test, which, according to the extensive analysis in section 3.4.2 of chapter 3 on the matter of subjective evaluation, is generally the most preferable assessment practice in the research field of generative modeling. Our survey is divided into two parts, with each one corresponding to a specific music generation task implemented by our proposed framework. More specifically, the section of Unconditional Generation, which constitutes the principal topic of this chapter, aims at a comprehensive comparison between our Convolutional GAN-based system and the official re-implementation of the baseline project MuseGAN. To this end, we employ an A/B testing[9] format for the structure of our questionnaire, under which musical segments, all generated from scratch, i.e. without any prior knowledge or additional information, are paired in such a way that every created listening couple involves results from both the aforementioned models. The evaluator is required to choose from each pair the sample that prefers in terms of:

- **Musical Naturalness:** Could the musical segment be composed by a human?
- **Harmonic Consistency:** Are the sounds produced by different instruments in musical consonance? Is the result acoustically pleasant?
- **Musical Coherence:** Are the various musical phrases associated someway through time?

It can be easily affirmed that the first question is established on the intuitive concept of the typical Turing test, which constitutes the simplest form of listening assessment and is applied in several studies of generative music systems [231, 15]. This strategy evaluates whether a machine is able to exhibit behavior indistinguishable from humans, in terms of generating musical pieces which can be considered man-made by the users. Since our examined research problem concerns the generation of aesthetic music in an autonomous manner, i.e. with minimum human intervention, musical naturalness is undoubtedly essential.

As regards the second question, harmonicity is another fundamental aspect of music compositions and hence is examined in a huge variety of assessment studies in the research field of Automatic Music Synthesis, including our reference project [2, 15, 140, 232]. It is interrelated to the acoustical result, as it defines how euphonic and cohesive is the composite product of individual musical voices. Thus, it constitutes an easily perceivable feature of the produced music samples from the perspective of the human listeners.

Finally, in order to acquire human feedback over the deeper structure of our generated examples, we investigate the property of musical coherence. This term refers to vaguely defined qualities in musical pieces that create a sense of connectivity and cohesion among the various parts of a musical entity. Musical coherence ensures that the arrangement of a music

[9] A/B testing constitutes an experimental methodology for the controlled comparison between two variants, A and B, established on individual user choices over pairs of samples representing the two alternatives.

composition represents an abstract musical idea and does not just involve random disjoint musical phrases.

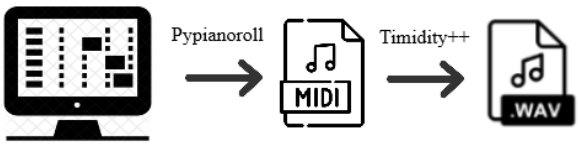


Figure 5.6.1: File Conversion Diagram

As regards the implementation details, we conduct our evaluation experiment online by designing the respective application based on the Python web development framework Flask. All the utilized samples are transformed into the proper auditory format, according to the diagram of Figure 5.6.1. As can be seen, the produced pianorolls are initially converted into MIDI files using the Python library Pypianoroll[10] and then into WAV files via the software synthesizer Timidity++[11]. After performing preprocessing by pruning longer audio samples, the duration of each resulting audio clip, regardless of its derivation model, is equal to approximately 12 seconds, a time period that corresponds to one 4-bar musical phrase under the framework of our proposed system and the employed multitrack pianoroll representation. The samples of each listening pair are randomly selected from two pools of 200 audio clips each, produced by our proposed system using the experimental configuration C_2 and the baseline project using the composer variant accordingly and presented to the users in random order. In order to avoid bias, we use *LPD-cleansed* version as the training dataset for both models.

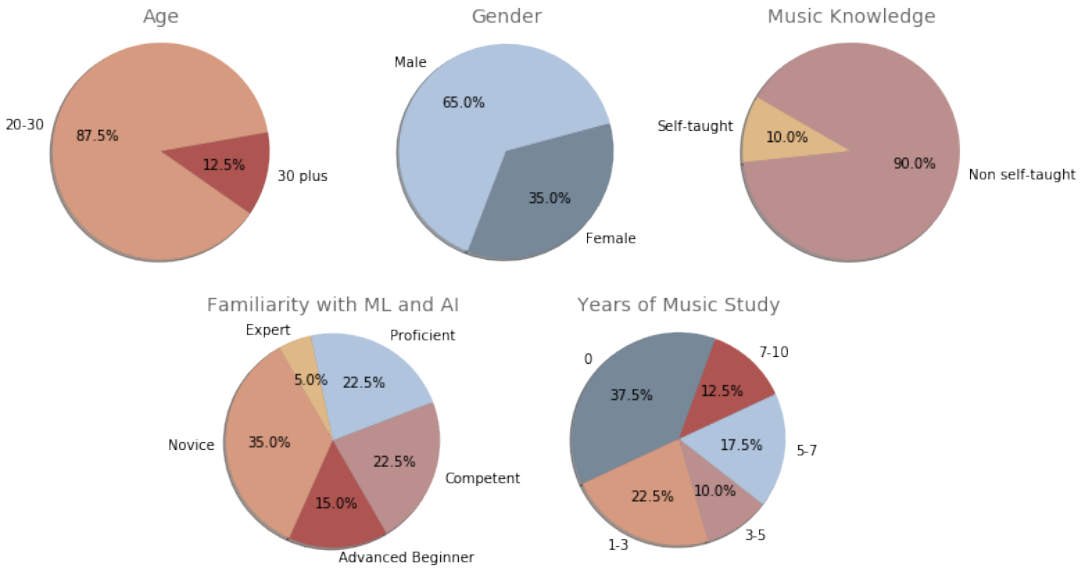


Figure 5.6.2: User Study Demographics

The participants in our survey are 40 subjects, mainly recruited through social circles. Each

[10] https://salu133445.github.io/pypianoroll/
[11] https://wiki.archlinux.org/title/Timidity++

subject evaluates 2 pairs of music samples, resulting in a total of 80 comparisons between our generative framework and the reference model. Identical instructions and stimuli are given to every evaluator involved in the procedure. The participants are also informed that some of the music examples "might be" real and some might be generated by machine, although in this case all samples are actually automatically produced.

The respective demographic analysis is graphically illustrated in Figure 5.6.2 in the form of pie charts. As can be observed, we have recruited a sufficient number of qualified listeners, characterized by adequate diversity regarding various, not necessarily musical, aspects. More specifically, the subjects' music knowledge level as well as their degree of familiarity with Machine Learning and Artificial Intelligence follow an approximately uniform distribution, including amateurs who lack relevant background, experts in the respective field and even self-taught musicians. According to [213], our user study satisfies the majority of requirements concerning the design of a proper listening test and therefore can lead to valid, reliable and replicable scientific evidence.

5.6.2 Subjective Results & Discussion

The results of our subjective testing are graphically illustrated in the barplot of Figure 5.6.3.

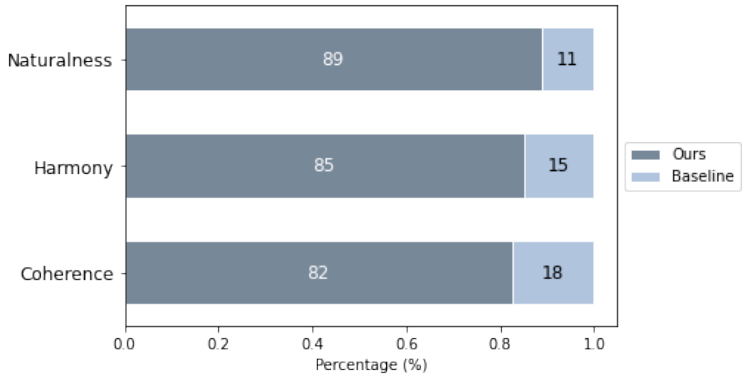


Figure 5.6.3: Results of Subjective Evaluation

As can be easily observed, our developed music generation system outperforms MuseGAN with respect to all the examined musical aspects. This fact indicates that our proposed parameterized architecture, which is based on a shared-private design for both the Generator and the Discriminator modules and enables us to emphasize on rhythmic attributes, undoubtedly contributes to the Naturalness and the Coherence of the generated musical pieces. Since a huge majority of human-composed songs follow an almost evident beat motif, which is also capable of creating a sense of cohesion and connectivity among the various parts of a musical entity, we confirm that rhythm is actually the key for both the aforementioned properties. In addition, it is also subjectively proven that our customization practice, which allows the experimentation over various generative configurations, such as the employed one, induces an improvement of tonal characteristics and texture elements that determine the overall harmony in a musical composition.

Chapter 6

Conditional Generation

This chapter aims at providing a complete overview of our proposed GAN-based framework for the task of Conditional Generation. In particular, section 6.1 introduces the main characteristics of the generation problem that we attempt to tackle. Section 6.2 includes a detailed description of the system architecture, the implementation of the various structural components, as well as the their respective training mechanism. Section 6.3 elaborates on the utilized form of data representation, the dataset and the required preprocessing steps. Section 6.4 focuses on the employed experimental protocol, while section 6.5 engages on a thorough analysis of the results produced using the aforementioned setup. Lastly, section 6.6 presents our user study related to this task and discusses its subjective findings.

6.1 Task Description

In order to expand our research into the undoubtedly vast field of Automatic Music Synthesis beyond the task of Unconditional Generation, which is considered a fundamental practice for composing novel musical pieces from scratch, we further focus on a different generation approach that involves some kind of conditions, typically in the form of prior knowledge about the produced musical pieces or supplementary information from the human user. More specifically, as graphically illustrated in Figure 6.1.1, the creation of novel music samples, under the framework of the **Conditional Generation** mechanism, is based on additional human-provided data of varying modalities, such as lyrics [233], lead sheets, chord progressions, primary melodic lines, tags and even video clips [234].

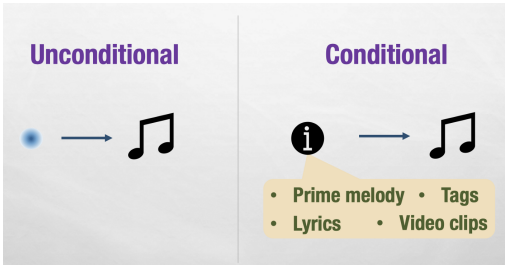


Figure 6.1.1: Unconditional vs Conditional Generation [227]

In our developed GAN-based generative system, this procedure is modeled as follows:

- One of the included tracks, derived from the ground-truth distribution of human-composed music samples, is provided to the network as conditional information. Therefore, this track is called *conditional*.
- The model learns to generate the four remaining tracks, which are considered as the accompaniment of the conditional one in terms of rhythmic and harmonic support.

The aforementioned task entails a lot of challenges, since it involves capturing the underlying structure of the conditional instrumental piece, discovering the intrinsic relations between the main theme and the accompaniment and creating the counterparts and the secondary melodic lines according to the selected accompaniment figure. Moreover, it constitutes a method for reconstructing and reconceptualizing existing musical pieces with new compositional techniques, such as the introduction of novel thematic material into the accompaniment tracks or their reharmonization in terms of chord progression and orchestration. Thus, it enables us to explore also the research area of Music Arrangement, as a different generation approach in the field of AI music.

6.2 Model

Our proposed model for automatically generating polyphonic musical segments that accompany human-composed parts of user-specific tracks, is conceptually established on our Unconditional Generation mechanism, which is extensively presented in the previous chapter 5, and basically developed from scratch. The respective modifications and also extensions required for the adjustment of our original framework to the examined generative approach in the context of human-AI cooperation will be meticulously discussed in the following subsections.

6.2.1 Architecture

The core architecture of our proposed system generally retains its convolutional GAN-based infrastructure from the Unconditional task, which is analytically described in the respective section 5.2.1 of chapter 5. However, the inclusion of conditions into the generation process naturally entails the alteration of some of the included modules from a structural perspective and also the incorporation of additional networks as well. To this end, the specific details will be examined for each component individually, as follows:

Generator

Under the context of the Conditional approach, the Generator module is responsible for producing novel accompaniment instances, by mapping an input vector to the output target data space of musical representations, via consecutive upsampling convolutional operations. As graphically demonstrated in the diagram of Figure 6.2.1, it structurally preserves the shared-private design of the Unconditional Generator, which has been thoroughly presented in section 5.2.1 of chapter 5), since it consists of a *shared* network G_s, followed by M *private* subnetworks G_p^i $(i = 1, \dots M)$. However in this case M equals to 4, as it signifies the number of accompaniment tracks. In order to ensure that the Generator takes into consideration the conditional track for the creation of the remaining ones, we properly modify its shared part so that it receives 2 distinct inputs:

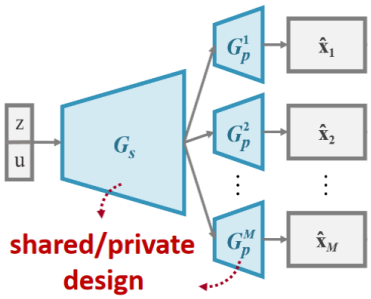


Figure 6.2.1: Conditional Generator

- A random noise vector $\mathbf{z}$, sampled from a prior distribution $p_{\mathbf{z}}$.
- An embedding of the conditional track into the latent space of noise, denoted as $\mathbf{u}$.

The two equally-sized vectors are concatenated and provided to the network in a unified form.

"Global" Discriminator

As mentioned in previous chapters of this thesis, the Discriminator module evaluates the input data instances $\mathbf{x}$ in terms of authenticity, by predicting the label of their respective origin class. Under the framework of our proposed generative system for music of

symbolic format, this process is performed via successive operations of typical convolution. According to the respective analysis in section 5.2.1 of chapter 5, our Unconditional Discriminator, which is developed for the homonym task, assesses 5-track pianorolls as a whole. To this end, in order to acquire a general critic capable of measuring the fitness of the accompaniment parts for the corresponding conditional track, we incorporate the aforementioned existing Discriminator unit into the architecture of our conditional generation model and referred to it as "Global". As graphically illustrated in the diagram of Figure 6.2.2, its design from a structural perspective remains unaltered. In particular, it consists of M private subnetworks D_p^i $(i = 1, \ldots, M)$, with each one corresponding to a specific track included in the music setting, followed by a shared network D_s. In this case M equals to 5, since our "Global Discriminator" receives all tracks as input in order to judge if they can collectively form a real musical composition.

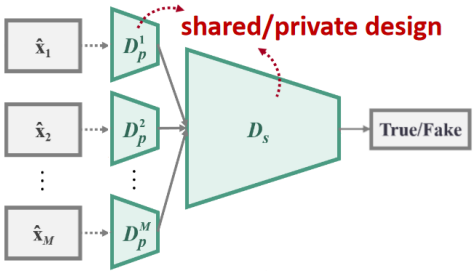


Figure 6.2.2: Conditional Discriminator

"Local" Discriminator

We further expand our original framework for the previously examined task of Unconditional Generation, via the inclusion of a second Discriminator, called "Local". As the name suggests, this Discriminator module is responsible for evaluating all the other tracks jointly, except for the conditional one, in order to provide feedback over the quality of the accompaniment as an independent musical piece. Structurally, it follows the shared-private design of the "Global" Discriminator, which is graphically demonstrated in the diagram of Figure 6.2.2. The only difference is that in this case only 4 private subnetworks D_p are involved, with each one corresponding to a specific accompaniment track.

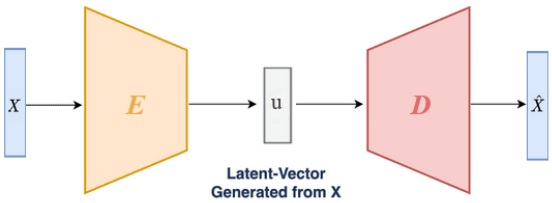


Figure 6.2.3: Encoder and Decoder

Encoder

Apart from the typical components of a GAN framework, our proposed system for automatically generating accompaniments of man-made musical pieces also includes an additional Encoder module. As graphically illustrated in the diagram of Figure 6.2.3, the Encoder network E receives as input the representation vector of a specific conditional track $\mathbf{x}$ and produces an embedding $\mathbf{u}$ in the latent space of the noise distribution. The resulting encoding is then fed into the shared part of the Generator as conditional information.

Decoder

In order to be able to experiment over the training mode of our Encoder module, we incorporate into the architecture of our proposed generative system the corresponding Decoder network, which is schematically displayed in the diagram of Figure 6.2.3 too. As the name suggests, the Decoder D acts as an interpreter of the produced encoding $\mathbf{u}$ in terms of decompressing this hidden representation into a vector $\hat{\mathbf{x}}$ in the original data space. In this way, it attempts to reconstruct the initial conditional track, based on the latent attributes of its low-order representation.

6.2.2 Implementation

Similar to the case of Unconditional Generation, which is thoroughly discussed in the previous chapter of this thesis, all structural modules included in our developed conditional framework are designed as deep Convolutional Neural Networks [2, 19] and implemented using the open source Machine Learning framework PyTorch. We also follow the same customization practice with the previously examined task, since, according to the produced results, has been proven extremely valuable regarding the experimentation over different generative configurations and the comparison of their effect on the music quality. The involved parameters are presented in Table 6.1 along with their respective notation.

s	number of samples
l	latent dimension
t	number of tracks
r	bar resolution
p	number of pitches
m	number of measures
$o\ (= m \cdot r)$	number of total timesteps
t'	number of accompaniment tracks
b	beat resolution
i	lowest pitch

Table 6.1: Parameter Notation

Tables 6.2 and 6.3 demonstrate the parametric network architectures for the shared Generator G_s and the private Generator G_p accordingly. As can be seen, both conditional units successively augment the dimensions of the respective input vector via transposed convolutional operations, which are initially applied along the time axis and afterwards along the pitch axis. Following the reference implementation of [2], a Batch Normalization layer (BN) is added before each non-linear activation function.

Tables 6.4 and 6.5 display the customized network configurations for the structural components of our "Global Discriminator", while Tables 6.6 and 6.7 provide the equivalent overview of the "Local Discriminator". It can be easily affirmed that all the aforementioned Discriminator modules act in reverse mode, compared to the Generator ones, in terms of gradually compressing the spatial dimensions of the corresponding input vector, first along the pitch axis and then along the time one, via the utilization of typical convolutional layers. In this case, the Layer Normalization method is applied before the non-linearity.

Input: $\mathbf{z} \in \mathbb{R}^{s \times l}$, $\mathbf{u} \in \mathbb{R}^{s \times l}$					
concatenate input vectors along horizontal axis: $\mathbf{z} \parallel \mathbf{u} \in \mathbb{R}^{s \times 2l}$					
reshape to $s \times 2l \times 1 \times 1 \times 1$					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
transconv	$2 \cdot l$	$m \times 1 \times 1$	$(m, 1, 1)$	Batch	ReLU
transconv	l	$1 \times r/2 \times 1$	$(1, 1, 1)$	Batch	ReLU
transconv	$l/2$	$1 \times 1 \times p/4$	$(1, 1, 1)$	Batch	ReLU
transconv	$l/4$	$1 \times 1 \times (p/4 + 1)$	$(1, 1, 1)$	Batch	ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times l/4 \times m \times r/2 \times p/2}$					

Table 6.2: Shared Generator G_s

Input: $\mathbf{x} \in \mathbb{R}^{s \times l/4 \times m \times r/2 \times p/2}$					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
transconv	$l/8$	$1 \times (r/2 + 1) \times 1$	$(1, 1, 1)$	Batch	ReLU
transconv	1	$1 \times 1 \times (p/2 + 1)$	$(1, 1, 1)$	Batch	ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times m \times r \times p}$ (stack along track axis for final vector)					

Table 6.3: Private Generator G_p

Input: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times o \times p}$ (reshape to $s \times 1 \times m \times r \times p$)					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
conv	$l/8$	$1 \times 1 \times (p/2 + 1)$	$(1, 1, 1)$	Layer	Leaky ReLU
conv	$l/8$	$1 \times (r/2 + 1) \times 1$	$(1, 1, 1)$	Layer	Leaky ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times l/8 \times m \times r/2 \times p/2}$ (stack along track axis for next layer)					

Table 6.4: Global Private Discriminator D_p

Input: $\mathbf{x} \in \mathbb{R}^{s \times tl/8 \times m \times r/2 \times p/2}$					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
conv	$l/2$	$1 \times 1 \times (p/4 + 1)$	$(1, 1, 1)$	Layer	Leaky ReLU
conv	$l/2$	$1 \times 1 \times p/4$	$(1, 1, 1)$	Layer	Leaky ReLU
conv	l	$1 \times r/2 \times 1$	$(1, r/2, 1)$	Layer	Leaky ReLU
conv	l	$(r/2 + 1) \times 1 \times 1$	$(1, 1, 1)$	Layer	Leaky ReLU
conv	$2 \cdot l$	$r/2 \times 1 \times 1$	$(1, 1, 1)$	Layer	Leaky ReLU
dense	1	(reshape to $s \times 2 \cdot l$ before)			
Output: $\mathbf{y} \in \mathbb{R}$					

Table 6.5: Global Shared Discriminator D_s

Input: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times o \times p}$ (reshape to $s \times 1 \times m \times r \times p$)					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
conv	$l/8$	$1 \times 1 \times (p/2 + 1)$	$(1, 1, 1)$	Layer	Leaky ReLU
conv	$l/8$	$1 \times (r/2 + 1) \times 1$	$(1, 1, 1)$	Layer	Leaky ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times l/8 \times m \times r/2 \times p/2}$ (stack along track axis for next layer)					

Table 6.6: Local Private Discriminator D_p

Input: $\mathbf{x} \in \mathbb{R}^{s \times t'l/8 \times m \times r/2 \times p/2}$					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
conv	$l/2$	$1 \times 1 \times (p/4+1)$	$(1,1,1)$	Layer	Leaky ReLU
conv	$l/2$	$1 \times 1 \times p/4$	$(1,1,1)$	Layer	Leaky ReLU
conv	l	$1 \times r/2 \times 1$	$(1,r/2,1)$	Layer	Leaky ReLU
conv	l	$(r/2+1) \times 1 \times 1$	$(1,1,1)$	Layer	Leaky ReLU
conv	$2 \cdot l$	$r/2 \times 1 \times 1$	$(1,1,1)$	Layer	Leaky ReLU
dense	1	(reshape to $s \times 2 \cdot l$ before)			
Output: $\mathbf{y} \in \mathbb{R}$					

Table 6.7: Local Shared Discriminator D_s

Tables 6.8 and 6.9 demonstrate the internal structure of our employed Encoder and Decoder modules with respect to the involved generative parameters. As can be seen, the Encoder transforms the initial input vector into a low-order representation in a latent space, by performing a gradual dimensionality reduction through consecutive convolutional layers. On the other hand, the Decoder mirrors the architecture of the corresponding Encoder, in terms of performing an upsampling process that successively expands the dimensions of the produced encoding unto the target data space of conditional tracks. This is accomplished by the utilization of transposed convolutional layers, which are considered as the near-complements of the typical convolutional ones.

Input: $\mathbf{x} \in \mathbb{R}^{s \times o \times p}$ (reshape to $s \times 1 \times m \times r \times p$)					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
conv	$l/8$	$1 \times 1 \times (p/2+1)$	$(1,1,1)$	Batch	Leaky ReLU
conv	$l/4$	$1 \times (r/2+1) \times 1$	$(1,1,1)$	Batch	Leaky ReLU
conv	$l/2$	$1 \times 1 \times (p/4+1)$	$(1,1,1)$	Batch	Leaky ReLU
conv	l	$1 \times 1 \times p/4$	$(1,1,1)$	Batch	Leaky ReLU
conv	$2 \cdot l$	$1 \times r/2 \times 1$	$(1,1,1)$	Batch	Leaky ReLU
conv	l	$m \times 1 \times 1$	$(m,1,1)$	Batch	Leaky ReLU
Output: $\mathbf{u} \in \mathbb{R}^{s \times l}$					

Table 6.8: Encoder E

Input: $\mathbf{u} \in \mathbb{R}^{s \times l}$ (reshape to $s \times l \times 1 \times 1 \times 1$)					
Layer Type	Number of Filters	Kernel Size	Stride	Normalization	Activation
transconv	$2 \cdot l$	$m \times 1 \times 1$	$(m,1,1)$	Batch	ReLU
transconv	l	$1 \times r/2 \times 1$	$(1,1,1)$	Batch	ReLU
transconv	$l/2$	$1 \times 1 \times p/4$	$(1,1,1)$	Batch	ReLU
transconv	$l/4$	$1 \times 1 \times (p/4+1)$	$(1,1,1)$	Batch	ReLU
transconv	$l/8$	$1 \times (r/2+1) \times 1$	$(1,1,1)$	Batch	ReLU
transconv	1	$1 \times 1 \times (p/2+1)$	$(1,1,1)$	Batch	ReLU
Output: $\mathbf{x} \in \mathbb{R}^{s \times 1 \times m \times r \times p}$ (reshape to $s \times o \times p$)					

Table 6.9: Decoder D

6.2.3 Training Process

Similar to the case of Unconditional Generation, the learning mechanism of our proposed framework for creation of novel multi-track accompaniments is based on the individual training procedures of the included network components. In particular:

- **Global Discriminator**
  Since this module retains its infrastructure from the previously examined task, its corresponding training process remains unaltered. Thus, according to the respective analysis in section 5.2.3 of chapter 5, our Global Discriminator learns to distinguish the real data from the fake samples, created by the Generator, using a properly selected cost function that quantifies the misclassification errors over both positive and negative training examples. As mentioned before, in this case, the examined music samples, derived from either the ground-truth or the Generator distribution, include the conditional track along with the accompaniment ones in a unified form.

- **Local Discriminator**
  Naturally, our introduced Local Discriminator follows the training procedure of the Global one. However, the main difference is that this module evaluates in terms of authenticity only the accompaniment parts, without the conditional track.

- **Generator**
  Similar to the unconditional case, the Generator learns to create novel realistic data instances, based on the predictions of the Discriminator over the class of its produced candidates. Therefore, under the framework of our proposed system, this module discovers underlying properties of the ground-truth accompaniment distribution indirectly via the feedback of the employed Discriminator scheme. More specifically, if both Discriminators are included in the model architecture, the loss of the Generator is computed as the mean value of the corresponding output probabilities over fake accompaniments and 5-track samples accordingly. Otherwise, only the predictions of the involved Discriminator are utilized.

- **Encoder**
  We experiment with two distinct practices regarding the training mode of our Encoder network:

  - *1-phase training:* In this case, the Encoder is trained jointly with the GAN system. More specifically, it follows the individual learning process of the Generator, which is previously described in detail, since these two structural units contribute to the creation of novel music samples collectively.

  - *2-phase training:* As the name suggests, in this case the training procedure of our proposed framework for Conditional Generation is divided into two definite parts. The first one involves the Encoder, while the other one corresponds to the GAN components. In particular, our Encoder module is initially pretrained along with the respective Decoder as a unified AutoEncoder system, using the MSE loss between the initial and the reconstructed conditional tracks and the Kullback–Leibler divergence, which represents the statistical distance between the Standard Normal distribution of white noise $\mathcal{N}(0, \mathbb{I})$ and the one modeling the

latent space of the produced embeddings. In this way, we ensure that the latent encodings follow the distribution of the input random noise. During this process, our incorporated Early-Stopping and Checkpointing mechanism, which employs as monitoring metric the mean sum of the aforementioned loss values calculated at the validation stage of each training step, indicates the version of the Encoder that will be utilized unchanged in the upcoming training of the GAN system.

Following our primary implementation for the task of Unconditional Generation, the overall training of our GAN-based model proceeds in consecutive interchanges between k steps of optimizing the included Discriminators and one step of optimizing the Generator. Algorithms 4 and 5 summarize all the aforementioned training details in a pseudocode format. As regards the utilized notation, the index t denotes the conditional track, while the index a symbolizes the accompaniment. Furthermore p_t indicates the distribution that models the latent space of the conditional embeddings, p_d represents the ground-truth distribution of the real music samples and $p_{\mathbf{z}}$ denotes the prior distribution from which the input random noise vector $\mathbf{z}$ is sampled. Lastly, $U[0, 1]$ symbolizes the continuous uniform distribution, defined by the given boundaries.

Algorithm 4: AutoEncoder Training

Input:

- Batch size m
- Adam hyperparameters α, β_1, β_2
- Number of total training steps N

for *number of training iterations* **do**
  for *i=1, . . . , m* **do**
    Sample real data $\boldsymbol{x} = (\boldsymbol{x}_t, \boldsymbol{x}_a) \sim p_d$
    $\boldsymbol{u} \leftarrow E_\gamma(\boldsymbol{x}_t)$
    $\hat{\boldsymbol{x}}_t \leftarrow D_\eta(\boldsymbol{u})$
    $L^{(i)} \leftarrow \texttt{MSE}\left(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t\right) + \texttt{KL}\left(p_t \parallel \mathcal{N}(0, 1)\right)$
  endfor
  $\gamma \leftarrow Adam\left(\nabla_\gamma \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \gamma, \alpha, \beta_1, \beta_2\right)$
  $\eta \leftarrow Adam\left(\nabla_\eta \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \eta, \alpha, \beta_1, \beta_2\right)$
endfor

Algorithm 5: Mini-batch Training Algorithm for Conditional GANs with Gradient Penalty

Input:

- Gradient penalty coefficient λ
- Number of Discriminator iterations per Generator iteration k
- Batch size m
- Adam hyperparameters α, β_1, β_2
- Number of total training steps N

for *number of training iterations* **do**

  `Update Global Discriminator` D_w `by ascending its stochastic gradient.`

  for k *steps* **do**

    for i*=1, . . . ,* m **do**

      `Sample`: real data $\boldsymbol{x} = (\boldsymbol{x}_t, \boldsymbol{x}_a) \sim p_d$
      latent variable $\boldsymbol{z} \sim p_{\boldsymbol{z}}$
      random number $\epsilon \sim U[0, 1]$

      $\boldsymbol{u} \leftarrow E_\gamma(\boldsymbol{x}_t)$

      $\bar{\boldsymbol{x}}_a \leftarrow G_\theta(\boldsymbol{z}, \boldsymbol{u})$

      $\bar{\boldsymbol{x}} \leftarrow (\boldsymbol{x}_t, \bar{\boldsymbol{x}}_a)$

      $\hat{\boldsymbol{x}} \leftarrow \epsilon \cdot \boldsymbol{x} + (1 - \epsilon) \cdot \bar{\boldsymbol{x}}$

      $L^{(i)} \leftarrow D_w(\bar{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$

    endfor

    $w \leftarrow Adam\left(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$

  endfor

  `Update Local Discriminator` D_ϕ `by ascending its stochastic gradient.`

  if *Local* **then**

    for k *steps* **do**

      for i*=1, . . . ,* m **do**

        `Sample`: real data $\boldsymbol{x} = (\boldsymbol{x}_t, \boldsymbol{x}_a) \sim p_d$
        latent variable $\boldsymbol{z} \sim p_{\boldsymbol{z}}$
        random number $\epsilon \sim U[0, 1]$

        $\boldsymbol{u} \leftarrow E_\gamma(\boldsymbol{x}_t)$

        $\bar{\boldsymbol{x}}_a \leftarrow G_\theta(\boldsymbol{z}, \boldsymbol{u})$

        $\hat{\boldsymbol{x}}_a \leftarrow \epsilon \cdot \boldsymbol{x}_a + (1 - \epsilon) \cdot \bar{\boldsymbol{x}}_a$

        $L^{(i)} \leftarrow D_\phi(\bar{\boldsymbol{x}}_a) - D_\phi(\boldsymbol{x}_a) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}_a} D_\phi(\hat{\boldsymbol{x}}_a)\|_2 - 1)^2$

      endfor

    endfor

    $\phi \leftarrow Adam\left(\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, \phi, \alpha, \beta_1, \beta_2\right)$

  end

```
31
32    Update Generator G_θ by descending its stochastic gradient.
33    for i=1, ..., m do
34        Sample: real data x = (x_t, x_a) ~ p_d
35                latent variable z ~ p_z
36        u ← E_γ(x_t)
37        x̄_a ← G_θ(z, u)
38        x̄ ← (x_t, x̄_a)
39        if Local then
40            L^(i) ← −(D_w(x̄) + D_φ(x̄_a)) / 2
41        else
42            L^(i) ← −D_w(x̄)
43        end
44    endfor
45    θ ← Adam( ∇_θ (1/m) Σ_{i=1}^{m} L^(i), θ, α, β_1, β_2 )
46    Update Encoder E_γ by descending its stochastic gradient.
47    if 1-phase then
48        for i=1, ..., m do
49            Sample: real data x = (x_t, x_a) ~ p_d
50                    latent variable z ~ p_z
51            u ← E_γ(x_t)
52            x̄_a ← G_θ(z, u)
53            x̄ ← (x_t, x̄_a)
54            if Local then
55                L^(i) ← −(D_w(x̄) + D_φ(x̄_a)) / 2
56            else
57                L^(i) ← −D_w(x̄)
58            end
59        endfor
60        γ ← Adam( ∇_γ (1/m) Σ_{i=1}^{m} L^(i), γ, α, β_1, β_2 )
61    end
62 endfor
```

6.3 Data

6.3.1 Data Representation

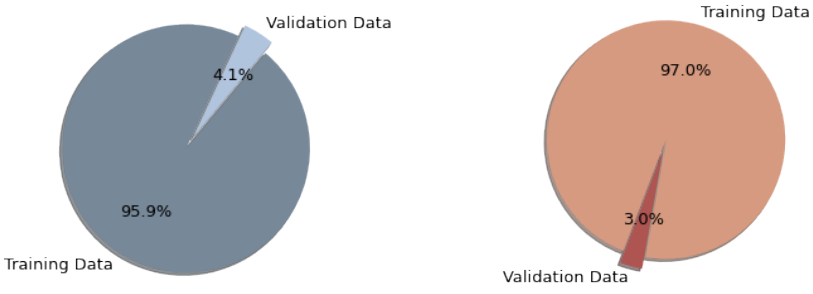
Following our primary approach to the research problem of automatic synthesis from scratch, we employ the *multi-track pianoroll* format for the representation of music samples, processed under the framework of our proposed conditional generation system. According to the detailed analysis in section 4.3.1 of chapter 4, a piano-roll is a binary-valued scoresheet-like matrix, which indicates the presence or absence of notes over different timesteps and consequently a multi-track piano-roll is defined as a set of piano-rolls corresponding to different musical instruments.

6.3.2 Dataset

The pianoroll dataset used for the training process of our conditional model is derived from the Lakh MIDI Dataset (LMD) [16] after the preprocessing procedure, which is thoroughly described in section 4.3.3 of chapter 4 and graphically illustrated in Figure 4.3.3. More specifically, we employ the *LPD-5-cleansed* version[1], which contains only those pianorolls with the higher matching confidence score to MSD entries [17], a "Rock" tag and 4/4 time.

6.3.3 Data Preprocessing

In order to acquire the final set of training examples, we apply the preprocessing steps that are explicitly presented in section 5.3.3 of chapter 5. In this way, the pianorolls included in the cleansed version of the utilized dataset are segmented into musical phrases of proper format and size, according to the input configuration. We further extend our data preparation routine via the inclusion of an additional criterion regarding the required note density in the conditional tracks. More specifically, we discard candidate samples with insufficient number of note instances in a specific user-defined track, which represents the conditional information. Finally, we perform a splitting of our processed dataset into training and validation subsets, using the ratios that are graphically illustrated in the pie-charts of Figure 6.3.1 for two distinct conditional instruments. The overall training duration is around 8 hours with a GeForce RTX 2080 Ti GPU.



(a) Piano (b) Guitar

Figure 6.3.1: Split Ratios

[1] https://salu133445.github.io/lakh-pianoroll-dataset/dataset.html

6.4 Experimental Protocol

6.4.1 Experimental Setup

In order to evaluate our proposed conditional system and thoroughly examine its effectiveness over the creation of aesthetic multi-track polyphonic accompaniments in a human-AI cooperative framework, we conduct a group of experiments that enable us to investigate various aspects of the generation process. The corresponding details as well as the produced results will be extensively presented in the following sections of this chapter.

		AutoEncoder	Local Discriminator
Piano	P_{00}	-	-
	P_{01}	-	✓
	P_{10}	✓	-
	P_{11}	✓	✓
Guitar	G_{00}	-	-
	G_{01}	-	✓
	G_{10}	✓	-
	G_{11}	✓	✓

Table 6.10: Conditional Models

Before proceeding further, we consider it useful to define the model variants that will be employed on the proximate analysis. As demonstrated in Table 6.10, these models mainly differ in terms of:

- **Included structural components**
  - *Only Global Discriminator ("-" at second column)*
  - *Both Global and Local Discriminators ("✓ " at second column)*
- **Training mode of Encoder**
  - *1-phase training ("-" at first column)*
  - *2-phase training ("✓ " at first column)*
- **Conditional Instrument**
  - *Piano*
  - *Guitar*

It is also worth mentioning that we employ the experiment configuration C_2, as presented in Table 5.6, for all the aforementioned conditional models, since the results of the previously examined task indicated that this particular combination of generation and training parameters can adequately capture rhythmic patterns, tonal characteristics and texture elements of man-made pieces and hence lead to artificial music of high quality.

6.4.2 Objective Metrics

In the context of objective assessment, we utilize our musical metric system consisting of the following 8 quantitative indices:

- **Empty Bars (EB):** ratio of empty bars included in the examined track (in %)
- **Qualified Notes (QN):** ratio of "qualified" notes (in %)
- **Drum Pattern (DP):** ratio of notes in beat patterns of 4/4 rhythm (in %)
- **Tonal Distance (TD):** measures the harmonicity between a pair of musical tracks
- **Used Pitch Classes (UPC):** mean number of pitch classes used per bar (from 0 to 12)
- **Used Pitches (UP):** mean number of unique pitches used per bar, including all octaves in the predefined range
- **Scale Ratio (SR):** ratio of notes in the given music scale
- **Polyphonic Rate (PR):** ratio of polyphonic timesteps

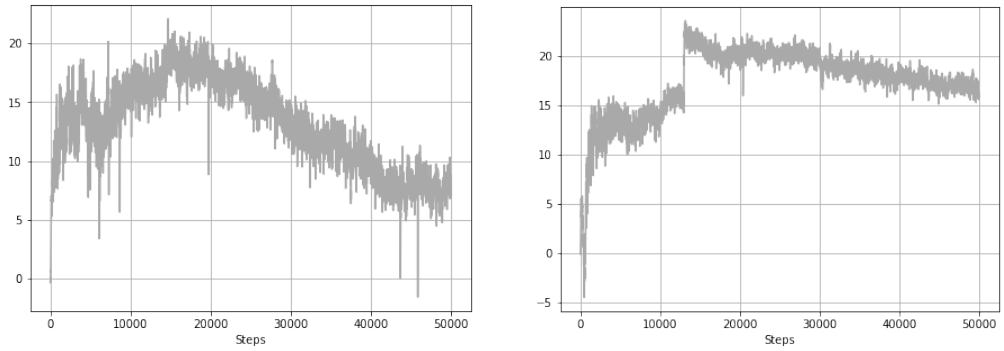
It is worth mentioning that all the aforementioned metrics can be computed for both real and generated samples.

Unlike music synthesis from scratch, the framework of Conditional Generation inherently provides two different accompaniments for each conditional track, one derived from the ground-truth distribution of human-composed musical pieces and the other created by our Generator module. Thus, we consider it useful to introduce an additional objective index that measures the corresponding distance between the two versions and in this way examine if our proposed model tends to musically imitate the real samples and also confine its creativity. For this purpose, we employ the Mean Squared Error, calculated between the original and the generated accompaniments of the provided conditional pieces.

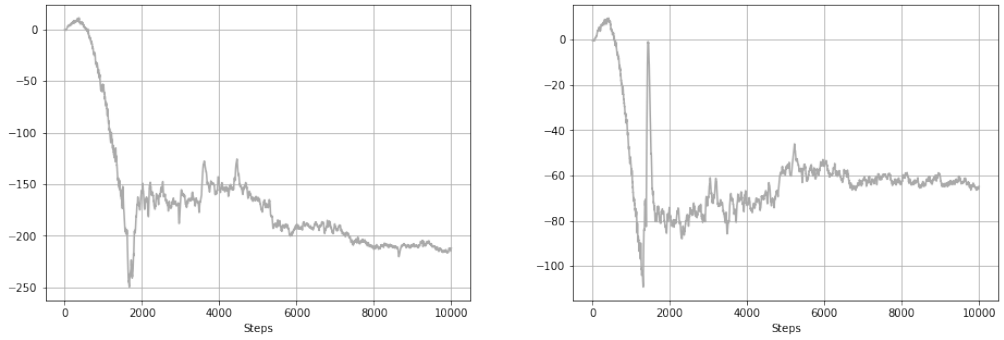
6.5 Results

6.5.1 Analysis of Training Process

As discussed in previous chapters of this thesis, the ideal training practice for a GAN-based system still remains an open problem, since it is particularly hard to explicitly identify the convergence state from a computational perspective, especially when additional components, such as a second Discriminator or an Encoder module, are also included in the architecture. Therefore, a closer inspection of the learning mechanism of our proposed model for the task of Conditional Generation, as well as a thorough examination of the individual behaviour of the involved networks, are considered essential. For this purpose, we employ the 4 model variants that use the Piano track as conditional information[2], since this constitutes the default case under our proposed music generation framework and also our baseline project [2].



(a) P_{00} (b) P_{01}



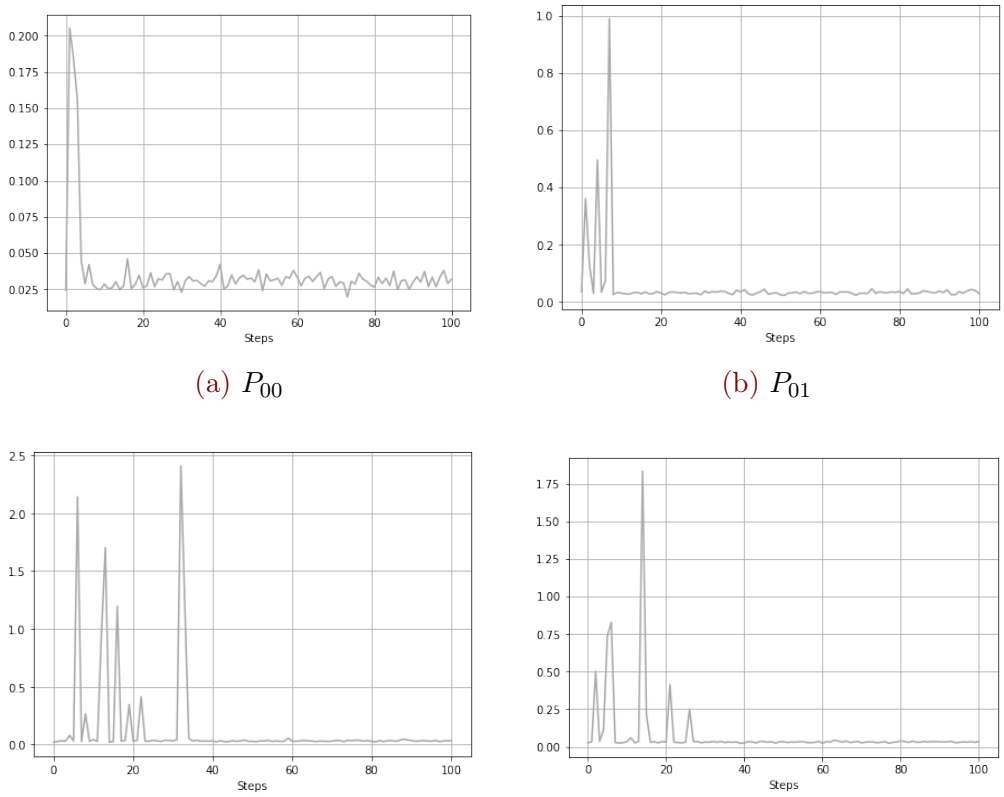
(c) P_{10} (d) P_{11}

Figure 6.5.1: Generator losses for the various piano-based conditional GANs

Figure 6.5.1 demonstrates the training loss of the Conditional Generator, formulated as function of training steps, for the examined model variants. In particular, the gray curves indicate the normalized loss values, which are smoothed via a moving average operation. As regards the figure layout, the plots in the first row correspond to the 1-phase training method, while the ones in the second row to the utilization of the AutoEncoder Pretraining (2-phase). Accordingly, the loss values resulting from the inclusion of only one Discriminator in the

[2]The Guitar models exhibit similar behaviour.

system architecture are depicted in the first column, while the effect of both Discriminators in the learning process of the Accompaniment Generator is graphically represented in the plots of the second column.



(a) P_{00} (b) P_{01}

(c) P_{10} (d) P_{11}

Figure 6.5.2: MSE losses between real and generated pianorolls for the various piano-based conditional GANs

First of all, it can be easily affirmed that the 2-phase learning practice requires significantly less training steps as compared to the joint fashion (1-phase). This observation is naturally expected, since in the first case the Encoder network is already trained to create embeddings of the conditional tracks following a consistent mapping between the input and the output latent space. This fact implies that its produced encodings throughout the complete training of the GAN part approximately follow the normal distribution. As a consequence, the Generator's training behaviour is stabilized rather early. Furthermore, in this way the GAN learning procedure is computationally lightened and also accelerated, as the Encoder state remains unaltered during this phase. As can be also observed, the Generator losses that are depicted in plots of the same column and hence correspond to identical GAN architectures, follow a similar trend, which is scaled according to the required number of training steps. More specifically, in the case of both Discriminators, a slight increase of the corresponding loss function is detected after the initial steep spikes, suggesting that the Generator probably can't handle the combined output predictions of its opponents and produces easily distinguishable candidates. However, it seems that after a particular training point, the Generator loss begins

to steadily decrease, indicating that the produced music samples become more plausible, until saturation.

Figure 6.5.2 displays the log of the MSE loss calculated between real and generated pianorolls during the training procedure of the 4 examined model variants for the first 100 steps. The plot layout is similar to the previous one, concerning the arrangement of the conditional models at rows and columns. As can be seen, the employment of the 1-phase training mode (first row) leads to faster convergence of the MSE function. This observation implies that, under this learning framework, our proposed generative system tends to imitate the real accompaniments by reproducing their tonal, rhythmic or texture features in quite early stages of training. However, some small oscillations are observed after the saturation point, indicating that a margin for creativity and differentiation from the original human-composed musical pieces is preserved. On the other hand, the utilization of the 2-phase learning practice (second row) also results in convergence of the MSE loss (≈ 0.025), but requires larger number of training steps. Similar to the previous case, this fact suggests that the generated and the corresponding real music segments become almost identical as training proceeds. From the perspective of our objective evaluation system, the similarity of the examined pianorolls is considered beneficial, but at the same time it limits the generation capabilities of our proposed model towards novel alternative accompaniments.

6.5.2 Qualitative Inspection

Figure 6.5.3 illustrates two multi-track polyphonic pianorolls, which correspond to the same Piano track as conditional information and represent one 4-bar musical phrase. The first one is a real human-composed music sample, derived from the ground-truth dataset, while the other one is created by our proposed GAN-based framework during the process of Inference. In particular, we employ the model variant P_{11} and utilize the Generator version of the last training step, as discussed in section 5.5.2 of chapter 5.

It is worth pointing out the following qualitative observations:

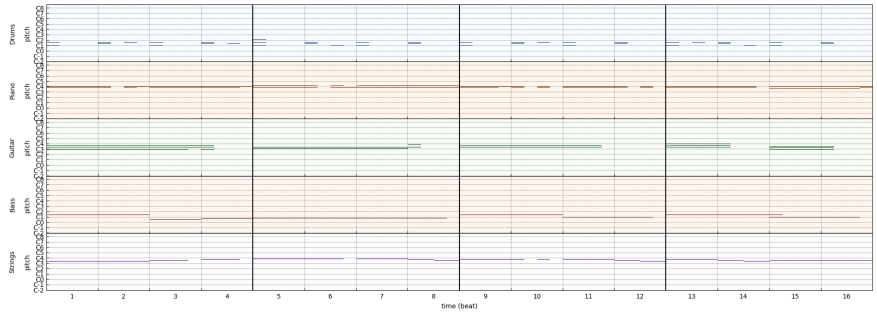
- The Drum track of the generated pianoroll follows an almost steady rhythmic pattern, which resembles the original to a significantly large degree. Notes in 8- or 16-beat motifs are evident, indicating the presence of 4/4 rhythm, while the strong beats of each bar can also be easily detected. This apparent rhythmicity is probably the result of our parameterized implementation, which enables us to emphasize on texture elements of this type during the creation of novel musical pieces.

- The Bass track is principally monophonic, playing a single melodic line. As can be seen, our proposed generative system is capable of capturing musical features of the real bass line, such as the duration of the notes and also the utilized pitch range, which is approximately from $C0$ to $C2$.

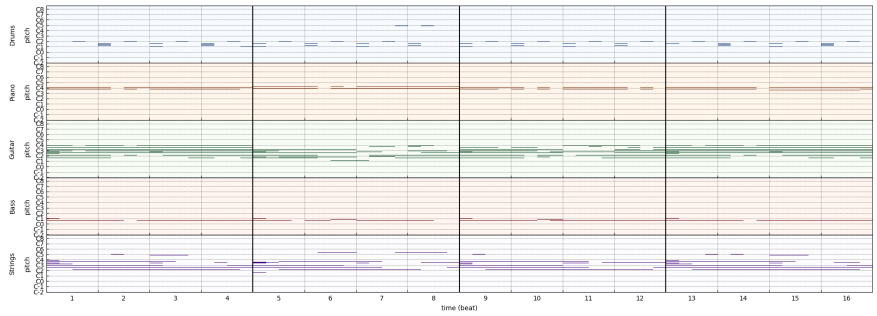
- The Strings and the Guitar, which are the mainly polyphonic tracks in the overall composition and usually tend to play the chord-like parts, are quite noisy compared to the original ones, both in terms of note density and number of utilized pitch classes. This fact may lead to weaker harmonic relations among the included instruments, which can negatively affect the acoustic result. On the other hand, it may also contribute to

a more musically interesting alternative.

- The melodic tracks (i.e. all except Drums) of the fake pianoroll usually play in the same music scale, as indicated by the overlapping pitches. This fact implies strong harmonic relations among the included instruments, that can contribute to an acoustically pleasant result.



(a) Real sample



(b) Fake sample

Figure 6.5.3: Qualitative analysis of generated pianorolls

6.5.3 Objective Evaluation

In order to examine if the structural modifications and extensions of our proposed GAN-based framework, as well as the different training modes can actually improve the quality of artificial music and lead to innovative results in the research field of Automatic Accompaniment Generation, we proceed in a comprehensive comparison of the model variants presented in Table 6.10, using our objective metric system. To this end, we follow the assessment practice of our baseline project. In particular, for each conditional model involved in the evaluation, we create, using the respective Generator of the last training step, 20000 bars organised in 4-bar musical phrases. Afterwards we apply our objective metrics on the produced musical segments and calculate the respective mean values.

Piano case

Table 6.11 summarizes the intra-track evaluation results of the 4 Piano models, i.e those variants that employ piano tracks as conditions, only for objective metrics shared between our implementation and the baseline project accordingly. As mentioned before, in this case values closer to the ones representing musical properties of the ground-truth distribution correspond to the desirable generative behavior and hence are considered better. At first glance, we can easily detect the absence of a particular model variation capable of improving all the employed objective musical metrics. However, a closer inspection indicates that the utilization of the 2-phase training mode (model P_{10}) benefits some musical characteristics of the produced samples, such as the intended note density as measured by `EB` metric, the contained beat patterns in 4/4 rhythm represented by `DP` value and also the proper number of used pitch classes (`UPC`). Nevertheless, it seems to negatively affect the form of the Bass track in terms of making it more sparse than the original (`EB` around 18 %). On the other hand, the inclusion of the Local Discriminator in the system architecture under both training practices has a positive impact on all the examined quantitative indicators of musical attributes. This result confirms that the extra feedback over the authenticity of the accompaniment parts actually helps the Generator to produce more plausible candidates.

		EB (%)					**UPC**				**QN** (%)				**DP** (%)
		B	D	G	P	S	B	G	P	S	B	G	P	S	D
training data	*baseline*	8.06	8.06	19.4	24.8	10.1	1.71	3.08	3.28	3.38	90.0	81.9	88.4	89.6	88.6
	ours	1.6	1.0	5.0	5.6	3.7	2.47	4.09	4.19	4.5	91.6	85.6	90.0	89.7	82.9
Ours	P_{00}	0.6	0.0	2.2	-	2.4	2.71	3.93	-	**4.33**	51.4	56.5	-	58.9	**86.1**
	P_{01}	0.2	0.0	1.8	-	1.5	2.57	**4.09**	-	4.76	**58.2**	56.1	-	**61.7**	86.3
	P_{10}	17.4	**0.2**	**3.0**	-	**4.4**	1.68	3.9	-	4.3	50.7	49.2	-	55.1	87.0
	P_{11}	**1.6**	0.0	0.7	-	0.9	**2.56**	4.19	-	5.16	54.8	**56.6**	-	51.0	86.2
Baseline	jamming	4.60	3.47	13.3	-	3.44	2.05	3.79	-	4.23	73.9	58.8	-	62.3	91.6
	composer	0.65	20.7	1.97	-	1.49	2.51	4.57	-	5.10	53.5	48.4	-	59.0	84.5
	hybrid	2.09	4.53	10.3	-	4.05	2.86	4.43	-	4.32	43.3	55.6	-	67.1	71.8

Table 6.11: Results of inter-track evaluation of the Piano-based models

For the purpose of completeness, we also present in Table 6.11 the respective results of the intra-track evaluation for the 3 multitrack interdependency models included in MuseGAN and thoroughly examined in section 4.2.2 of chapter 4. However, it can be easily observed that there is a substantial divergence between the training data statistics, as measured under the two involved frameworks. This difference may result from the randomized rule that determines the collection of training samples from the songs included in the employed pianoroll dataset, as explained in section 5.3.3. Thus, the two generative systems cannot be exactly compared in the context of intra-track evaluation metrics. Nevertheless, despite the statistical bias, all the calculated values are in the same order of magnitude, which indicates that our metric implementation actually provides a meaningful interpretation of the produced music.

The results of the inter-track evaluation, which is represented by `TD` metric, are demonstrated in Table 6.12 for all models involved in the comparative analysis of piano conditions. As referred to earlier, in this case smaller values correspond to smaller Euclidean distances in the interior space of a 6D polytope and hence imply stronger harmonic relations between the examined pairs of tracks. It can be easily affirmed that the model P_{10}, which is derived from the the inclusion of only the Global Discriminator under the 2-phase training practice,

		TD					
		B-G	B-S	B-P	G-S	G-P	S-P
Ours	P_{00}	0.82	0.83	0.88	0.87	0.95	**0.94**
	P_{01}	0.79	0.81	0.85	**0.85**	**0.94**	**0.94**
	P_{10}	**0.74**	**0.73**	**0.81**	0.94	1.02	1.01
	P_{11}	0.83	0.92	0.97	0.99	1.12	1.17
Baseline	jamming	1.51	1.53	1.50	1.04	0.95	1.00
	composer	1.41	1.36	1.40	0.96	1.01	0.95
	hybrid	1.39	1.36	1.38	0.96	0.94	0.95

Table 6.12: Results of inter-track evaluation of the Piano-based models

presents the best performance among our proposed variants (bold values) and also surpasses all the baseline architectures (coloured cells) in terms of harmonicity between a melody-like track, such as the Bass and a chord-like track, such as the Piano, Guitar and Strings. This observation comes in agreement with the role of our Global Discriminator as a general critic that evaluates all tracks collectively and hence assesses the harmonic quality between melody and chords. On the other hand, the model P_{01}, which results from the inclusion of both Discriminators in the system architecture under the 1-phase training mode, outperforms all the involved variants in terms of harmonicity between two chord-like tracks. This fact confirms the beneficial contribution of our Local Discriminator, which provides an additional feedback over the quality of the accompaniment tracks, to the generation efficiency of our proposed framework.

		UP				SR (%)				PR (%)				
		B	G	P	S	B	G	P	S	B	D	G	P	S
training data		2.71	5.68	5.85	6.71	75.9	74.4	74.1	72.8	1.1	15.2	55.7	61.8	62.3
Ours	P_{00}	2.94	5.79	-	**6.28**	81.7	**75.8**	-	77.1	**1.2**	13.3	40.6	-	44.2
	P_{01}	2.94	**5.77**	-	7.17	**77.1**	76.3	-	75.6	1.5	**15.2**	**48.7**	-	**59.9**
	P_{10}	1.74	5.05	-	6.07	82.2	80.6	-	79.0	0.2	10.1	22.2	-	30.2
	P_{11}	**2.84**	5.43	-	7.3	80.7	77.6	-	**72.3**	1.9	9.7	38.2	-	56.3

Table 6.13: Additional results on the evaluation of the Piano-based models

Lastly, the supplementary Table 6.13 displays the additional intra-track evaluation results, only for our 4 model variants that employ Piano as conditional track. As can be seen, all conditional models accomplish to significantly approximate the statistics of the real distribution in the context of tonal characteristics as quantified by UP and SR metrics and other texture elements, such as PR. Especially in the case of PR, the most distinguishable performance corresponds to model P_{01}, indicating that the incorporation of a Local Discriminator in the architecture of our proposed system helps the Generator to uncover more properties of the human-composed music, including the proper polyphony of the chord-like tracks and accordingly the monophony of the melodic ones, such as the Bass, which actually tends to play a single melodic line (PR around 1.5%).

Guitar case

Table 6.14 provides an overview of the objective evaluation results for the 4 model variants that employ Guitar as conditional instrument and are presented in Table 6.10 along with their respective notation. As before, bold values denote greater proximity to the ground-

	EB (%)					**UPC**				**QN** (%)				**UP**			
	B	D	G	P	S	B	G	P	S	B	G	P	S	B	G	P	S
training data	1.8	0.9	4.3	5.2	3.6	2.47	4.21	4.14	4.49	91.8	87.5	91.6	90.5	2.7	5.85	5.84	6.75
G_{00}	0.8	0.0	-	2.1	1.8	**2.51**	-	5.04	**4.59**	**62.5**	-	49.3	**60.3**	**2.77**	-	7.31	6.91
G_{01}	0.0	0.0	-	3.1	0.0	3.05	-	4.31	5.28	57.6	-	52.4	59.6	3.36	-	6.18	7.69
G_{10}	**1.6**	0.0	-	1.8	**3.5**	2.35	-	**4.28**	4.01	50.2	-	**59.5**	58.6	2.59	-	**6.13**	5.88
G_{11}	0.4	**0.2**	-	**3.3**	0.6	2.32	-	4.62	4.66	55.6	-	47.8	57.9	2.46	-	6.4	**6.68**

	TD						**SR** (%)				**PR** (%)					**DP** (%)
	B-G	B-S	B-P	G-S	G-P	S-P	B	G	P	S	B	D	G	P	S	D
training data	0.71	0.72	0.7	0.69	0.66	0.66	75.4	73.5	73.4	73.1	0.8	15.5	59.7	61.0	62.6	85.0
G_{00}	**0.83**	0.85	0.9	0.96	1.01	0.98	84.7	-	80.9	**77.0**	1.1	10.9	-	53.9	53.4	87.1
G_{01}	0.87	0.87	**0.83**	**0.93**	**0.92**	**0.86**	86.7	-	83.6	83.9	2.8	**14.9**	-	**55.3**	**60.8**	**86.0**
G_{10}	0.84	**0.84**	0.84	**0.93**	0.95	0.89	82.0	-	79.8	85.4	**0.7**	6.0	-	37.5	44.0	91.7
G_{11}	0.89	0.87	0.88	1.06	1.09	0.97	**78.0**	-	**76.9**	80.5	**0.9**	9.7	-	42.1	54.4	83.7

Table 6.14: Results of objective evaluation of the Guitar-based models

truth distribution, expect for the inter-track TD metric, where smaller ones are considered better. Similar to the Piano case, we can easily detect the absence of a particular model variation capable of improving all the employed objective musical metrics simultaneously. However, a closer inspection indicates that the utilization of the Local Discriminator under the 1-phase training practice (model G_{01}) results in stronger harmonic interrelations among the included tracks, as suggested by the corresponding TD values. Moreover, the additional evaluation feedback over the authenticity of the accompaniment parts as an independent musical composition benefits the rhythmic attributes of the generated musical phrases, as indicated by the high DP rate and also other texture elements, such as the desired PR for each track individually. On the other hand, the utilisation of the 2-phase training mode over both architectural approaches has a positive impact on the note density of the generated samples, as measured by EB metric, the number of contained "qualified" notes, particularly in the Piano track and tonal characteristics, such as UP, UPC and SR, especially for the chord-like instruments. However, in case of QN rates there is still space for further improvement, since there is a substantial distance between the resulting values and the original ones.

6.6 User Study

6.6.1 Experimental Setup

As discussed in section 5.6 of chapter 5, in the context of subjective assessment, we follow our baseline project [2] and conduct a user study in the form of listening test. Our survey is divided into two parts, with each one corresponding to a specific music generation task implemented by our proposed framework. In particular, the section of Conditional Generation, which constitutes the principal topic of this chapter, aims at a comprehensive comparison among the variants of our developed system presented in Table 6.10 along with their respective notation.

The conditional part of our questionnaire follows the A/B testing format, which has been also applied in the section of Generation from scratch. However, in this case, the two alternative choices A and B represent two possible accompaniments, generated by different models or even derived from the ground-truth distribution of human-composed musical pieces, for the same conditional track. Thus, each testing group of samples consists of 3 distinct audio clips. The first one is the conditional track and the others contain matching accompaniments for it. Similar to the unconditional case, the evaluator is required to choose from each listening pair the accompaniment version that best fits the conditional track in terms of:

- **Musical Naturalness:** Could the musical segment be composed by human?
- **Harmonic Consistency:** Are the sounds produced by different instruments in musical consonance? Is the result acoustically pleasant?
- **Musical Coherence:** Are the various musical phrases associated someway through time?

As regards the implementation details, all the involved models are trained using the same set of training examples for each conditional instrument. The produced samples and also the real musical segments are then transformed into the proper auditory format, according to the diagram of Figure 5.6.1 in section 5.6. The duration of each resulting audio clip, regardless of its derivation model, is pruned to approximately 12 seconds, a time period that corresponds to one 4-bar musical phrase under the framework of our proposed system and the employed multitrack pianoroll representation. The conditional track of each listening case and consequentially the respective accompaniment versions are randomly selected from pools of 32 audio clips. Both the order of the testing groups and also the sample order within each group are randomized for each user.

Table 6.15 demonstrates the examined comparisons among the involved conditional models. As can be easily observed, musical segments derived from each variant are placed in juxtaposition with fake data instances generated by two other models of the same conditional instrument and also the real accompaniments that correspond to their common conditional track (denoted with the letter R). If we consider symmetric comparisons as equivalent cases, this results in a total of 16 unique testing pairs.

As mentioned in section 5.6 of chapter 5, the participants in our survey are 40 subjects, mainly recruited via social circles. In this case, each subject evaluates 18 listening group of samples, where 16 of them correspond to the unique model pairs and the remaining 2 are randomly

Piano	P_{00}	P_{01}	P_{10}	R
	P_{01}	P_{00}	P_{11}	R
	P_{10}	P_{00}	P_{11}	R
	P_{11}	P_{01}	P_{10}	R
Guitar	G_{00}	G_{01}	G_{10}	R
	G_{01}	G_{00}	G_{11}	R
	G_{10}	G_{00}	G_{11}	R
	G_{11}	G_{01}	G_{10}	R

Table 6.15: Conditional Comparisons

selected. The overall occurrences for each examined couple are graphically illustrated in the diagrams of Figure 6.6.1. As can be seen, we have recruited a sufficient number of qualified listeners and also distributed the involved comparisons in a almost uniform fashion among them. Thus, we can conclude that our user study can provide statistically significant results and hence lead to valid, reliable and replicable scientific evidence.

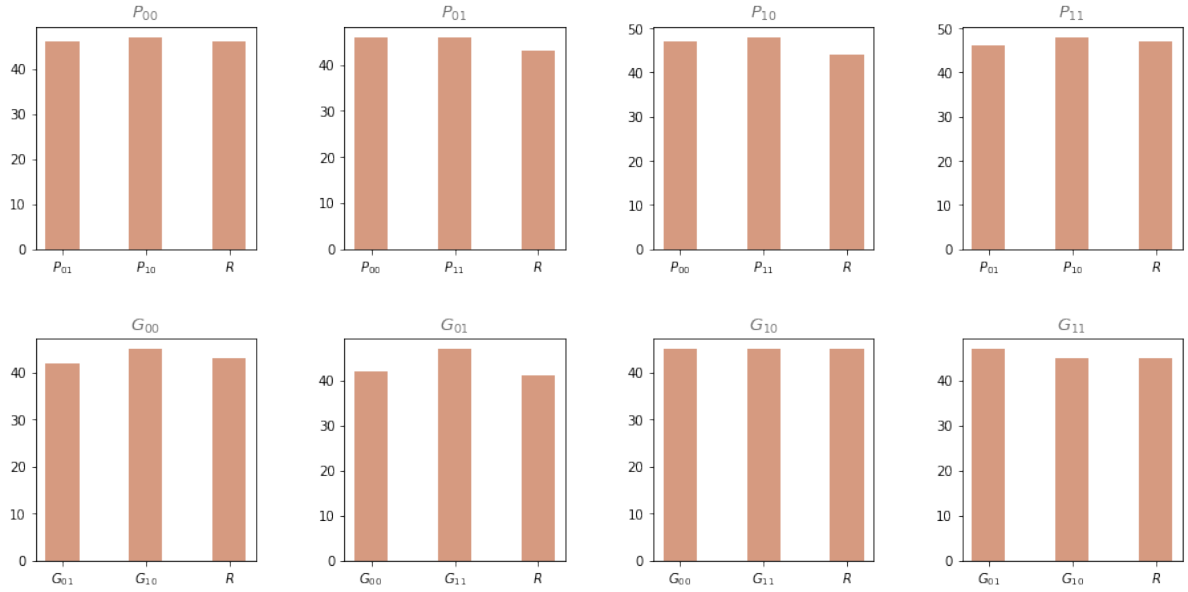


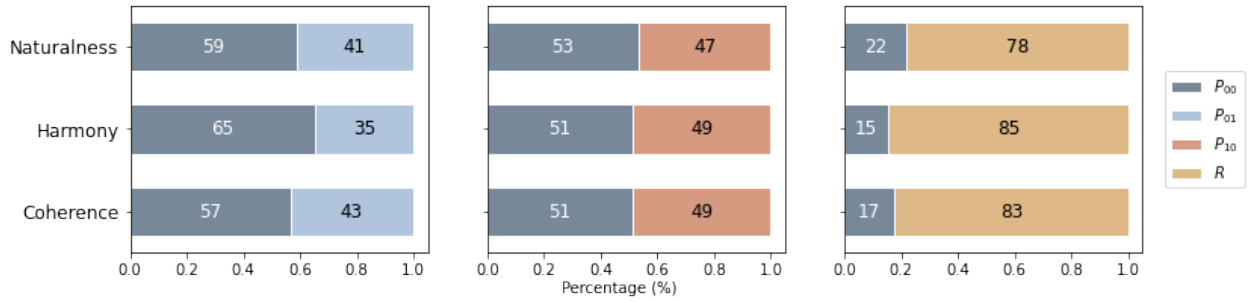
Figure 6.6.1: Total number of comparisons for each model pair

6.6.2 Subjective Results & Discussion

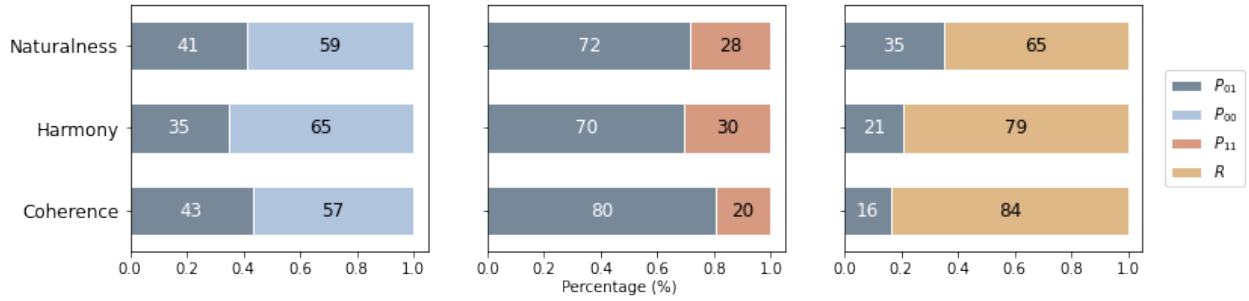
Piano case

The results of our subjective testing for the model variants that employ Piano as conditional track are graphically illustrated in Figure 6.6.2. Each bar-plot represents the evaluators' preferences between the compared models under the examined musical criteria in the form of percentages. As can be seen, in the case of comparison with the real music segments, the majority of fake samples are easily distinguishable, regardless of their derivation model, indicating that AI music is still far from the level of human compositions in terms of Naturalness, Harmony and Coherence. The highest favor proportion against human performance corresponds to model P_{01} for the first question in our survey and is equal to 35%. This fact

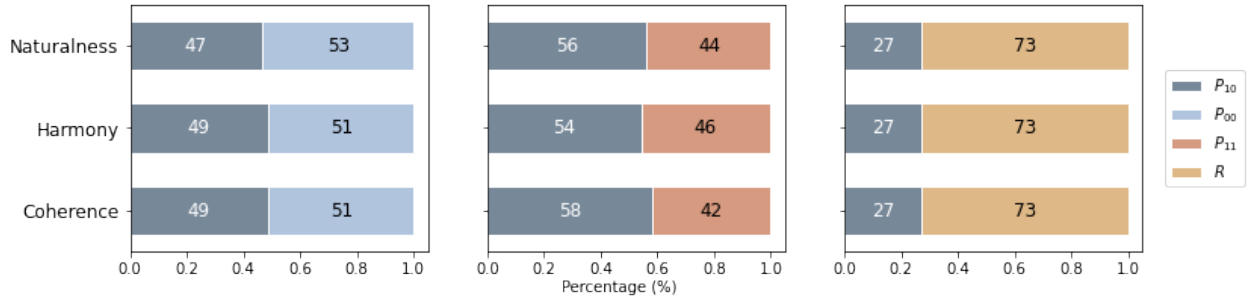
suggests that the additional evaluation feedback provided by the Local Discriminator over the accompaniment parts, can actually help the Generator to create samples that sound more natural to the human subjects.



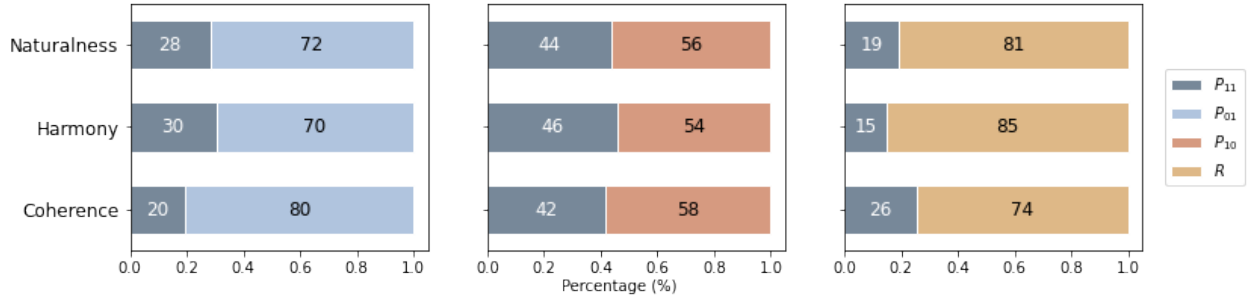
(a) P_{00}



(b) P_{01}



(c) P_{10}



(d) P_{11}

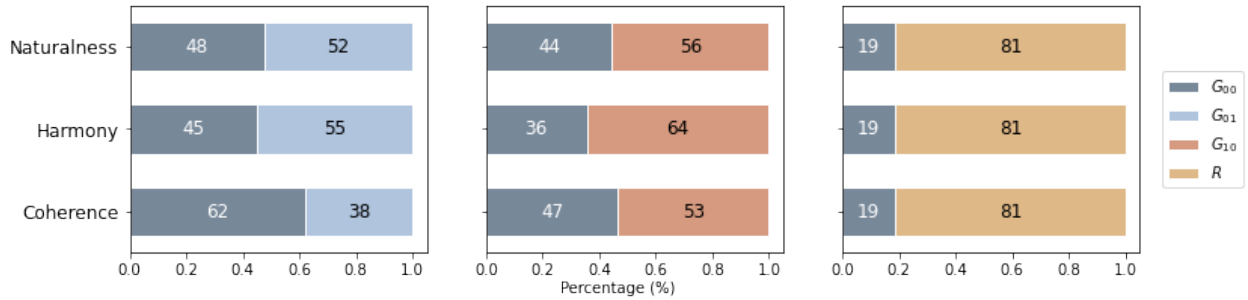
Figure 6.6.2: Results of Subjective Evaluation for Piano

As regards the comparison among our developed varying frameworks, we observe that model

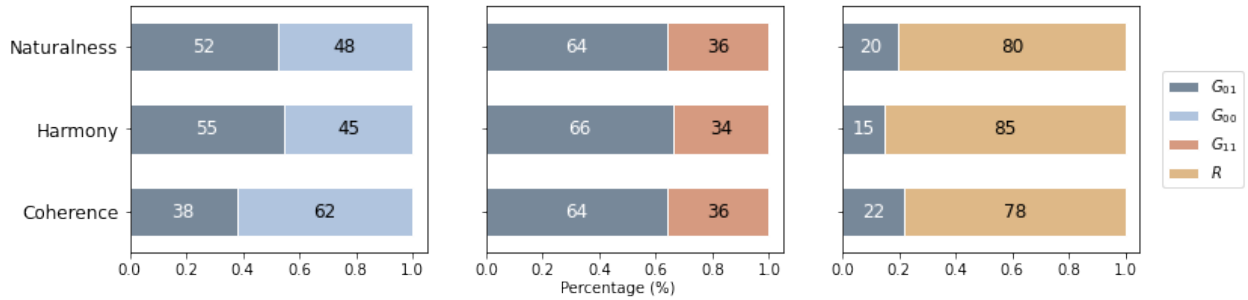
P_{01} significantly outperforms P_{11} with respect to all the examined musical aspects, especially Coherence. This result comes in agreement with the outcomes of the objective assessment, as it suggests that the most suitable training practice for the system architecture of both Discriminators is the 1-phase mode. Moreover, it can be easily affirmed that variant P_{10} also surpasses P_{11} in terms of all 3 criteria, indicating that the proper structural design for the 2-phase training mode includes only the Global Discriminator, as pointed out in the objective analysis too. Lastly, we can further observe a slight preference for P_{00} as compared to P_{01} and P_{10}. This fact suggests that the basic implementation of our proposed generative system can also lead to the creation of aesthetic musical pieces.

Guitar case

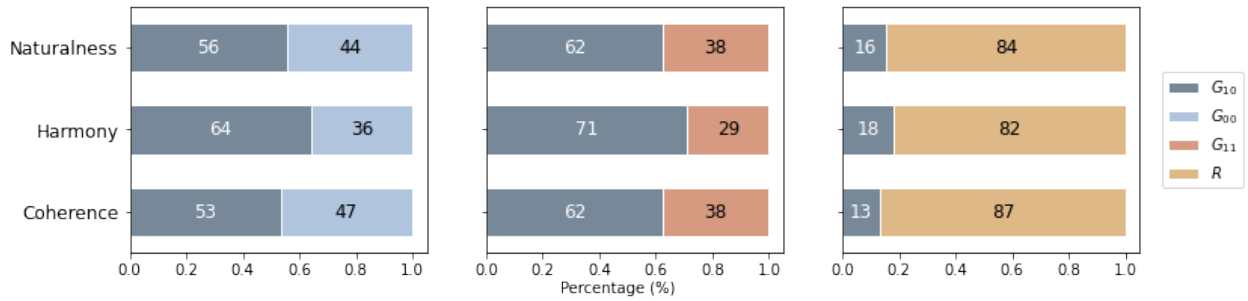
The results of our subjective testing for the model variants that employ Guitar as conditional instrument are graphically illustrated in Figure 6.6.3.



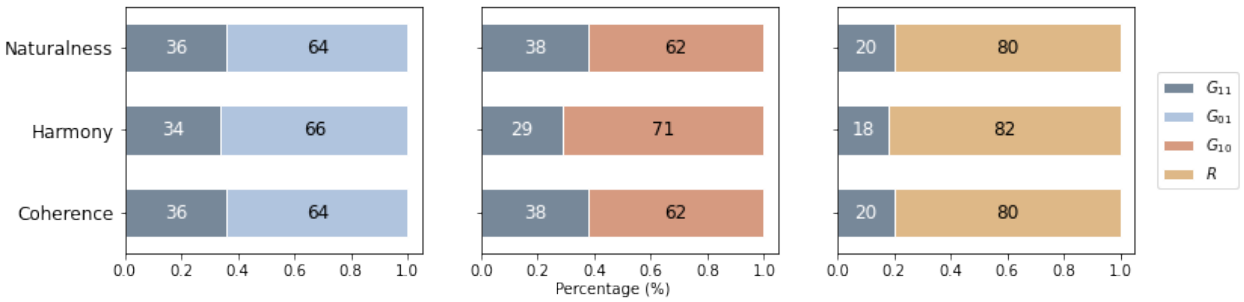
(a) G_{00}



(b) G_{01}



(c) G_{10}



(d) G_{11}

Figure 6.6.3: Results of Subjective Evaluation for Guitar

Similar to the previous conditional case, the majority of fake samples are easily distinguishable from the real ones in terms of all the examined musical aspects. As can be seen, all the favor proportions are in the range of 13 to 20%, probably indicating that Guitar tracks provide less conditional information than the Piano ones. This is mostly due to the fact that Guitar in Rock songs usually plays the chords, while Piano typically includes some melodic patterns as well.

As regards the comparison among the 4 generative frameworks, it can be easily affirmed that the corresponding outcomes are similar to the ones of the Piano case. More specifically, we observe that model G_{10} outperforms both G_{00} and G_{11} with respect to all the aforementioned musical criteria. This result suggests that the most effective combination of training practice and architectural design regarding the quality of produced music, is the 2-phase mode applied in a GAN system that comprises only the Global Discriminator. Furthermore, we can see that the variant G_{01} surpasses to a large extent G_{11} in terms of all the examined musical properties. This fact indicates that the most suitable training practice for the architecture of both Discriminators is the 1-phase mode. However, in the case of comparison between G_{01} and G_{00}, where the 1-phase learning method is applied, the utilization of the Global Discriminator only seems to have a beneficial impact on the coherence of the generated musical pieces, as indicated by the preference proportion of 62%.

Chapter 7

Conclusions

7.1 Synopsis

In the past decade, automatic music generation has undoubtedly achieved rapid progress. Compared to traditional methods and former approaches, deep learning has shown its powerful capabilities. However, the generative results still deviate from the real ones in terms of structure and innovation, highlighting the open challenges in the area, such as the modeling of music expressive performance, the incorporation of the emotion, the limited interaction with users and the absence of a unified music evaluation standard. Thus, due to the multidisciplinary nature of this research field, it is sometimes hard to define precise goals and keep track of which tasks can be considered solved by state-of-the-art systems and which instead require further developments.

In the context of this thesis, we begin with a hierarchical review of existing approaches to the examined research problem, emphasizing on recent studies that make use of deep neural networks and machine learning techniques. We advocate that such a detailed and organized analysis could provide means to face many of the open challenges listed above and possibly allow for easier comparison among varying methods in terms of the learning process improvement and hence the creation of more aesthetic human-like music.

Moving on to the experimental part of our research, we designed a convolutional GAN-based generative framework that implements the automatic creation of novel polyphonic musical content in the pianoroll format, under 2 different approaches:

- **Unconditional Generation:** Automatic generation of musical phrases, composed of 5 distinct tracks (*Drums*, *Piano*, *Guitar*, *Bass* and *Strings*), from scratch, i.e. without subjecting to any prior knowledge or supplementary information provided from the human user.
- **Conditional Generation:** Automatic generation of 4-track accompaniments for human-composed track samples that are provided to the model as conditional information.

We made our model even more flexible and structurally adaptable to different generative configurations and practices, by performing a customization of our implementation with respect to a group of various parameters that determine musical attributes and also training features. As regards the learning process, we incorporated into our system auxiliary monitoring mechanisms, based on an additional validation phase, for closer inspection of the individual structural components behavior during training. In the context of assessment, we developed an alternative implementation for the existing musical metrics and further expanded our employed objective evaluation system via the introduction of 3 additional quantitative indicators that emphasize on tonal characteristics and texture attributes of the generated samples per track. We also conducted a qualitative study in the form of listening test across 40 subjects, in order to include human auditory feedback into our analysis.

In the case of *Unconditional Synthesis*, we extensively experimented over multiple generative configurations and examined the effect of the respective modifications on the musical quality of the created pianorolls, investigating at the same time various aspects of the generation mechanism. For this purpose, we applied our proposed objective evaluation system, which consists of 7 intra-track and 1 inter-track metrics. The produced results indicated the absence

of a particular parameter arrangement capable of improving all the employed quantitative indicators simultaneously. This outcome highlighted the lack of a unified objective criterion for the designation of the best model in terms of generation efficiency and produced musical quality. Nevertheless, we were able to derive some interesting conclusions, such as the correlation between the utilized beat resolution and the resulting harmonic relations among the included tracks or the impact of higher values of k (number of training steps per Generator update) on tonal characteristics of the generated samples. As regards the subjective assessment part, we observed that our developed system for music generation from scratch significantly outperforms MuseGAN with respect to 3 examined musical aspects: Musical Naturalness, Harmonic Consistency, Musical Coherence. This fact demonstrates that our proposed parameterized architecture, which is based on a shared-private design for both the Generator and the Discriminator modules and enables us to emphasize on rhythmic attributes, undoubtedly contributes to the creation of novel aesthetic music.

In the case of *Conditional Synthesis*, we experimentally focused on 8 variants of our proposed conditional generative system that differ in terms of the included structural components (Global Discriminator, Global and Local Discriminators), the training algorithm of the Encoder module (1-phase mode, 2-phase-mode) and the type of conditional instrument (Piano, Guitar). We thoroughly examined the impact and the effectiveness of our proposed modifications over the creation of aesthetic multi-track polyphonic accompaniments in a human-AI cooperative framework from objective as well as subjective aspects. The produced results from both assessment methods indicated that the most suitable training practice for the system architecture of both Discriminators is the 1-phase mode, confirming that the extra feedback over the authenticity of the accompaniment parts as an independent musical composition along with a shared learning fashion between the utilized Encoder and the GAN, actually helps the Generator to produce more plausible candidates. On the other hand, we observed that the proper structural design for the 2-phase training mode includes only the Global Discriminator. This fact suggests that the AutoEncoder Pretraining contributes to the improvement of the generated accompaniment quality when the supervisory signals come from a single Discriminator. The aforementioned outcomes refer to both conditional instruments. However, the qualitative study demonstrated that Guitar provides less conditional information than Piano, probably due to the fact that in Rock songs it usually plays the chords, while Piano typically includes some melodic patterns as well. Lastly, the agreement between the results of the objective evaluation and the outcomes of human assessment indicates that our proposed implementation for the employed metrics provides a meaningful interpretation of the produced music from a computational perspective.

7.2 Thoughts on Future Work

As thoroughly discussed in the previous section, the contributions of our research study in the area of Automatic Music Synthesis cover 2 distinct generation tasks, corresponding to different approaches and hence capabilities. Although our developed framework musically and aesthetically still falls behind the level of human musicians, it demonstrates a few desirable properties that pave the way for further investigation in the field. To this end, some interesting potential directions for future research are the following:

- *User Melodies*: model extension to an interactive framework between human and machine that automatically produces accompaniments for user-defined inputs of variable length and track type.
- *Full Song Generation*, not by just concatenating independently produced musical phrases but in a more human-like compositional fashion, related to overall structure and musical coherence.
- *Cross-Modal Generation*: enrichment of conditions with different modalities or supplementary information derived from other sources (e.g. Music + Video, Music + Lyrics, Video + Text).
- *Monitoring Metric*: Implementation of a network that extracts features from pianorolls, so that FID is applicable to symbolic music generation [229, 230].

Bibliography

[1] Briot, J.-P., Hadjeres, G., and Pachet, F.-D. "Deep Learning Techniques for Music Generation – A Survey". In: *arXiv preprint arXiv:1709.01620* (2017). URL: Survey on Music Generation.

[2] Dong, H.-W. et al. "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), New Orleans, LA, USA*. 2018. URL: MuseGAN.

[3] Herremans, D., Chuan, C.-H., and Chew, E. "A Functional Taxonomy of Music Generation Systems". In: *ACM Computing Surveys (CSUR)* 50 (2017), pp. 1–30. URL: Music Generation Systems.

[4] Herremans, D. and Chew, E. "MorpheuS: Generating Structured Music with Constrained Patterns and Tension". In: *arXiv preprint arXiv:1812.04832* (2017). URL: MorpheuS.

[5] Huang, C.-Z. A. et al. "Music Transformer: Generating Music with Long-Term Structure". In: *Proceedings of International Conference on Learning Representations (ICLR), Vancouver, British Columbia, Canada*. 2018. URL: Music Transformer.

[6] Kang, S., Ok, S.-Y., and Kang, Y.-M. "Automatic Music Generation and Machine Learning based Evaluation". In: *Proceedings of International Conference on Multimedia Systems and Signal Processing (ICMSSP), Shanghai, China*. 2012. URL: Automatic Music Generation and Machine Learning based Evaluation.

[7] *Convolutional Neural Networks (CNNs / ConvNets)*. URL: CNN.

[8] Levine, M. W and Shefner, J. M. *Fundamentals of Sensation and Perception*. Oxford University Press, 1991. URL: Visual perception.

[9] *Convolutional Neural Networks*. URL: CNN.

[10] Vint, D. et al. "Automatic Target Recognition for Low Resolution Foliage Penetrating SAR Images using CNNs and GANs". In: *Remote Sensing* 13 (2021), p. 596. URL: GAN.

[11] Bang, D. and Shim, H. "Improved Training of Generative Adversarial Networks using Representative Features". In: *Proceedings of International Conference on Machine Learning (ICML), Stockholm, Sweden*. 2018. URL: Nash equilibrium.

[12] *From Autoencoder to Beta-VAE*. URL: AutoEncoder.

[13] *MuseGAN slides*. URL: MuseGAN tracks.

[14] Saito, M., Matsumoto, E., and Saito, S. "Temporal Generative Adversarial Nets with Singular Value Clipping". In: *Proceedings of International Conference on Computer Vision (ICCV), Venice, Italy*. 2017. URL: Temporal GANs.

[15] Yang, L.-C., Chou, S.-Y., and Yang, Yi-Hsuan. "MIDINET: A Convolutional Generative Adversarial Network for Symbolic-Domain Music Generation". In: *arXiv preprint arXiv:1703.10847* (2017). URL: MIDINET.
[16] *The Lakh MIDI Dataset v0.1*. URL: LMD.
[17] Bertin-Mahieux, T. et al. "The Million Song Dataset". In: *Proceedings of International Conference on World Wide Web (WWW), Lyon, France*. 2012. URL: MSD.
[18] Harte, C., Sandler, M., and Gasser, M. "Detecting Harmonic Change in Musical Audio". In: *Proceedings of International Conference on Audio and Music Computing Multimedia (AMCMM), Santa Barbara, California, USA*. 2006. URL: Detecting harmonic change in musical audio.
[19] Dong, H.-W. and Yang, Y.-H. "Convolutional Generative Adversarial Networks with Binary Neurons for Polyphonic Music Generation". In: *arXiv preprint arXiv:1804.09399* (2018). URL: BMuseGAN.
[20] Gulrajani, I. et al. "Improved Training of Wasserstein GANs". In: *arXiv preprint arXiv:1704.00028* (2017). URL: Improved Training of Wasserstein GANs.
[21] Goodfellow, I. et al. "Generative Adversarial Nets". In: *arXiv preprint arXiv:1406.2661* (2014). URL: GANs.
[22] *LMD Statistics*. URL: LMD.
[23] *Want to Generate your own Music using Deep Learning? Here's a Guide to do just that!* URL: AI artists.
[24] Morley, Iain. *The Prehistory of Music: Human Evolution, Archaeology, and the Origins of Musicality*. Oxford University Press, 2013. URL: Prehistory of Music.
[25] *Mozart's Musical Dice Game*. URL: Dice Game.
[26] Alpern, A. "Techniques for Algorithmic Composition of Music". In: *CiteSeer* 95 (1995), p. 120. URL: Algorithmic Music Composition.
[27] Hiller, A., Lejaren, J., and Isaacson, L. M. "Musical Composition with a High Speed Digital Computer". In: *Audio Engineering Society Convention 9. Audio Engineering Society* 6 (1957), pp. 154–160. URL: Musical composition with digital computer.
[28] *MusiComp*. URL: MusiComp.
[29] Xenakis, I. *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, 1992. URL: Formalized music.
[30] *Illiac Suite*. URL: Illiac Suite.
[31] *John Cage and his Musical Chess Pieces: Part One*. URL: Chess Music.
[32] Schwartz, E. and Godfrey, D. *Music Since 1945: Issues, Materials, and Literature*. Schirmer Books: New York, 1993. URL: Music Since 1945: Issues, Materials, and Literature.
[33] *How Is Artificial Intelligence Transforming The Music Industry*. URL: AI Music.
[34] *Music and the Brain*. URL: Music and Brain.
[35] *Music: The Last Thing We Forget*. URL: Music and Memory.
[36] *Orchestra Stock Vectors, Clipart and Illustrations*. URL: Orchestra.
[37] *Rock Band Figure*. URL: Band.
[38] *Chords and Arpeggios*. URL: Texture elements.
[39] *What is Artificial Intelligence?* URL: Artificial Intelligence.
[40] *Supervised and Unsupervised Learning [Differences & Examples]*. URL: Supervised and Unsupervised Learning.
[41] *Supervised Machine Learning*. URL: Supervised Learning.

[42] *Linear Regression t-test: Formula, Example*. URL: Linear regression.
[43] *Regression Analysis in Machine Learning*. URL: Regression Analysis.
[44] *Email Spam Classifier using Naive Bayes*. URL: Email Spam Classifier.
[45] *Top 6 Machine Learning Algorithms for Classification*. URL: Classification Algorithms.
[46] *Underfitting Vs Just right Vs Overfitting in Machine Learning*. URL: Underfitting vs Overfitting.
[47] *Unsupervised Learning*. URL: Unsupervised Learning.
[48] *Unsupervised Machine Learning*. URL: Unsupervised Machine Learning.
[49] Ezugwu, A. E et al. "Automatic Clustering Algorithms: A Systematic Review and Bibliometric Analysis of Relevant Literature". In: *Neural Computing and Applications* 33 (2021), pp. 6247–6306. URL: Automatic clustering algorithms.
[50] *17 Clustering Algorithms used in Data Science and Mining*. URL: Clustering Algorithms.
[51] *Market Basket Analysis*. URL: Market Basket.
[52] *Reinforcement Learning Applications: A Brief Guide on How to Get Business Value from RL*. URL: Reinforcement Learning.
[53] *More Effective and Efficient Reinforcement Learning*. URL: Reinforcement Learning.
[54] *Artificial Neural Network Tutorial*. URL: Artificial Neuron.
[55] *The Unit that Makes Neural Networks Neural: Perceptron*. URL: Perceptron.
[56] *Perceptron: Explanation, Implementation and a Visual Example*. URL: Perceptron.
[57] *Multi-layer Perceptron in TensorFlow*. URL: Multi-layer Perceptron.
[58] Feng, J. et al. "Reconstruction of Porous Media from Extremely Limited Information using Conditional Generative Adversarial Networks". In: *Physical Review E* 100 (2019), p. 033308. URL: Reconstruction using CGAN.
[59] Rumelhart, D. E, Hinton, G. E, and Williams, R. J. "Learning Representations by Back-Propagating Errors". In: *Nature* 323 (1986), pp. 533–536. URL: Learning representations by back-propagating errors.
[60] Bottou, L. "Online Algorithms and Stochastic Approximations". In: *Online Learning and Neural Networks*. Cambridge University Press, 1998. URL: SGD.
[61] Jais, I. K. M., Ismail, A. R., and Nisa, S. Q. "Adam Optimization Algorithm for Wide and Deep Neural Network". In: *Knowledge Engineering and Data Science* 2 (2019), pp. 41–46. URL: Adam for wide and deep neural network.
[62] Xu, K. et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of International Conference on Machine Learning (ICML), Lille, France*. 2015. URL: Image caption generation.
[63] Gregor, K. et al. "Draw: A Recurrent Neural Network for Image Generation". In: *Proceedings of International Conference on Machine Learning (ICML), Lille, France*. 2015. URL: Draw.
[64] Kingma, D. P and Ba, J. "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980* (2014). URL: Adam.
[65] *Convolutional Neural Networks for Visual Recognition*. URL: CNN for Visual Recognition.
[66] Chauhan, R., Ghanshala, K. K., and Joshi, RC. "Convolutional Neural Network (CNN) for Image Detection and Recognition". In: *Proceedings of International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India*. 2018. URL: Image Recognition.

[67] Van den Oord, A., Dieleman, S., and Schrauwen, B. "Deep Content-based Music Recommendation". In: *Proceedings of International Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, Nevada, USA*. 2013. URL: Recommendation System.

[68] Collobert, R. and Weston, J. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of International Conference on Machine Learning (ICML), Helsinki, Finland*. 2008. URL: NLP.

[69] Avilov, O. et al. "Deep Learning Techniques to Improve Intraoperative Awareness Detection from Electroencephalographic Signals". In: *Proceedings of International Conference of Engineering in Medicine & Biology Society (EMBS), Montreal, Canada*. 2020. URL: Brain-Computer interfaces.

[70] Tsantekidis, A. et al. "Forecasting Stock Prices from the Limit Order Book using Convolutional Neural Networks". In: *Proceedings of International Conference on Business Informatics (CBI), Thessaloniki, Greece*. 2017. URL: Financial Time Series.

[71] *Convolutional Neural Networks, Explained*. URL: Image.

[72] *A Comprehensive Guide to Convolutional Neural Networks*. URL: CNN.

[73] *Convolutional Neural Networks*. URL: Stride.

[74] *What is Transposed Convolutional Layer?* URL: Transposed Convolution.

[75] *Convolutional Neural Networks Cheatsheet*. URL: Pooling.

[76] *Normalization*. URL: Normalization.

[77] Ba, J. L., Kiros, J. R., and Hinton, G. E. "Layer Normalization". In: *arXiv preprint arXiv:1607.06450* (2016). URL: Layer Normalization.

[78] Graves, A. et al. "A Novel Connectionist System for Unconstrained Handwriting Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2008), pp. 855–868. URL: Unconstrained Handwriting Recognition.

[79] Sak, H., Senior, A., and Beaufays, F. "Long Short-Term Memory based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition". In: *arXiv preprint arXiv:1402.1128* (2014). URL: Speech Recognition.

[80] Li, X. and Wu, X. "Constructing Long Short-Term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition". In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, Queensland, Australia*. 2015. URL: Vocabulary Speech Recognition.

[81] *Language Translation with RNNs*. URL: RNN.

[82] Khan, R. et al. "A Deep Neural Framework for Image Caption Generation using GRU-based Attention Mechanism". In: *arXiv preprint arXiv:2203.01594* (2022). URL: Image Captioning.

[83] *Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More*. URL: RNN.

[84] *Recurrent Neural Network*. URL: RNN.

[85] Hochreiter, S. and Schmidhuber, J. "Long Short-Term Memory". In: *Neural Computation* 9 (1997), pp. 1735–1780. URL: LSTM.

[86] Cao, J., Qi, M., and Fiaidhi, J. "A Review of Automatic Music Generation based on Performance RNN". In: *TechRxiv Preprint Server* (2020). URL: Performance RNN.

[87] *Generative Adversarial Networks could be most Powerful Algorithm in AI*. URL: LeCun's statement.

[88] Taif, K., Ugail, H., and Mehmood, I. "Cast Shadow Generation using Generative Adversarial Networks". In: *Proceedings of International Conference on Computational Science (ICCS), Amsterdam, The Netherlands*. 2020. URL: Cast Shadow Generation.

[89] Yu, J. et al. "Generative Image Inpainting with Contextual Attention". In: *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, Utah, USA*. 2018. URL: Image Inpainting.

[90] *The Rise of AI Supermodels*. URL: AI Supermodels.

[91] Schawinski, K. et al. "Generative Adversarial Networks Recover Features in Astrophysical Images of Galaxies beyond the Deconvolution Limit". In: *Monthly Notices of the Royal Astronomical Society: Letters* 467 (2017), pp. L110–L114. URL: Astrophysical Images.

[92] Mustafa, M. et al. "CosmoGAN: Creating High-Fidelity Weak Lensing Convergence Maps using Generative Adversarial Networks". In: *Computational Astrophysics and Cosmology* 6 (2019), pp. 1–13. URL: CosmoGAN.

[93] Oliveira, L. de, Paganini, M., and Nachman, B. "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis". In: *Computing and Software for Big Science* 1 (2017), pp. 1–24. URL: Physics Synthesis.

[94] Paganini, M., Oliveira, L. de, and Nachman, B. "Accelerating Science with Generative Adversarial Networks: an Application to 3D Particle Showers in Multilayer Calorimeters". In: *Physical Review Letters* 120 (2018), p. 042003. URL: 3D Particle Showers.

[95] Wang, X. et al. "EsrGAN: Enhanced Super-Resolution Generative Adversarial Networks". In: *Proceedings of International Conference on Computer Vision (ICCV), Munich, Germany*. 2018. URL: ESRGAN.

[96] Nistal, J. et al. "VQCPC-GAN: Variable-Length Adversarial Audio Synthesis using Vector-Quantized Contrastive Predictive Coding". In: *Proceedings of International Conference on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, New York, USA*. 2021. URL: VQCPC-GAN.

[97] *Deep Understanding of Discriminative and Generative Models in Machine Learning*. URL: Generative and Discriminative Models.

[98] *Understanding Generative Adversarial Networks (GANs)*. URL: Generator.

[99] *Generative Adversarial Networks*. URL: GANs.

[100] *Complete Guide to Generative Adversarial Networks (GANs)*. URL: Discriminator.

[101] Hinton, G. E. "Learning Translation Invariant Recognition in a Massively Parallel Networks". In: *Proceedings of International Conference on Parallel Architectures and Languages Europe (PARLE), Eindhoven, The Netherlands*. 1987. URL: Parallel networks.

[102] Hinton, G. E, Krizhevsky, A., and Wang, S. D. "Transforming Auto-Encoders". In: *Proceedings of International Conference on Artificial Neural Networks (ICANN), Espoo, Finland*. 2011. URL: Transforming auto-encoders.

[103] Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. " O'Reilly Media, Inc.", 2019. URL: Feature Detection.

[104] Liou, C.-Y., Huang, J.-C., and Yang, W.-C. "Modeling Word Perception using the Elman Network". In: *Neurocomputing* 71 (2008), pp. 3150–3157. URL: Modeling word perception.

[105] Liou, C.-Y. et al. "Autoencoder for Words". In: *Neurocomputing* 139 (2014), pp. 84–96. URL: Autoencoder for words.
[106] *The MNIST Database of Handwritten Digits*. URL: MNIST.
[107] Lu, X. et al. "Speech Enhancement based on Deep Denoising Autoencoder". In: *Interspeech* 2013 (2013), pp. 436–440. URL: Speech enhancement.
[108] Gondara, L. "Medical Image Denoising using Convolutional Denoising Autoencoders". In: *Proceedings of International Conference on Data Mining (ICDM), Barcelona, Spain*. 2016. URL: Medical image denoising.
[109] Deng, J. et al. "Sparse Autoencoder-based Feature Transfer Learning for Speech Emotion Recognition". In: *Proceedings of International Conference on Affective Computing and Intelligent Interaction (ACII), Geneva, Switzerland*. 2013. URL: Speech emotion recognition.
[110] Al-Qatf, M. et al. "Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection". In: *IEEE Access* 6 (2018), pp. 52843–52856. URL: Network intrusion detection.
[111] Hennig, J. A., Umakantha, A., and Williamson, R. C. "A Classifying Variational Autoencoder with Application to Polyphonic Music Generation". In: *arXiv preprint arXiv:1711.07050* (2017). URL: Classifying Variational Autoencoder.
[112] Gillick, J., Roberts, A., and Engel, J. "GrooVAE: Generating and Controlling Expressive Drum Performances". In: *Proceedings of International Conference on Machine Learning (ICML), Long Beach, California, USA*. 2019. URL: GrooVAE.
[113] Pratella, D. et al. "A Survey of Autoencoder Algorithms to Pave the Diagnosis of Rare Diseases". In: *International Journal of Molecular Sciences* 22 (2021), p. 10891. URL: Diagnosis of Rare Diseases.
[114] *Symbolic Format: MIDI*. URL: MIDI Format.
[115] *MIDI*. URL: MIDI.
[116] *Symbolic Format: MusicXML*. URL: MusicXML.
[117] Wang, Z. et al. "POP909: A Pop-Song Dataset for Music Arrangement Generation". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Montréal, Canada*. 2020. URL: POP909.
[118] *Humdrum File Format*. URL: Humdrum.
[119] Sapp, C. S. "Online Database of Scores in the Humdrum File Format". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), London, UK*. 2005. URL: Humdrum file format.
[120] *WAVE PCM Soundfile Format*. URL: WAV.
[121] *Monophony*. URL: Monophonic piece.
[122] Ji, S., Luo, J., and Yang, X. "A Comprehensive Survey on Deep Music Generation: Multi-Level Representations, Algorithms, Evaluations, and Future Directions". In: *arXiv preprint arXiv:2011.06801* (2020). URL: A Comprehensive Survey on Deep Music Generation.
[123] Bretan, M., Weinberg, G., and Heck, L. "A Unit Selection Methodology for Music Generation using Deep Neural Networks". In: *arXiv preprint arXiv:1612.03789* (2016). URL: A Unit Selection Methodology for Music Generation.
[124] Sturm, B. L. et al. "Music Transcription Modelling and Composition using Deep Learning". In: *arXiv preprint arXiv:1604.08723* (2016). URL: Music transcription.

[125] Hadjeres, G. and Nielsen, F. "Interactive Music Generation with Positional Constraints using Anticipation-RNNs". In: *arXiv preprint arXiv:1709.06404* (2017). URL: Anticipation-RNN.

[126] Roberts, A. et al. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *Proceedings of International Conference on Machine Learning (ICML), Stockholm, Sweden*. 2018. URL: MusicVAE.

[127] Yu, L. et al. "SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), San Francisco, California, USA*. 2017. URL: SeqGAN.

[128] *Polyphony*. URL: Polyphonic piece.

[129] Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription". In: *arXiv preprint arXiv:1206.6392* (2012). URL: Polyphonic Music Generation and Transcription.

[130] Hadjeres, G., Pachet, F., and Nielsen, F. "DeepBach: A Steerable Model for Bach Chorales Generation". In: *Proceedings of International Conference on Machine Learning (ICML), Sydney, Australia*. 2017. URL: DeepBach.

[131] Mao, H. H., Shin, T., and Cottrell, G. W. "DeepJ: Style-Specific Music Generation". In: *Proceedings of International Conference on Semantic Computing (ICSC), Laguna Hills, California, USA*. 2018. URL: DeepJ.

[132] Johnson, D. D. "Generating Polyphonic Music using Tied Parallel Networks". In: *Proceedings of International Conference on Evolutionary and biologically Inspired Music and Art (EvoMUSART), Amsterdam, The Netherlands*. 2017. URL: Biaxial LSTM.

[133] Wang, Z. et al. "PIANOTREE VAE: Structured Representation Learning for Polyphonic Music". In: *arXiv preprint arXiv:2008.07118* (2020). URL: PIANOTREE VAE.

[134] Chu, H., Urtasun, R., and Fidler, S. "Song from Pi: A Musically Plausible Network for Pop Music Generation". In: *arXiv preprint arXiv:1611.03477* (2017). URL: Song from Pi.

[135] *Song from π*. URL: Song from π.

[136] Guan, F., Yu, C., and Yang, S. "A GAN model with Self-Attention Mechanism to Generate Multi-Instruments Symbolic Music". In: *Proceedings of International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary*. 2019. URL: DMB-GAN.

[137] Donahue, C. et al. "LakhNES: Improving Multi-Instrumental Music Generation with Cross-Domain Pre-training". In: *arXiv preprint arXiv:1907.04868* (2019). URL: LakhNES.

[138] Valenti, A., Carta, A., and Bacciu, D. "Learning Style-Aware Symbolic Music Representations by Adversarial Autoencoders". In: *arXiv preprint arXiv:2001.05494* (2020). URL: MusAE.

[139] Makhzani, A. et al. "Adversarial Autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015). URL: AAE.

[140] Simon, I., Morris, D., and Basu, S. "MySong: Automatic Accompaniment Generation for Vocal Melodies". In: *Proceedings of International Conference on Human Factors in Computing Systems (CHI), Florence, Italy*. 2008. URL: MySong.

[141] Wang, Z. and Xia, G. "A Framework for Automated Pop-Song Melody Generation with Piano Accompaniment Arrangement". In: *arXiv preprint arXiv:1812.10906* (2018). URL: Automated Pop-song Melody Generation with Piano Accompaniment Arrangement.

[142] Shumway, R. H, Stoffer, D. S, and Stoffer, D. S. *Time Series Analysis and its Applications*. Springer, 2000. URL: ARMA.

[143] Jiang, N. et al. "RL-Duet: Online Music Accompaniment Generation using Deep Reinforcement Learning". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), New Delhi, India*. 2020. URL: RL-Duet.

[144] Ren, Y. et al. "PopMAG: Pop Music Accompaniment Generation". In: *Proceedings of International Conference on Multimedia (ACM-MM), New York, USA*. 2020. URL: PopMAG.

[145] Benetos, E. et al. "Automatic Music Transcription: An Overview". In: *IEEE Signal Processing Magazine* 36 (2018), pp. 20–30. URL: Automatic Music Transcription.

[146] Percival, G., Fukayama, S., and Goto, M. "Song2Quartet: A System for Generating String Quartet Cover Songs from Polyphonic Audio of Popular Music". In: *Proceedings of International Conference on Music Information Retrieval Conference (ISMIR), Málaga, Spain*. 2015. URL: Song2Quartet.

[147] Nakamura, E. and Sagayama, S. "Automatic Piano Reduction from Ensemble Scores based on Merged-Output Hidden Markov Model". In: *Proceedings of International Conference on Computer Music (ICMC), Denton, Texas, USA*. 2015. URL: Automatic Piano Reduction from Ensemble Scores.

[148] Nakamurai, E. and Yoshii, K. "Statistical Piano Reduction Controlling Performance Difficulty". In: *APSIPA Transactions on Signal and Information Processing* 7 (2015), p. 13. URL: Piano Reduction.

[149] Gatys, L. A., Ecker, A. S., and Bethge, M. "Image Style Transfer using Convolutional Neural Networks". In: *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Nevada, USA*. 2016. URL: Image Style Transfer.

[150] Dai, S., Zhang, Z., and Xia, G. G. "Music Style Transfer: A Position Paper". In: *arXiv preprint arXiv:1803.06841* (2018). URL: Music Style Transfer.

[151] Brunner, G. et al. "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer". In: *arXiv preprint arXiv:1809.07600* (2018). URL: MIDI-VAE.

[152] Yang, R. et al. "Deep Music Analogy via Latent Representation Disentanglement". In: *arXiv preprint arXiv:1906.03626* (2019). URL: Latent representation disentanglement.

[153] Chen, Z. et al. "Learning to Fuse Music Genres with Generative Adversarial Dual Learning". In: *Proceedings of International Conference on Data Mining (ICDM), Phuket, Thailand*. 2017. URL: Fuse Music Genres.

[154] Huang, S. et al. "TIMBRETRON: A Wavenet(cycleGAN(CQT(audio))) Pipeline for Musical Timbre Transfer". In: *arXiv preprint arXiv:1811.09620* (2019). URL: TIMBRETRON.

[155] Barry, S. and Kim, Y. "Style Transfer for Musical Audio using Multiple Time-Frequency Representations". In: *Proceedings of International Conference on Learning Representations (ICLR), Vancouver, British Columbia, USA*. 2018. URL: Style transfer for musical audio.

[156] Peng, X. et al. "A Lightweight Music Texture Transfer System". In: *arXiv preprint arXiv:1810.01248* (2020). URL: Music Texture Transfer System.

[157] Sismana, B. et al. "SINGAN: Singing Voice Conversion with Generative Adversarial Networks". In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Lanzhou, China*. 2019. URL: SINGAN.

[158] Zhang, L. et al. "DurIAN-SC: Duration Informed Attention Network based Singing Voice Conversion System". In: *arXiv preprint arXiv:2008.03009* (2020). URL: DurIAN-SC.

[159] Hung, Y.-N. et al. "Musical Composition Style Transfer via Disentangled Timbre Representations". In: *arXiv preprint arXiv:1905.13567* (2019). URL: Music Style Transfer.

[160] Zalkow, F., Brand, S., and Graf, B. "Musical Style Modification as an Optimization Problem". In: *Proceedings of International Conference on Computer Music (ICMC), Utrecht, Netherlands*. 2016. URL: Musical Style Modification.

[161] Pati, A., Lerch, A., and Hadjeres, G. "Learning to Traverse Latent Spaces for Musical Score Inpainting". In: *arXiv preprint arXiv:1907.01164* (2019). URL: Musical score inpainting.

[162] Chen, K. et al. "Music Sketchnet: Controllable Music Generation via Factorized Representations of Pitch and Rhythm". In: *arXiv preprint arXiv:2008.01291* (2020). URL: Music Sketchnet.

[163] Chi, W. et al. "Generating Music with a Self-Correcting, Non-Chronological Autoregressive Model". In: *arXiv preprint arXiv:2008.08927* (2020). URL: EsNET.

[164] Marafioti, A. et al. "A Context Encoder for Audio Inpainting". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27 (2019), pp. 2362–2372. URL: Audio Inpainting.

[165] Marafioti, A. et al. "GACELA: A Generative Adversarial Context Encoder for Long Audio Inpainting of Music". In: *IEEE Journal of Selected Topics in Signal Processing* 15 (2020), pp. 120–131. URL: GACELA.

[166] *JSB Chorales*. URL: Bach chorales.

[167] Cuthbert, M. S. and Ariza, C. "music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Utrecht, Netherlands*. 2010. URL: music21.

[168] Ferreira, L. N. and Whitehead, J. "Learning to Generate Music with Sentiment". In: *arXiv preprint arXiv:2103.06125* (2019). URL: Learning to generate music with sentiment.

[169] Crestel, L. et al. "A Database Linking Piano and Orchestral MIDI Scores with Application to Automatic Projective Orchestration". In: *arXiv preprint arXiv:1810.08611* (2018). URL: A Database linking piano and orchestral MIDI scores.

[170] *International Piano e-Competition*. URL: e-Piano.

[171] Kong, Q. et al. "GiantMIDI-Piano: A Large-Scale MIDI Dataset for Classical Piano Music". In: *arXiv preprint arXiv:2010.07061* (2020). URL: GiantMIDI-Piano.

[172] Kong, Q. et al. "High-Resolution Piano Transcription with Pedals by Regressing Onsets and Offsets Times". In: *arXiv preprint arXiv:2010.01815* (2020). URL: Piano Transcription with Pedals.

[173] *TheoryTab DB*. URL: TheoryTab.

[174] Yeh, Y.-C. et al. "Automatic Melody Harmonization with Triad Chords". In: *Journal of New Music Research* 50 (2021), p. 20. URL: Automatic Melody Harmonization with Triad Chords: A Comparative Study.

[175] *Lead Sheet Dataset*. URL: HLSD.

[176] Lim, H., Rhyu, S., and Lee, K. "Chord Generation from Symbolic Melody using BLSTM Networks". In: *arXiv preprint arXiv:1712.01011* (2017). URL: Chord generation from symbolic melody.

[177] Jeong, D. et al. "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Delft, The Netherlands*. 2019. URL: VirtuosoNet.

[178] Dong, H.-W., Hsiao, W.-Y., and Yang, Y.-H. "Pypianoroll: Open source Python Package for Handling Multitrack Pianoroll". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Paris, France*. 2018. URL: Pypianoroll.

[179] *ABC version of the Nottingham Music Database*. URL: NMD.

[180] *Going to use the Nottingham Music Database?* URL: Cleansed NMD.

[181] *Henrik Norbeck's ABC Tunes*. URL: Norbeck's ABC Tunes.

[182] Cherla, S. et al. "Hybrid Long- and Short-Term Models of Folk Melodies". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Málaga, Spain*. 2015. URL: Folk melodies.

[183] Engel, J. et al. "Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders". In: *Proceedings of International Conference on Machine Learning (ICML), Sydney, Australia*. 2017. URL: Neural Audio Synthesis.

[184] Defferrard, M. et al. "FMA: A Dataset for Music Analysis". In: *arXiv preprint arXiv:1612.01840* (2017). URL: FMA.

[185] Duan, Z. et al. "The NUS Sung and Spoken Lyrics Corpus: A Quantitative Comparison of Singing and Speech". In: *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Kaohsiung, Taiwan*. 2013. URL: The NUS Sung and Spoken Lyrics Corpus.

[186] Hawthorne, C. et al. "Enabling Factorized Piano Music Modeling and Generation with Maestro Dataset". In: *arXiv preprint arXiv:1810.12247* (2019). URL: Maestro Dataset.

[187] Gillick, J. et al. "Learning to Groove with Inverse Sequence Transformations". In: *Proceedings of International Conference on Machine Learning (ICML), Long Beach, California, USA*. 2019. URL: Learning to Groove.

[188] Foscarin, F. et al. "ASAP: A Dataset of Aligned Scores and Performances for Piano Transcription". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Montreal, Canada*. 2020. URL: ASAP.

[189] Yu, Y. and Canales, S. "Conditional LSTM-GAN for Melody Generation from Lyrics". In: *arXiv preprint arXiv:1908.05551* (2019). URL: Melody Generation from Lyrics.

[190] Dorfer, M. et al. "Learning Audio-Sheet Music Correspondences for Cross-Modal Retrieval and Piece Identification". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Paris, France*. 2018. URL: Cross-modal retrieval.

[191] Zeng, Donghuo, Yu, Yi, and Oyama, Keizo. "MTM Dataset for Joint Representation Learning among Sheet Music, Lyrics and Musical Audio". In: *arXiv preprint arXiv:2012.00290* (2020). URL: MTM Dataset.

[192] Yang, L.-C. and Lerch, A. "On the Evaluation of Generative Models in Music". In: *Neural Computing and Applications* 32 (2018), pp. 4773–4784. URL: Evaluation of generative models in music.

[193] Theis, L., Oord, A., and Bethge, M. "A Note on the Evaluation of Generative Models". In: *arXiv preprint arXiv:1511.01844* (2015). URL: Evaluation of generative models.

[194] Sturm, B. L. and Ben-Tal, O. "Taking the Models back to Music Practice: Evaluating Generative Transcription Models built using Deep Learning". In: *Journal of Creative Music Systems* 2 (2017). URL: Evaluating Generative Transcription Models.

[195] Casella, G. and Berger, R. L. *Statistical Inference*. Cengage Learning, 2021. URL: Statistical Inference.

[196] Huang, C.-Z. A. et al. "Counterpoint by Convolution". In: *arXiv preprint arXiv:1903.07227* (2019). URL: Counterpoint by convolution.

[197] Johnson, D. D. "Generating Polyphonic Music using Tied Parallel Networks". In: *Proceedings of International Conference on Evolutionary and Biologically Inspired Music and Art (EvoMUSART), Amsterdam, The Netherlands*. 2017. URL: Generating Polyphonic Music Using Tied Parallel Networks.

[198] Donahue, C., McAuley, J., and Puckette, M. "Adversarial Audio Synthesis". In: *arXiv preprint arXiv:1802.04208* (2018). URL: Adversarial Audio Synthesis.

[199] Marafioti, A. et al. "Adversarial Generation of Time-Frequency Features with Application in Audio Synthesis". In: *Proceedings of International Conference on Machine Learning (ICML), Long Beach, California, USA*. 2019. URL: Audio Synthesis.

[200] Engel, J. et al. "GANSynth: Adversarial Neural Audio Synthesis". In: *arXiv preprint arXiv:1902.08710* (2019). URL: GANSynth.

[201] Roberts, A. et al. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music". In: *Proceedings of International Conference on Machine Learning (ICML), Stockholm, Sweden*. 2018. URL: Long-Term Structure in Music.

[202] Brunner, G. et al. "Symbolic Music Genre Transfer with CycleGAN". In: *Proceedings of International Conference on Tools with Artificial Intelligence (ICTAI), Volos,Greece*. 2018. URL: CycleGAN.

[203] Chuan, C.-H. and Herremans, D. "Modeling Temporal Tonal Relations in Polyphonic Music through Deep Networks with a Novel Image-based Representation". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), New Orleans, Louisiana, USA*. 2018. URL: Modeling Temporal Tonal Relations in Polyphonic Music.

[204] Sabathé, R., Coutinho, E., and Schuller, B. "Deep Recurrent Music Writer: Memory-Enhanced Variational Autoencoder-based Musical Score Composition and an Objective Measure". In: *Proceedings of International Joint Conference on Neural Networks (IJCNN), Anchorage, Alaska, USA*. 2017. URL: Deep Recurrent Music Writer.

[205] Choi, K. et al. "Encoding Musical Style with Transformer Autoencoders". In: *Proceedings of International Conference on Machine Learning (ICML), Vienna, Austria*. 2020. URL: Encoding Musical Style .

[206] Wang, C.-i and Dubnov, S. "Guided Music Synthesis with Variable Markov Oracle". In: *Proceedings of International Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Raleigh, North Carolina, USA*. 2014. URL: Variable Markov Oracle.

[207] Allauzen, C., Crochemore, M., and Raffinot, M. "Factor Oracle: A New Structure for Pattern Matching". In: *Proceedings of International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Milovy, Czech Republic*. 1999. URL: Factor Oracle.

[208] Dubnov, S., Assayag, G., and Cont, A. "Audio Oracle: A New Algorithm for Fast Learning of Audio Structures". In: *Proceedings of International Conference on Computer Music (ICCM), Copenhagen, Denmark*. 2007. URL: Audio Oracle.

[209] Chen, K. et al. "The Effect of Explicit Structure Encoding of Deep Neural Networks for Symbolic Music Generation". In: *Proceedings of International Workshop on Multilayer Music Representation and Processing (MMRP), Milano, Italy*. 2019. URL: Symbolic Music Generation.

[210] Hakimi, S. H., Bhonker, N., and El-Yaniv, R. "BEBOPNET: Deep Neural Models for Personalized Jazz Improvisations". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Montreal, Canada*. 2020. URL: BEBOPNET.

[211] Jin, C. et al. "A Style-Specific Music Composition Neural Network". In: *Neural Processing Letters* 52 (2020), pp. 1893–1912. URL: A Style-Specific Music Composition Neural Network.

[212] Parekh, J., Rao, P., and Yang, Y.-H. "Speech-to-Singing Conversion in an Encoder-Decoder Framework". In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain*. 2020. URL: Speech-to-singing conversion.

[213] Ribeiro, F. et al. "CROWDMOS: An Approach for Crowdsourcing Mean Opinion Score Studies". In: *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czechia*. 2011. URL: CROWDMOS.

[214] Turing, A. M. "Computing Machinery and Intelligence". In: *Parsing the Turing Test*. Springer, 2009. URL: Computing machinery and intelligence.

[215] Liang, F. et al. "Automatic Stylistic Composition of Bach Chorales with Deep LSTM". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Suzhou, China*. 2017. URL: Bach chorales.

[216] Haque, A., Guo, M., and Verma, P. "Conditional End-to-End Audio Transforms". In: *arXiv preprint arXiv:1804.00047* (2018). URL: Conditional End-to-End Audio Transforms.

[217] Zhao, K. et al. "An Emotional Symbolic Music Generation System based on LSTM Networks". In: *Proceedings of International Conference on Information Technology, Networking, Electronic and Automation Control (ITNEC), Chengdu, China*. 2019. URL: Emotional Symbolic Music Generation.

[218] Wei, I-C., Wu, C.-W., and Su, L. "Generating Structured Drum Pattern using Variational Autoencoder and Self-Similarity Matrix". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Delft, The Netherlands*. 2019. URL: Generating structured drum pattern.

[219] Lu, C.-Y. et al. "Play as You Like: Timbre-Enhanced Multi-Modal Music Style Transfer". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), Honolulu, Hawaii, USA*. 2019. URL: Play as You Like.

[220] Yi, Y.-H. et al. "Singing Voice Synthesis using Deep Autoregressive Neural Networks for Acoustic Modeling". In: *arXiv preprint arXiv:1906.08977* (2019). URL: Singing Voice Synthesis .

[221] *Anomaly Detection using GAN*. URL: Convolutional GAN.

[222] *Generative Adversarial Networks - A Deep Learning Architecture*. URL: GANs.

[223] Arjovsky, M. and Bottou, L. "Towards Principled Methods for Training Generative Adversarial Networks". In: *arXiv preprint arXiv:1701.04862* (2017). URL: Training generative adversarial networks.

[224] Arjovsky, M., Chintala, S., and Bottou, L. "Wasserstein Generative Adversarial Networks". In: *Proceedings of International Conference on Machine Learning (ICML), Sydney, Australia*. 2017. URL: WGANs.

[225] Raffel, C. and Ellis, D. PW. "Intuitive Analysis, Creation and Manipulation of MIDI Data with Pretty_midi". In: *Proceedings of International Conference on Music Information Retrieval (ISMIR), Taipei, Taiwan*. 2014. URL: Pretty_Midi.

[226] Serra, J. et al. "Unsupervised Detection of Music Boundaries by Time Series Structure Features". In: *Proceedings of International Conference on Artificial Intelligence (AAAI), Toronto, Ontario, Canada*. 2012. URL: Unsupervised detection of music boundaries.

[227] *AI Music Generation - Lead Sheet Composition and Arrangement*. URL: Unconditional Generation.

[228] *ISMIR 2019 tutorial: Generating Music with Generative Adversarial Networks (GANs)*. URL: Generating Music with GANs.

[229] Heusel, M. et al. "GANs Trained by a two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *arXiv preprint arXiv:1706.08500* (2017). URL: FID.

[230] Szegedy, C. et al. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Nevada, USA*. 2016. URL: Inception-v3.

[231] Wang, Z. et al. "Learning Interpretable Representation for Controllable Polyphonic Music Generation". In: *arXiv preprint arXiv:2008.07122* (2020). URL: Controllable Generation.

[232] Zhao, J. and Xia, G. "AccoMontage: Accompaniment Arrangement via Phrase Selection and Style Transfer". In: *arXiv preprint arXiv:2108.11213* (2021). URL: AccoMontage.

[233] Yu, Y., Srivastava, A., and Canales, S. "Conditional LSTM-GAN for Melody Generation from Lyrics". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17 (2021), pp. 1–20. URL: Melody generation from lyrics.

[234] Gan, C. et al. "Foley Music: Learning to Generate Music from Videos". In: *Proceedings of European Conference on Computer Vision (ECCV), Glasgow, UK*. 2020. URL: Foley Music.