ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Χρήση Γεννητικών Ανταγωνιστικών Δικτύων ως Τεχνική Εμπλουτισμού Δεδομένων σε Ιατρικές Εφαρμογές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΦΙΛΙΠΠΟΣ Π. ΚΟΝΙΔΑΡΗΣ

Επιβλέπων : Ανδρέας - Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Συνεπιβλέποντες : Θάνος Τάγαρης, Υποψήφιος Διδάκτορας Ε.Μ.Π.
Μάρα Σδράκα, Υποψήφια Διδάκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Χρήση Ανταγωνιστικών Γεννητικών Δικτύων ως Τεχνική Εμπλουτισμού Δεδομένων σε Ιατρικές Εφαρμογές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΦΙΛΙΠΠΟΣ Π. ΚΟΝΙΔΑΡΗΣ

Επιβλέπων : Ανδρέας - Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.

Συνεπιβλέποντες : Θάνος Τάγαρης,
Υποψήφιος Διδάκτορας Ε.Μ.Π.

Μάρα Σδράκα,
Υποψήφια Διδάκτορας Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Ιουλίου 2018.

.....	.....	.....
Ανδρέας-Γεώργιος Σταφυλοπάτης	Γεώργιος Στάμου	Κωνσταντίνα Νικήτα
Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

……

Φίλιππος Π. Κονιδάρης
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια έχει διαπιστωθεί μία ραγδαία ανάπτυξη στις τεχνικές της Όρασης Υπολογιστών (Computer Vision) και ιδιαίτερα σε αυτές που σχετίζονται με τη βαθιά μάθηση (Deep Learning). Οι τεχνικές αυτές όμως απαιτούν τεράστιο πλήθος οπτικών δεδομένων για να εκπαιδευτούν επαρκώς, κάτι που σε πολλά προβλήματα ενδέχεται να μην είναι διαθέσιμο. Για το λόγο αυτό στην πράξη χρησιμοποιούνται τεχνικές εμπλουτισμού δεδομένων (data augmentation), οι οποίες, πραγματοποιώντας απλούς μετασχηματισμούς στο επίπεδο των pixels, παράγουν εικόνες που το δίκτυο εκλαμβάνει ως νέα δεδομένα. Οι δυνατότητες της πρακτικής αυτής όμως είναι περιορισμένες, καθώς δεν μπορούν να αλλάξουν σε ουσιαστικό βαθμό την είσοδο, εξαρτώνται από τη φύση του προβλήματος, αλλά κυρίως επειδή τα δεδομένα λαμβάνονται υπόψη ξεχωριστά, και όχι ως μέρη ενός μεγαλύτερου συνόλου. Στα πλαίσια αυτής της διπλωματικής εργασίας προτείνεται μία τεχνική που επιχειρεί να ξεπεράσει τους προαναφερθέντες περιορισμούς, εμπλουτίζοντας τα δεδομένα με χρήση πιο ισχυρών τεχνικών.

Τα Γεννητικά Ανταγωνιστικά Δίκτυα (Generative Adversarial Networks, GANs) είναι μία οικογένεια μοντέλων ικανών να παράγουν ρεαλιστικές, συνθετικές εικόνες. Ένα GAN αποτελείται από δύο δίκτυα, με το ένα να προσπαθεί να παράξει όσο το δυνατόν πιο ρεαλιστικές εικόνες (δημιουργός), και το άλλο να προσπαθεί να ξεχωρίσει όσο το δυνατόν καλύτερα τις τεχνητές από τις αληθινές (διευκρινιστής). Μέσω του συνεχούς ανταγωνισμού μεταξύ τους, τα δίκτυα βελτιώνονται μέχρι να επέλθει η ισορροπία, όπου οι αληθινες και οι τεχνητές εικόνες φαίνονται πανομοιότυπες στον διευκρινιστή. Ο δημιουργός τότε παράγει τις πιο αληθοφανείς εικόνες.

Όσον αφορά το πειραματικό κομμάτι της εργασίας, αρχικά λήφθηκαν εικόνες μαγνητικών τομογραφιών υγιών ατόμων και ασθενών που πάσχουν από τη νόσο του Alzheimer. Τα δεδομένα από ένα ποσοστό των ατόμων έμειναν κρυφά κατά τη διάρκεια της εκπαίδευσης, ώστε να μπορεί να εκτιμηθεί όσο το δυνατόν καλύτερα η δυνατότητα των δικτύων που θα δοκιμάζονταν για γενίκευση των γνώσεών τους. Στη συνέχεια, επιλέχθηκε και εκπαιδεύτηκε μία αρχιτεκτονική GAN πάνω στα δεδομένα αυτά, η οποία στη συνέχεια χρησιμοποιήθηκε για την παραγωγή τεχνητών εικόνων. Οι τεχνητές αυτές εικόνες προστέθηκαν στο αρχικό σύνολο εκπαίδευσης και χρησιμοποιήθηκαν για την εκπαίδευση συνελικτικών δικτύων ταξινόμησης. Η επίδοσή τους συγκρίθηκε με την αντίστοιχη των δικτύων εκπαιδευμένων στο αρχικό σύνολο, με και χωρίς καμία εφαρμογή των γνωστών τεχνικών εμπλουτισμού.

Παρ' όλο που στη βιβλιογραφία οι κλασικές τεχνικές εμπλουτισμού έχουν αποδειχθεί ότι στις περισσότερες περιπτώσεις ενισχύουν την απόδοση των μοντέλων, κάτι τέτοιο δεν παρατηρήθηκε στην παρούσα εργασία.Η προσθήκη όμως συνθετικών εικόνων στο σύνολο δεδομένωνεπέφερε σημαντική βελτίωση στην επίδοσή τους. Τέλος, καλύτερα μοντέλα αποδείχθηκαν αυτά που συνδύαζαν τον εμπλουτισμό μέσω GAN με τις κλασικές τεχνικές.

Λέξεις Κλειδιά

Τεχνητή Νοημοσύνη, Βαθιά Μάθηση, Εμπλουτισμός Δεδομένων, Γεννητικά Ανταγωνιστικά Δίκτυα, Ανταγωνιστική Μάθηση

Abstract

Over the past years, there has been a rapid development in the field of Computer Vision, especially through techniques involving Deep Learning. These techniques, however, require large amounts of data to be trained properly, which in practice, are rarely available. For this reason, data augmentation is used extensively to avoid such limitations. The most common form of augmentation involves performing simple, affine transformations on each image. This both increases the size of the training set and theoretically allows the Deep Neural Networks to generalize better. These methods, however, have several drawbacks, such as their inability to drastically change the input without damaging its semantic content, their ad-hoc application depending on the nature of each problem and most importantly their inability to regard the input data as a whole. This diploma thesis proposes a novel technique that attempts to overcome said limitations by augmenting the data using more powerful methods.

Generative Adversarial Networks (GANs) are a family of models capable of producing realistic artificial images. Each GAN is composed of two networks, one attempting to produce artificial images (generator) and the other trying to distinguish the real from the fake ones (discriminator). These two networks compete against each other, through adversarial training, until they reach an equilibrium. At this point the images from the two distributions appear indistinguishable to the discriminator, as the generator produces sufficiently realistic images.

The experimental part of this thesis was performed on a dataset consisting of MR images from patients suffering from Alzheimer's Disease and healthy control subjects. A GAN was trained on the above dataset and was used to produce synthetic images from both categories. These images were added to the original data and were subsequently used to train Convolutional Neural Networks for classification.

The use of traditional data augmentation methods did not seem to have much of an effect in improving the performance of any model it was tested upon, contradicting most findings in related work. The inclusion, however, of artificial images in the training set, resulted in considerable improvements. Finally, the best models proved to be the ones that combined the traditional augmentation methods with the synthetic images produced by the GAN.

Key words

Artificial Intelligence, Deep Learning, Data Augmentation, Generative Adversarial Networks, Adversarial Learning

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο του Προπτυχιακού Προγράμματος Σπουδών της Σχολής Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου και σηματοδοτεί την ολοκλήρωση των σπουδών μου. Πριν όμως από οποιαδήποτε αναφορά στη διαδικασία που ακολουθήθηκε και στα αποτελέσματα που προέκυψαν, θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους με τους οποίους συνεργάστηκα και συνέβαλαν στην ολοκλήρωση της εργασίας αυτής.

Κατ' αρχάς απευθύνω τις ευχαριστίες μου στον επιβλέποντα κ. Ανδρέα-Γεώργιο Σταφυλοπάτη, Καθηγητή Ε.Μ.Π, για την δυνατότητα που μου προσέφερε να εργαστώ σε ένα αντικείμενο ιδιαίτερα ελκυστικό για μένα και να διευρύνω τις γνώσεις μου. Παράλληλα, θα ήθελα να ευχαριστήσω τους κ. Γεώργιο Στάμου, Αναπληρωτή Καθηγητή Ε.Μ.Π και κ. Κωνσταντίνα Νικήτα, Καθηγήτρια Ε.Μ.Π που με τίμησαν με την παρουσία τους στην τριμελή επιτροπή εξέτασης.

Επίσης οφείλω ιδιαίτερες ευχαριστίες στους Θάνο Τάγαρη και Μάρα Σδράκα, Υποψήφιους Διδάκτορες Ε.Μ.Π., για το χρόνο που αφιέρωσαν και τη θεμελιώδη συνεισφορά τους στην εκπόνηση της συγκεκριμένης εργασίας. Τόσο η επιστημονική όσο και η πνευματική τους στήριξη, ιδιαίτερα στα τελευταία στάδια της εργασίας, ήταν ιδιαίτερα σημαντικές για εμένα. Η εμπειρία και οι γνώσεις τους στάθηκαν καθοριστικές και η συνεργασία μας θεωρώ πως ήταν άκρως επιτυχημένη και εποικοδομητική.

Τέλος, σε προσωπικό επίπεδο θα ήθελα απλά να πω ένα μεγάλο ευχαριστώ σε όλους όσοι ήταν δίπλα μου και με στήριξαν τα τελευταία έξι χρόνια, ο καθένας με το δικό του ξεχωριστό τρόπο.

Φίλιππος Π. Κονιδάρης,
Αθήνα, 24η Ιουλίου 2018

Πίνακας Περιεχομένων

Πίνακας Σχημάτων

Πίνακας Πινάκων

Κεφάλαιο 1: Εισαγωγή

Κανείς δεν μπορεί να αμφισβητήσει την επιρροή της μηχανικής μάθησης, και γενικότερα της τεχνητής νοημοσύνης στις ζωές μας τα τελευταία πέντε με δέκα χρόνια. Από προβλήματα όπως η ταξινόμηση εικόνων σε κλάσεις [1], η εύρεση της τοποθεσίας πολλαπλών αντικειμένων σε φωτογραφίες [2] και η πρόταση των πιο πιθανών προιόντων που είναι πιθανό να αγοράσει ένας χρήστης μέχρι τη νίκη σε παιχνίδια όπως το Go [3] και το Dota 2 [4] με αντιπάλους παγκόσμιους πρωταθλητές, τα αυτόνομα αυτοκίνητα και την αποθορυβοποίηση σημάτων από όργανα μικροσκοπίας ατομικής κλίμακας [5], η μηχανική μάθηση δε σταματά να μας εκπλήσσει με τις δυνατότητές της.

1.1: Ιστορικές Σημειώσεις

Η περιοχή της τεχνητής νοημοσύνης είναι σχεδόν τόσο παλιά όσο και η ίδια η επιστήμη υπολογιστών. Αν και υπήρχαν από πριν μερικά εξαιρετικά δείγματα, με κύριο το άρθρο που εισήγαγε το Turing test [6], στο οποίο εξετάζεται το αν μία μηχανή μπορεί να παράξει δικές της σκέψης τόσο περίπλοκες όσο ο ανθρώπινος νους, ως ακαδημαϊκό πεδίο αναγνωρίστηκε στα μέσα της δεκαετίας του 1950. Αρχικές προσπάθειες εστιάστηκαν κυρίως στο συμβολικό συλλογισμό, στην παραγωγή δηλαδή νέας γνώσης από μερικές δοσμένες, αληθείς προτάσεις, στη σημασιολογική αναπαράσταση του κόσμου από οντότητες και σχέσεων μεταξύ τους (π.χ. John-IsA-human), και στους αλγορίθμους αναζήτησης. Παράλληλα, υπήρξαν σημαντικές επιτυχίες στον τομέα της ρομποτικής, η σημαντικότερη από τις οποίες ήταν η επιτυχής εκπαίδευση ενός ρομπότ στο να στοιβάζει κύβους μεγάλου μεγέθους.

Το 1958 ένα άρθρο του Frank Roseblatt [7] εισήγαγε τα πρώτα νευρωνικά δίκτυα, τα perceptrons. Εμπνευσμένο από τη λειτουργία των ανθρώπινων νευρώνων, ένα perceptron είναι ένας γραμμικός δυαδικός ταξινομητής. Η είσοδος πολλαπλασιάζεται με έναν πίνακα βαρών, που αντιπροσωπεύουν τη σημασία που θα δοθεί στα επιμέρους χαρακτηριστικά της, και προστίθεται με έναν αριθμό που αντιπροσωπεύει την εγγενή πόλωση του δικτύου. Αν το αποτέλεσμα της πράξης αυτής είναι μεγαλύτερο από ένα όριο, συνήθως το μηδέν, τότε το perceptron ενεργοποιείται, έχει δηλαδή έξοδο 1. Διαφορετικά, έχει έξοδο 0.

Οι πρώτες αυτές επιτυχίες ώθησαν τους ερευνητές να δηλώνουν ότι "σε δέκα το πολύ χρόνια το πρόβλημα της τεχνητής νοημοσύνης θα έχει λυθεί", "σε πέντε χρόνια οι υπολογιστές θα μεταφράζουν αυτόματα" κ.λ.π. Ακόμη, τα perceptrons διαφημίστηκαν ως η απάντηση σε όλα τα προβλήματα ("ένα σύστημα perceptron θα έχει τη νοημοσύνη ενός μέσου ανθρώπου"). Ωστόσο οι Seymour Papert και Marvin Minsky στο βιβλίο τους, Perceptrons [8], έδειξαν ότι ένα perceptron είναι ανίκανο από τη φύση του να λύσει προβλήματα που δεν είναι γραμμικά διαχωρίσιμα, όπως π.χ. το exclusive-or (XOR). Τα perceptrons έλαβαν τόσο μεγάλο πλήγμα που πρακτικά ξεχάστηκαν. Τον ίδιο περίπου καιρό δημοσιεύτηκαν οι εργασίες των Cook [9] και Karp [10], που απέδειξαν ότι τα πιο σημαντικά προβλήματα ίσως να μη λύνονται σε πρακτικό χρόνο (intractability), εκτός από τις περιπτώσεις όπου το μέγεθός τους είναι πρακτικά ασήμαντο. Τέλος, υπήρχε η φιλοσοφική άποψη ότι η ιδέα μηχανών που εμφανίζουν νοημοσύνη είναι εκ φύσεως καταδικασμένη να αποτύχει, λόγω του θεωρήματος της μη πληρότητας του Kurt Gödel [11].

Η αποτυχία των ερευνητών να εκτιμήσουν ρεαλιστικά το μέγεθος των προβλημάτων που αντιμετώπιζαν, η δημοσιότητα που έδιναν στις ανακαλύψεις τους και κατά συνέπεια η απογοήτευση που κυριαρχούσε όταν οι τεχνικές της εποχής αδυνατούσαν να λύσουν τα προβλήματα που είχαν υποσχεθεί οδήγησε σιγά σιγά σε διακοπή των χρηματοδοτήσεων και στη γενικότερη επικράτηση του φαινομένου που ονομάστηκε A.I. Winter, όπου ήταν σχεδόν ντροπιαστικό να παραδέχεται κανείς ότι ασχολείται με την περιοχή.

Ο πρώτος A.I. Winter τελείωσε στις αρχές της δεκαετίας του 1980, με την σταδιακή καθιέρωση μοντέλων που ονομάστηκαν συστήματα ειδικών (expert systems). Τα expert systems απαντούσαν σε ερωτήσεις και έλυναν προβλήματα σε μία συγκεκριμένη περιοχή, χρησιμοποιώντας λογικούς κανόνες που είχαν οριστεί εκ των προτέρων από ειδικούς της περιοχής αυτής. Το γεγονός ότι τα expert systems, σε αντίθεση με τις προηγούμενες ιδέες, σημείωναν πολύ καλές επιδόσεις σε πρακτικές εφαρμογές του "πραγματικού κόσμου" είχε ως αποτέλεσμα αυτά να χρησιμοποιηθούν εκτενώς από εταιρείες σε όλο τον κόσμο. Ακόμη, βρέθηκε ότι ο περιορισμός των απλών perceptrons σχετικά με την ανάγκη για γραμμικότητα μπορούσε να παρακαμφθεί αν απλά χρησιμοποιούνταν πάνω από ένας νευρώνες.

Έτσι, άρχισαν για πρώτη φορά να αναπτύσσονται δίκτυα που όχι μόνο τροφοδοτούσαν την είσοδό τους σε πολλαπλούς νευρώνες, αλλά χρησιμοποιούσαν και νευρώνες των οποίων η είσοδος ήταν η έξοδος προηγούμενων νευρώνων. Τα δίκτυα αυτά ονομάστηκαν perceptrons πολλαπλών στρωμάτων (multi-layer perceptrons, MLP), ενώ ένας νέος αλγόριθμος που αναπτύχθηκε την εποχή αυτή, ο αλγόριθμος της προς τα πίσω διάδοσης (back propagation) [12], κατέστησε εφικτή την αποδοτική εκπαίδευση των δικτύων αυτών μέσω gradient descent.

Η υπερβολική πίστη που είχε τοποθετήσει η βιομηχανία στα expert systems οδήγησε τελικά στον δεύτερο A.I. Winter, όταν ανέβηκαν στην επιφάνεια θεμελιώδη προβλήματα σχετικά με τη λειτουργία τους (αδυναμία μάθησης νέων κανόνων, κατάρρευση όταν η είσοδος παρέκλινε από το συνηθισμένο κ.λ.π.). Ο "χειμώνας" αυτός όμως οφειλόταν απλά στον τρόπο με τον οποίο οι κυβερνήσεις, οι εταιρείες και οι χρηματοδοτικοί οργανισμοί αντιλαμβάνονταν την περιοχή, και όχι σε έλλειψη προόδου, αφού την περίοδο αυτή, μεταξύ άλλων, αναπτύχθηκαν τα Support Vector Machines (SVM) [13], τα Random Forests [14] και τα Long Short-Term Memory δίκτυα [15], μοντέλα που χρησιμοποιούνται ευρέως ακόμα και σήμερα.

Ωστόσο, η έρευνα στην περιοχή των νευρωνικών δικτύων, εκτός από κάποιες εξαιρέσεις [16], είχε πρακτικά σταματήσει, κυρίως λόγω της αδυναμίας των ερευνητών να λύσουν τα προβλήματα των εξαφανιζόμενων παραγώγων (vanishing gradients, βλ. Ενότητα 1.2.3). Πιο πρόσφατα, στα μέσα της προηγούμενης δεκαετίας, οπότε και το πρόβλημα άρχισε να παρακάμπτεται, εκπαιδεύτηκαν τα πρώτα πραγματικά βαθιά δίκτυα, ενώ η νίκη της ομάδας των Krizhensky et al. [1] στο ILSVRC του 2012 πυροδότησε την επανάσταση στη βαθιά μάθηση που βιώνουμε μέχρι και σήμερα.

1.2: Κάποιες Βασικές Έννοιες

1.2.1: Ορισμός της Μηχανικής Μάθησης

Ο ορισμός της μηχανικής μάθησης που χρησιμοποιείται περισσότερο είναι ο ακόλουθος: «Ένα πρόγραμμα υπολογιστή λέγεται ότι μαθαίνει από εμπειρία E ως προς μια κλάση εργασιών T και ένα μέτρο επίδοσης P, αν η επίδοσή του σε εργασίες της κλάσης T, όπως αποτιμάται από το μέτρο P, βελτιώνεται με την εμπειρία E». Διατυπώθηκε από τον Tom M. Mitchell [17].

1.2.2: Gradient Descent

Το gradient descent είναι επαναληπτικός αλγόριθμος βελτιστοποίησης για την εύρεση ελάχιστων, όχι απαραίτητα ολικών, σημείων μίας συνάρτησης, έστω J, που στην περιοχή της μηχανικής μάθησης ονομάζεται συνάρτηση απώλειας, με παραμέτρους θ. Είναι μέθοδος πρώτης τάξης, δηλαδή λαμβάνει υπόψη μόνο τις πρώτες μερικές παραγώγους της συνάρτησης, και εκμεταλλεύεται το γεγονός ότι το διάνυσμά τους δείχνει την κατεύθυνση στον n-διάστατο χώρο προς την οποία θα αυξηθεί περισσότερο η τιμή της. Γι'αυτό και για την εύρεση των ελάχιστων σημείων ο αλγόριθμος

κάνει βήματα προς την αντίθετη κατεύθυνση, αυτή δηλαδή που θα οδηγήσει στη μεγαλύτερη μείωση.

Αρχικά οι μερικές παράγωγοι πολλαπλασιάζονται με έναν μικρό αριθμό που ονομάζεται ρυθμός μάθησης (learning rate). Ο learning rate καθορίζει το μέγεθος των βημάτων και είναι αρκετά μικρός για λόγους σταθερότητας στην εκπαίδευση. Στη συνέχεια το αποτέλεσμα αφαιρείται από τις τιμές των παραμέτρων. Οι παράγωγοι μπορούν να ληφθούν με διάφορους τρόπους, όπως back propagation, χρήση αριθμητικών μεθόδων ή μεθόδων που βασίζονται στην τυχαιότητα [18].

Υπάρχουν τρεις παραλλαγές του gradient descent και διαφέρουν στην ποσότητα των δεδομένων που χρειάζεται για να υπολογιστούν οι μερικές παράγωγοι. Όσο πιο πολλά τα δεδομένα, τόσο πιο ακριβείς θα είναι και οι παράγωγοι που θα προκύψουν, αλλά παράλληλα θα απαιτείται και τόσο περισσότερος χρόνος για τον υπολογισμό τους.

1.2.2.1: Batch Gradient Descent

Στην αρχική υλοποίηση του gradient descent, που ονομάστηκε batch gradient descent, οι μερικές παράγωγοι της συνάρτησης απώλειας ως προς τις παραμέτρους του δικτύου υπολογίζονται λαμβάνοντας υπόψη όλα τα δεδομένα εκπαίδευσης:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$$

Η ανάγκη για υπολογισμό των παραγώγων για όλα τα δεδομένα για την πραγματοποίηση ενός μόνο βήματος έχει ως αποτέλεσμα το batch gradient descent να είναι πιθανώς πολύ αργό, και πρακτικά άχρηστο για σύνολα δεδομένων (datasets) που δε χωρούν στη μνήμη. Ένα ακόμη μειονέκτημά του είναι ότι δεν επιτρέπει την επανεκπαίδευση του μοντέλου με νέα δεδομένα (online learning). Παρ'όλ'αυτά, σε περίπτωση που η επιφάνεια της συνάρτησης απώλειας είναι κυρτή, το batch gradient descent θα φτάσει αποδεδειγμένα σε ολικό ελάχιστο, κάτι που δεν ισχύει για τις άλλες δύο παραλλαγές.

1.2.2.2: Stochastic Gradient Descent

Το stochastic gradient descent πάει στο άλλο άκρο, και ενημερώνει τις παραμέτρους για κάθε δεδομένο, x_i, και το αντίστοιχο label, y_i, ξεχωριστά:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta, x_i, y_i)$$

Η λογική πίσω από το stochastic gradient descent είναι ότι σε μεγάλα datasets το batch gradient descent πραγματοποιεί παραπάνω υπολογισμούς απ'όσους χρειάζονται, καθώς υπολογίζει ξανά και ξανά σχεδόν ίδιες παραγώγους για παρόμοια δεδομένα. Το stochastic gradient descent είναι προφανώς η πιο γρήγορη από τις δύο παραλλαγές, ενώ μπορεί να χρησιμοποιηθεί για online επανεκπαίδευση μοντέλων. Όμως, οδηγεί σε σχετική αστάθεια στην εκπαίδευση, αφού η έλλειψη πληροφοριών για την στατιστική κατανομή των δεδομένων καθιστά τον αλγόριθμο ευάλωτο σε ακραία, μη αντιπροσωπευτικά δεδομένα (outliers). Αυτό προκαλεί αισθητά μεγαλύτερη διακύμανση στις τιμές της απώλειας κατά τη διάρκεια της εκπαίδευσης, κάτι που όμως με τη σειρά του μπορεί να έχει το ευεργετικό, αναπάντεχο αποτέλεσμα το stochastic gradient descent να οδηγηθεί σε νέα, καλύτερα τοπικά ελάχιστα μίας μη-κυρτής επιφάνειας, που το batch gradient descent δε θα έφτανε. Τέλος, πειράματα έχουν δείξει ότι αν ο learning rate μειώνεται σταδιακά κατά τη διάρκεια της εκπαίδευσης το stochastic gradient descent παρουσιάζει την ίδια ακριβώς συμπεριφορά σύγκλισης με το batch gradient descent.

1.2.2.3: Mini-batch Gradient Descent

Το mini-batch gradient descent συνδυάζει τα πλεονεκτήματα των δύο παραπάνω προσεγγίσεων, ενημερώνοντας τις παραμέτρους για κάθε δέσμη (mini-batch) n δεδομένων και labels:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta, x^{(i:i+n)}, y^{(i:i+n)})$$

Έτσι, από τη μία μειώνεται η ανεπιθύμητη διακύμανση των ενημερώσεων του stochastic gradient descent, και από την άλλη, αφού οι ενημερώσεις συνεχίζουν να είναι προσεγγίσεις αυτών του batch gradient descent, δεν εξαφανίζεται εντελώς η πιθανότητα της εύρεσης ενός καλύτερου τοπικού ελάχιστου σε μία μη-κυρτή επιφάνεια. Παράλληλα, λόγω της εκτενέστατης χρήσης τα τελευταία χρόνια, έχουν αναπτυχθεί βιβλιοθήκες που χρησιμοποιούν βελτιστοποιήσεις για εξαιρετικά γρήγορες πράξεις μεταξύ πινάκων. Τα τελευταία χρόνια έχει επικρατήσει στη βιβλιογραφία η χρήση του ονόματος stochastic gradient descent (SGD) ακόμα και όταν εφαρμόζεται mini-batch bradient descent.

Το SGD δεν εγγυάται πάντοτε τη βέλτιστη σύγκλιση, και κατά την εφαμογή του ενδέχεται να προκύψουν τα εξής προβλήματα:

- Επιλογή του σωστού learning rate: Αν είναι πολύ μικρό, η σύγκλιση θα είναι απαγορευτικά αργή. Αν από την άλλη είναι πολύ μεγάλο, είναι πολύ πιθανό προς το τέλος της εκπαίδευσης η συνάρτηση απώλειας να ταλαντώνεται γύρω από κάποιο ελάχιστο σημείο χωρίς ποτέ να το φτάνει, ή ακόμα και να προκύψει απόκλιση.
- Για το λόγο αυτό έχουν αναπτυχθεί εμπειρικές τεχνικές που ρυθμίζουν το learning rate κατά τη διάρκεια της εκπαίδευσης. Ένας προφανής τρόπος είναι η μείωσή του σύμφωνα με κάποιο προκαθορισμένο πρόγραμμα ή όταν η διαφορά της απώλειας μεταξύ δύο εποχών γίνει μικρότερη από κάποιο όριο. Το κακό με αυτές τις τεχνικές είναι ότι πρέπει να έχουν οριστεί πριν ξεκινήσει η εκπαίδευση, οπότε είναι αδύνατο να προσαρμοστούν στα χαρακτηριστικά των δεδομένων.
- Όλες οι παράμετροι ενημερώνονται με βάση το ίδιο learning rate. Αυτό ίσως είναι προβληματικό όταν έχουμε "αραιά" δεδομένα (sparse data), όταν δηλαδή η πλειοψηφία των δεδομένων έχει μηδενικές τιμές, ή όταν τα χαρακτηριστικά των δεδομένων δεν εμφανίζονται με την ίδια συχνότητα. Ιδανικά, θα θέλαμε οι ενημερώσεις να είναι πιο "γενναίες" για μοτίβα που εμφανίζονται σπάνια.
- Όσον αφορά συγκεκριμένα τα νευρωνικά δίκτυα, είναι συνηθισμένο οι συναρτήσεις απώλειας που χρησιμοποιούνται να είναι σημαντικά μη-κυρτές, κάτι που έχει ως αποτέλεσμα το SGD πολύ συχνά να παγιδεύεται σε τοπικά, μη βέλτιστα ελάχιστα σημεία. Σύμφωνα με τους Dauphin et al. [19], η συμπεριφορά αυτή οφείλεται στα σημεία σέλας (saddle points), σημεία που η κλίση μίας διάστασης είναι θετική και μίας άλλης αρνητική. Τα σημεία αυτά συνήθως περιβάλλονται από ένα "πλάτωμα" που η συνάρτηση απώλειας έχει την ίδια τιμή, με αποτέλεσμα πρακτικά όλες οι παράγωγοι να έχουν τιμές κοντά στο μηδέν και το SGD να εγκλωβίζεται.

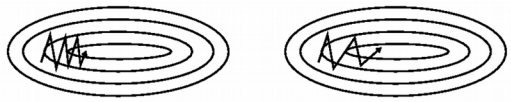
1.2.2.4: Παραλλαγές και Βελτιώσεις του Απλού SGD

Έχουν προταθεί πολλές αλγοριθμικές παραλλαγές και βελτιώσεις του SGD για την αντιμετώπιση των παραπάνω προβλημάτων. Μερικές από τις πιο γνωστές είναι ο αλγόριθμος ορμής (momentum), ο αλγόριθμος ορμής με επιταχυνόμενη παράγωγο Nesterov (Nesterov accelerated gradient), ο Adagrad, ο RMSProp και ο Adam. Οι πληροφορίες για τους αλγορίθμους αυτούς λήφθηκαν από το [20].

1.2.2.4.1: Momentum

Ο SGD δεν αποδίδει τόσο καλά όταν πρέπει να διασχίσει μία "κοιλάδα", δηλαδή μία περιοχή που μία διάσταση έχει πολύ μεγαλύτερη κλίση από κάποια άλλη. Συνήθως ταλαντώνεται έντονα μεταξύ των "πλαγιών" της κοιλάδας και κάνει πολύ μικρά βήματα προς το ζητούμενο ελάχιστο σημείο. Η ορμή [21], σε μία προσπάθεια αντιμετώπισης του πρώτου προβλήματος, επιταχύνει το SGD προς τη σωστή κατεύθυνση και μειώνει τις ταλαντώσεις (βλ. Σχήμα 1.1). Το καταφέρνει αυτό προσθέτοντας στο διάνυσμα ενημερώσεων v_t έναν όρο ορμής, που ορίζεται ως το διάνυσμα του προηγούμενου βήματος πολλαπλασιασμένο με ένα παράγοντα γ:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta)$$
$$\theta = \theta - v_t$$



Σχήμα 1.1: Επιρροή του όρου ορμής στο SGD.

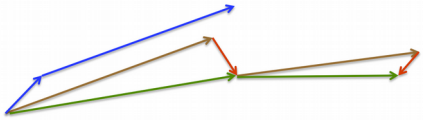
Μπορούμε να σκεφτούμε την επιρροή της ορμή ως κύληση μίας μπάλας σε μία πλαγιά. Η μπάλα συγκεντρώνει όλο και περισσότερη ορμή, και γίνεται όλο και ταχύτερη. Με την ίδια λογική, ο όρος ορμής αυξάνεται για τις παραμέτρους των οποίων οι παράγωγοι έχουν σταθερά την ίδια κατεύθυνση και μειώνεται για αυτές που οι παράγωγοί τους αλλάζουν συχνά. Έτσι, πετυχαίνουμε ταχύτερη σύγκλιση και λιγότερη ταλάντωση. Στις περισσότερες περιπτώσεις επιλέγεται $\gamma = 0.9$.

1.2.2.4.2: Momentum (Nesterov Accelerated Gradient)

Αν επανέλθουμε στη μεταφορά της ορμής ως μπάλα που κυλάει σε μία πλαγιά θα διαπιστώσουμε ότι μπορούμε να βελτιώσουμε την επίδοση του αλγορίθμου εάν, αντί η μπάλα να ακολουθεί τυφλά την κλίση της πλαγιάς, της δώσουμε την ευκαιρία να "βλέπει" που πηγαίνει, ώστε για παράδειγμα να μπορεί να ελαττώσει την ταχύτητά της όταν διαπιστώσει ότι η κλίση αρχίζει να ανεβαίνει.

Ο αλγόριθμος της ορμής με Nesterov accelerated gradient (NAG) [22] δίνει ως ένα βαθμό στον όρο της ορμής ακριβώς αυτή τη δυνατότητα. Ξέρουμε ότι θα χρησιμοποιήσουμε τον όρο γv_{t-1} για την ενημέρωση των παραμέτρων θ, οπότε υπολογίζοντας το $\theta - \gamma v_{t-1}$ παίρνουμε μία σχετικά καλή προσέγγιση για την επόμενη θέση των παραγώγων. Έτσι, ο αλγόριθμος μπορεί να δει καλύτερα την πορεία του αν υπολογίσει τις παραγώγους όχι ως προς τις τρέχουσες παραμέτρους θ, αλλά τη μελλοντική προσέγγισή τους:

$$v_t = \gamma v_{t-1} + \eta \cdot \nabla_\theta J(\theta - \gamma v_{t-1})$$
$$\theta = \theta - v_t$$



Σχήμα 1.2: Διαφορές μεταξύ απλής και NAG ορμής.

Η διαφορά μεταξύ της "απλής" ορμής και της ορμής NAG φαίνεται στο Σχήμα 1.2. Η "απλή" ορμή υπολογίζει τις τρέχουσες παραγώγους (μικρό μπλέ διάνυσμα) και στη συνέχεια εκτελεί ένα μεγάλο βήμα στην κατεύθυνση των ενημερωμένων και αθροισμένων παραγώγων (μεγάλο μπλε διάνυσμα). Αντίθετα, η ορμή NAG εκτελεί ένα μεγάλο βήμα στην κατεύθυνση των αθροισμένων παραγώγων του προηγούμενου βήματος (καφέ διάνυσμα) και στη συνέχεια εκτελεί ένα βήμα διόρθωσης που εμποδίζει τον αλγόριθμο να παρασυρθεί υπερβολικά (κόκκινο διάνυσμα), με το τελικό αποτέλεσμα να είναι η ολοκληρωμένη ενημέρωση (πράσινο διάνυσμα). Και εδώ ως παράγοντας ορμής συνήθως επιλέγεται $\gamma = 0.9$.

1.2.2.4.3: Adagrad

Ο αλγόριθμος Adagrad [23] προτάθηκε για την αντιμετώπιση του τρίτου προβλήματος: Αντί όλες οι παράμετροι θ να ανανεώνονται με έναν ενιαίο, πιθανώς μη-βέλτιστο learning rate, κάθε παράμετρος θ_i τη χρονική στιγμή t διαθέτει τον δικό της, ο οποίος προσαρμόζεται ανάλογα με τη συσχέτισή της με χαρακτηριστικά μεγάλης συχνότητας. Υψηλή συσχέτιση σημαίνει συχνά χαρακτηριστικά, οπότε έχουμε μικρά learning rates, και αντίστοιχα χαμηλή συσχέτιση σημαίνει σπάνια χαρακτηριστικά και μεγάλα learning rates. Αυτό επιτυγχάνεται με τη διαίρεση του learning rate με την τετραγωνική ρίζα του αθροίσματος των τετραγώνων των παραγώγων ως προς τη θ_i μέχρι τη στιγμή t:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot \nabla_\theta J(\theta_{t,i})$$

Το ε είναι ένας παράγοντας εξομάλυνσης που αποτρέπει τη διαίρεση με το μηδέν (συνήθως $\varepsilon = 1e-8$) και G_t είναι ένας διαγώνιος πίνακας στον οποίο αποθηκεύονται τα ζητούμενα αθροίσματα.

Παρ'όλο που ο Adagrad οδήγησε σε αισθητή αύξηση της απόδοσης των μοντέλων, η κεντρική ιδέα πίσω από τον αλγόριθμο έχει το πρόβλημα ότι στον παρονομαστή αθροίζονται θετικές, αύξουσες ποσότητες. Έτσι, οι learning rates δεν μπορούν παρά να μικραίνουν συνέχεια, μέχρι να φτάσουν στο σημείο να γίνουν απειροστά μικροί και ουσιαστικά το δίκτυο να μην μπορεί να αποροφήσει επιπλέον πληροφορίες. Ως λύσεις στο πρόβλημα αυτό προτάθηκαν αρκετοί αλγόριθμοι, τρεις από τους οποίους, ο RMSProp, ο Adadelta και ο Adam, παρουσιάζονται παρακάτω.

1.2.2.4.4: RMSprop

Η λύση που προτείνει ο RMSprop [24] είναι αρκετά απλή: Αντί να αποθηκεύονται οι παράγωγοι σε όλα τα βήματα από την αρχή της εκπαίδευσης, αποθηκεύονται μόνο οι τελευταίοι w. Έτσι, τυχόν μεγάλες ανανεώσεις στα πρώτα βήματα της εκπαίδευσης, όπου τα βάρη είναι στην ουσία τυχαία, σταματούν να έχουν σημασία καθώς το δίκτυο συγκλίνει, ενώ ως επιπλέον πλεονέκτημα του αλγορίθμου οι μικρές ανανεώσεις στα επόμενα στάδια της εκπαίδευσης τον καθιστούν πολύ πιο κατάλληλο για "τελειοποίηση" (fine tuning) των παραμέτρων. Οι προηγούμενες παράγωγοι όμως δεν αποθηκεύονται σε κάποια δομή δεδομένων μήκους w, αλλά ως ένας εκθετικά φθίνων μέσος όρος (exponentially decaying average), όπου και εδώ συνήθως $\gamma = 0.9$, $\varepsilon = 1e-8$, ενώ ο προτεινόμενος learning rate ισούται με 0.001:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) \cdot g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g_t$$

1.2.2.4.5: Adadelta

Ο Adadelta [25] προτάθηκε τον ίδιο περίπου καιρό με τον RMSprop, από ανεξάρτητες ομάδες. Βασίζεται στην ίδια κεντρική ιδέα, τη χρήση μόνο των τελευταιών w τιμών σε ένα exponentially decaying average, μόνο που αντί να την εφαρμόζει μόνο στο τετράγωνο των μερικών παραγώγων, την εφαρμόζει και στο τετράγωνο των ανανεώσεων των παραμέτρων:

$$
\begin{aligned}
E[g^2]_t &= \gamma E[g^2]_{t-1} + (1 - \gamma)\cdot g_t^2 \\
RMS[g]_t &= \sqrt{E[g^2]_t + \varepsilon} \\
E[\Delta\theta^2]_t &= \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma)\cdot \Delta\theta_t^2 \\
RMS[\Delta\theta]_t &= \sqrt{E[\Delta\theta^2]_t + \varepsilon} \\
\Delta\theta_t &= -\frac{RMS[\Delta\theta]_t}{RMS[g]_t}\cdot g_t \\
\theta_{t+1} &= \theta_t + \Delta\theta_t
\end{aligned}
$$

Αξίζει να σημειωθεί ότι ο Adadelta δεν απαιτεί καν τον ορισμό ενός learning rate, αφού δεν τον χρησιμοποιεί στην ανανέωση. Εμφανίζει παρόμοια συμπεριφορά με τον RMSprop, ενώ όπως πάντα $\gamma = 0.9$, $\varepsilon = 1e - 8$.

1.2.2.4.6: Adam

Ο Adam [26] συνδυάζει την κεντρική ιδέα των RMSprop και Adadelta με την ορμή, καθώς μαζί με τον γνωστό exponentially decaying average του τετραγώνου των παραγώγων, έστω v_t, αποθηκεύει και έναν exponentially decaying average των παραγώγων, έτσω m_t, παρόμοια με την ορμή:

$$
\begin{aligned}
m_t &= \beta_1\cdot m_{t-1} + (1 - \beta_1)\cdot g_t \\
v_t &= \beta_2\cdot v_{t-1} + (1 - \beta_2)\cdot g_t^2
\end{aligned}
$$

Τα m_t και v_t είναι εκτιμήσεις της πρώτης και δεύτερης ροπής των παραγώγων, αντίστοιχα (μέση τιμή και μη προσανατολισμένη διακύμανση). Τα διανύσματα αυτά αρχικοποιούνται με μηδενικά, κάτι που τα καθιστά "προδιαθετημένα" να λάβουν τιμές κοντά στο μηδέν, ειδικά στις αρχικές στιγμές ή όταν τα β_1 και β_2 είναι πολύ κοντά στο 1, όπως ισχύει συνήθως. Οι πολώσεις αυτές αντιμετωπίζονται με τον υπολογισμό "διορθωμένων" εκτιμήσεων των ροπών, που χρησιμοποιούνται τελικά για την ανανέωση των παραμετρων:

$$
\begin{aligned}
\hat{m}_t &= \frac{m_t}{(1 - \beta_1^t)} \\
\hat{v}_t &= \frac{v_t}{(1 - \beta_2^t)} \\
\theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon}\cdot \hat{m}_t
\end{aligned}
$$

Προτείνονται $\beta_1 = 0.9$, $\beta_2 = 0.999$ και $\varepsilon = 1e - 8$.

1.2.3: Back Propagation

Ο αλγόριθμος του back propagation χρησιμοποιείται στην εκπαίδευση νευρωνικών δικτύων, και συγκεκριμένα για τον υπολογισμό των μερικών παραγώγων της συνάρτησης απώλειας ως προς τα βάρη και τις πολώσεις τους, ώστε να πραγματοποιηθεί στη συνέχεια ένα βήμα gradient descent. Το back propagation μπορεί να εφαρμοστεί εαν η συνάρτηση απώλειας ικανοποιεί δύο συνθηκες: Πρώτον, να μπορεί να γραφτεί ως μέσος όρος των επιμέρους απωλειών των δεδομένων εισόδου,

και δεύτερον να μπορεί να γραφτεί ως συνάρτηση των εξόδων του δικτύου. Η πρώτη συνθήκη πρέπει να ισχύει διότι το back propagation υπολογίζει τις μερικές παραγώγους της συνάρτησης απώλειας για μία μόνο είσοδο, οπότε οι τελικές τιμές πρέπει να ανακατασκευαστούν με αυτόν τον τρόπο. Η δεύτερη συνθήκη πρέπει να ισχύει για να είναι καλά ορισμένες οι παράγωγοι του πρώτου από το τέλος στρώματος, από όπου και ξεκινάει ο υπολογισμός τους.

Εάν εξετάσουμε τον αλγόριθμο από καθαρά μαθηματική σκοπιά, το back propagation απλά κάνει χρήση του κανόνα της αλυσίδας. Αυτό όμως που το καθιέρωσε ήταν οι λεπτομέρειες της υλοποίησης των Rumelhart et al. [12], η οποία, αντί να υπολογίζει τις παραγώγους κατευθείαν ή με κάποια τεχνική από την αριθμητική ανάλυση, λάμβανε υπόψη της τον τρόπο λειτουργίας των υπολογιστών. Πιο συγκεκριμένα, ήταν ιεραρχική: Αρχικά υπολογιζόταν μία ενδιάμεση ποσότητα που είχε οριστεί ως το λάθος των νευρώνων στο layer εξόδου, ως η μερική παράγωγος της συνάρτησης απώλειας ως προς την είσοδό τους. Στη συνέχεια, με γνωστό το λάθος του layer i, υπολογιζόταν το λάθος του προηγούμενου layer, μέχρι να φτάσουμε στην είσοδο του δικτύου. Οι ζητούμενες παράγωγοι υπολογίζονταν ως εύκολα υπολογίσιμες συναρτήσεις των παραπάνω λαθών.

1.2.4: Vanishing Gradients

Στα δίκτυα πολλαπλών στρωμάτων της δεκαετίας του 1980 ξεκίνησε η πρακτική της προώθησης των εξόδων των νευρώνων σε μία μη-γραμμική συνάρτηση, που ονομάστηκε συνάρτηση ενεργοποίησης. Η χρήση της ήταν αναγκαία, διότι διαφορετικά η ολική συμπεριφορά του δικτύου θα ήταν ουσιαστικά γραμμική και όλο το δίκτυο θα μπορούσε να αντικατασταθεί από έναν πίνακα. Το δίκτυο θα ήταν δηλαδή ισοδύναμο με ένα δίκτυο ενός στρώματος, με όλα τα υπόλοιπα να είναι κυριολεκτικά άχρηστα. Ως συνάρτηση ενεργοποίησης κυριάρχησαν η σιγμοειδής και η υπερβολική εφαπτομένη, λόγω της ομοιότητάς τους με την πραγματική συμπεριφορά των βιολογικών νευρώνων.

Ωστόσο, η παράγωγος των συναρτήσεων αυτών είναι προβληματική: Πρώτον, η μέγιστη τιμή της είναι 0.25 για τη σιγμοειδή και 0.5 για την υπερβολική εφαπτομένη, δηλαδή στην καλύτερη περίπτωση θα έχουμε μείωση του μεγέθους των ανανεώσεων στο ένα τέταρτο και στο μισό, αντίστοιχα. Ακόμη, σε πιο κάπως ακραία σημεία η παράγωγος είναι αισθητά μικρότερη, κάτι που σημαίνει ότι οι ενημερώσεις στα βάρη ενός βαθιού δικτύου θα είναι όλο και μικρότερες καθώς προχωράμε προς τα πρώτα layers, λόγω των συνεχών πολλαπλασιασμών με ποσότητες που είναι πολύ κοντά στο μηδέν. Το πρόβλημα αυτό ήταν ο κύριος λόγος που δεν υπήρξε ουσιαστική πρόοδος στον τομέα των νευρωνικών δικτύων για περίπου είκοσι χρόνια, ενώ τα πρώτα βήματα για τη λύση του προβλήματος έγιναν μόλις το 2006 από τους Nair και Hinton [27].

1.2.5: Υπερπροσαρμογή (overfitting)

Το φαινόμενο της υπερπροσαρμογής (overfitting) αποτελεί ίσως την μεγαλύτερη πρόκληση στην εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Συμβαίνει όταν η ισχύς του είναι τόσο μεγάλη σε σχέση με την ποσότητα των δεδομένων εκπαίδευσης, που λέμε ότι αποτελούν το σύνολο εκπαίδευσης (training set), που αντί να μαθαίνει χρήσιμα χαρακτηριστικά και σχέσεις μεταξύ τους, μαθαίνει να "αποστηθίζει" τυφλά τα δεδομένα αυτά καθεαυτά. Το overfitting είναι απολύτως λογικό να συμβαίνει, αφού το gradient descent, αν εντοπίσει ότι στο μοντέλο υπάρχει "ελεύθερος χώρος", δηλαδή παράμετροι που δε χρησιμοποιούνται με αποδοτικό τρόπο καθώς δεν υπάρχουν αρκετά δεδομένα για να ανακαλυφθούν καινούρια μοτίβα, θα το οδηγήσει να μάθει "απ'έξω" τα δεδομένα εκπαίδευσής του, αφού έτσι ελαχιστοποιείται η απώλειά του.

Για τον εντοπισμό του overfitting αρχικά ένα μικρό αλλά όχι ασήμαντο ποσοστό των δεδομένων εκπαίδευσης, τις περισσότερες φορές 10% με 20%, αποκρύπτεται από το μοντέλο. Τα δεδομένα αυτά αποτελούν αυτό που ονομάζεται σύνολο επιβεβαίωσης (validation set) και σε τακτά χρο-

νικά διαστήματα, συνήθως στο τέλος κάθε εποχής, το μοντέλο κάνει προβλέψεις πάνω στο σύνολο αυτό. Overfitting υπάρχει όταν η μέση απώλεια επιβεβαίωσης (validation loss) είναι αισθητά μεγαλύτερη από την απώλεια εκπαίδευσης (training loss), ή παρόμοια όταν η επίδοση επιβεβαίωσης του μοντέλου υπό κάποια μετρική (accuracy, precision κ.λ.π.) είναι αισθητά χειρότερη από την επίδοση εκπαίδευσης. Αυτό πρακτικά σημαίνει ότι το δίκτυο δεν έχει δυνατότητα γενίκευσης, δεν μπορεί δηλαδή να "επεκτείνει" τη γνώση του σε δεδομένα στα οποία δεν έχει εκπαιδευτεί.

Οι τεχνικές που έχουν αναπτυχθεί για την αντιμετώπιση του overfitting εντάσσονται σε δύο γενικές κατηγορίες: Στην πρώτη ανήκουν αυτές που προσπαθούν να περιορίσουν την ισχύ του εκάστοτε μοντέλου, με μία προφανή ιδέα να είναι το σταμάτημα της εκπαίδευσης όταν παρατηρηθεί άυξηση της διαφοράς των δύο απωλειών ή σταθεροποίηση του validation loss (early stopping). Άλλες γνωστές τεχνικές είναι η l_1 και l_2 ομαλοποιήσεις (regularization), που εισάγουν στη συνάρτηση απώλειας επιπλέον όρους, και συγκεκριμένα τις l_1 και l_2 νόρμες των παραμέτρων του μοντέλου. Έτσι, στο μοντέλο δίνονται λιγότεροι βαθμοί ελευθερίας, και άρα είναι δυσκολότερο να υπερπροσαρμοστεί στα δεδομένα του.

Παράλληλα, υπάρχουν τεχνικές που προσπαθούν να δώσουν στο μοντέλο την εντύπωση ότι εκπαιδεύεται με περισσότερα δεδομένα από αυτά που υπάρχουν στο training set, και αποτελούν τη δεύτερη κατηγορία, τον εμπλουτισμό δεδομένων (data augmentation). Για παράδειγμα, στην περίπτωση ενός συνελικτικού νευρωνικού δικτύου που εκπαιδεύεται σε δεδομένα εικόνων (βλ. Κεφάλαιο 2) μερικές από τις πιο γνωστές τεχνικές είναι η τυχαία αλλαγή των χρωμάτων, η περιστροφή της εικόνας κατά μερικές μοίρες (rotation), ο "καθρεφτισμός" της εικόνας κατά τον οριζόντιο ή τον κάθετο άξονα (horizontal και vertical flipping), η εστίαση (zooming) κατά ένα μικρό ποσοστό, η οριζόντια ή κάθετη μετακίνηση της εικόνας κατά μερικά pixels και η αλλαγή της αντίθεσης.

Παρά τα τεράστιά πιθανά τους οφέλη, οι τεχνικές αυτές έχουν έναν πολύ σημαντικό περιορισμό: Το εύρος των τιμών των διάφορων τεχνικών, αλλά πολλές φορές ακόμα και η ίδια η δυνατότητα χρήσης τους καθορίζεται από τη φύση των δεδομένων εκπαίδευσης. Για παράδειγμα, αν ένα συνελικτικό δίκτυο εκπαιδεύεται να αναγνωρίζει ανθρώπινα πρόσωπα, το κάθετο flipping στην πραγματικότητα είναι πολύ πιθανό να χειροτερέψει την επίδοση, αφού το μοντέλο θα έχει εκπαιδευτεί πάνω σε δεδομένα που δεν έχουν φυσική σημασία, και άρα θα έχει σπαταλήσει πολύτιμο "χώρο" για την αποθήκευσή των μοτίβων τους.

Κεφάλαιο 2: Συνελικτικά Νευρωνικά Δίκτυα

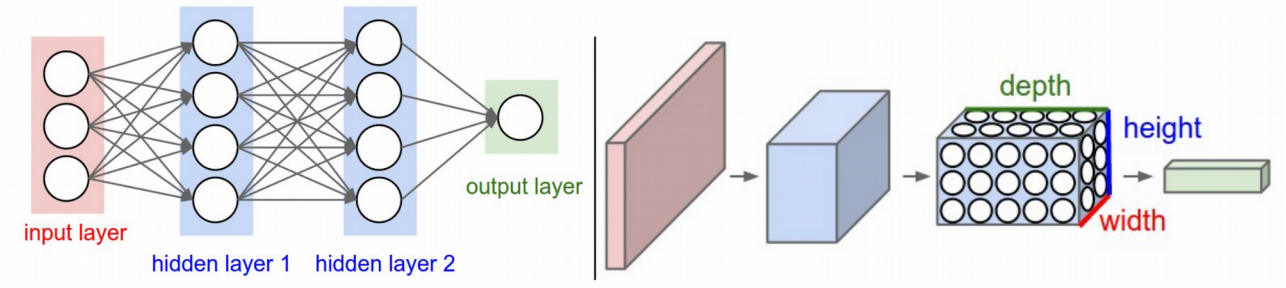
Τα συνελικτικά νευρωνικά δίκτυα (convolutional neural networks) χρησιμοποιούνται τυπικά σε εφαρμογές όπου τα δεδομένα είναι εικόνες και εκμεταλλεύονται το γεγονός αυτό κάνοντας κάποιους συμβιβασμούς που μειώνουν δραματικά τον αριθμό των παραμέτρων, διευκολύνοντας έτσι σημαντικά την εκπαίδευση και αυξάνοντας την αποδοτικότητα της υλοποίησης. Οι ιδέες που οδήγησαν στα δίκτυα αυτά διατυπώθηκαν αρχικά με βάση δύο ανακαλύψεις από την περιοχή της βιολογίας, την τοπικότητα των νευρώνων του οπτικού συστήματος, καθώς και την ιεραρχική αποθήκευση οπτικών πληροφοριών από τον εγκέφαλο.

Συγκεκριμένα, οι έρευνες των Hubel και Wiesel τη δεκαετία του 1960 [28], [29], έδειξαν ότι το οπτικό σύστημα των γατών και των μαϊμούδων αποτελείται από νευρώνες που ανταποκρίνονται σε αλλαγές σε συγκεκριμένες, πολύ μικρές περιοχές του οπτικού πεδίου, οι οποίες ονομάστηκαν περιοχές ευαισθητοποίησης (receptive fields). Διαχώρισαν τους νευρώνες αυτούς σε απλούς (S cells) και περίπλοκους (C cells). Οι απλοί νευρώνες έχουν μικρότερες περιοχές ευαισθητοποίησης και ενεργοποιούνται εάν αντιληφθούν την ύπαρξη ενός απλού γεωμετρικού χαρακτηριστικού, π.χ. γραμμή, γωνία, ακμή κ.λ.π. σε ένα συγκεκριμένο σημείο του χώρου και υπό συγκεκριμένη γωνία. Αντίθετα, η ενεργοποίηση ενός περίπλοκου νευρώνα, η περιοχή ευαισθητοποίησης του οποίου είναι αισθητά μεγαλύτερη και ανιχνεύει πιο σύνθετα χαρακτηριστικά, δεν εξαρτάται από τη θέση τους στο χώρο. Οι πληροφορίες από τους νευρώνες στη συνέχεια συνδυάζονται σε πολλαπλά επίπεδα από το οπτικό σύστημα για τη δημιουργία χρήσιμων πληροφοριών.

2.1: Αρχές Λειτουργίας

Ένα βαθύ συνελικτικό δίκτυο, στην κλασική του υλοποίηση [30], αποτελείται από μία σειρά συνελικτικών στρωμάτων (layers) και layers υποδειγματοληψίας, ακολουθούμενα από μία σειρά κλασικών fully connected layers. Το πρώτο μέρος του δικτύου είναι υπεύθυνο για την εκμαίευση των χρήσιμων χαρακτηριστικών από τις εικόνες, ενώ το δεύτερο, με χρήση των χαρακτηριστικών αυτών, πραγματοποιεί κάποιο άλλο task, όπως για παράδειγμα ταξινόμηση σε κάποια κλάση.

Σε αντίθεση με ένα παραδοσιακό νευρωνικό δίκτυο που αποθηκεύει τα δεδομένα του σε 1 διάσταση, ένας συνελικτικός νευρώνας τα αποθηκεύει σε 3, συνήθως μήκος, πλάτος, βάθος ή βάθος, μήκος, πλάτος (βλ. Σχήμα 2.1), όπου ο όρος βάθος αναφέρεται στον αριθμό των χαρακτηριστικών (features) της εικόνας και όχι στο βάθος του ίδιου του δικτύου (π.χ. μία έγχρωμη εικόνα 32×32 έχει 3 features: red, green, blue, και αναπαρίσταται ως ένας [32,32,3] πίνακας). Σε ένα συνελικτικό layer κεντρικό ρόλο παίζουν μικροί πίνακες που ονομάζονται φίλτρα ή πυρήνες (filters/kernels), ενώ η έξοδος του layer ονομάζεται χάρτης χαρακτηριστικών (feature map).



Σχήμα 2.1: Τρόπος λειτουργίας ενός fully connected δικτύου (αριστερά), και ενός συνελικτικού (δεξιά).

Τα φίλτρα ουσιαστικά δρουν ως μάσκες, αφού κάθε φορά πραγματοποιούν μία συνέλιξη με την εκάστοτε περιοχή των δεδομένων εισόδου σε όλο της το βάθος, και στη συνέχεια προχωρούν στην επόμενη. Η συνέλιξη αυτή καθεαυτή αποτελείται από δύο βήματα: Αρχικά πολλαπλασιάζο-

νται ανά στοιχείο το φίλτρο με τα δεδομένα, και στη συνέχεια προστίθενται τα στοιχεία του πίνακα που προκύπτει. Το φίλτρο έχει και αυτό 3 διαστάσεις, από τις οποίες μόνο 2 μπορούν να αλλαχθούν από το χρήστη, αυτές που ορίζουν το μέγεθος της συνέλιξης στο μήκος και το πλάτος της εικόνας εισόδου. Η τρίτη διάσταση, το βάθος, ισούται υποχρεωτικά με τον αριθμό των features του προηγούμενου επιπέδου.

Καθώς το φίλτρο "γλιστρά" κατά το μήκος και το πλάτος των δεδομένων εισόδου προκύπτει τελικά ένας δισδιάστατος χάρτης ενεργοποίησης, που αντιπροσωπεύει την "ανταπόκριση" του φίλτρου σε κάθε περιοχή. Σε κάποιο από τα πρώτα επίπεδα το φίλτρο ενεργοποιείται με την παρουσία ενός σχετικά απλού χαρακτηριστικού κάπου στην εικόνα, π.χ. μίας ακμής ή μίας γωνίας. Σε κάποιο από τα επόμενα επίπεδα όμως, καθώς τα χαρακτηριστικά συνδυάζονται και αυξάνονται σε πολυπλοκότητα, μπορεί να δείχνει την παρουσία ενός προσώπου, ενός αυτιού, ενός τροχού κλπ. Επαναλαμβάνοντας τη διαδικασία αυτή με πολλά φίλτρα δημιουργούνται διαφορετικοί χάρτες ενεργοποίησης, και βάζοντάς τους στη σειρά προκύπτει ο τελικός feature map του στρώματος.

Η έξοδος του συνελικτικού layer προωθείται ως είσοδος σε μία συνάρτηση ενεργοποίησης, ώστε να επιτευχθεί η απαιτούμενη μη-γραμμικότητα. Πλέον ως συνάρτηση ενεργοποίησης συνελικτικών layers χρησιμοποιείται σχεδόν εξ' ολοκλήρου η ReLU [27] ή κάποια παραλλαγή της (Leaky ReLU [31], PRELU [32], ELU [33]).

Δύο σημαντικά χαρακτηριστικά ενός συνελικτικού layer είναι το βήμα (stride) και το "γέμισμα" (padding). Το βήμα ορίζει την απόσταση που μεσολαβεί ανάμεσα σε δύο διαδοχικές εφαρμογές μίας συνέλιξης. Βήμα 1 σημαίνει ότι το φίλτρο θα κινείται κάθε φορά κατά 1 pixel. Βήμα μεγαλύτερο της μονάδας σημαίνει ότι υπάρχει μικρότερη επικάλυψη μεταξύ κοντινών pixels, ενώ το γεγονός ότι η έξοδος της συνέλιξης θα έχει αρκετά μικρότερες διαστάσεις από την είσοδο αναγκάζει το δίκτυο να μάθει ουσιώδη χαρακτηριστικά.

Στην περίπτωση που το μέγεθος του φίλτρου είναι μεγαλύτερο από τη μονάδα, η έξοδος της συνέλιξης θα έχει μικρότερες διαστάσεις από την είσοδο. Για το λόγο αυτό η είσοδος μπορεί να επεκταθεί (padding), συνήθως με μηδενικά (zero padding) ώστε αφ'ενός η είσοδος και η έξοδος να έχουν ίδιες διαστάσεις, και αφ'ετέρου για να υπάρχει μία δικλείδα ασφαλείας, ώστε να μη δημιουργηθούν θέματα λόγω επιλογής λάθος μεγέθους φίλτρου ή βήματος.

Η χρήση των φίλτρων ως μάσκες οφείλεται σε πρακτικούς λόγους: Εάν ακολουθούνταν τυφλά η λογική της βιολογίας και κάθε νευρώνας λάμβανε ως είσοδο μόνο ένα μικρό κομμάτι της εικόνας και ανίχνευε την παρουσία ενός χαρακτηριστικού μόνο στην περιοχή αυτή, τότε ο αριθμός των παραμέτρων θα γινόταν πολύ γρήγορα απαγορευτικός για χρήση συνελικτικών τεχνικών σε ουσιαστικές εφαρμογές. Αντ'αυτού, γίνεται η λογική υπόθεση ότι αν το δίκτυο κρίνει χρήσιμο να υπολογίσει ένα οπτικό χαρακτηριστικό σε μία περιοχή, το χαρακτηριστικό αυτό θα είναι χρήσιμο και σε άλλες περιοχές. Έτσι, όλοι οι νευρώνες του στρώματος μοιράζονται τις παραμέτρους τους στη διάσταση του βάθους, και ουσιαστικά έχουμε έναν νευρώνα ανά feature (parameter sharing).

Για παράδειγμα, έστω ότι έχουμε ένα συνελικτικό στρώμα με μέγεθος φίλτρου 5×5, βήματος 1, χωρίς χρήση padding, 16 χαρακτηριστικών, με είσοδο μία 32×32×3 εικόνα. Η έξοδος θα είναι ένας [27,27,16] πίνακας. Αν κάθε νευρώνας είχε τις δικές του, εκπαιδεύσιμες παραμέτρους, τότε θα απαιτούνταν 27·27·16·(5·5·3 + 1) = 886.464 παράμετροι. Χρησιμοποιώντας όμως parameter sharing απαιτούνται μόνο 16·(5·5·3 + 1) = 1216 παράμετροι.

Μετά από ένα ή περισσότερα συνελικτικά στρώματα τοποθετείται ένα στρώμα δειγματοληψίας (pooling layer), ώστε να μειωθεί η διάσταση των δεδομένων, αφ'ενός για πρακτικούς λόγους μείωσης της απαιτούμενης υπολογιστικής ισχύος και του χρόνου εκπαίδευσης, και αφ'ετέρου για μείωση του overfitting. Το πιο συνηθισμένο είδος pooling είναι το max-pooling, στο οποίο επιλέγεται και τροφοδοτείται η μέγιστη τιμή σε μία περιοχή των δεδομένων (συνήθως 2×2). Παρ'όλ'αυτά, υπάρχει μία τάση για εγκατάληψη των ντετερμινιστικών τεχνικών δειγματοληψίας και αντικατάστασής τους από συνελίξεις βήματος μεγαλύτερου της μονάδας, στη λογική των fully convolutional networks [34], ώστε το ίδιο το δίκτυο να μάθει μέσω back propagation τη δειγματο-

ληψία που οδηγεί στα καλύτερα αποτελέσματα. Τα layers συνελίξεων και δειγματοληψίας τυπικά ακολουθούνται από κάποια fully connected layers που αναλαμβάνουν το κομμάτι του classification.

Παρακάτω ακολουθούν οι πιο ιστορικά σημαντικές συνελικτικές αρχιτεκτονικές.

2.2: LeNet-5

Η πρώτη επιτυχής εφαρμογή των συνελικτικών δικτύων επιτεύχθηκε το 1998 από τους LeCun, Bottou, Bengio και Haffner [16], οι οποίοι εκπαίδευσαν ένα δίκτυο να αναγνωρίζει χειρόγραφους ASCII χαρακτήρες. Το δίκτυο, που ονόμασαν LeNet-5, χρησιμοποιήθηκε αργότερα μεταξύ άλλων σε τράπεζες, για την αυτοματοποίηση του περάσματος επιταγών στο ηλεκτρονικό σύστημα, και σε ταχυδρομεία, για διάβασμα ταχυδρομικών κωδικών.

Το LeNet-5 είχε 7 στρώματα. Ως είσοδο δεχόταν 32×32 ασπρόμαυρες εικόνες, τις οποίες προωθούσε σε ένα συνελικτικό layer (C1) 6 χαρακτηριστικών και 5×5 φίλτρων, χωρίς να γίνεται χρήση κάποιας μη γραμμικής συνάρτησης ενεργοποίησης. Το C1 παρήγαγε έναν πίνακα εξόοδου μεγέθους [28,28,6] και είχε (5·5 + 1)·6 = 156 εκπαιδεύσιμες παραμέτρους. Το επόμενο layer (S2) ήταν 2×2 υποδειγματοληψίας, που παρήγαγε ως έξοδο έναν [14,14,6] πίνακα. Τα στοιχεία κάθε 2×2 περιοχής αθροίζονταν, πολλαπλασιάζονταν με μία εκπαιδεύσιμη παράμετρο κλίμακας (1 για κάθε χαρακτηριστικό), αθροίζονταν με μία εκπαιδεύσιμη παράμετρο πόλωσης (ομοίως, συνολικά 6 + 6 = 12) και τέλος περνούσαν από τη σιγμοειδή συνάρτηση ενεργοποίησης.

Το δεύτερο συνελικτικό layer (C3) είχε μέγεθος φίλτρου 5 και 16 χαρακτηριστικά, με τη διαφορά ότι οι νευρώνες κάθε χαρακτηριστικού συνδέονταν μόνο με ένα υποσύνολο των χαρακτηριστικών του προηγούμενου επιπέδου, εκτός από του τελευταίου, που συνδεόταν με όλα. Τα 6 πρώτα συνδέονταν με όλες τις συνεχείς τριάδες χαρακτηριστικών (0-1-2, 1-2-3, …, 5-0-1), τα 6 επόμενα με τις συνεχείς τετράδες, τα επόμενα 3 με κάποιες μη συνεχείς τετράδες επιλεγμένες στην τύχη. Συνολικά το layer αυτό είχε 6·(3·5·5 + 1) + 6·(4·5·5 + 1) + 3·(4·5·5 + 1) +1·(6·5·5 + 1) = 1516 εκπαιδεύσιμες παραμέτρους. Η απόφαση αυτή λήφθηκε εν μέρει για να για να κρατηθεί το μέγεθος του δικτύου σε διαχειρίσιμα για το 1998 επίπεδα, κυρίως όμως για να καταπολεμηθεί το overfitting με τη λογική ότι εφ'όσον τα χαρακτηριστικά θα έχουν διαφορετικές εισόδους, θα μάθουν και διαφορετικές, εύρωστες εσωτερικές αναπαραστάσεις των δεδομένων. Ο πίνακας που προκύπτει έχει διαστάσεις [10,10,16].

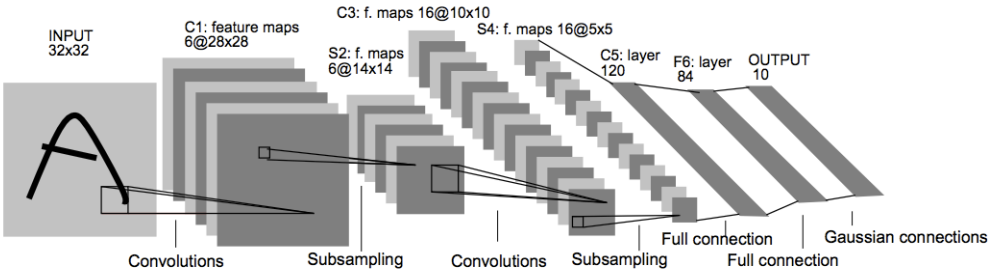
Το C3 ακολουθεί το δειγματοληπτικό layer S4, που λειτουργεί ακριβώς με τον ίδιο τρόπο με το S2, με 32 εκπαιδεύσιμες παραμέτρους, και παράγει έναν [5,5,16] πίνακα.

Το επόμενο συνελικτικό layer (C5) έχει 120 χαρακτηριστικά και μέγεθος φίλτρου 5×5, και το αποτέλεσμά του είναι ένας [1,1,120] πίνακας. Κάθε νευρώνας συνδέεται με όλα τα χαρακτηριστικά του S4, γι'αυτό και το C5 διαθέτει 120·(5·5·16 + 1) = 48.120 εκπαιδεύσιμες παραμέτρους. Στη λειτουργία του μοιάζει με ένα fully connected layer, αυτό συμβαίνει όμως επειδή οι δύο πρώτες διαστάσεις του S4 έτυχε να είναι ίδιες με το μέγεθος του φίλτρου του C5.

Το C5 ακολουθείται από το fully connected layer F6 των 84 νευρώνων, με 84·120 = 10.164 εκπαιδεύσιμες παραμέτρους. Ως συνάρτηση ενεργοποίησης χρησιμοποιείται η tanh. Το layer εξόδου έχει Radial Basis Function (RBF) units, ένα για κάθε κλάση, και κάθε unit μετρά το Mean Squared Error ανάμεσα στην έξοδο του προηγούμενου layer και σε ένα διάνυσμα 84 παραμέτρων. Το διάνυσμα αυτό ήταν μοναδικό για κάθε κλάση και προέκυψε από τη δειγματοληψία ενός αντιπροσωπευτικού χαρακτήρα της κλάσης αυτής σε ανάλυση 7×12 = 84.

Το LeNet-5 είχε ποσοστό λάθους 0.8% στο MNIST, ενώ το αντίστοιχο άρθρο ήταν και αυτό στο οποίο χρησιμοποιήθηκε για πρώτη φορά το dataset αυτό.

Στο Σχήμα 2.2 φαίνεται λεπτομερώς η δομή του LeNet-5.



Σχήμα 2.2: Η δομή του LeNet-5.

2.3: AlexNet

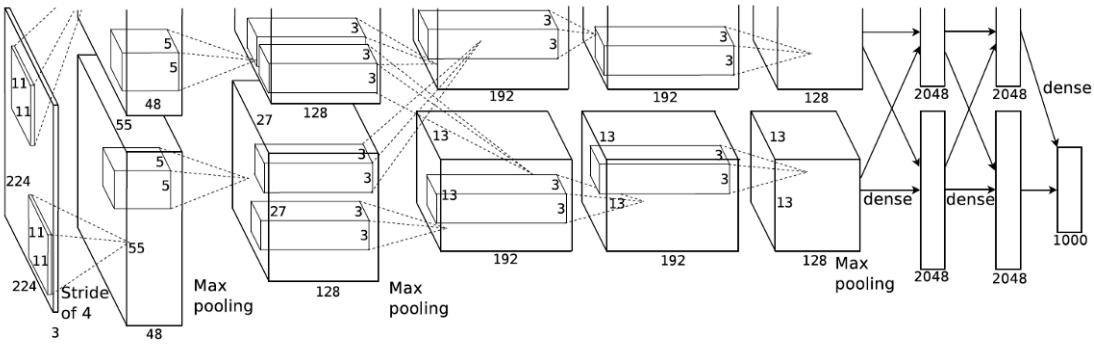
Το AlexNet [1] αγωνίστηκε στο ImageNet Large Scale Visual Recognition Challenge (ILSVRC) του 2012 και κατέκτησε την πρώτη θέση, με top-5 error 16% (η δεύτερη θέση είχε top-5 error 26%). Είχε παρόμοια αρχιτεκτονική με το LeNet-5, με τη διαφορά ότι ήταν πολύ μεγαλύτερο και πολύ πιο βαθύ, και περιείχε συνελικτικά layers το ένα πίσω από το άλλο, ενώ μέχρι τότε το συνηθισμένο ήταν κάθε συνελικτικό layer να ακολουθείται από layer δειγματοληψίας. Ακόμη, εφαρμόστηκαν νέες γνώσεις σχετικά με τα χαρακτηριστικά της κατανομής της τυχαίας αρχικοποίησης των βαρών για γρηγορότερη σύγκλιση [35]. Επίσης, σε αυτό χρησιμοποιήθηκε πρώτη φορά ευρέως η τεχνική του dropout [36], το τυχαίο σβήσιμο δηλαδή ενός ποσοστού των νευρώνων σε κάθε layer κατά τη διάρκεια της εκπαίδευσης, ώστε το δίκτυο να αναγκαστεί να μάθει χρήσιμα χαρακτηριστικά των δεδομένων του υπό αντίξοες συνθήκες.

Η πιο σημαντική ίσως διαφορά όμως ήταν η αντικατάσταση συναρτήσεων ενεργοποίησης όπως η σιγμοειδής ή η υπερβολική εφαπτομένη από την ReLU (Rectified Linear Unit), η οποία ορίζεται ώς εξής: $\mathrm{ReLU}(x) = \max(0, x)$. Η ReLU όπως φαίνεται είναι από τις πιο απλές μη γραμμικές συναρτήσεις και δεν απαιτεί περίπλοκους υπολογισμούς, ενώ παράλληλα το γεγονός ότι η παράγωγός της, σε περίπτωση που έχει ενεργοποιηθεί ο νευρώνας, είναι παντού ίση με τη μονάδα βοηθάει πάρα πολύ στην καταπολέμηση του προβλήματος των vanishing gradients. Τα δύο αυτά χαρακτηριστικά έχουν ως αποτέλεσμα η χρήση της ReLU να επιταχύνει την εκπαίδευση ενός δικτύου 5 με 6 φορές. Στο AlexNet η ReLU εφαρμοζόταν μετά από κάθε συνέλιξη.

Το AlexNet, η δομή του οποίου φαίνεται στο Σχήμα 2.3, δεχόταν ως είσοδο 224×224 εικόνες και είχε 8 layers, 5 convolutional και 3 fully connected. Το πρώτο convolutional είχε 96 φίλτρα, μέγεθος φίλτρου 11 και βήμα 4, με 96·(11·11·3 + 1) = 34.944 εκπαιδεύσιμες παραμέτρους, παρήγαγε πίνακα [55,55,96] και ακολουθούνταν από 2×2 max-pooling layer, με αποτέλεσμα πίνακα [27,27,96]. Το επόμενο συνελικτικό layer είχε 256 φίλτρα, μέγεθος φίλτρου 5, βήμα 1 και padding 2, είχε 256·(5·5·96 + 1) = 614.656 εκπαιδεύσιμες παραμέτρους, παρήγαγε πίνακα [27,27,256] και ακολουθούνταν από 2×2 max-pooling layer, με αποτέλεσμα πίνακα [13,13,256].

Στη συνέχεια έχουμε 3 συνελικτικά layers στη σειρά. Όλα είχαν μέγεθος φίλτρου 3, βήμα 1 και padding 1, με το πρώτο και το δεύτερο να έχουν 384 φίλτρα (384·(3·3·256 + 1) = 885.120 και 384·(3·3·384 + 1) = 1.327.488 εκπαιδεύσιμες παράμετροι αντίστοιχα), και το τρίτο 256 (256·(3·3·384 + 1) = 884.992 εκπαιδεύσιμες παράμετροι), με το τελικό αποτέλεσμα να είναι ένας [13,13,256] πίνακας. Στη συνέχεια υπήρχε 2×2 max-pooling layer που παρήγαγε [6,6,256] πίνακα. Στη συνέχεια εφαρμοζόταν dropout με ποσοστό απενεργοποίησης 0.5, και ακολουθούσαν 3 fully connected επίπεδα 4096, 4096 και 1000 νευρώνων, με (6·6·256)·4096 = 37.748.736, 4096·4096 = 16.777.216, 1000·4096 = 4.096.000 εκπαιδεύσιμες παραμέτρους αντίστοιχα. Εφαρμοζόταν dropout με ποσοστό απενεργοποίησης 0.5 ανάμεσα στα δύο πρώτα fully connected layers.

Συνολικά λοιπόν το δίκτυο είχε 62.3 εκατομμύρια παραμέτρους, από τις οποίες μόνο 3.7 εκατομμύρια (6%) ανήκαν στα συνελικτικά layers. Τα υπόλοιπα 58.6 εκατομμύρια (94%) ανήκαν στα fully connected.



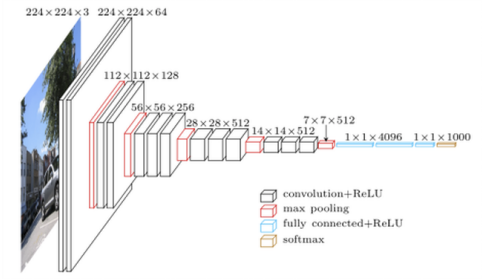
Σχήμα 2.3: Η δομή του AlexNet.

2.4: VGG-16

Το VGG-16 [37] κατέκτησε τη δεύτερη θέση στο ILSVRC του 2014 με top-5 error 7.3%, και έδειξε ότι το βάθος του δικτύου παίζει καθοριστική σημασία στη συνολική επίδοση. Είχε 16 συνελικτικά και fully connected layers, ενώ εμφανίζει μία πολύ πιο ομοιογενή αρχιτεκτονική από το AlexNet, αφού όλες οι συνελίξεις είχαν μέγεθος φίλτρου 3 και βήμα 1. Το πρώτο συνελικτικό μπλοκ είχε δύο layers 64 φίλτρων, το δεύτερο δύο 128 φίλτρων, το τρίτο τρία 256 φίλτρων και τέλος το τέταρτο τρία των 512 φίλτρων. Μεταξύ δύο μπλοκ υπήρχε max-pooling layer, ενώ ως συνάρτηση ενεργοποίησης χρησιμοποιήθηκε η ReLU. Έτσι, ξεκινώντας από μία εικόνα [224,224,3] φτάνουμε σε πίνακα [7,7,512] που συνδεόταν με το fully connected κομμάτι του δικτύου, το οποίο αποτελούνταν από 3 layers, 4096, 4096 και 1000 νευρώνων αντίστοιχα. Συνολικά το δίκτυο είχε 140 εκατομμύρια παραμέτρους, 123 από τα οποία ανήκαν στο fully connected μέρος. Συγκεκριμένα, 102 ανήκαν στο πρώτο layer, 16 στο δεύτερο και 4 στο τρίτο. Η δομή του VGG-16 φαίνεται στο Σχήμα 2.4.

Βλέπουμε λοιπόν ότι οι αρχιτεκτονικές αυτές ήταν πολύ απλοϊκές στη δομή τους: Αποτελούνταν από μία σειρά συνελικτικών layers με ενδιάμεσα max-pooling layers, ακολουθούμενα από κάποια fully connected στο τέλος. Όλες οι συνδέσεις ήταν σειριακές, ενώ το συντριπτικό ποσοστό των παραμέτρων χρησιμοποιούνταν για τα fully connected layers.

Έτσι, από πολύ νωρίς ξεκίνησε έρευνα για την εύρεση πιο περίπλοκων δομικών στοιχείων που θα προσφέρουν στο δίκτυο δυνατότητες μάθησης ισχυρότερων αναπαραστάσεων. Αυτό θα μείωνε και ιδανικά θα εξάλειφε την ανάγκη για τα τεράστια fully connected layers, κάτι που με τη σειρά του σημαίνει ότι ο χώρος που θα ελευθερωνόταν θα μπορούσε να χρησιμοποιηθεί με πολύ καλύτερα αποτελέσματα σε βαθύτερες και πιο ισχυρές αρχιτεκτονικές.

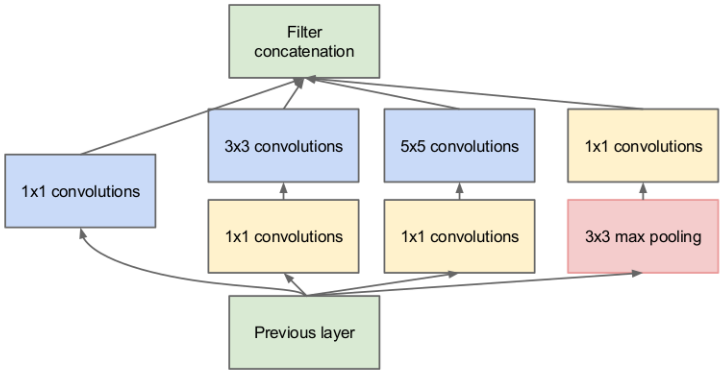


Σχήμα 2.4: Η δομή του VGG-16.

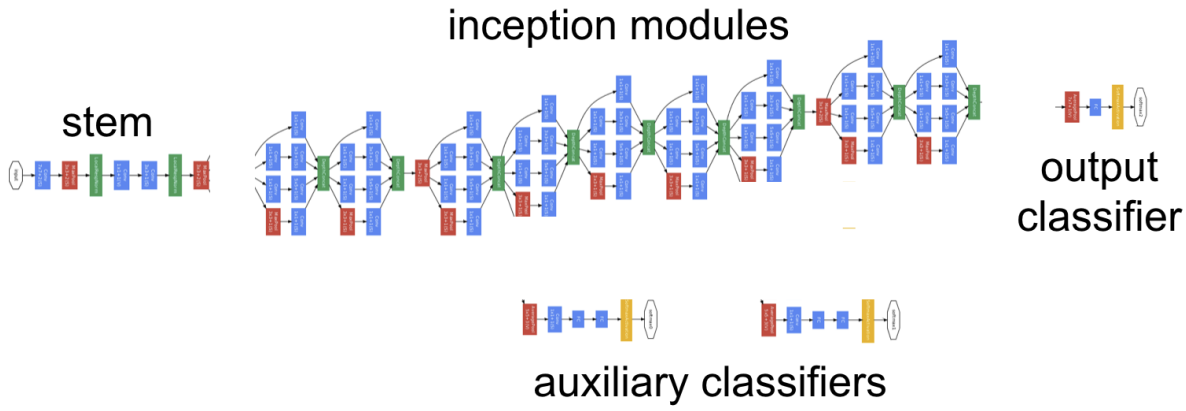
2.5: Inception

Η κεντρική ιδέα πίσω από τα Inception δίκτυα [38] είναι η εξής: Αντί ο σχεδιαστής της αρχιτεκτονικής να επιλέγει ο ίδιος το μέγεθος του πυρήνα ενός συνελικτικού layer (1×1 ή 3×3 για μοντελοποίηση λεπτομερειών σε μικρές περιοχές, 5×5, 7×7 κλπ για εύρεση χαρακτηριστικών αυξημένης οπτικής πολυπλοκότητας), χρησιμοποιούνται συνελίξεις με διαφορετικο μέγεθος πυρήνα πάνω στην ίδια είσοδο, οι οποίες στο τέλος ενώνονται σε έναν ενιαίο πίνακα, σε μία δομή που ονομάστηκε Inception module (βλ. Σχήμα 2.5). Η αρχιτεκτονική αυτή (βλ. Σχήμα 2.6), χρησιμοποιήθηκε για πρώτη φορά στο ILSVRC του 2014, κερδίζοντας την πρώτη θέση με top-5 validation error, και μέχρι σήμερα έχουν βγει 5 εκδοχές (v1, v2, v3, v4, Inception-ResNet [39]), και στην αρχική χρησιμοποιούνταν συνελίξεις 1×1, 3×3 και 5×5, μαζί με ένα 3×3 max-pooling layer με padding και βήμα 1 ακολουθούμενο από 1x1 συνέλιξη.

Η μεγαλύτερη καινοτομία του Inception module ήταν η εκτενής χρήση 1×1 συνελίξεων με αισθητά μικρότερο αριθμό φίλτρων από την είσοδο πριν τις 3×3 και 5×5 συνελίξεις. Αυτό είχε ως αποτέλεσμα τη μείωση του υπολογιστικού κόστους στο ένα δέκατο, καθιστώντας έτσι εφικτές από τη μία την εκπαίδευση των δικτύων σε λιγότερο χρόνο, και από την άλλη, ίσως πιο σημαντικά, τη γρήγορη δοκιμή εναλλακτικών στρατηγικών για περαιτέρω μείωση του κόστους και αύξηση της ακρίβειας (π.χ. οι 5×5 συνελίξεις αντικαταστάθηκαν από δύο 3×3 στη σειρά). Πράγματι, η εκδοχή 1 είχε 22 layers, και με μόλις 4 εκατομμύρια παραμέτρους είχε top-5 validation error 6.67%, ενώ οι εκδοχές 2 και 3 εκδόθηκαν στο ίδιο άρθρο με την πρώτη.



Σχήμα 2.5: Το inception module του Inception-v1, a.k.a. GoogLeNet.



Σχήμα 2.6: Η δομή του Inception-v1 a.k.a GoogLeNet.

2.6: Residual Αρχιτεκτονικές

Το VGG-16 έδειξε ότι η αύξηση του βάθους του δικτύου είναι καίριος παράγοντας για τη συνολική του επίδοση, οπότε η λογική συνέχεια ήταν η προσπάθεια εκπαίδευσης όλο και πιο βαθιών δικτύων. Όσο πιο βαθύ είναι όμως ένα δίκτυο, τόσο πιο δύσκολο είναι να εκπαιδευτεί. Η κοινή γνώμη ήταν ότι η δυσκολία αυτή οφειλόταν στο μεγαλύτερο βαθμό στο πρόβλημα των vanishing gradients, ωστόσο βρέθηκε ότι εξίσου σημαντικό ήταν και το πρόβλημα του identity mapping.

Έστω πως έχουμε ένα δίκτυο Α, συγκεκριμένου βάθους, που παράγει ποσοστό λάθους x. Έστω πως "παγώνουμε" τα βάρη του Α και στο τέλος του τοποθετούμε μερικά επιπλέον layers, ώστε να προκύψει το δίκτυο Β. Η διαίσθηση λέει ότι το ποσοστό λάθους του Β δε θα έπρεπε να ξεπερνάει το x, αφού στη χειρότερη περίπτωση, όταν δηλαδή το δίκτυο δεν μπορεί να μάθει τίποτα αξιόλογο, τα layers του Β μπορούν απλά να αντιγράψουν την είσοδό τους (F(x) = x, identity mapping). Πειραματικές μετρήσεις όμως ανάμεσα στο VGG-19 (μία λίγο πιο ισχυρή εκδοχή του VGG-16) και σε ένα δίκτυο παρόμοιας σειριακής συνελικτικης αρχιτεκτονικής 34 στρωμάτων, εκπαιδευμένα και τα δύο στο ImageNet, έδειξαν ότι το validation error του βαθύτερου δικτύου ήταν μεγαλύτερο από το αντίστοιχο του ρηχότερου (28.54% έναντι 27.94%), οπότε το πρόβλημα του identity mapping αποδείχθηκε πολύ πιο δύσκολο απ'ότι αρχικά φαινόταν.

Ως πιθανή λύση για τα δύο προβλήματα οι He et al. [40], που διαγωνίστηκαν στο ILSVRC του 2015, πρότειναν τη χρήση δομικών στοιχείων που ονόμασαν residual blocks (βλ. Σχήμα 2.7). Υπέθεσαν ότι, σε περίπτωση που ένα layer πρέπει να υλοποιεί μία συνάρτηση $x \rightarrow y$, όπου $y = H(x)$, το Η είναι πολύ πιο εύκολο να οριστεί ως $H(x) = F(x) + x$, να συνδεθεί δηλαδή απευθείας η είσοδος του layer με την έξοδο (shortcut), και στη συνέχεια, εάν απαιτείται identity mapping, το F να αναγκαστεί από το gradient descent στο 0.

Πράγματι, μετρήσεις έδειξαν ότι η εισαγωγή residual συνδέσεων σε μοντέλα της κλασικής αρχιτεκτονικής οδηγεί σε πολύ καλά αποτελέσματα: Το validation error του residual δικτύου 18 στρωμάτων ήταν 27.88%, ενώ, πιο σημαντικά, το αντίστοιχο του residual δικτύου 34 στρωμάτων ήταν 25.03%, αισθητά δηλαδή χαμηλότερο.

Στο άρθρο αναφέρθηκαν 5 residual αρχιτεκτονικές (ResNet) με 18, 34, 50, 101 και 152 layers αντίστοιχα. Ένα ensemble μοντέλο 6 ResNet διαφορετικών βαθών είχε 3.57% top-5 validation error, κερδίζοντας την πρώτη θέση του διαγωνισμού.

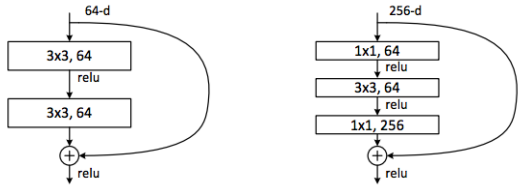
Ένα μεγάλο ποσοστό της επιτυχίας στην εκπαίδευση τόσο βαθιών αρχιτεκτονικών ήταν ότι τα ResNet ήταν οι πρώτες αρχιτεκτονικές στις οποίες έγινε εκτενής χρήση του batch normalization [41], μίας τεχνικής που χρησιμοποιείται για την καταπολέμηση του covariate shift, της αλλαγής δηλαδή της κατανομής των δεδομένων αφού περάσουν από ένα layer, κάτι που καθιστά την εκπαίδευση εξαιρετικά ασταθή. Αρχικά υπολογίζονται η μέση τιμή και η τυπική απόκλιση του τρέχοντος batch, μετά τα δεδομένα κανονικοποιούνται και στη συνέχεια προστίθενται και πολλαπλασιάζονται με δύο εκπαιδεύσιμες παραμέτρους, ώστε το ίδιο το δίκτυο να αποφασίσει, μέσω gradient descent, τα βέλτιστα χαρακτηριστικά της κατανομής των δεδομένων του. Το batch normalization αυξάνει δραματικά τη σταθερότητα του δικτύου, κάτι που επέτρεψε για πρώτη φορά την εκπαίδευση πραγματικά βαθιών δικτύων. Παράλληλα, χρησιμοποιήθηκε για πρώτη φορά η τυχαία αρχικοποίηση He (He initialization) των βαρών κάθε layer [32].

Προτάθηκαν δύο είδη residual blocks ανάλογα με το είδος του shortcut που υλοποιούν. Σε περίπτωση που οι διαστάσεις εισόδου και εξόδου ενός layer είναι ίδιες, τότε η είσοδος απλά προστίθεται στην έξοδο, χωρίς κάποια επιβάρυνση στην πολυπλοκότητα. Διαφορετικά, η είσοδος περνάει από 1x1 συνελικτικό layer με τα κατάλληλα χαρακτηριστικά, με μία πολύ μικρή αύξηση στο συνολικό αριθμό παραμέτρων.

Το κυρίως κομμάτι του residual block στις αρχιτεκτονικές των 50, 101, και 152 layers αποτελείται από 3 συνελικτικά layers στη σειρά. Το πρώτο χρησιμοποιεί 1×1 συνελίξεις για να περιορίσει τον αριθμό των χαρακτηριστικών από το προηγούμενο layer, λειτουργώντας δηλαδή ως

bottleneck για τον περιορισμό του αριθμού των παραμέτρων, με τον ίδιο τρόπο με το Inception module. Το επόμενο πραγματοποιεί 3×3 συνελίξεις με zero padding, με βήμα 2 εάν πραγματοποιείται pooling, διαφορετικά με 1, και ίδιο αριθμό φίλτρων με το προηγούμενο layer, και τέλος το τρίτο layer πραγματοποιεί 1×1 συνελίξεις με τέσσερις φορές τα φίλτρα των προηγούμενων layers, ώστε να συνδυαστούν τα χαρακτηριστικά με ωφέλιμο τρόπο.

Στις μικρότερες αρχιτεκτονικές που δεν υπάρχει τόση ανάγκη για περιορισμό του αριθμού των παραμέτρων το σειριακό κομμάτι του residual block αποτελείται από 2 συνελικτικά layers, ίδιου αριθμού φίλτρων, μέγεθος φίλτρου 3×3 με εφαρμογή zero padding. Τα δύο είδη residual blocks φαίνονται στην παρακάτω εικόνα.



Σχήμα 2.7: Τα δύο είδη των residual blocks.

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Πίνακας 2.1: Επιδόσεις των ResNet στο validation set του ImageNet.

Στον Πίνακα 2.2 υπάρχουν περιλήψεις των αρχιτεκτονικών που εξετάστηκαν, ενώ στον Πίνακα 2.1 φαίνονται οι επιδόσεις διάφορων ResNet στο validation set του ImageNet, με το πιο σημαντικό συμπέρασμα να είναι ότι υπάρχει μία σταθερή μείωση στο error καθώς αυξάνουμε τον αριθμό των layers. Το γεγονός αυτό οδήγησε τους συγγραφείς να υποθέσουν ότι η παράμετρος που επηρεάζει περισσότερο την επίδοση είναι το βάθος του δικτύου. Ωστόσο, οι Zagoruyko και Komodakis [42] μέσα από εκτενή πειράματα έδειξαν ότι ακόμα και η παραμικρή βελτίωση του state of the art όσον αφορά την ακρίβεια στην ταξινόμηση απαιτεί το διπλασιασμό του αριθμού των layers, κάτι που δείχνει την φθίνουσα επαναχρησιμοποίηση χαρακτηριστικών σε πολύ βαθιά residual δίκτυα, με αποτέλεσμα τα δίκτυα αυτά να απαιτούν απαγορευτικά πολύ χρόνο για να εκπαιδευτούν.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Πίνακας 2.2: Δομές των διάφορων ResNet που εξετάστηκαν αρχικά.

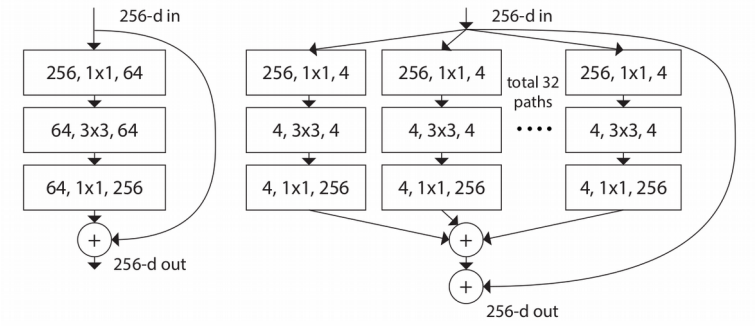
Ως λύση στο προβλημα αυτό πρότειναν τη χρήση μεγαλύτερου αριθμού καναλιών, την αύξηση δηλαδή του "πλάτους" των δικτύων και εκπαίδευσαν ένα Wide ResNet (WRN) με 16 layers που είχε καλύτερα αποτελέσματα στα CIFAR-10 και CIFAR-100 από βαθιά ResNet με περισσότερα από 1000 layers και 8 φορές λιγότερα κανάλια ανά layer. Επίσης, εκπαίδευσαν ένα WRN με 50 layers και 2 φορές περισσότερα κανάλια ανά layer που ξεπερνούσε σε ακρίβεια τα αρχικά δίκτυα των 50, 101 και 152 layers. Έδειξαν δηλαδή ότι η δύναμη των ResNet οφείλεται περισσότερο στα residual blocks και ότι το βάθος τους απλά προκύπτει ως συνέπεια της χρήσης τους.

Οι Xie et al. [43] με τη σειρά τους έδειξαν ότι, όπως η συνεχής αύξηση του πλάτους οδηγεί σε όλο και λιγότερη επαναχρησιμοποίηση χαρακτηριστικών, έτσι και η συνεχής αύξηση του πλάτους μπορεί να οδηγήσει το δίκτυο να μάθει άχρηστες ή και λάθος πληροφορίες (θόρυβος). Εισήγαγαν μία ακόμα παράμετρο, την πολλαπλότητα του δικτύου C (cardinality), εξίσου σημαντική με το βάθος και το πλάτος του, αφού αφ'ενός η αύξησή της οδηγεί σε καλύτερα αποτελέσματα σε σχέση με την αύξηση των παραπάνω παραδοσιακών παραμέτρων για την ίδια αύξηση της χωρητικότητας του δικτύου, και αφ'ετέρου γιατί συνεχίζει να προσφέρει βελτίωση όταν το βάθος και το πλάτος σταματούν.

Το ResNeXt block (βλ. Σχήμα 2.8) εμπνεύστηκε στον σχεδιασμό του από το Inception module, και συγκεκριμένα τη λογική split-transform-merge, ακολουθώντας μία δομή που οι συγγραφείς ονόμασαν split-transform-aggregate: Υπάρχουν C στον αριθμό paths μέσα στο block, και όλα ακολουθούν την ίδια τοπολογία. Αποτελούνται από 3 συνελικτικά layers στη σειρά, με το πρώτο να λειτουργεί ως bottleneck, πραγματοποιώντας 1×1 συνελίξεις με έναν αρκετά μικρό αριθμό φίλτρων (στο άρθρο χρησιμοποιείται και προτείνεται 4). Το δεύτερο layer πραγματοποιεί 3×3 συνελίξεις με τον ίδιο αριθμό φίλτρων, και τελικά το τρίτο πραγματοποιεί 1×1 συνελίξεις, με τη διαφορά ότι ο αριθμός των φίλτρων είναι πολύ μεγαλύτερος. Σκοπός κάθε path είναι να μάθει να αναπαριστά πολύ καλά έναν συγκεκριμένο υποχώρο της κατανομής των δεδομένων, με τη γενικότερη δύναμη του μοντέλου να προκύπτει από το άθροισμα των επιμέρους paths.

Με άλλα λόγια, το δίκτυο συνδυάζει τις δύο κύριες ιδέες στο σχεδιασμό συνελικτικών δικτύων: τις residual συνδέσεις, που διευκολύνουν τη βελτιστοποίηση των παραμέτρων, και τους συναθροιστικούς (aggregated) μετασχηματισμούς, που προσφέρουν μεγαλύτερη ισχύ στις αναπαραστάσεις. Επιπλέον, η ομοιόμορφη τοπολογία των επιμέρους paths σημαίνει ότι πρακτικά η μόνη υπερπαράμετρος που έχει σημασία να εξεταστεί και να επιλεχθεί με προσοχή είναι η πολλαπλότητα του δικτύου, αντίθετα με τα Inception modules που απαιτούν το fine tuning αρκετών παραμέτρων, όπως το μέγεθος των διάφορων πυρήνων του κάθε layer.

Ένα ensemble μοντέλο από ResNeXt διαφόρων μεγεθών έλαβε την δεύτερη θέση στο ILSVRC του 2016, με top-5 error 3.03%, ενώ την πρώτη θέση έλαβε η αρχιτεκτονική DenseNet [44], που πήγε τη χρήση παρακάμψεων ένα βήμα παραπέρα εισάγοντας ως δομικό στοιχείο το dense block, με κάθε dense block να συνδέεται με όλα τα προηγούμενα.



Σχήμα 2.8: Ένα ResNeXt block πολλαπλότητας 32 (δεξιά) σε σύγκριση με ένα απλό ResNet block (αριστερά).

Κεφάλαιο 3: Γεννητικά Μοντέλα

Όλα τα μοντέλα που έχουν αναφερθεί μέχρι τώρα είναι "διακριτικά" (discriminative), με την έννοια ότι παίρνουν ως είσοδο δεδομένα και παράγουν ως έξοδο πληροφορίες για τα δεδομένα αυτά, π.χ. σε ποια κλάση ανήκουν, ποιο είναι το πιο κοντινό δεδομένο σε αυτά ως προς κάποια μετρική απόστασης κ.λ.π. Υπάρχουν όμως και τα λεγόμενα γεννητικά (generative) μοντέλα, κύριος στόχος των οποίων είναι, με είσοδο κάποια πληροφορία, η δημιουργία νέων, τεχνητών δεδομένων που μοντελοποιούν όσο το δυνατόν καλυτερα την πληροφορία αυτή. Πιο συγκεκριμένα, τα διακριτικά μοντέλα είναι υπό συνθήκη πιθανοτικά μοντέλα του στόχου Y με δοσμένο το δεδομένο x που ακολουθεί την κατανομή X, P(Y | X = x), ενώ τα generative μοντέλα είναι υπό συνθήκη πιθανοτικά μοντέλα του X με δοσμένο το y που ακολουθεί την κατανομή Y, P(X | Y = y). Στο πεδίο της μηχανικής μάθησης έχουν αναπτυχθεί τρεις μεγάλες οικογένειες γεννητικών μοντέλων: οι μηχανές Boltzmann, οι αυτοκωδικοποιητές (autoencoders) και τα γεννητικά ανταγωνιστικά δίκτυα (Generative Adversarial Networks, GANs).

3.1: Boltzmann Machines

Οι μηχανές Boltzmann (Boltzmann machines, BM) [45], [46] αποτελούν είδος στοχαστικού επαναλαμβανόμενου/αναδρομικού (recurrent) δικτύου, μπορούν να θεωρηθούν ειδική περίπτωση των δικτύων Hopfield [47] και οφείλουν το όνομά τους στην κατανομή Boltzmann, στην οποία και βασίζουν τη λειτουργία τους. Η δομή μίας τυπικής μηχανής Boltzmann φαίνεται στο Σχήμα 3.1.

Οι νευρώνες μίας BM λαμβάνουν δύο μόνο τιμές, 0 και 1, ενώ χωρίζονται σε ορατούς (visible), το στρώμα εισόδου, και κρυμμένους (hidden). Κεντρικό ρόλο στην εκπαίδευσή της παίζει η λεγόμενη συνάρτηση ενέργειας, την οποία το δίκτυο προσπαθεί να ελαχιστοποιήσει, η οποία ορίζεται ως εξής: $E = \sum_{i<j} (w_{ij} s_i s_j) + \sum_i \theta_i s_i$, όπου $w_{ij} = w_{ji}$ η δύναμη της σύνδεσης μεταξύ των νευρώνων i και j, s_i η κατάσταση του νευρώνα i και θ_i η πόλωση του νευρώνα i. Αν θεωρήσουμε τη διαφορά ΔE_i της ενέργειας του συστήματος όταν ο νευρώνας i είναι ενεργοποιημένος από την αντίστοιχη όταν είναι απενεργοποιημένος, και αντικαταστήσουμε τις δύο με τις σχετικές τους πιθανότητες (σύμφωνα με την ιδιότητα της κατανομής Boltzmann ότι η ενέργεια μίας κατάστασης είναι ανάλογη του αρνητικού λογαρίθμου της πιθανότητας αυτής) προκύπτει ότι η πιθανότητα να είναι ενεργοποιημένος ο νευρώνας i ισούται με $p_{i=1} = (1 + \exp(-\Delta E_i/T))^{-1}$, όπου το T είναι μία παράμετρος που αντιπροσωπεύει τη "θερμοκρασία" του συστήματος, ή αλλιώς την ευαισθησία του σε αλλαγές της ενέργειας. Η μορφή της παραπάνω πιθανότητας οδήγησε στην ευρεία χρήση της σιγμοειδούς συνάρτησης στις BMs.

Το δίκτυο εκπαιδεύεται με την επαναλαμβανόμενη τυχαία επιλογή ενός νευρώνα, την απόπειρα για επαναφορά της κατάστασής του και την εξέταση της αλλαγής αυτής στην ενέργεια του δικτύου. Ευνοούνται αλλαγές που οδηγούν σε μείωσή της, ενώ μετά από κάποιες επαναλήψεις σε μία συγκεκριμένη θερμοκρασία, η πιθανότητα μίας κατάστασης του δικτύου εξαρτάται μόνο από την ολική ενέργεια της κατάστασης αυτής, και όχι από την αρχική κατάσταση στην αρχή της εκπαίδευσης. Τότε λέμε ότι το δίκτυο έχει φτάσει σε θερμική ισορροπία. Στη συνέχεια η θερμοκρασία του συστήματος πολλαπλασιάζεται με μία σταθερά $c < 1$, πχ $c = 0.95$, και η διαδικασία ξεκινά από την αρχή, είτε μέχρι η θερμοκρασία να περάσει κάποιο επιθυμητό όριο, είτε μέχρι να μην μπορεί να γίνει κάποια αλλαγή στην κατάσταση του συστήματος για κάποιο αριθμό εποχών.

Πιο συγκεκριμένα, η ενημέρωση των βαρών των συνδέσεων γίνεται με gradient descent, μέσω της ελαχιστοποίησης της KL απόκλισης μεταξύ της κατανομής των ορατών νευρώνων και της κατανομής των κρυμμένων όταν έχει επιτευχθεί θερμική ισορροπία.

Το βασικό θεωρητικό πλεονέκτημα των BMs απέναντι σε άλλες αρχιτεκτονικές (π.χ. multi-layer perceptrons) από άποψη βιολογίας και νευροεπιστήμης ήταν ότι η κατάσταση ενός νευρώνα εξαρτάται μόνο από τις καταστάσεις των νευρώνων που συνδέονται με αυτόν, και όχι από όλους τους υπόλοιπους. Παρά όμως τα θεωρητικά τους πλεονεκτήματα, υπήρξαν πρακτικά προβλήματα που τις εμπόδισαν, στη γενική περίπτωση, από το να χρησιμοποιηθούν ευρέως: Πρώτον, ο χρόνος που απαιτείται για να φτάσει κάθε φορά το δίκτυο σε κατάσταση θερμικής ισορροπίας φάνηκε εμπειρικά να αυξάνεται εκθετικά αναφορικά με το μέγεθος του δικτύου, ενώ σε περίπτωση που οι πιθανότητες ενεργοποίησης των νευρώνων βρίσκονται στη μέση του διαστήματος [0,1] τα βάρη των συνδέσεων έτειναν να ταλαντώνονται έντονα χωρίς κάποιο νόημα, με αποτέλεσμα στην ουσία να αλλάζουν τυχαία, μέχρι να φτάσουν σε κατάσταση κορεσμού.

3.1.1: Restricted Boltzmann machines

Η περιορισμένη μηχανή Boltzmann (RBM) [48] είναι μία μηχανή Boltzmann με τον περιορισμό ότι δεν επιτρέπονται συνδέσεις μεταξύ νευρώνων του ίδιου επιπέδου. Η RBM λοιπόν μοιάζει στη δομή της με ένα fully connected νευρωνικό δίκτυο δύο επιπέδων (συγκεκριμένα, η RBM έχει τη μορφή ενός διμερούς γράφου). Η διαφορά όμως με τα κλασσικά νευρωνικά δίκτυα είναι ότι η έξοδος του κρυμμένου επιπέδου της RBM τροφοδοτείται πιθανοτικάπίσω στο δίκτυο ως είσοδος σε ένα αντίστροφο πέρασμα που καταλήγει στο ορατό στρώμα, οι έξοδοι του οποίου είναι ανακατασκευές των δεδομένων εισόδου.

Οι διαφορές των RBMs με τους autoencoders εντοπίζονται στο ότι οι πρώτες διαθέτουν διάνυσμα πόλωσης (bias) όχι μόνο στο κρυμμένο στρώμα, αλλά και στο στρώμα εισόδου, για τις ανάγκες του αντίστροφου περάσματος, καθώς και στον τύπο δεδομένων που δέχονταν, τουλάχιστον στις αρχικές τους υλοποιήσεις, και τον τρόπο εκπαίδευσής τους. Οι RBMs, όπως και οι γενικές BMs αρχικά λειτουργούσαν μόνο με δυαδικά δεδομένα, ενώ εκπαιδεύονταν με τον αλγόριθμο contrastive divergence [18], που χρησιμοποιεί σε πολύ μεγάλο βαθμό την τυχαιότητα, σε αντίθεση με τους τυπικούς αλγορίθμους gradient descent με χρήση back propagation που χρησιμοποιούνται για την εκπαίδευση autoencoders.

Συγκεκριμένα, αρχικά υπολογίζονται οι ενεργοποιήσεις του κρυμμένου επιπέδου με είσοδο ένα δείγμα v, με χρήση της σιγμοειδούς συνάρτησης, ώστε να μπορούν να χρησιμοποιηθούν ως πιθανότητες. Με βάση τις πιθανότητες αυτές δειγματολειπτείται ένα διάνυσμα ενεργοποίησης h. Από το h ανακτάται μία ανακατασκευή v' του δεδομένου εισόδου, με βάση την οποία υπολογίζεται το διάνυσμα ενεργοποίησης h'. Στη συνέχεια, υπολογίζονται τα εξωτερικά γινόμενα μεταξύ των v και h (θετική παράγωγος) και των v' και h' (αρνητική παράγωγος). Η διαφορά της αρνητικής παραγώγου από τη θετικη, πολλαπλασιασμένη με μία σταθερά, το ρυθμό μάθησης, προστίθεται στον πίνακα βαρών $\Delta W=\varepsilon\cdot(vh^T-v'h'^T)$, ενώ οι πολώσεις ενημερώνονται ανάλογα: $\Delta bias_v=\varepsilon(v-v')$, $\Delta bias_h=\varepsilon\cdot(h-h')$. Η παραπάνω διαδικασία μπορεί να εφαρμοστεί πολλές φορές, αλλά τυπικά εφαρμόζεται μόνο μία.

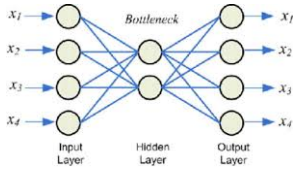
Όπως και στην περίπτωση των autoencoders, μπορούμε να ενώσουμε πολλές RBMs για τη δημιουργία ενός λεγόμενου βαθιού δικτύου πίστης (deep belief network) [49], ως εξής: Μόλις το δίκτυο μάθει τη δομή των δεδομένων ως προς το πρώτο κρυμμένο layer, το layer αυτό γίνεται το ορατό και ως είσοδοι στο δίκτυο θεωρούνται πλέον οι ενεργοποιήσεις του. Το δεύτερο κρυμμένο layer, για να ανακατασκευάσει τις ενεργοποιήσεις του πρώτου, μαθαίνει ακόμα πιο περίπλοκες αναπαραστάσεις, υπάρχει δηλαδή μία ιεραρχία στην πολυπλοκότητα των χαρακτηριστικών που μαθαίνονται καθώς προχωράμε στα κρυμμένα επίπεδα. Αποδοτικοί αλγόριθμοι για την εκπαίδευση των deep belief networks αναπτύχθηκαν στα μέσα της προηγούμενης δεκαετίας από τους Hinton et al. [50].

3.2: Autoencoders

Οι αυτοκωδικοποιητές (autoencoders) χρησιμοποιούνται στην μη επιβλεπόμενη μάθηση και συγκεκριμένα για τη μάθηση αποδοτικών αναπαραστάσεων (codings) των δεδομένων εισόδου. Οι αναπαραστάσεις αυτές, τουλάχιστον στις αρχικές υλοποιήσεις των αρχιτεκτονικών αυτών, είχαν συνήθως αισθητά λιγότερες διαστάσεις απ'ότι τα δεδομένα εισόδου, κάτι που ήταν αναγκαίο, λόγω των περιορισμένων υπολογιστικών πόρων. Το γεγονός ότι οι πληροφορίες των δεδομένων αναγκάζονται να συμπτυχθούν σε έναν πολύ περιορισμένο χώρο καθιστά τους autoencoders ισχυρούς ανιχνευτές χαρακτηριστικών [51], και άρα κατάλληλους για μη επιβλεπόμενη προεκπαίδευση νευρωνικών δικτύων [52]. Ακόμη, οι autoencoders, ως γεννητικά μοντέλα, είναι ικανοί να δημιουργούν με τυχαίο τρόπο νέα, τεχνητά δεδομένα, που μοιάζουν με αυτά στα οποία έχουν εκπαιδευτεί.

Ένας autoencoder αποτελείται από δύο μέρη, τον encoder, που μετασχηματίζει τα δεδομένα εισόδου στην εσωτερική αναπαράσταση του δικτύου, και τον decoder, ο οποίος μετασχηματίζει την αναπαράσταση αυτή στην έξοδο. Η έξοδος ενός autoencoder ονομάζεται ανακατασκευή (reconstruction), αφού ο στόχος του μοντέλου είναι αυτή να μοιάζει όσο το δυνατόν περισσότερο στη είσοδο. Η δομή ενός τυπικού autoencoder φαίνεται στο Σχήμα 3.3. Η συνάρτηση απώλειας ονομάζεται απώλεια ανακατασκευής (reconstruction loss), διότι τιμωρεί το μοντέλο όταν οι έξοδοί του είναι πολύ διαφορετικές από τις εισόδους.

Στην πιο απλή του μορφή, ένας autoencoder μοιάζει στη δομή του με ένα μικρό multi-layer perceptron, με τους encoder και decoder να αποτελούνται από ένα fully connected layer έκαστος, με τον περιορισμό ότι ο αριθμός των νεωρώνων του layer εξόδου πρέπει να είναι ίδιος με αυτόν του layer εισόδου. Ένας autoencoder του οποίου η εσωτερική αναπαράσταση έχει μικρότερο αριθμό διαστάσεων απ'ότι οι είσοδοί του χαρακτηρίζεται ως undercomplete. Ένας undercomplete autoencoder είναι ανίκανος απλά να αντιγράψει τυφλά τις εισόδους του στις κωδικοποιήσεις του, και αντ'αυτού αναγκάζεται να μάθει τα πιο σημαντικά χαρακτηριστικά των δεδομένων, αγνοώντας τα υπόλοιπα. Εδώ πρέπει να αναφερθεί ότι σε περίπτωση που ο autoencoder χρησιμοποιεί μόνο γραμμικές συναρτήσεις ενεργοποίησης και ως απώλεια ανακατασκευής έχει οριστεί το mean squared error (MSE), το encoder κομμάτι του μοντέλου ουσιαστικά εφαρμόζει principal component analysis (PCA).



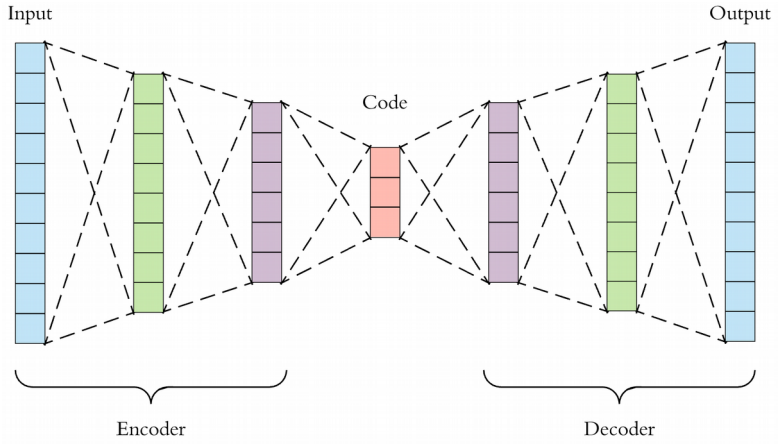
Σχήμα 3.3: Ένας τυπικός undercomplete autoencoder ενός layer.

3.2.1: Stacked Autoencoders

Οι stacked/deep autoencoders είναι απλά autoencoders των οποίων τα δομικά στοιχεία έχουν περισσότερα από ένα hidden layers (βλ. Σχήμα 3.4). Αυτό από τη μία επιτρέπει στο δίκτυο να μάθει αρκετά πιο περίπλοκες εσωτερικές αναπαραστάσεις, ωστόσο από την άλλη μπορεί να δημιουργήσει έναν autoencoder αρκετά ισχυρό ώστε απλά να αντιγράφει τα δεδομένα του, χωρίς καμία δηλαδή ικανότητα γενίκευσης. Συνήθως ένας stacked autoencoder είναι συμμετρικός ως προς το μεσαίο hidden layer (το layer των αναπαραστάσεων) και τα δύο δίκτυα μοιράζονται μεταβλητές,

για λόγους εξοικονόμησης μνήμης, με τους πίνακες των decoding layers απλά να είναι ανάστροφοι πίνακες των encoding layers.

Μία συνηθισμένη τεχνική είναι να εκπαιδεύεται κάθε φορά ένας "ρηχός" autoencoder, και να προστίθεται κάθε φορά ένα ζευγάρι layers: Ο πρώτος autoencoder θα μάθει να ανακατασκευάζει την είσοδο, ο δεύτερος την έξοδο του πρώτου κ.ο.κ, με το αποτέλεσμα να είναι τελικά μία στοίβα (stack) από autoencoders. Η τεχνική αυτή, αν εφαρμοστεί σωστά, μπορεί να οδηγήσει σε πολύ βαθιούς και ισχυρούς autoencoders, και χρησιμοποιήθηκε με επιτυχία πρώτη φορά από τους Bengio et al. [53] το 2007.

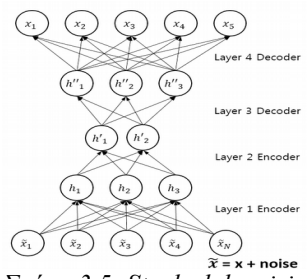


Σχήμα 3.4: Ένας τυπικός Stacked autoencoder

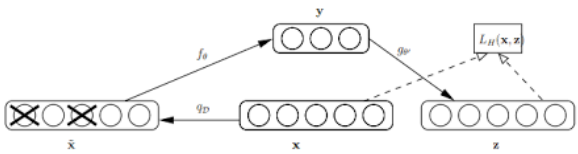
Παρακάτω παρουσιάζονται μερικές πιο προχωρημένες τεχνικές που αποτρέπουν έναν autoencoder από το να μαθαίνει τυφλά τις εισόδους του. Οι τεχνικές αυτές επιτρέπουν τη χρήση μεγαλύτερων και ισχυρότερων layers, καθιστώντας εφικτή ακόμα και τη χρήση overcomplete autoencoders, χωρίς να τίθενται θέματα γενίκευσης.

3.2.2: Denoising Autoencoders

Μία από τις τεχνικές αυτές είναι η προσθήκη θορύβου στην είσοδο του autoencoder και η εκπαίδευσή του ώστε να ανακτήσει τα αρχικά δεδομένα. Ο θόρυβος μπορεί να είναι Γκαουσιανός που έχει προστεθεί στις εισόδους (Σχήμα 3.5), τυχαία απενεργοποιημένα τμήματα των εισόδων, παρόμοια με το dropout (Σχήμα 3.6), ώστε ο autoencoder να μάθει να συμπληρώνει ελλειπή δεδομένα, κ.λ.π. Οι ιδέες αυτές αναφέρθηκαν πρώτη φορά στη διδακτορική διατριβή του Yann LeCun [54], ενώ αργότερα βρέθηκε ότι οι denosing autoencoders ήταν πολύ καλοί ανιχνευτές χαρακτηριστικών [55]. Οι λεγόμενοι stacked denoising autoencoders άρχισαν να εμφανίζονται το 2010 [56].



Σχήμα 3.5: Stacked denoising autoencoder Γκαουσιανού θορύβου.



Σχήμα 3.6: Denoising autoencoder απενεργοποίησης τμημάτων της εισόδου.

3.2.3: Sparse Autoencoders

Σε έναν sparse autoencoder [57] ο αριθμός των ενεργών νευρώνων σε κάθε layer περιορίζεται ώστε να αναγκαστεί το μοντέλο να αναπαριστά κάθε είσοδο ως έναν συνδυασμό μικρού αριθμού νευρώνων, και έτσι να μάθει χρήσιμα χαρακτηριστικά.

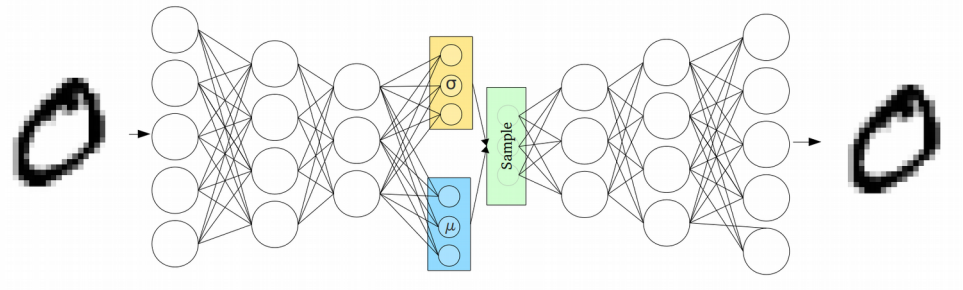
Πρέπει να δοθεί ιδιαίτερη προσοχή στο ποσοστό των ενεργών νευρώνων: αν είναι υπερβολικά μικρό, τότε είναι πολύ πιθανό να υπάρξει underfit, ενώ προφανώς αν είναι πολύ μεγάλο είναι πολύ πιθανό να υπάρξει overfit. Η λύση στο θέμα αυτό είναι η παρακολούθηση της αραιότητας του coding layer κατά τη διάρκεια της εκπαίδευσης και στη συνέχεια η τιμωρία των νευρώνων που είναι υπερβολικά ενεργοί. Το πρώτο επιτυγχάνεται με τον υπολογισμό της μέσης ενεργοποίησης του κάθε νευρώνα του coding layer πάνω σε κάθε batch δεδομένων (κάτι που προφανώς σημαίνει ότι το μέγεθος του batch δεν πρέπει να είναι υπερβολικά μικρό, γιατί τότε τα αποτελέσματα κατά πάσα πιθανότητα δε θα αντικατοπτρίζουν την πραγματικότητα).

Το δεύτερο επιτυγχάνεται με την προσθήκη μίας "απώλειας αραιότητας" στη συνάρτηση κόστους. Μία πιθανή συνάρτηση κόστους είναι το MSE, στην πράξη όμως επιλέγεται η Kullback-Leibler απόκλιση λόγω ισχυρότερων παραγώγων, και άρα μεγαλύτερης ταχύτητας σύγκλισης.

3.2.4: Variational Autoencoders

Η πιο διαδεδομένη αρχιτεκτονική autoencoder σήμερα είναι οι variational autoencoders (VAE) [58]. Οι VAE διαφέρουν από τις άλλες υλοποιήσεις που έχουν εξεταστεί μέχρι τώρα στο ότι χρησιμοποιούν την τυχαιότητα όχι μόνο κατά τη διάρκεια, αλλά και μετά το στάδιο της εκπαίδευσης, στο ότι δίνουν μεγαλύτερη έμφαση στη δημιουργία τεχνητών δεδομένων παρά στην εκμάθηση χαρακτηριστικών για σκοπούς προεκπαίδευσης και στο ότι δεν παράγουν απευθείας το coding layer, αλλά παράγουν κωδικοποιήσεις για τη μέση τιμή (μ) και την τυπική απόκλιση (σ) (βλ. Σχήμα 3.7). Το coding αυτό καθεαυτό, οι πληροφορίες δηλαδή που αποκωδικοποιεί ο decoder, δειγματολειπτείται στη συνέχεια από μία από τις συνηθισμένες κατανομές (Gaussian, ομοιόμορφη) μέσης τιμής μ και τυπικής απόκλισης σ. Έτσι, όσο περίπλοκη και να είναι η κατανομή των δεδομένων εισόδου, αυτή μετασχηματίζεται σε ένα σύνολο από σημεία που μπορούν να διαχωριστούν και να δειγματοληφθούν εύκολα, ενώ η δημιουργία νέων δεδομένων απαιτεί μόνο τη δειγματοληψία codings με τα δοσμένα στατιστικά χαρακτηριστικά.

Η συνάρτηση απώλειας αποτελείται από δύο μέρη: Το πρώτο μέρος είναι η απώλεια ανακατασκευής, ώστε τα δεδομένα που παράγει ο decoder να μοιάζουν με τα αληθινά, ενώ το δεύτερο είναι η λανθάνουσα απώλεια (latent loss), για να δοθεί κίνητρο στον autoencoder ώστε οι αναπαραστάσεις του να δίνουν την εντύπωση ότι είναι δείγματα που έχουν ληφθεί από την κατανομή εκπαίδευσης. Χρησιμοποιείται η KL απόκλιση μεταξύ της κατανομής-στόχου (κανονική) και της υπάρχουσας κατανομής (κατανομή δεδομένων). Η ύπαρξη του θορύβου λειτουργεί ως μηχανισμός κανονικοποίησης, περιορίζοντας ωφέλιμα την ποσότητα και την ποιότητα των πληροφοριών που μπορεί να περάσει στο coding layer, κάτι που αναγκάζει τον autoencoder να μάθει χρήσιμα και ισχυρά χαρακτηριστικά.

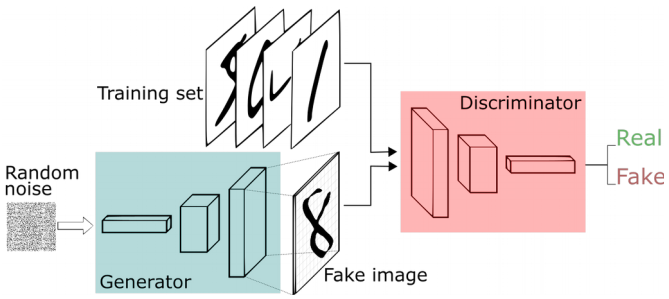


Σχήμα 3.7: Ένας τυπικός variational autoencoder

3.3: Generative Adversarial Networks

Τα Γεννητικά Ανταγωνιστικά Δίκτυα (Generative Adversarial Networks, GANs) προτάθηκαν το 2014 από τους Goodfellow et al. [59] στα πλαίσια της μη επιβλεπόμενης μάθησης, με σκοπό τη δημιουργία τεχνητών δεδομένων.

Μία αρχιτεκτονική GAN αποτελείται από δύο δίκτυα, τον δημιουργό (generator) και τον διευκρινιστή (discriminator). Η είσοδος του generator είναι συνήθως ένα διάνυσμα τυχαίων μεταβλητών που ακολουθούν μία συγκεκριμένη κατανομή, ενώ η έξοδός του, τα τεχνητά δεδομένα, προωθείται μαζί με αληθινά δεδομένα για αξιολόγηση στον discriminator, η έξοδος του οποίου, τουλάχιστον στην αρχική υλοποίηση της αρχιτεκτονικής, ήταν η πιθανότητα ένα δείγμα να προέρχεται από τα δεδομένα εκπαίδευσης. Σκοπός του discriminator είναι να μάθει λοιπόν να ξεχωρίζει με όσο το δυνατόν περισσότερη ακρίβεια τα αληθινά δεδομένα από αυτά που παράγει ο generator, ενώ ο σκοπός του generator είναι με τη σειρά του να μάθει την κατανομή των δεδομένων και να "ξεγελάσει" τον discriminator, να παράγει δεδομένα δηλαδή τα οποία ο discriminator δεν μπορεί να ξεχωρίσει από τα αληθινά. Η τυπική δομή μίας αρχιτεκτονικής GAN ακολουθεί παρακάτω:



Σχήμα 3.8: Η δομή μίας τυπικής GAN αρχιτεκτονικής.

3.3.1: Μερικές Εφαρμογές των GANs

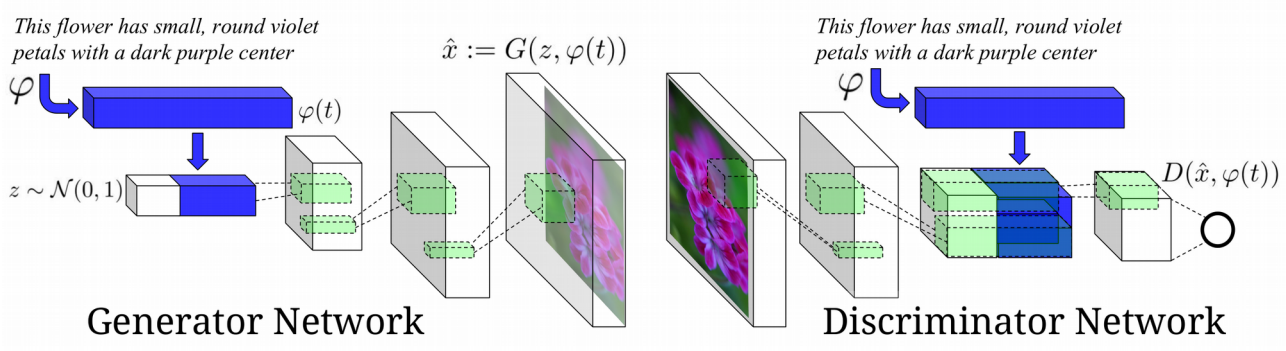
Μία από τις κύριες χρήσεις των GAN είναι στην μη επιβλεπόμενη μάθηση, και ειδικότερα στη μάθηση επαναχρησιμοποιούμενων αναπαραστάσεων χαρακτηριστικών. Πιο συγκεκριμένα, στην περιοχή της όρασης υπολογιστών, χρησιμοποιώντας τον τεράστιο όγκο unlabeled δεδομένων εικόνας και βίντεο μπορεί κανείς να εκπαιδεύσει δίκτυα ώστε να μάθουν ισχυρές ενδιάμεσες αναπαραστάσεις και να τα χρησιμοποιήσει στη συνέχεια σε μία πληθώρα επιβλεπόμενων έργων, όπως π.χ. στην ταξινόμηση (classification). Αυτό οφείλεται στη φύση της συνάρτησης-στόχου του μοντέλου, η οποία δίνει έμφαση μόνο στη στατιστική ποιότητα των συνθετικών δεδομένων και όχι σε κάποια συνάρτηση κόστους που λειτουργεί στο επίπεδο των pixels (π.χ. σφάλμα ελάχιστων τετραγώνων), όπως π.χ. οι autoencoders που εξετάστηκαν προηγούμενα.

3.3.1.1: Παραγωγή Εικόνας από Λεζάντα

Ο τρόπος λειτουργίας των GANs τα καθιστά πολύ ευέλικτες αρχιτεκτονικές, και είχε ως αποτέλεσμα την χρήση τους σε περιοχές έρευνας πολύ διαφορετικές μεταξύ τους, που σε κάποιες περιπτώσεις δεν είχαν καμία σχέση με οπτικά δεδομένα, την αρχική δηλαδή περιοχή εφαρμογής. Μία αρκετά γνωστή εφαρμογή των GANs ήταν η χρήση τους για τη δημιουργία εικόνων με είσοδο μία λεζάντα (caption) [60]. Το πρόβλημα αυτό αποτελείται από δύο υποπροβλήματα, αρχικά το μετασχηματισμό της λεζάντας σε μία εσωτερική αναπαράσταση χαρακτηριστικών που συγκρατεί τα σημαντικά οπτικά χαρακτηριστικά που αυτή περιγράφει, και στη συνέχεια τη μετατροπή της αναπαράστασης αυτής στην τελική εικόνα. Χρησιμοποιήθηκαν τεχνικές από την περιοχή της επεξεργασίας φυσικής γλώσσας (Natural Language Processing, NLP) για τη συμπύκνωση της λεζάντας σε

ένα διάνυσμα, το οποίο, αφού περάσει από ένα fully-connected layer για λόγους μείωσης διαστάσεων, ενώνεται με το διάνυσμα του προγενέστερου θορύβου για να τροφοδοτηθεί ως είσοδος στον generator. Τα υπόλοιπα μέρη της εκπαίδευσης ακολουθούν τις συνηθισμένες τεχνικές, ενώ η ίδια τεχνική εφαρμόζεται στο τελευταίο layer του discriminator. Παρακάτω φαίνονται η αρχιτεκτονική του εν λόγω μοντέλου, καθώς και μερικές εικόνες που προέκυψαν με είσοδο τις εξής λεζάντες:

Λεζάντα 1: This small bird has a pink breast and crown, and black primaries and secondaries.
Λεζάντα 2: This magnificent fellow is almost all black with a red crest, and white cheek patch.
Λεζάντα 3: The flower has petals that are bright pinkish purple with white stigma.
Λεζάντα 4: This white and yellow flower have thin white petals and a round yellow stamen.



Σχήμα 3.9: Η δομή της GAN αρχιτεκτονικής του [60].

Σχήμα 3.10: Εικόνα που προκύπτει από τη Λεζάντα 1.

Σχήμα 3.11: Εικόνα που προκύπτει από τη Λεζάντα 2.

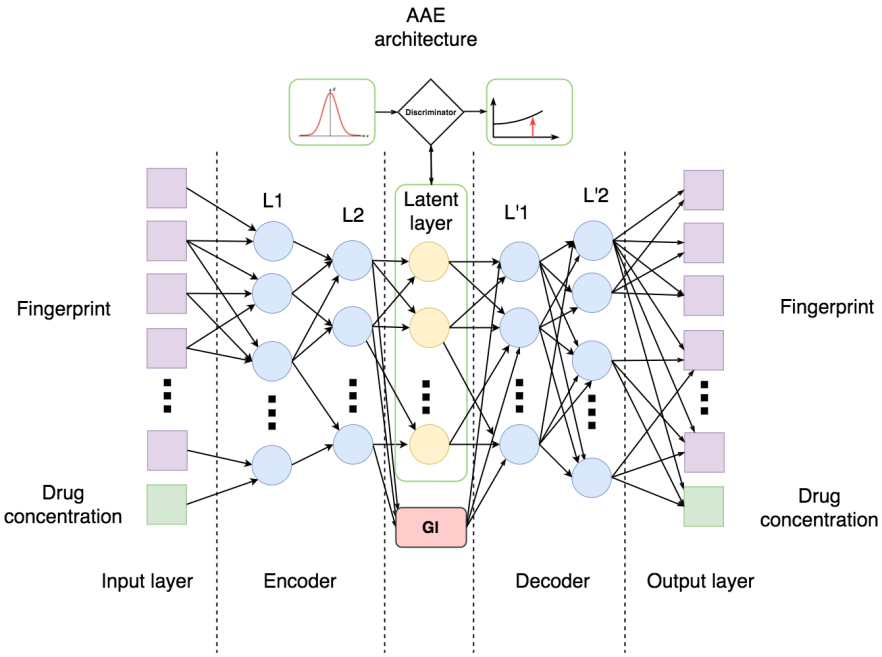
Σχήμα 3.12: Εικόνα που προκύπτει από τη Λεζάντα 3.

Σχήμα 3.13: Εικόνα που προκύπτει από τη Λεζάντα 4.

3.3.1.2: Ανακάλυψη Φαρμάκευτικών Ουσιών

Οι Kadurin et al. [61] συνδύασαν τα GAN με τους autoencoders για τη δημιουργία νέων ενεργών ουσιών για την καταπολέμηση του καρκίνου. Πιο συγκεκριμένα, τα δεδομένα εισόδου κατά την εκπαίδευση ήταν 6252 δυαδικά διανύσματα 166 χαρακτηριστικών: το 0 συμβόλιζε την απουσία ενός σημαντικού χαρακτηριστικού και το 1 την παρουσία του χαρακτηριστικού αυτού. Μαζί με τα δυαδικά αυτά χαρακτηριστικά, ως επιπλέον πληροφορίες εισόδου δίνονταν η συγκέντρωση της ουσίας, στο layer εισόδου, καθώς και η ισχύς της (το ποσοστό της μείωσης των καρκινικών κυττάρων μετά τη χορηγία της), στο latent layer, που αποτελούνταν από 4 + 1 νευρώνες. Το encoder κομμάτι του δικτύου ήταν ένα 2-layer MLP (128 και 64 νευρώνων, αντίστοιχα) με το decoder, που μπορεί να θεωρηθεί ως ο generator, να είναι συμμετρικό του, και η εσωτερική αναπαράσταση του δικτύου συγκρινόταν στον discriminator με μία κανονική κατανομή ίδιων διαστάσεων (4, αφού δε λαμβανόταν υπόψη η συγκέντρωση της ουσίας). Η έξοδος του decoder κομματιού ήταν ένα διάνυ-

σμα πιθανοτήτων παρουσίας του εκάστοτε χαρακτηριστικού. Σκοπός δηλαδή ήταν το δίκτυο να μάθει να συμπυκνώνει τα δεδομένα του με ανταγωνιστικό τρόπο σε μία Γκαουσιανή κατανομή, ώστε να είναι εύκολη η δημιουργία νέων δειγμάτων.



Σχήμα 3.14: Η δομή της αρχιτεκτονικής των adversarial autoencoders.

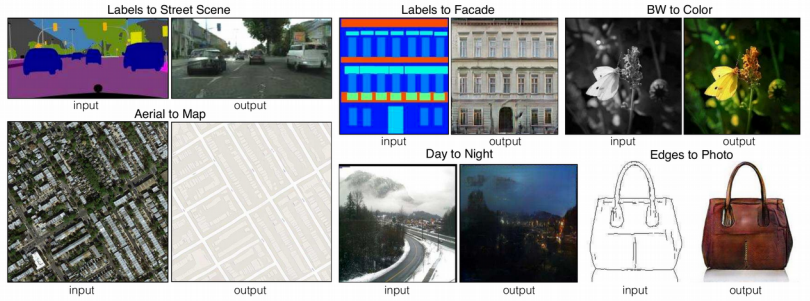
Μετά την εκπαίδευση, στο δίκτυο δόθηκαν 640 Γκαουσιανά διανύσματα και ποσοστά μειώσεων, και από όλα τα διανύσματα πιθανοτήτων που παράχθηκαν κρατήθηκαν 32 που δεν είχαν υπερβολικά υψηλή συγκέντρωση. Τα διανύσματα αυτά δόθηκαν ως είσοδος σε βάση δεδομένων ενεργών ουσιών, και με μεγιστοποίηση της λογαριθμικής πιθανότητας επιλέχθηκαν κάθε φορά οι 10 πιο ταιριαστές ουσίες. Συνολικά βρέθηκαν 69 ξεχωριστές ουσίες, από τις οποίες αρκετές είτε χρησιμοποιούνταν ήδη για τη θεραπεία του καρκίνου είτε ήταν σε δοκιμαστικό στάδιο. Οι καινούριες ουσίες που βρέθηκαν θα δοκιμάζονταν για την αποτελεσματικότητά τους.

3.3.1.3: Μεταφορά Περιεχομένου Εικόνας σε Διαφορετικό Στυλ

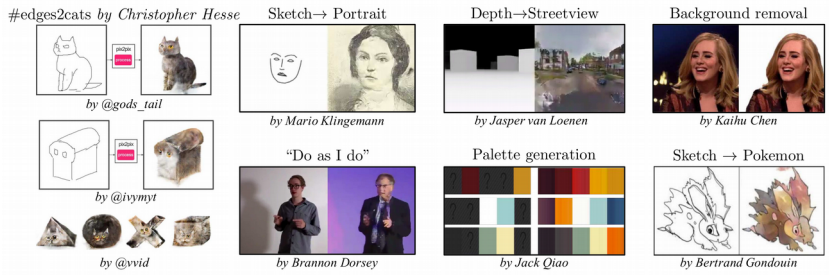
Τέλος, μία από τις πιο γνωστές εφαρμογές των GANs ήταν το Isola et al. [62], όπου ουσιαστικά λύθηκε το πρόβλημα της αλλαγής του στυλ μίας εικόνας με είσοδο την ίδια την εικόνα: Με είσοδο π.χ. μία αεροφωτογραφία το δίκτυο κατάφερε να κατασκευάσει το χάρτη της περιοχής, ή με εισοδο ένα σκίτσο μίας τσάντας το δίκτυο δημιούργησε μία ρεαλιστική μορφή του σκίτσου. Αυτό επιτεύχθηκε με τη χρήση των εξής τεχνικών: Εφ'όσον οι αρχικές εικόνες είναι δοσμένες, στη συνάρτηση απώλειας του GAN προστίθεται και ένας όρος L_1 απόστασης, ώστε η εικόνα που θα παραχθεί όχι μόνο να είναι αρκετά ρεαλιστική, αλλά και να διατηρεί τα χαμηλού επιπέδου γεωμετρικά χαρακτηριστικά της αρχικής. Ως generator χρησιμοποιείται ένας stacked autoencoder, ενώ για τον ίδιο λόγο η έξοδος κάθε layer του encoding μέρους του δικτύου προσκολλάται στο αντίστοιχο συμμετρικό layer του decoding μέρους, μαζί με την έξοδο του layer, παρόμοια με τον τρόπο λειτουργίας του U-Net [63]. Αντίθετα με τη συνηθισμένη υλοποίηση των GANs, δε χρησιμοποιείται διάνυσμα θορύβου ως είσοδος, αλλά χρησιμοποιείται dropout στον generator, όχι μόνο κατά τη διάρκεια της εκπαίδευσης, αλλά και μετά το πέρας της.

Όσον αφορά τον discriminator, ακριβώς επειδή η L_1 απόσταση οδηγεί τον generator να μάθει τη γενική γεωμετρία της εικόνας, ο discriminator περιορίζεται μόνο σε τοπικά τεμάχια, ώστε

να επικεντρωθεί στον εντοπισμό των λεπτομερειών που κάνουν μία τεχνητή εικόνα να ξεχωρίζει από μία αληθινή, που είναι αδύνατο να μοντελοποιηθούν με χρήση της L_1, που τείνει να παράγει αρκετά θολές εικόνες. Έτσι, ο discriminator λειτουργεί παρόμοια με ένα συνελικτικό layer, ταξινομώντας κάθε φορά ένα N × N κομμάτι της εικόνας ως αληθινό ή τεχνητό, και το αποτέλεσμα λαμβάνεται τελικά ως ο μέσος όρος των επιμέρους αποτελεσμάτων. Μερικές από τις εικόνες που παράχθηκαν ακολουθούν παρακάτω:



Σχήμα 3.15: Μερικοί από τους μετασχηματισμούς που εξετάστηκαν από τους συγγραφείς του άρθρου.



Σχήμα 3.16: Μερικοί από τους μετασχηματισμούς που αναπτύχθηκαν από χρήστες του Διαδικτύου, μετά τη δημοσίευση του λογισμικού pix2pix.

Ο μόνος περιοριστικός παράγοντας στην pix2pix αρχιτεκτονική είναι ότι πρέπει να υπάρχουν ήδη τα δεδομένα για τη μεταφορά από το ένα στυλ στο άλλο, περιορίζοντας έτσι τις πιθανές περιοχές εφαρμογής του μοντέλου. Οι Zhu et al. [64] αντιμετώπισαν το πρόβλημα αυτό με τη χρήση δύο ζευγαριών discriminator-generator που μετασχηματίζουν την εικόνα από τη μία περιοχή στην άλλη και το αντίθετο, ώστε το δίκτυο να μάθει να ανακατασκευάζει την αρχική εικόνα μετά από δύο αλλαγές στο στυλ.

Τα GANs έχουν επίσης εφαρμοστεί στην αλλαγή της ηλικίας ενός φωτογραφημένου προσώπου [65], στη βελτίωση φωτογραφιών ώστε να φαίνονται πιο επαγγελματικές [66], στη μετατροπή σκίτσου σε πίνακα [67] κ.λ.π.

3.3.2: Θεωρητικές Βάσεις

Για να μάθουμε την κατανομή του generator, p_g, πάνω στα δεδομένα **x,** αρχικά ορίζουμε μία προγενέστερη κατανομή $p_z(z)$ μεταβλητών θορύβου και αντιπροσωπεύουμε την μεταφορά στον χώρο των δεδομένων ως G(z), όπου G είναι μία παραγωγίσιμη συνάρτηση. Παράλληλα, ορίζουμε δεύτερη συνάρτηση D(**x**) με έξοδο έναν αριθμό, την πιθανότητα το **x** να προέρχεται από τα αληθινά δεδομένα και όχι από την κατανομή p_g. Σκοπός του D είναι να ταξινομεί σωστά τόσο δεδομένα εκπαίδευσης, όσο και δείγματα του G. Αντίστοιχα, σκοπός του G είναι να ελαχιστοποιήσει το log(1 – D(G(**z**)). Έτσι, τα D και G παίζουν ένα minimax παίγνιο δύο παικτών, με συνάρτηση αξίας V(G, D):

$$V(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

Στη θεωρητική περίπτωση που οι D και G αντιπροσωπεύονται από νευρωνικά δίκτυα άπειρης χωρητικότητας, το παίγνιο έχει ένα σημείο ισορροπίας, για $p_g = p_{data}$, στο οποίο ο D δίνει παντού έξοδο ½.

Απόδειξη:

Για γνωστό G, ο βέλτιστος D είναι ο $D^*_G = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$

Απόδειξη: Στόχος του D, με δοσμένο έναν G, είναι να μεγιστοποιήσει την συνάρτηση

$$\begin{aligned} V(G,D) &= \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz \\ &= \int_x p_{data}(x)\log(D(x))dx + p_g(x)\log(1-D(x))dx \end{aligned}$$

Για κάθε $(a,b)\in\mathbb{R}^2\setminus\{0,0\}$ η συνάρτηση $y\to a\log(y)+b\log(1-y)$ έχει ολικό μέγιστο στο [0,1] στο σημείο $\frac{a}{a+b}$, ενώ ο discriminator δεν χρειάζεται να οριστεί εκτός του διαστήματος $Supp(p_{data})\cup Supp(p_g)$.

Εδώ πρέπει να ληφθεί υπόψη ότι ο στόχος του D στο παίγνιο μπορεί να ερμηνευθεί ως μεγιστοποίηση της λογαριθμικής πιθανότητας, της σωστής εκτίμησης δηλαδή της $P(Y=y|\mathbf{x})$, όπου y = 1 εάν το x προέρχεται από το p_{data} και y = 0 εάν προέρχεται από το p_g. Έτσι, το παίγνιο μπορεί τώρα να εκφραστεί ως εξής:

$$\begin{aligned} C(G) &= \max_D V(G,D) \\ &= \mathbb{E}_{x\sim p_{data}}[\log D^*_G(x)] + \mathbb{E}_{z\sim p_z}[\log(1-D^*_G(G(z)))] \\ &= \mathbb{E}_{x\sim p_{data}}[\log D^*_G(x)] + \mathbb{E}_{x\sim p_g}[\log(1-D^*_G(x))] \\ &= \mathbb{E}_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}] + \mathbb{E}_{x\sim p_g}[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}] \end{aligned} \quad (3.1)$$

Θεώρημα 3.1:

Το C(G) έχει μόνο ένα ολικό ελάχιστο, στο $p_g = p_{data}$, και είναι -log(4)

Απόδειξη:

Για $p_g = p_{data}$, $D^*_G = 1/2$. Σύμφωνα με την εξίσωση (3.1) λοιπόν, C(G) = log(1/2) + log(1/2) = -log(4). Αφαιρώντας την έκφραση αυτή από την αρχική προκύπτει ότι

$$C(G) = -\log(4) + KL(p_{data}\|\frac{p_{data}+p_g}{2}) + KL(p_g\|\frac{p_{data}+p_g}{2})$$

όπου KL είναι η απόκλιση Kullback-Leibler. Η παραπάνω εξίσωση μπορεί να γραφτεί ως:

$$C(G) = -\log(4) + 2 \cdot JS(p_{data} \| p_g)$$

όπου JS είναι η απόκλιση Jensen-Shannon (Jensen-Shannon divergence), η οποία είναι πάντοτε μη αρνητική, και ίση με μηδέν αν και μόνο αν οι δύο κατανομές είναι ίσες. Έτσι, $C^* = -\log(4)$ είναι το ολικό ελάχιστο του C(G) και η μόνη τιμή για την οποία επιτυγχάνεται είναι η $p_g = p_{data}$, όταν δηλαδή ο δημιουργός μαθαίνει να αντιγράφει την κατανομή των αληθινών δεδομένων.

Παρακάτω ακολουθεί ο αλγόριθμος εκπαίδευσης των GANs:

Αλγόριθμος 1: Αλγόριθμος εκπαίδευσης GAN με minibatch stochastic gradient descent. Υπερπαράμετροι είναι το k, που ορίζεται ως ο αριθμός των φορών που εκπαιδεύεται ο discriminator ανά εκπαίδευση του generator (στο αρχικό άρθρο επιλέχθηκε k = 1, για λόγους υπολογιστικού κόστους) και το m, το batch size.

Για αριθμό επαναλήψεων:

Για k:

- Κάνουμε δειγματοληψία m διανυσμάτων θορύβου $\{z^{(1)}, \ldots, z^{(m)}\}$ από την προγενέστερη κατανομή, p_z.
- Κάνουμε δειγματοληψία m δεδομένων $\{x^{(1)}, \ldots, x^{(m)}\}$ από την κατανομή των αληθινών δεδομένων, p_{data}.
- Ανανεώνουμε τα βάρη του discriminator κάνοντας stochastic gradient ascent:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$$

- Κάνουμε δειγματοληψία m διανυσμάτων θορύβου $\{z^{(1)}, \ldots, z^{(m)}\}$ από την προγενέστερη κατανομή, p_z.
- Ανανεώνουμε τα βάρη του generator κάνοντας stochastic gradient descent:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z^{(i)})))]$$

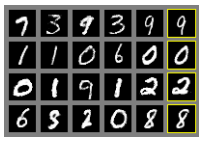
Αν οι G και D έχουν αρκετή χωρητικότητα, σε κάθε βήμα ο D αφήνεται να φτάσει το βέλτιστό του σημείο με δεδομένο τον G, και το p_g ανανεώνεται ώστε να ελαχιστοποιείται το παρακάτω κριτήριο, τότε το p_g συγκλίνει στο p_{data}, κάτι που προκύπτει από την κυρτότητα της παραπάνω συνάρτησης και από το ότι αυτή έχει μόνο ένα ολικό ελάχιστο στο $p_g = p_{data}$.

3.3.3: MLP-GAN

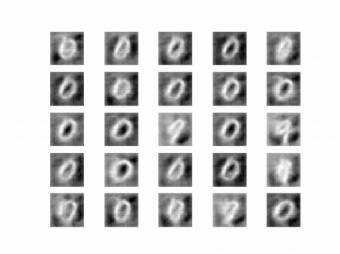
Πρακτικά, οι παραπάνω αποδείξεις δεν ισχύουν, διότι αντί να βελτιστοποιούμε το p_g αυτό καθεαυτό, βελτιστοποιούμε το θ_g (όπου θ_g οι παράμετροι της επιλεγμένης αρχιτεκτονικής) μέσω του $G(z, \theta_g)$. Έτσι λοιπόν, ειδικά στην αρχική διατύπωση του παιγνίου, ήταν πολύ συχνό το φαινόμενο της κατάρρευσης, στο οποίο ο generator περιορίζεται σε μερικά μόνο σημεία από όλη την κατανομή, τα οποία και παράγει σε μεγάλο βαθμό ανεξάρτητα από την είσοδο (βλ. Σχήμα 3.18).

Αυτό ήταν ιδιαίτερα αισθητό στην αρχική υλοποίηση του μοντέλου, όπου τα δύο δίκτυα ήταν multi-layer perceptrons, όχι τόσο ισχυρές δηλαδή αρχιτεκτονικές. Η απόφαση αυτή λήφθηκε κατά πάσα πιθανότητα ώστε να μπορεί να χρησιμοποιηθεί η επίδοσή τους ως μέτρο σύγκρισης με άλλες, πιο περίπλοκες αρχιτεκτονικές, αλλά και για λόγους ευκολίας, λόγω της "τυποποιημένης" φύσης της εκπαίδευσης ενός MLP με back propagation. Όπως φαίνεται και παρακάτω, ενώ σε απλά

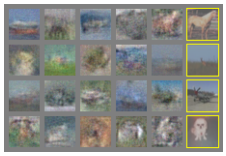
datasets όπως το MNIST το μοντέλο φαίνεται να παράγει ικανοποιητικά δείγματα, σε πιο περίπλοκα οι περιορισμοί του γίνονται εμφανείς, αφού ακόμα και όταν αποφεύγεται η κατάρρευση, οι τεχνητές εικόνες ξεχωρίζουν πολύ εύκολα από τις αληθινές.



Σχήμα 3.17: Το MLP-GAN εκπαιδεύεται επιτυχώς στο MNIST.



Σχήμα 3.18: Κατάρρευση στο MNIST, σε διαφορετικό πείραμα.



Σχήμα 3.19: Κακής ποιότητας εικόνες του MLP-GAN στο CIFAR-10.

3.3.4: DCGAN

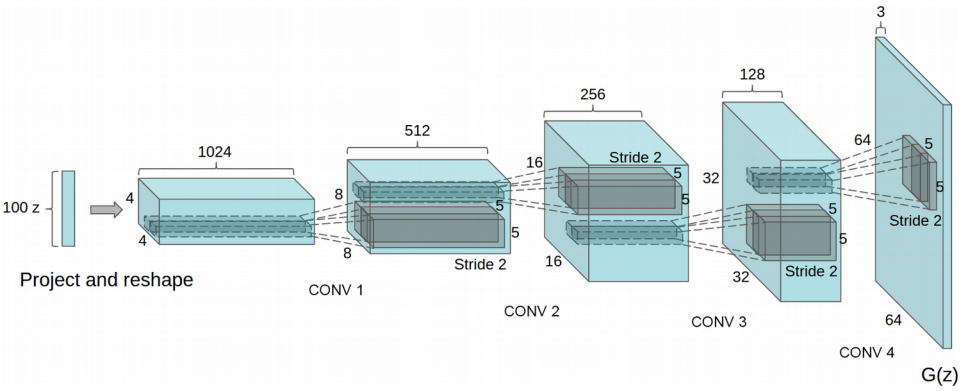
Ένα προφανές αρχικό βήμα για τη βελτίωση της ποιότητας των δημιουργούμενων εικόνων και την καταπολέμηση του φαινομένου της αστάθειας κατά της εκπαίδευση ήταν η χρήση ισχυρότερων αρχιτεκτονικών, όπως τα βαθιά συνελικτικά δίκτυα. Οι Radford et al. [68] μετά από πολλές δοκιμές πρότειναν:

1) Χρήση ολικών συνελικτικών δικτύων και στα δύο δίκτυα, ώστε να αντικατασταθεί το ντετερμινιστικό χωρικό pooling (π.χ max-pooling) που μπορεί να έχει αποτέλεσμα κατώτερο του βέλτιστου με συνελίξεις βήματος μεγαλύτερου από 1, ώστε να δοθεί η ευκαιρία στα δίκτυα να μάθουν τα ίδια τη βέλτιστη υποδειγματοληψία τους.

2) Εξάλειψη των fully connected layers μετά από τα layers συνελίξεων. Αντ'αυτού, αν το ζητούμενο είναι η σταθερότητα του μοντέλου σε βάρος της ταχύτητας σύγκλισης, οι συγγραφείς προτείνουν την τεχνική του global average pooling. Εναλλακτικά προτείνεται η απευθείας σύνδεση των συνελικτικών χαρακτηριστικών των δύο δικτύων.

3) Χρήση batch normalization. Η σταθερότητα που προσφέρει η χρήση του batch norm καθιστά εφικτή τη χρήση βαθύτερων, και άρα ισχυρότερων δίκτυων στα δύο μέρη της αρχιτεκτονικής.

Παρακάτω ακολουθούν η δομή του generator, καθώς και κάποιες εικόνες από το LSUN dataset [LSUN] (dataset υπνοδωματίων), που δείχνουν τη γενικότερη βελτίωση της ποιότητας των παραγόμενων εικόνων:



Σχήμα 3.20: Δομή του generator μέρους της DCGAN αρχιτεκτονικής.

Σχήμα 3.21: Τεχνητές εικόνες από το DCGAN στο LSUN dataset.

3.3.5: Προχωρημένες Εμπειρικές Τεχνικές

Περαιτέρω εμπειρικές τεχνικές για την αντιμετώπιση της απόκλισης των δικτύων κατά την εκπαίδευση αναπτύχθηκαν από τους Salimasn et al. [69]:

3.3.5.1: Feature Matching

Η τεχνική του feature matching ορίζει έναν επιπλέον στόχο στον generator, που τον εμποδίζει να εκπαιδευτεί παραπάνω από όσο πρέπει στον τρέχοντα discriminator. Αντί να μεγιστοποιέι απευθείας την έξοδο του discriminator, ο νέος στόχος απαιτεί από τον generator να δημιουργήσει δεδομένα που έχουν παρόμοια στατιστικά χαρακτηριστικά με τα αληθινά δεδομένα, με τον discriminator να χρησιμοποιείται μόνο για τον προσδιορισμό των χαρακτηριστικών που θεωρούνται σημαντικά.

Συγκεκριμένα, ο generator εκπαιδεύεται ώστε να προβλέπει τη μέση τιμή των features σε ένα ενδιάμεσο layer του discriminator, ώστε να κατανοήσει ποια χαρακτηριστικά είναι αυτά που βοηθούν περισσότερο τον discriminator να πάρει την απόφασή του. Ο discriminator εκπαιδεύεται με τον γνωστό τρόπο, ενώ η συνάρτηση απώλειας του generator, αν f το ενδιάμεσο layer που θέλουμε να προσεγγίσουμε, είναι η εξής: $\|\mathbb{E}_{x\sim p_{data}} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z))\|_2^2$. Το feature matching χρησιμοποιήθηκε για αύξηση της σταθερότητας του μοντέλου και οδήγησε σε σύγκλιση σε περιπτώσεις που ένα τυπικό μοντέλο GAN κατέρρευσε.

3.3.5.2: Minibatch Discrimination

Λίγο πριν την κατάρρευση σε ένα σημείο οι παράγωγοι του discriminator έχουν παρόμοιες, σχεδόν ίδιες κατευθύνσεις για πολλά σημεία. Εφ'όσον τώρα ο discriminator επεξεργάζεται κάθε δεδομένο του generator ανεξάρτητα από τα υπόλοιπα, δεν υπάρχει κάποιος τρόπος να "ειδοποιηθεί" ο generator ώστε να προσπαθήσει να κάνει τις εξόδους του περισσότερο ανόμοιες μεταξύ τους. Έτσι, όλες οι έξοδοι κατευθύνονται προς το σημείο αυτό, το οποίο ο discriminator προσωρινά θεωρεί αρκετά ρεαλιστικό, και στη συνέχεια εγκλωβίζονται.

Το gradient descent είναι ανίκανο να διαχωρίσει τις πανομοιότυπες εξόδους, οπότε το μόνο που κάνουν οι παράγωγοι του discriminator είναι απλά να μεταφέρουν γύρω γύρω τη θέση του μοναδικού σημείου που παράγεται από τον generator, με το μοντέλο ανίκανο να συγκλίνει σε μία κατανομή με τη σωστή εντροπία.

Μία προφανής τακτική για την αποφυγή της παθολογικής αυτής κατάστασης είναι να επιτρέψουμε στον discriminator να λαμβάνει υπόψη πολλαπλούς συνδυασμούς δεδομένων (υπό αυτή τη λογική, το minibatch discrimination είναι περισσότερο οικογένεια τεχνικών, με το batch normalization να μπορεί να θεωρηθεί ως υλοποίηση minibatch discrimination). Σημασία δόθηκε

στην αναγνώριση δειγμάτων που βρίσκονται πολύ κοντά μεταξύ τους: Έστω $f(x_i) \in \mathbb{R}^A$ ένα διάνυσμα χαρακτηριστικών της εισόδου x_i σε κάποιο layer του discriminator. Το $f(x_i)$ πολλαπλασιάζεται με έναν tensor $T \in \mathbb{R}^{A \times B \times C}$ και προκύπτει ο πίνακας $M_i \in \mathbb{R}^{B \times C}$. Υπολογίζεται η L_1 – απόσταση μεταξύ των γραμμών των M_i, $i \in \{0, 1, ..., n\}$, η οποία τροφοδοτείται σε αρνητικό εκθετικό $c_b(x_i, x_j) = \exp(-\|M_{i,b} - M_{j,b}\|_{L1}) \in \mathbb{R}$.

Στη συνέχεια, η έξοδος του minibatch στρώματος $o(x_i)$ ορίζεται ως το άθροισμα όλων των $c_b(x_i, x_j)$ των υπόλοιπων δειγμάτων:

$$o(x_i)_b = \sum_{j=1}^{n} c_b(x_i, x_j) \in \mathbb{R}$$

$$o(x_i) = [o(x_1), o(x_2), ..., o(x_n)] \in \mathbb{R}^B$$

$$o(\mathbf{X}) \in \mathbb{R}^{n \times B}$$

Προωθούμε την έξοδο $o(x_i)$ του minibatch layer μαζί με την έξοδο του ενδιάμεσου στρώματος $f(x_i)$ στο επόμενο στρώμα του discriminator. Τα minibatch χαρακτηριστικά υπολογίζονται ξεχωριστά για τα δείγματα του generator και ξεχωριστά για τα αληθινά δεδομένα εκπαίδευσης. Έτσι, αφ'ενός ο discriminator διευκολύνεται, εφ'όσον τώρα έχει στη διάθεσή του επιπλέον παράπλευρες πληροφορίες, αφ'ετέρου ο generator μπορεί να αποφύγει ευκολότερα την κατάρρευση.

Πράγματι, πειραματικά διαπιστώθηκε ότι όντως η χρήση τεχνικών minibatch discrimination οδήγησε σε οπτικά ανώτερες εικόνες γρηγορότερα από το feature matching, το οποίο όμως ήταν πολύ καλύτερο εάν ο σκοπός ήταν να δημιουργηθεί ένας ισχυρός ταξινομητής για χρήση σε περιβάλλοντα ημι-επιβλεπόμενης μάθησης.

3.3.5.3: Ομαλοποίηση Ιστορικού Μέσου Όρου

Στην τεχνική αυτή τα κόστη των δικτύων τροποποιούνται, ώστε να συμπεριλάβουν και τον όρο $\|\boldsymbol{\theta} - \frac{1}{t}\sum_{i=1}^{t} \theta[i]\|^2$, όπου θ[i] είναι η τιμή των παραμέτρων μία προηγούμενη χρονική στιγμή i. Ο ιστορικός μέσος όρος ανανεώνεται απευθείας, ώστε να μπορεί να χρησιμοποιηθεί η τεχνική αυτή σε περιπτώσεις που θα χρειαστούν πολλές εποχές εκπαίδευσης. Πειραματικά βρέθηκε ότι η τεχνική αυτή μπορεί να βρει σημεία ισορροπίας σε συνεχή, μη κυρτά παίγνια χαμηλών διαστάσεων, στα οποία το gradient descent αποτυγχάνει.

3.3.5.4: Μονόπλευρη Ομαλοποίηση Ετικετών

Η ομαλοποίηση ετικετών αντικαθιστά τις τιμές-στόχους 1 και 0 με τιμές α και β (π.χ. α = 0.9 και β = 0.1). Στην περίπτωση των GANs, αυτό έχει ως αποτέλεσμα ο βέλτιστος discriminator να γίνεται $D(x) = \frac{\alpha p_{data}(\mathbf{x}) + \beta p_{model}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{model}(\mathbf{x})}$. Η παρουσία του p_{model} στον αριθμητή είναι προβληματική, διότι σε περιοχές όπου το p_{data} είναι σχεδόν μηδενικό και το p_{model} μεγάλο δείγματα του generator θα ταξινομηθούν με μεγάλη σιγουριά ως αληθινά από τον discriminator, με αποτέλεσμα να μην υπάρχει κίνητρο για το p_{model} να προσεγγίσει το p_{data}. Για το λόγο αυτό ομαλοποίηση εφαρμόζεται μόνο στα αληθινά δεδομένα, η παράμετρος β δηλαδή ισούται με 0.

3.3.5.5: Εικονικό Batch Normalization

Παρά την αύξηση στη σταθερότητα που προσφέρει το batch normalization, έχει ως αποτέλεσμα η έξοδος ενός νευρωνικού δικτύου σε είσοδο x να εξαρτάται σε μεγάλο βαθμό από άλλες εισόδους στο ίδιο batch. Στο εικονικό batch normalization κάθε είσοδος κανονικοποιείται με βάση τις στατιστικές τιμές ενός batch αναφοράς, που επιλέγεται στην αρχή της εκπαίδευσης.

3.3.6: Θεωρητική Θεμελίωση των Προβλημάτων Εκπαίδευσης

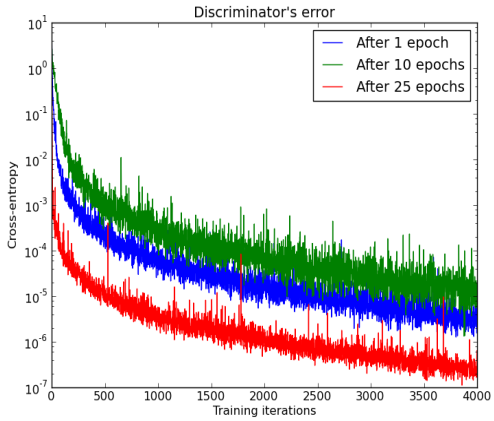
Παρά την πολυπλοκότητα μερικών από τις παραπάνω τεχνικές, δεν έλυσαν τα προβλήματα των GANs όσον αφορά την αστάθεια και τη δυσκολία στην εκπαίδευση, ακριβώς επειδή ήταν εμπειρικές τεχνικές χωρίς κάποιο αξιόλογο μαθηματικό υπόβαθρο. Για το λόγο αυτό, οι Arjovsky και Bottou [70] ερεύνησαν θεωρητικά τον τρόπο λειτουργίας, τις ιδιότητες και τα προβλήματα των GANs υπό την αρχική υλοποίηση.

Συγκεκριμένα, εφ'όσον ο D προσπαθεί να μειώσει την JS απόκλιση μεταξύ των δύο κατανομών, η θεωρία λέει ότι θα μπορούσαμε να τον εκπαιδεύουμε μέχρι τη βελτιστότητα, και μετά να εκπαιδεύουμε εναλλάξ για λίγο τους G και D. Στην πράξη όμως, όσο ο D βελτιώνεται, τόσο χειρότερα γίνονται τα updates των βαρών του G. Οι Goodfellow et al. [59] υπέθεσαν ότι αυτό οφείλεται σε κορεσμό και πρότειναν τη χρήση μίας εναλλακτικής, παρόμοιας συνάρτησης κόστους, η οποία ωστόσο πειραματικά εμφάνισε την ίδια μακροσκοπική συμπεριφορά.

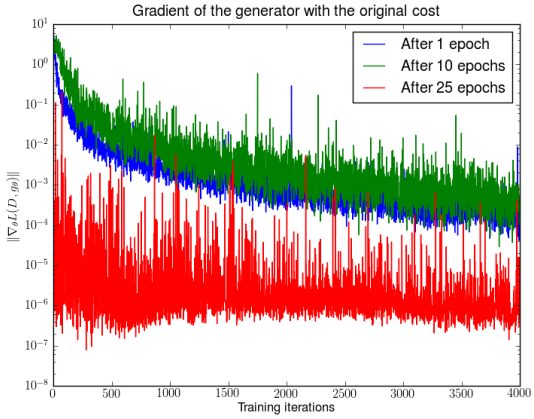
Θεωρητικά, ο D έχει κόστος το πολύ $2\log 2 - JS(\mathbb{P}_r \,||\, \mathbb{P}_g)$. Ωστόσο, πειράματα έδειξαν (Εικόνες 3.22 και 3.23) ότι αν ο D εκπαιδευτεί μέχρι τη βελτιστότητα το κόστος αυτό γίνεται 0, δηλαδή η JS είναι στο μέγιστό της, κάτι που συμβαίνει μόνο αν ισχύει μία συγκεκριμένη συνθήκη (οι κατανομές να μην είναι συνεχείς ή να έχουν supports ξένα μεταξύ τους). Αποδείχθηκε ότι, σε περίπτωση που οι D και G μοντελοποιούνται από νευρωνικά δίκτυα, η συνθήκη αυτή ισχύει, με αποτέλεσμα να υπάρχει ανά πάσα στιγμή τέλειος discriminator μεταξύ των δύο κατανομών, ένας discriminator δηλαδή που όχι μόνο μαντεύει σωστά με ακρίβεια 1, αλλά και στις περισσότερες περιπτώσεις έχει μηδενική παράγωγο. Αυτό είναι προφανώς καταστροφικό, γιατί καθιστά αδύνατη την εκπαίδευση με back propagation, και οφείλεται στο ότι αν οι κατανομές δεν ταυτίζονται 100%, όσο κοντά και να είναι, οι JS και KL αποκλίσεις είναι σταθερές ($\log 2$ και $+\infty$ αντίστοιχα).

Ό,τι έχει αναφερθεί μέχρι τώρα ισχύει στο θεωρητικό πλαίσιο που ο D εκπαιδεύεται μέχρι τη βελτιστότητα. Δυστυχώς, αποδείχθηκε ότι και σε πρακτικές καταστάσεις, όπου δηλαδή ο D είναι μία προσέγγιση του βέλτιστου discriminator D^*, όσο πιο κοντά είναι ο D στον D^*, τόσο οι παράγωγοι του G εξαφανίζονται. Έτσι, οι ανανεώσεις του D είτε θα είναι μηδενικές, είτε θα είναι ανακριβείς, με αποτέλεσμα η ποιότητα των τεχνητών δεδομένων να κρίνεται χωρίς κάποιο θεμελιωμένο μαθηματικό κριτήριο από τον άνθρωπο, κάτι που, πιο σημαντικά, έχει ως αποτέλεσμα την αυθαίρετη λήψη της απόφασης για το τέλος της εκπαίδευσης.

Προτάθηκε η χρήση της Wasserstein απόστασης [71] μεταξύ των κατανομών λόγω των πολύ καλών θεωρητικών ιδιοτήτων της.



Σχήμα 3.22: Μεγιστοποίηση της JS απόκλισης μεταξύ των δύο κατανομών με την εκπαίδευση του discriminator.



Σχήμα 3.23: Vanishing gradients στον generator υπό την JS απόκλιση.

3.3.7: Wasserstein GAN

Η μεγαλύτερη διαφορά ανάμεσα στις συναρτήσεις απόστασης είναι η επίδραση που έχουν στη σύγκλιση μεταξύ δύο κατανομών. Μία αλληλουχία κατανομών $\mathbb{P}_t$, $t \in \mathbb{N}$ λέγεται ότι συγκλίνει αν και μόνο αν υπάρχει κατανομή $\mathbb{P}_\infty$ τέτοια ώστε το $\rho(\mathbb{P}_t, \mathbb{P}_\infty)$ να τείνει στο μηδέν, καθώς το t τείνει στο ∞, κάτι που προφανώς εξαρτάται άμεσα από τον τρόπο με τον οποίο έχει οριστεί το ρ. Μία απόσταση ρ λέγεται ότι επιφέρει μία πιο αδύναμη τοπολογία όταν διευκολύνει μία αλληλουχία κατανομών να συγκλίνει. Πιο τεχνικά, η τοπολογία που επιφέρεται από την ρ είναι πιο αδύναμη από αυτή που επιφέρεται από την ρ' όταν το σύνολο των αλληλουχιών που συγκλίνουν υπό την ρ είναι υπερσύνολο του αντίστοιχου υπό την ρ'.

Είναι επιθυμητό, για να είναι εφικτή η βελτιστοποίηση των παραμέτρων θ, η κατανομή του μοντέλου μας, $\mathbb{P}_\theta$, να είναι τέτοια ώστε η αντιστοίχιση $\theta \rightarrow \mathbb{P}_\theta$ να είναι συνεχής. Έτσι, όταν μία αλληλουχία παραμέτρων θ_t συγκλίνουν στην θ, οι $\mathbb{P}_{\theta t}$ επίσης συγκλίνουν στην $\mathbb{P}_\theta$. Αυτό είναι ισοδύναμο με το να είναι συνεχής η συνάρτηση ρ με την οποία μετράμε την απόσταση δύο κατανομών.

Οι πιο ευρέως χρησιμοποιούμενες αποστάσεις είναι οι:

- Total Variation (TV) απόσταση:

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

- Kullback-Leibler (KL) απόκλιση:

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x)\, d\mu(x)$$, όπου οι $\mathbb{P}_r$ και $\mathbb{P}_g$ θεωρούνται συνεχείς, ώστε να δέχονται πυκνότητες σχετικά με κάποια μετρική μ ορισμένη στο X^2. Η KL απόκλιση είναι εξαιρετικά ασύμμετρη, και πιθανώς άπειρη όταν υπάρχουν σημεία τέτοια ώστε $\mathbb{P}_g(x) = 0$ και $\mathbb{P}_r(x) > 0$.

- Jensen-Shannon (JS) απόκλιση:

$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r || \mathbb{P}_m) + KL(\mathbb{P}_g || \mathbb{P}_m)$, όπου $\mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2$ η "ανάμεικτη" κατανομή. Η απόκλιση αυτή είναι συμμετρική και ορισμένη παντού, επειδή ως μετρική μ στις δύο KL αποκλίσεις μπορούμε να ορίσουμε την $\mu = \mathbb{P}_m$.

Οι Arjovsky et al. [72] εξέτασαν τις παραπάνω συναρτήσεις απόστασης και εισήγαγαν μία νέα συνάρτηση απόστασης που προσφέρει πολύ μεγαλύτερη ευελιξία, την Earth-Mover (EM) ή Wasserstein απόσταση: $W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$, όπου το $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ ορίζει το σύνολο όλων των κοινών κατανομών γ(x, y) των οποίων τα άκρα είναι οι $\mathbb{P}_r$ και $\mathbb{P}_g$ αντίστοιχα. Ενστικτωδώς, η EM απόσταση εκφράζει το ποσό της "μάζας" που πρέπει να μεταφερθεί από το x στο y ώστε η κατανομή $\mathbb{P}_r$ να μετασχηματιστεί στην $\mathbb{P}_g$. Όσον αφορά τη δύναμη των παραπάνω αποστάσεων, η KL απόκλιση είναι η δυνατότερη, ακολουθούμενη από τις JS και TV αποκλίσεις, με τελευταία και πιο αδύναμη την EM απόσταση.

Παρακάτω ακολουθεί παράδειγμα που δείχνει ότι ακόμα και πολύ απλές ακολουθίες κατανομών πιθανότητας που συγκλίνουν υπό την EM απόσταση αποκλίνουν υπό όλες τις υπόλοιπες:

Παράδειγμα 3.1:

Έστω Z ~ U[0, 1] η ομοιόμορφη κατανομή στο μοναδιαίο διάστημα. Έστω $\mathbb{P}_0$ η κατανομή του $(0, Z) \in \mathbb{R}^2$ (0 στον άξονα x και η τυχαία μεταβλητή στον άξονα y). Έστω τώρα $g_\theta(z) = (\theta, z)$ με θ μία πραγματική παράμετρο. Τότε:

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{αν } \theta \neq 0 \\ 0 & \text{αν } \theta = 0 \end{cases}$
- $KL(\mathbb{P}_0 || \mathbb{P}_\theta) = KL(\mathbb{P}_\theta || \mathbb{P}_0) = \begin{cases} \infty & \text{αν } \theta \neq 0 \\ 0 & \text{αν } \theta = 0 \end{cases}$
- $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{αν } \theta \neq 0 \\ 0 & \text{αν } \theta = 0 \end{cases}$

Καθώς $\theta_t \to 0$, η ακολουθία $(\mathbb{P}_{\theta t})$, $t \in \mathbb{N}$, συγκλίνει στο $\mathbb{P}_0$ υπό την EM απόσταση, αλλά όχι υπό τις JS, KL και TV αποκλίσεις. Έτσι, η κατανομή είναι εφικτό να μαθευτεί με gradient descent υπό την EM απόσταση, κάτι αδύνατο με κάποια από τις άλλες αποκλίσεις, αφού δεν είναι καν συνεχείς.

Θεώρημα 3.2:

Έστω $\mathbb{P}_r$ αμετάβλητη κατανομή πιθανότητας στο χώρο **X**. Έστω Z τυχαία μεταβλητή στο χώρο **Z**. Έστω g: $\mathbf{Z} \times \mathbb{R}^d \to \mathbf{X}$ συνάρτηση στην οποία θα αναφερόμαστε ως $g_\theta(z)$, με $z \in \mathbf{Z}$ και $\theta \in \mathbb{R}^d$. Έστω $\mathbb{P}_\theta$ η κατανομή της $g_\theta(z)$. Τότε, αποδεικνύεται ότι:

1) Αν g συνεχής στο θ, τότε και συνεχής είναι και η $W(\mathbb{P}_r, \mathbb{P}_\theta)$

2) Αν g τοπικά Lipschitz και ικανοποιεί τη συνθήκη 1, τότε η $W(\mathbb{P}_r, \mathbb{P}_\theta)$ είναι συνεχής παντού και παραγωγίσιμη σχεδόν παντού.

3) Τα (1) και (2) δεν ισχύουν για τις JS και KL αποκλίσεις.

Στη συνέχεια, οι [WGAN] έδειξαν ότι η εκπαίδευση νευρωνικών δικτύων με χρήση της Wasserstein απώλειας, τουλάχιστον από άποψη θεωρίας, έχει νόημα, αφού αν g_θ νευρωνικό δίκτυο με παραμέτρους θ, και p(z) μία προγενέστερη κατανομή τέτοια ώστε $\mathbb{E}_{z\sim p(z)}[||z||]<\infty$, τότε η συνθήκη (1) ικανοποιείται, οπότε η $W(\mathbb{P}_r, \mathbb{P}_\theta)$ είναι συνεχής παντού και παραγωγίσιμη σχεδόν παντού.

Τα παραπάνω μας οδηγούν στο συμπέρασμα ότι η EM απόσταση είναι πολύ πιο λογική συνάρτηση κόστους από την JS απόκλιση. Ωστόσο, το infimum της εξίσωσης [X] είναι υπολογιστικά ασύμφορο (intractable), γι' αυτό και στη θέση του χρησιμοποιείται η δυαδικότητα Kantorovich-Rubinstein, σύμφωνα με την οποία: $W(\mathbb{P}_r,\mathbb{P}_\theta)=\sup_{\|f\|_L\leq 1}\mathbb{E}_{x\sim\mathbb{P}_r}[f(x)]-\mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$, όπου το supremum λαμβάνεται υπόψη για όλες τις 1-Lipschitz (μία συνάρτηση πολλών μεταβλητών λέγεται K-Lipschitz αν και μόνο αν το μέτρο των παραγώγων της είναι φραγμένο από το K) συναρτήσεις $f : X \to \mathbb{R}$. Εάν αντικαταστήσουμε τη συνθήκη $||f||_L \leq 1$ με $||f||_L \leq K$, για κάποιο K, τότε οδηγούμαστε στο $K\cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$. Έτσι, σε μία παραμετροποιημένη οικογένεια συναρτήσεων $\{f_w\}$, $w \in \mathbf{W}$, K-Lipschitz, για κάποιο K, μπορούμε να λύσουμε το πρόβλημα $\max_{w\in W}\mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)]-\mathbb{E}_{z\sim\mathbb{P}_z}[f_w(g_\theta(z))]$ (3.2), και αν το supremum επιτυγχάνεται για κάποιο $w \in \mathbf{W}$, το παραπάνω μέγιστο οδηγεί σε μία προσέγγιση της $W(\mathbb{P}_r, \mathbb{P}_\theta)$ μέχρι μία πολλαπλασιαστική σταθερά. Ακόμη, μπορούμε να παραγωγίσουμε την $W(\mathbb{P}_r, \mathbb{P}_\theta)$, πάλι έως μία σταθερά, με χρήση back propagation, εκτιμώντας το $\mathbb{E}_{z\sim\mathbb{P}_z}[\nabla_\theta f_w(g_\theta(z))]$ (3.3). Έτσι, το WGAN μπορεί να εκπαιδευτεί, με τον discriminator να προσπαθεί να μεγιστοποιήσει την παράγωγο της Εξίσωσης (3.2) και τον generator να προσπαθεί να ελαχιστοποιήσει την Εξίσωση (3.3).

Το μόνο θέμα που μένει είναι η εύρεση της κατάλληλης συνάρτησης f. Η λύση που προτάθηκε αρχικά ήταν η χρήση ενός δικτύου του οποίου τα βάρη βρίσκονται σε έναν συμπαγή χώρο W και η εκπαίδευσή του με back propagation μέσω της Εξίσωσης (3.2). Το γεγονός ότι ο W είναι συμπαγής έχει ως αποτέλεσμα όλες οι συναρτήσεις f_w να είναι K-Lipschitz, για κάποιο K που εξαρτάται μόνο από το W και όχι από τα εκάστοτε βάρη, οπότε η συνάρτηση απώλειας προσεγγίζεται έως έναν πολλαπλασιαστικό παράγοντα χωρίς επίδραση. Μία προφανής λύση είναι να ψαλιδίζουμε τα βάρη του δικτύου μετά από κάθε ανανέωση των παραγώγων, ώστε να είναι πάντα ανάμεσα σε ένα "κουτί", π.χ. $W = [-0.01, 0.01]^l$. Ο αλγόριθμος εκπαίδευσης του WGAN ακολουθεί παρακάτω.

Η τεχνική του ψαλιδίσματος χρησιμοποιήθηκε αφ'ενός λόγω απλότητας, και αφ'ετέρου επειδή όλες οι υπόλοιπες, πιο περίπλοκες τεχνικές που δοκιμάστηκαν αρχικά, όπως η προβολή των βαρών σε μία επιφάνεια σφαίρας, είχαν παρόμοια εμπειρικά αποτελέσματα. Είναι προφανές ότι η τεχνική αυτή δεν οδηγεί σε βέλτιστα αποτελέσματα, αφού η επιλογή των ορίων του κουτιού είναι εντελώς αυθαίρετη και άκαμπτη. Μια βελτιωμένη τεχνική αναπτύχθηκε από τους Gulrajani et al. [73].

Αλγόριθμος 3.2: Αλγόριθμος εκπαίδευσης WGAN. Υπερπαράμετροι είναι το n_{critic}=5, οι φορές που εκπαιδεύεται ο critic ανά εκπαίδευση του generator, το α=0.00005, ο learning rate, το c=0.01, τα όρια του "κουτιού" και το m=64, το batch size.
Για αριθμό επαναλήψεων: Για n_{critic}: • Κάνουμε δειγματοληψία m δεδομένων $\{x^{(1)}, \ldots, x^{(m)}\}$ από την κατανομή των αληθινών δεδομένων, $\mathbb{P}_r$. • Κάνουμε δειγματοληψία m διανυσμάτων θορύβου $\{z^{(1)}, \ldots, z^{(m)}\}$ από την προγενέστερη κατανομή, p_z. • $g_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$

- $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
- $w \leftarrow \text{clip}(w, -c, c)$

- Κάνουμε δειγματοληψία m διανυσμάτων θορύβου $\{z^{(1)}, \ldots, z^{(m)}\}$ από την προγενέστερη κατανομή, p_z.
- $g_\theta \leftarrow -\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
- $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

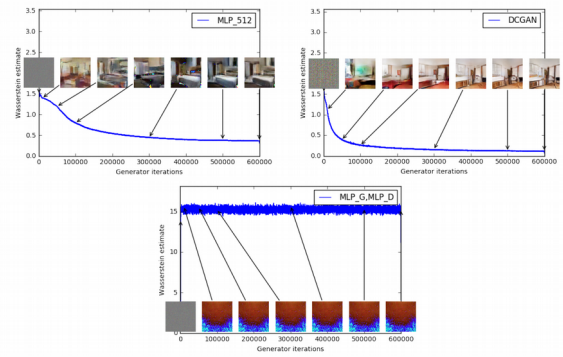
Το γεγονός ότι η EM απόσταση είναι συνεχής και παραγωγίσιμη σχεδόν παντού σημαίνει ότι όχι μόνο μπορούμε, αλλά και πρέπει να εκπαιδεύσουμε τον discriminator μέχρι τη βελτιστότητα: Όσο περισσότερο εκπαιδεύουμε τον discriminator τόσο πιο αξιόπιστες γίνονται οι παράγωγοι της EM απόστασης, κάτι που με τη σειρά του βοηθάει τον generator. Όσον αφορά την JS απόκλιση, ναι μεν με την εκπαίδευση του discriminator οι παράγωγοι γίνονται πιο αξιόπιστες, αλλά κοντά στο βέλτιστο σημείο οι αληθινές παράγωγοι είναι μηδενικές λόγω τοπικού κορεσμού, όπως εξηγείται στο [TOWARDS], οπότε είναι πιο εύκολο να οδηγηθούμε σε κατάρρευση. Το φαινόμενο αυτό φαίνεται στις Εικόνες 3.24 (Wasserstein) και 3.25 (JS).

Πιο συγκεκριμένα, στην πάνω αριστερά γραφική παράσταση του Σχήματος 3.24 ο generator είναι ένα MLP 4 layers, με 512 hidden units ανά layer και πάνω δεξιά είναι ένα DCGAN, ενώ και στις δύο περιπτώσεις ο critic είναι DCGAN. Βλέπουμε ότι καθώς η απώλεια μειώνεται η ποιότητα των τεχνητών εικόνων αυξάνεται. Ακόμη, μπορούμε να εκτιμήσουμε ποιοτικά πολύ πιο αξιόπιστα τόσο την πορεία, αλλά και την ισχύ των μοντέλων απλά παρατηρώντας την γραφική παράσταση της απώλειας, αφού ο DCGAN generator παράγει εμφανώς καλύτερης ποιότητας εικόνες, ενώ η απώλειά του αφ'ενός συγκλίνει πολύ πιο γρήγορα στο μηδέν, αφ'ετέρου παραμένει σταθερά σε πιο χαμηλά επίπεδα από την αντίστοιχη του MLP, όπως φαίνεται στις δύο πάνω γραφικές παραστάσεις του Σχήματος 3.24. Στην κάτω εικόνα και τα δύο μέρη της αρχιτεκτονικής είναι MLP, η εκπαίδευση αποτυγχάνει και η απώλεια παραμένει σταθερή.

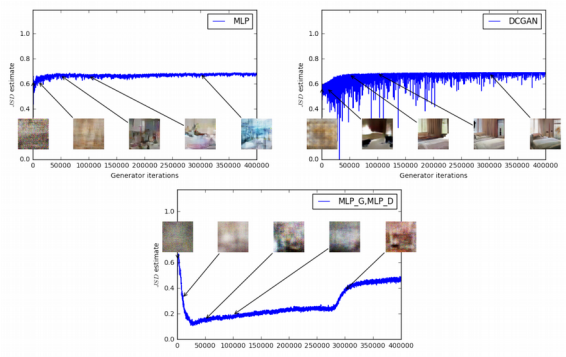
Αντίθετα, όταν τα δίκτυα εκπαιδεύονται υπό την JS απόκλιση βλέπουμε ότι παρά την αύξηση στην ποιότητα των εικόνων, η απώλεια και στις δύο αρχιτεκτονικές του πάνω μέρους παραμένει σταθερή, και μάλιστα ίση με ~0.69 = log2, τη μέγιστη τιμή της JS απόκλισης. Τα πράγματα είναι ακόμα χειρότερα για τη γραφική παράσταση του κάτω μέρους η οποία ανεβοκατεβαίνει φαινομενικά τυχαία, ανεξάρτητα από την ποιότητα των εικόνων.

Τα δίκτυα εκπαιδεύτηκαν κυρίως στο LSUN dataset, και πειραματικά φάνηκε ότι υπήρχε συσχέτιση ανάμεσα στην μείωση της EM απώλειας και στη βελτίωση της ποιότητας των παραγόμενων εικόνων, κάτι που δεν ίσχυε υπό την JS απόκλιση. Το γεγονός αυτό ήταν καίριο για την καθιέρωση των GANs, αφού πλέον ήταν και πειραματικά επιβεβαιωμένο πως δεν ήταν απαραίτητη η ανθρώπινη εκτίμηση της ποιότητας των παραγόμενων εικόνων.

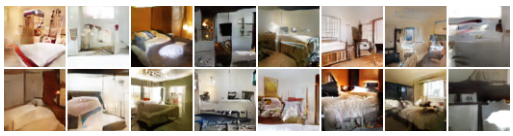
Ακολουθούν μερικές τεχνητές εικόνες που δημιουργήθηκαν από DCGAN generator υπό την EM απώλεια. Η ποιότητά τους είναι σαφώς ανώτερη από τις αντίστοιχες των Radford et al. [68].



Σχήμα 3.24: Γραφικές παραστάσεις απώλειας του discriminator υπο την Wasserstein απόσταση.



Σχήμα 3.25:Γραφικές παραστάσεις απώλειας του discriminator υπο την JS απόκλιση.



Σχήμα 3.26: Τεχνητές εικόνες του LSUN από DCGAN generator, εκπαιδευμένο με την Wasserstein απώλεια.

3.3.8: Improved Wasserstein GAN (Gradient Penalty)

Παρά τα θεωρητικά πλεονεκτήματα της EM απόστασης, οι Gulrajani et al. [73] έδειξαν ότι η ικανοποίηση της συνθήκης Lipschitz με ψαλίδισμα των βαρών του discriminator ήταν πιθανό να οδηγήσει σε προβληματικά σενάρια, και πρότειναν ως εναλλακτική λύση μία τεχνική που ονόμασαν gradient penalty.

Στην τεχνική αυτή το μέτρο της παραγώγου του discriminator ως προς την είσοδο x περιορίζεται ενσωματώνοντάς το στη συνάρτηση κόστους. Παρ'όλ'αυτά, η απευθείας εφαρμογή της τεχνικής αυτής θα ήταν υπολογιστικά ασύμφορη, οπότε για το λόγο αυτό πραγματοποιούμε κάθε φορά τυχαία δειγματοληψία $\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}$:

$$L = \mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} [D(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbb{P}_r} [D(\mathbf{x})] - \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} [(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]$$

Μερικές διευκρινήσεις:

- **Κατανομή της δειγματοληψίας**: Η $\mathbb{P}_{\hat{\mathbf{x}}}$ ορίζεται έμμεσα, λαμβάνοντας δείγματα ομοιόμορφα κατά μήκος ευθείων γραμμών μεταξύ ζευγαριών σημείων από τις κατανομές $\mathbb{P}_r$ και $\mathbb{P}_g$ του discriminator και του generator αντίστοιχα. Η επιλογή αυτή λήφθηκε με βάση το πιο σημαντικό αποτέλεσμα του άρθρου, που είναι το γεγονός ότι ο γράφος του βέλτιστου discriminator, λόγω της συνθήκης 1-Lipschitz που ικανοποιεί, αποτελείται από ευθείες γραμμές που συνδέουν σημεία μεταξύ των δύο κατανομών. Η εφαρμογή του περιορισμού του μέτρου της παραγώγου σε όλο το χώρο δεν είναι εφικτή, οπότε χρησιμοποιείται η προσέγγιση αυτή, η οποία φαίνεται να αρκεί και να οδηγεί σε πολύ καλή πειραματική συμπεριφορά.
- **Συντελεστής τιμωρίας**: Ο συντελεστής αυτός ορίζει τη σημασία που θα δοθεί στον περιορισμό του μέτρου της παραγώγου σε μία αντικρουόμενη κατάσταση, π.χ. η αλλαγή μίας παραμέτρου να οδηγεί σε μείωση του μέτρου, αλλά ταυτόχρονα σε αύξηση της EM απόστα-

σης. Οι συγγραφείς επέλεξαν εμπειρικά λ = 10 λόγω καλής πειραματικής συμπεριφοράς σε μεγάλο εύρος αρχιτεκτονικών και datasets.

- **Όχι χρήση batch normalization στον discriminator**: Το batch normalization μετασχηματίζει το στόχο του discriminator από το "ταίριασμα" μίας εισόδου σε μία έξοδο, στο "ταίριασμα" μίας ολόκληρης δέσμης εισόδων σε μία άλλη δέσμη εξόδων. Στο πλαίσιο αυτό δεν μπορεί να εφαρμοστεί το gradient penalty, αφού το μέτρο της παραγώγου του discriminator τιμωρείται με βάση κάθε είσοδο ανεξάρτητα από τις άλλες, και όχι ολόκληρη τη δέσμη. Για το λόγο αυτό το batch normalization παραλείπεται, και στη θέση του προτείνεται η χρήση του layer normalization [74], όπου αντί να γίνεται κανονικοποίηση στον άξονα των features με βάση τη μέση τιμή και την τυπική απόκλιση όλων των δεδομένων του batch, γίνεται με βάση τα στατιστικά χαρακτηριστικά του ίδιου του δεδομένου που εξετάζεται. Το layer normalization προσφέρει τα πλεονεκτήματα του batch normalization μαζί με στατιστική ανεξαρτησία, ενώ μπορεί να λειτουργήσει αξιόπιστα και με πιο μικρά batch sizes.
- **Αμφίπλευρη τιμωρία**: Το μέτρο της παραγώγου του discriminator ενθαρρύνεται να κατευθύνεται προς τη μονάδα, και όχι απλά να την προσεγγίζει από κάτω, διότι πειραματικά βρέθηκε ότι έτσι αυξάνεται όχι μόνο η ταχύτητα σύγκλισης, αλλά η ποιότητα των βέλτιστων σημείων.

Πειράματα έδειξαν ότι σε απλά datasets (βλ. Σχήμα 3.27 αριστερά) με δεδομένο generator ($\mathbb{P}_g = \mathbb{P}_r$ + Γκαουσιανός θόρυβος μοναδιαίας διασποράς) ο discriminator στον οποίο εφαρμοζόταν το ψαλίδισμα (WGAN) αδυνατεί να μάθει χαρακτηριστικά της κατανομής με υψηλότερο επίπεδο και αντ'αυτού περιορίζεται σε γραμμικές προσεγγίσεις της βέλτιστης λύσης. Αυτό συμβαίνει επειδή ο βέλτιστος k-Lipschitz discriminator έχει σχεδόν παντού μέτρο παραγώγου k (βλ. Σχήμα 3.28 δεξιά πάνω), κάτι που στην περίπτωση του ψαλιδίσματος μπορεί να επιτευχθεί μόνο εάν οι τιμές του περιοριστούν στις ακραίες περιοχές του "κουτιού". Αντίθετα, ο discriminator στον οποίο εφαρμοζόταν το gradient penalty (WGAN-GP) δεν εμφανίζει τέτοια παθολογική συμπεριφορά, με την κατανομή των τιμών των βαρών του να εμφανίζει πολύ μεγαλύτερο εύρος, όπως φαίνεται και στο δεξιά κάτω μέρος του Σχήματος 3.28.

Επιπλέον, πειραματικά φάνηκε ότι η αυθαίρετη επιλογή της τιμής ψαλιδίσματος μπορεί πολύ εύκολα να οδηγήσει σε προβλήματα vanishing/exploding gradients, κάτι που αναγνώρισαν και οι συγγραφείς του αρχικού WGAN άρθρου. Πιο συγκεκριμένα, εκπαίδευσαν στο toy dataset του Ελβετικού Ρολό MLP 12 στρωμάτων, με συνάρτηση ενεργοποίησης την ReLU, χωρίς batch normalization, με τιμές ψαλιδίσματος [10^{-1}, 10^{-2}, 10^{-3}], και εκτός από την αδυναμία του discriminator να συλλάβει και σε αυτή την περίπτωση τα πιο λεπτά χαρακτηριστικά της κατανομής (Σχήμα 3.27 δεξιά), διαπίστωσαν ότι οι παράγωγοι των δικτύων είτε αυξάνονταν είτε συρρικνώνονταν εκθετικά καθώς προχωρούσαν όλο και πιο πίσω στο δίκτυο. Αντίθετα, τα δίκτυα που εκπαιδεύτηκαν με gradient penalty είχαν εμφανώς πιο σταθερές παραγώγους, κάτι που εν γένει κάνει εφικτή την εκπαίδευση πιο βαθιών και περίπλοκων δικτύων. Οι διαφορές αυτές εντοπίζονται στο αριστερό μέρος του Σχήματος 3.28

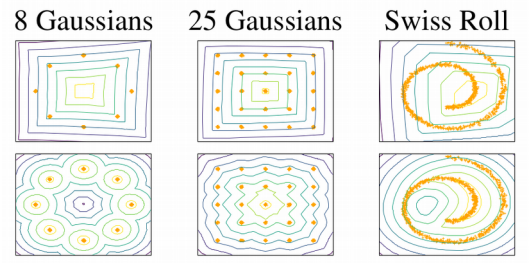
Όσον αφορά τα πειράματα σε κάποιο από τα γνωστά datasets, εκπαιδεύτηκαν έξι αρχιτεκτονικές GAN στο LSUN, έχοντας ως σημείο αναφοράς την DCGAN των [DCGAN]. Αυτές ήταν:

- όχι batch normalization και χρήση σταθερού αριθμού φίλτρων στον generator, DCGAN discriminator
- ReLU MLP generator 4 στρωμάτων και 512 νευρώνων, DCGAN discriminator
- καμία χρήση normalization, ούτε στον generator ούτε στον discriminator
- χρήση μη-γραμμικοτήτων τύπου πολλαπλασιαστικών πυλών (sigmoid(x) · tanh(x))
- χρήση μη-γραμμικοτήτων τύπου tanh

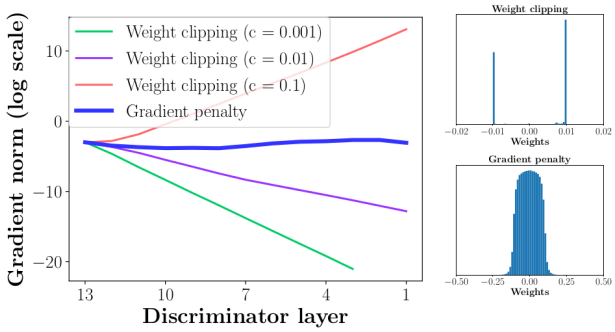
- ResNet 101 στρωμάτων σε generator και discriminator (η πρώτη φορά που επιτεύχθηκε ουσιώδης εκπαίδευση GAN με δομικά στοιχεία πολύ βαθιά residual δίκτυα.

Κάθε αρχιτεκτονική εκπαιδεύτηκε με τέσσερις διαδικασίες: DCGAN, LSGAN (Least Squares GAN), WGAN και WGAN-GP. Σε κάθε περίπτωση οι τιμές των παραμέτρων ήταν αυτές που προτείνονταν στα εκάστοτε άρθρα που τις εισήγαγαν, εκτός από την LSGAN, όπου και εφαρμόστηκε αναζήτηση για τη σύγκριση learning rates.

Τα αποτελέσματα παρουσιάζονται στο Σχήμα 3.29, όπου φαίνεται ότι το WGAN-GP είναι η μόνη τεχνική που καταφέρνει να εκπαιδεύσει επιτυχώς όλες τις αρχιτεκτονικές με τις προτεινόμενες, προκαθορισμένες παραμέτρους τους.



Σχήμα 3.27: Επίδοση του απλού (πάνω) και του βελτιωμένου (κάτω) discriminator σε toy datasets.



Σχήμα 3.28: Μέτρο των παραγώγων ως προς τη θέση των layers (αριστερά). Ιστόγραμμα των βαρών του απλού (πάνω) και βελτιωμένου (κάτω) discriminator.

Σχήμα 3.29: Αποτελέσματα της εκπαίδευσης των 6 αρχιτεκτονικών με τους 4 τρόπους στο LSUN.

Κεφάλαιο 4: Πειραματική Διαδικασία

4.1: Συλλογή και Προεπεξεργασία Δεδομένων

Σκοπός της παρούσας εργασίας είναι η αξιολόγηση της χρήσης των GANs ως προχωρημένη τεχνική εμπλουτισμού δεδομένων (data augmentation) σε tasks ιατρικών εφαρμογών. Συγκεκριμένα, εκπαιδεύτηκαν συνελικτικές αρχιτεκτονικές σε ένα task δυαδικής ταξινόμησης, αρχικά χωρίς εφαρμογή data augmentation, και στη συνέχεια με εφαρμογή των κλασικών τεχνικών (περιστροφή, μετακίνηση κατά μερικά pixels κ.λ.π). Η υπόθεσή μας ήταν ότι η δημιουργία τεχνητών δεδομένων μέσω μίας τεχνικής που δε λαμβάνει τις ήδη υπάρχουσες εικόνες απ' ευθείας ως είσοδο αλλά λειτουργεί στο πολύ πιο ισχυρό επίπεδο της στατιστικής κατανομής των δεδομένων θα οδηγούσε σε αύξηση της επίδοσης των αρχικών ταξινομητών. Όλος ο κώδικας που αφορούσε το κομμάτι της μηχανικής μάθησης της εργασίας γράφτηκε στο Keras, ένα API βαθιάς μάθησης σχεδιασμένο για ευκολία στη χρήση, με backend το TensorFlow.

Ως ασθένεια αρχικά επιλέχθηκε η ασθένεια του Parkinson, αλλά οι προσπάθειες αυτές εγκαταλείφθηκαν νωρίς, καθώς οι διαφορές μεταξύ ενός ασθενούς και ενός υγιούς ατόμου εντοπίζονται στα DaT scans. Αντίθετα, οι διαφορές των MRI τομών είναι ουσιαστικά ανεπαίσθητες, και για το λόγο αυτό επιλέχθηκε τελικά η ασθένεια του Alzheimer (Alzheimer's disease). Η ασθένεια αυτή είναι μία μη αναστρέψιμη νευροεκφυλιστική πάθηση, υπεύθυνη για την πλειοψηφία των περιστατικών άνοιας. Πλήττει κυρίως την μνήμη και τις γνωστικές ικανότητες μέσω της μόνιμης απενεργοποίησης νευρώνων και συνάψεων στον εγκεφαλικό φλοιό, τον κροταφικό και τον βρεγματικό λοβό, με τους ασθενείς να καταλήγουν στο τελικό στάδιο της ασθένειας να μην μπορούν να φέρουν εις πέρας ακόμα και τις πιο απλές δραστηριότητες.

Τα δεδομένα πάνω στα οποία πραγματοποιήθηκαν τα πειράματα αποτελούν υποσύνολο του ADNI dataset (Altzheimer's disease Neuroimaging Initiative) [75]. Συγκεκριμένα, χρησιμοποιήθηκαν T1 MRI τομές, ανεξάρτητα από την έρευνα στην οποία ανήκαν οι ασθενείς (ADNI1, ADNI2, ADNI3 κ.ο.κ.). Υπόψη λήφθηκαν μόνο οι ασθενείς που είχαν οριστεί ως Normal ή AD (Alzheimer's disease), αγνοήθηκαν δηλαδή οι ασθενείς που είχαν εκτιμηθεί ότι έπασχαν από MCI (mild cognitive impairment). Ακόμη, δε λήφθηκαν υπόψη οι επιδόσεις των ασθενών στα σχετικά τεστ (FAQ, GDSCALE, Global CDR, MMSE, NPI-Q). Χρησιμοποιήθηκαν μόνο οι axial τομές των ασθενών, αλλά το κριτήριο αυτό δεν καθορίστηκε κατά την αναζήτηση, καθώς η πλειοψηφία των δεδομένων ήταν σε ογκομετρικό format .nii (nifti). Για την καλύτερη δυνατή ποιότητα εικόνας επιλέχθηκαν προεπεξεργασμένα δεδομένα, ενώ για κάθε επίσκεψη ενός ασθενή ήταν διαθέσιμα τα δεδομένα σε κάθε στάδιο της προεπεξεργασίας (MPR-R, GradWarp, N3, Scaled/Scaled 2). Επιλέχθηκαν τα αρχεία MPR-R; GradWarp; N3; Scaled για τη μέγιστη ποιότητα, ενώ αγνοήθηκαν τα αρχεία MPR-R; GradWarp; N3; Scaled 2 για λόγους συνοχής των δεδομένων: Αφ'ενός ήταν πολύ λιγότερα, αφ'ετέρου δεν υπήρχε κάποια εμφανής διαφορά με τα Scaled.

Τελικά προέκυψαν 152 AD ασθενείς, με 475 επισκέψεις, και 179 Normal, με 823 επισκέψεις. Τα δεδομένα χωρίστηκαν τυχαία σε training, validation και test sets στο επίπεδο των ασθενών, ώστε να εκτιμηθεί η ικανότητα ή όχι των δικτύων που θα χρησιμοποιούνταν στη συνέχεια να γενικεύουν τις γνώσεις τους σε άτομα στα οποία δεν έχουν εκπαιδευτεί. Ωστόσο, προτού εφαρμοστεί ο χωρισμός του Normal συνόλου, λήφθηκε υπόψη μόνο το 58% των Normal ατόμων, για εξισορρόπηση του αριθμού των επισκέψεων, και άρα του τελικού αριθμού εικόνων, αφού κατά μέσο όρο ένα Normal άτομο είχε σχεδόν το διπλάσιο αριθμό επισκέψεων κατά μέσο όρο από έναν ασθενή AD. Στη συνέχεια οι axial τομές τύπου .png εκμαιεύθηκαν από τα ογκομετρικά δεδομένα .nii. Οι εικόνες που προέκυψαν όμως δεν είχαν όλες τις ίδιες διαστάσεις, αφού πιθανώς είχαν ληφθεί από διαφορετικά νοσοκομεία, με διαφορετικές πολιτικές λήψης, διαφορετικό διαθέσιμο χρόνο κ.λ.π. Στον Πίνακα 4.1 φαίνονται οι διάφορες διαστάσεις, καθώς και ο αριθμός των εικόνων με τις διαστάσεις αυτές. Ο αριθμός των Normal εικόνων λήφθηκε πριν την εξισορρόπηση των datasets.

AD		Normal	
Διαστάσεις	Αριθμός εικόνων	Διαστάσεις	Αριθμός εικόνων
(192, 192, 160)	186	(192, 192, 160)	335
(192, 192, 176)	1	(240, 256, 160)	61
(240, 256, 160)	49	(248, 256, 160)	1
(256, 256, 160)	1	(256, 256, 160)	2
(256, 256, 162)	1	(256, 256, 166)	347
(256, 256, 166)	195		

Πίνακας 4.1: Διαστάσεις και αριθμός εικόνων μετά από την μετατροπή των δεδομένων σε .png.

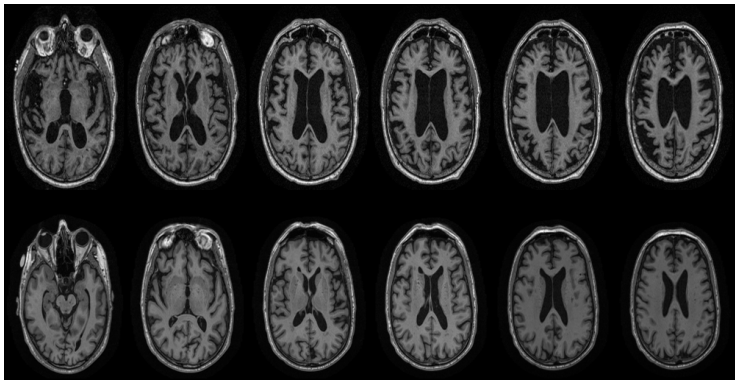
Σε μία επίσκεψη, το μηχάνημα, για λόγους πληρότητας, ξεκινάει να λαμβάνει δεδομένα από το ύψος του λαιμού του ασθενούς, και σταματά λίγο πιο πάνω από το τέλος του κεφαλιού. Έτσι, οι κάτω εικόνες (περίπου το πρώτο 40-45% των τομών) δεν απεικονίζουν τον εγκέφαλο, αλλά το στόμα, το λαιμό και τη μύτη των ασθενών, πληροφορίες δηλαδή που δεν περιέχουν χρήσιμες πληροφορίες για το αν πάσχουν από Alzheimer ή όχι, γι' αυτό και δε λήφθηκαν υπόψη. Παρόμοια, δε λήφθηκε υπόψη το τελευταίο 25% των εικόνων, που απεικονίζει το πάνω μέρος του κεφαλιού και το κρανίο του ασθενούς, αφού δεν επηρεάζονται καθόλου από την πάθηση.

Συγκεκριμένα, σε περίπτωση που η επίσκεψη αποτελούνταν από 192 axial τομές χρησιμοποιήθηκαν οι τομές 95-140, στην περίπτωση των 240 τομών οι 100-180 και τέλος στην περίπτωση των 256 τομών οι 120-200. Τέλος, για να υπάρχει ομοιομορφία στις διαστάσεις των τελικών εικόνων, αλλά και για λόγους υπολογιστικού κόστους, όλες οι εικόνες μετασχηματίστηκαν σε διαστάσεις 192×160 με χρήση Lanczos interpolation. Τέλος, τα δεδομένα τοποθετήθηκαν ενιαία σε φακέλους, ανάλογα με την κλάση στην οποία ανήκαν, για να είναι συμβατή η οργάνωση των αρχείων με τις απαιτήσεις του Keras. Η τελική κατανομή των εικόνων φαίνεται στον Πίνακα 4.2.

	training	validation	test
AD	25154	1975	1399
Normal	24298	2343	2139

Πίνακας 4.2: Κατανομή των δεδομένων

Παρακάτω ακολουθούν μερικές ενδεικτικές τομές από έναν AD και έναν Normal ασθενή:



Σχήμα 4.1: Ενδεικτικές MRI τομές ενός ασθενούς (πάνω) και ενός υγιούς ατόμου (κάτω).

4.2: Αρχιτεκτονικές GAN

Όσον αφορά το κομμάτι της δημιουργίας των τεχνητών εικόνων, εκπαιδεύτηκαν δύο GAN, ένα για κάθε κλάση. Η δομή των δικτύων φαίνεται στους Πίνακες 4.3 και 4.4.

Generator	
Layer	Αριθμός παραμέτρων
Input (128)	0
Dense (6·5·512)	1.981.440 + 61.440
Conv2DTrans_up (512)	6.554.112 + 2048
Conv2D (256)	3.277.056 + 1024
Conv2DTrans_up (256)	1.638.656 + 1024
Conv2D (128)	819.328 + 512
Conv2DTrans_up (128)	409.728 + 512
Conv2D (64)	204.864 + 256
Conv2DTrans_up (64)	102.464 + 256
Conv2D (32)	51.232 + 128
Conv2DTrans_up (32)	25.632 + 128
Conv2D (1)	801
	15.132.641 (Συνολικά)

Πίνακας 4.3: Δομή του generator μέρους της αρχιτεκτονικής.

Όλα τα Conv2D και Conv2DTrans_up layers εκτός από το τελευταίο ακολουθούνται από batch normalization layer στον άξονα των features, οι παράμετροι των οποίων είναι κάθε φορά το δεξί μέρος του αθροίσματος στην δεύτερη στήλη του Πίνακα 4.3, και χρησιμοποιούν ως συνάρτηση ενεργοποίησης την LeakyReLU. Ως συνάρτηση ενεργοποίησης του τελευταίου συνελικτικού layer επιλέχθηκε η υπερβολική εφαπτομένη (tanh), ώστε η τελική έξοδος του generator να είναι φραγμένη και να μπορεί έτσι να αναπαραστήσει εικόνες. Προτιμήθηκε η tanh γιατί το πεδίο τιμών της είναι κεντραρισμένο στο μηδέν, κάτι που βοηθάει στην εκπαίδευση [76], ενώ της σιγμοειδούς συνάρτησης (sigmoid) είναι κεντραρισμένο στο ½. Το Conv2D layer πραγματοποιεί συνέλιξη με μέγεθος πυρήνα 5 και χρήση zero padding, ενώ το Conv2DTrans_up πραγματοποιεί transposed συνέλιξη με μέγεθος πυρήνα 5, με χρήση zero padding και βήμα 2, διπλασιάζει δηλαδή τις χωρικές διαστάσεις της εικόνας.

Ως είσοδος στον generator δίνεται ένα διάνυσμα 128 τυχαίων μεταβλητών από 0 έως 1 που ακολουθούν την ομοιόμορφη κατανομή, ενώ η έξοδός του είναι ένας [192,160] πίνακας με τιμές από -1 έως 1. Δοκιμάστηκαν και ισχυρότεροι generators ως προς το μέγεθος του latent vector (256 και 512 τυχαίες μεταβλητές), αλλά όλες τις φορές το δίκτυο οδηγήθηκε σε κατάρρευση. Την είσοδο ακολουθεί fully connected layer 6·5·512 νευρώνων, που στη συνέχεια μετασχηματίζεται σε τρισδιάστατο πίνακα διαστάσεων [6,5,512], ουσιαστικά δηλαδή σε μία εικόνα διαστάσεων 6×5 και 512 χαρακτηριστικών. Από το σημείο αυτό και μετά το δίκτυο αποτελείται από ζεύγη layers που πραγματοποιούν αρχικά 2-upscaling transposed συνελίξεις και στη συνέχεια συνελίξεις με τον μισό αριθμό features. Συνολικά έχουμε πέντε διπλασιασμούς, οπότε προκύπτει μία εικόνα διαστάσεων $6 \cdot 2^5 \times 5 \cdot 2^5 = 192 \times 160$, ενός καναλιού.

Discriminator	
Layer	Αριθμός παραμέτρων
Input ((192,160,1))	0
Conv2D (32)	832
Conv2D_down(64)	51.264
Conv2D (64)	102.464
Conv2D_down (64)	102.464
Conv2D (64)	102.464
Conv2D_down (128)	204.928
Conv2D (128)	409.728
Conv2D_down (128)	409.728
Conv2D (128)	409.728
Dense(512)	7.864.832
Dense(1)	513
	9.658.945 (Συνολικά)

Πίνακας 4.4: Δομή του discriminator μέρους της αρχιτεκτονικής.

Οι παράμετροι του discriminator τώρα επιλέχθηκαν με κάπως αυθαίρετο τρόπο, για λόγους μνήμης. Ο discriminator ξεκινάει με ένα συνελικτικό layer 32 χαρακτηριστικών. Στη συνέχεια βρίσκονται 2 ζεύγη συνελίξεων βήματος 1 και βήματος 2 για υποδειγματοληψία, των 64 χαρακτηριστικών, τα οποία ακολουθούνται από 2 ίδια ζεύγη των 128 χαρακτηριστικών. Τέλος, υπάρχουν δύο fully connected layers, με το πρώτο να έχει 512 νευρώνες, και το δεύτερο 1. Το μέγεθος του πυρήνα των συνελίξεων ήταν 5, ενώ και στα δύο δίκτυα τα βάρη αρχικοποιήθηκαν σύμφωνα με την He αρχικοποίηση [32]. Εκεί που πρέπει να δοθεί σημασία είναι ότι η έξοδός του είναι ένα γραμμικό layer που δεν περνάει από κάποια συνάρτηση ενεργοποίησης. Η τελική τιμή δηλαδή δεν εκφράζει κάποια πιθανότητα, αλλά το πόσο καλά ή πόσο "σίγουρα" μπορεί να διαχωρίσει το δίκτυο τις δύο κατανομές.

Περιορισμοί του Keras δεν επέτρεπαν στα δίκτυα να εκπαιδευτούν απευθείας, κυρίως λόγω του gradient penalty, και για το λόγο αυτό δημιουργήθηκαν δύο "μοντέλα-μοντέλων", όπου ο generator είναι διασωληνωμένος (piped) στον discriminator. Στο generator-model είναι "παγωμένα" (μη εκπαιδεύσιμα) τα βάρη του discriminator, ενώ στο discriminator-model είναι παγωμένα τα βάρη του generator. Η είσοδος του generator-model είναι ένας πίνακας θορύβου μεγέθους [batch_size, latent_size] και η έξοδός του είναι οι προβλέψεις του discriminator στις εικόνες αυτές. Το discriminator-model δέχεται ως είσοδο δύο πίνακες. Ο πρώτος πίνακας είναι διαστάσεων [batch_size,192,160,1], αντιπροσωπεύει τις αληθινές εικόνες και δίνεται κατευθείαν ως είσοδος στο discriminator μέρος, ενώ ο δεύτερος είναι διαστάσεων [batch_size, latent_size], αντιπροσωπεύει τις τεχνητές εικόνες και προωθείται στο generator μέρος του discriminator-model. Η έξοδός του είναι μία λίστα τριών πινάκων προβλέψεων: Ο πρώτος πίνακας είναι οι προβλέψεις του discriminator μέρους για τις αληθινές εικόνες, ο δεύτερος για τις τεχνητές και ο τρίτος είναι ένας άχρηστος (dummy) πίνακας που αντιπροσωπεύει την "απώλεια" του discriminator στο gradient penalty στις interpolated εικόνες.

Παράλληλα, το Keras είναι αρκετά περιοριστικό στο ποιες συναρτήσεις επιτρέπει να λειτουργούν ως συναρτήσεις απώλειας. Συγκεκριμένα, για κάθε δεδομένο απαιτείται ένα δοσμένο label, η συνάρτηση δέχεται αποκλειστικά δύο παραμέτρους, την y_true και την y_pred, ενώ πράξεις επιτρέπονται μόνο μεταξύ των μεταβλητών αυτών. Για το λόγο αυτό στις αληθινές εικόνες δίνεται το label 1 και στις τεχνητές το -1, ενώ η Wasserstein απώλεια ορίζεται ως το γινόμενο των y_true και y_pred. Ο ορισμός αυτός με μία πρώτη ματιά μοιάζει λάθος, αλλά είναι ορθός αν αναλογιστούμε ότι η Wasserstein απώλεια στα GAN προσπαθεί να ελαχιστοποιήσει την απόσταση που βλέπει ο discriminator μεταξύ των δύο κατανομών. Έστω λοιπόν ότι ο discriminator με είσοδο τεχνητή εικόνα δίνει αρνητικό αποτέλεσμα με μεγάλη απόλυτη τιμή, μπορεί δηλαδή να την ξεχωρίσει πολύ καλά από μία αληθινή. Το αποτέλεσμα τότε πολλαπλασιάζεται με το -1 και γίνεται θετικό, οπότε θα δοθεί μεγάλη ποινή και ο discriminator ενθαρρύνεται στο backwards πέρασμα από το gradient descent να δώσει έξοδο πιο κοντά στο μηδέν. Με τον ίδιο ακριβώς τρόπο ενθαρρύνεται η σύγκλιση στο μηδέν και στις αληθινές εικόνες.

Όσον αφορά το gradient penalty, ορίζεται η συνάρτηση gradient_penalty_loss που δέχεται τέσσερις παραμέτρους: τα *y_true*, τα *y_pred*, τα *averaged_samples* και το *gradient_penalty_weight*. Το *gradient_penalty_weight* είναι υπερπαράμετρος και ορίστηκε ίσο με 10, όπως προτάθηκε στο αρχικό άρθρο, ενώ το *averaged_samples* δημιουργεί σε κάθε κλήση έναν τυχαίο αριθμό από το 0 έως το 1 και πραγματοποιεί το interpolation μεταξύ αληθινών και τεχνητών εικόνων. Για την παράκαμψη του περιορισμού των δύο παραμέτρων έγινε χρήση του partial, ενός εργαλείου της Python που καθιστά δυνατή την αντικατάσταση μερικών από τις παραμέτρους μίας συνάρτησης με "οντότητες" που έχουν ήδη οριστεί και τον ορισμό νέων συναρτήσεων με τις υπόλοιπες παραμέτρους. Συγκεκριμένα, ορίστηκε η συνάρτηση partial_gp_loss ως:

partial gp loss(*y_true, y_pred)* = gradient_penalty_loss(*y_true, y_pred,*
averaged_samples = *averaged_samples,*
gradient_penalty_weight= *gradient_penalty_weight)*

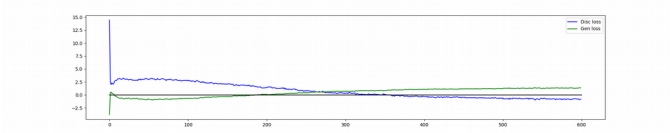
με τις δύο τελευταίες παραμέτρους να έχουν οριστεί σε προηγούμενο σημείο του κώδικα. Αρχικά υπολογίζονται οι μερικές παράγωγοι του αθροίσματος των y_pred ως προς το averaged_samples, και στη συνέχεια υπολογίζεται το l_2 μέτρο του πίνακα των παραγώγων (gradient_l2_norm). Η ποσότητα που προσπαθεί να ελαχιστοποιήσει το gradient descent είναι το gradient_penalty, που ορίζεται ως το γινόμενο του gradient_penalty_weight με το τετράγωνο της διαφοράς (1 – gradient_l2_norm).

Πριν ξεκινήσει η εκπαίδευση δημιουργούνταν ένας πίνακας θορύβου διαστάσεων [25, latent_size], ώστε να μπορεί να συγκριθεί η ποιότητα 25 τεχνητών εικόνων σε διαφορετικές εποχές με την ίδια είσοδο. Πραγματοποιείται κανονικοποίηση στο set των αληθινών εικόνων, που είναι αποθηκευμένα σε ένα μεγάλο NumPy array, ώστε το εύρος τους να είναι ίδιο με το πεδίο τιμών της υπερβολικής εφαπτομένης (-1,1), για συνοχή με την έξοδο του generator. Στην αρχή κάθε εποχής ο πίνακας ανακατεύεται τυχαία (shuffle) στον πρώτο άξονα, ώστε να μη βλέπει το δίκτυο τις ίδιες αληθινές εικόνες με την ίδια σειρά. Το discriminator-model εκπαιδεύεται 5 φορές για κάθε εκπαίδευση του generator-model, όπως προτείνεται και στο αρχικό άρθρο.

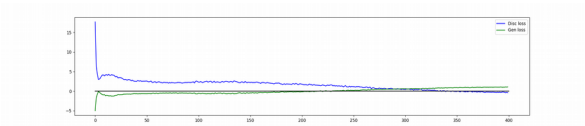
Το δίκτυο που ήταν υπεύθυνο για τις AD εικόνες εκπαιδεύτηκε πρώτο για εξακόσιες εποχές, και τα βάρη αποθηκεύονταν κάθε εκατό. Για τη δημιουργία των τελικών εικόνων ωστόσο χρησιμοποιήθηκαν τα βάρη της εποχής 400, καθώς ήταν το πιο κοντινό αποθηκευμένο σημείο στο μηδενισμό του μέσου όρου ανά εποχή της Wasserstein απώλειας του discriminator, το σημείο δηλαδή όπου σύμφωνα με τον discriminator ο generator προσεγγίζει περισσότερο την κατανομή των αληθινών δεδομένων (βλ. Σχήμα 4.2). Η συμπεριφορά του Normal δικτύου ήταν σχεδόν πανομοιότυπη, γι'αυτό και τελικά χρησιμοποιήθηκαν και σε αυτή την περίπτωση τα βάρη της εποχής 400 (βλ. Σχή-

μα 4.3). Τα δίκτυα εκπαιδεύτηκαν με batch_size = 32 και χρήση του Adam optimizer, με τις παραμέτρους που προτείνονται στο αρχικό άρθρο.

Στο σημείο αυτό πρέπει να αναφερθεί ότι δοκιμάστηκαν και άλλες αρχιτεκτονικές με ισχυρότερο discriminator οι οποίες δε συμπεριλήφθηκαν στην εργασία είτε λόγω χρονικών περιορισμών, είτε αποτυχίας στην εκπαίδευση. Συγκεκριμένα, εκπαιδεύτηκε δίκτυο υπεύθυνο για τη δημιουργία AD εικόνων για χίλιες εποχές, με την αρχιτεκτονική που φαίνεται στον Πίνακα 4.5. Βλέπουμε ότι το νέο δίκτυο έχει μία πιο λογική δομή, με τον αριθμό των χαρακτηριστικών να διπλασιάζεται μετά από κάθε υποδιπλασιασμό των διαστάσεων, σε έναν "καθρεφτισμό" της δομής του generator. Δεν ήταν δυνατή η χρήση του μοντέλου αυτού στο κομμάτι της ταξινόμησης για τον απλό λόγο ότι δε βρέθηκε αρκετός χρόνος για την εκπαίδευση του αντίστοιχου Normal δικτύου.



Σχήμα 4.2: Γραφικές παραστάσεις του Normal GAN.



Σχήμα 4.3: Γραφικές παραστάσεις του AD GAN.

Discriminator	
Layer	Αριθμός παραμέτρων
Input ((192,160,1))	0
Conv2D (32)	832
Conv2D_down(32)	25.632
Conv2D (64)	51.264
Conv2D_down (64)	102.464
Conv2D (128)	204.928
Conv2D_down (128)	409.728
Conv2D (256)	819.456
Conv2D_down (256)	1.638.656
Conv2D (512)	3.277.312
Conv2D_down (512)	6.554.112
Dense(512)	7.864.832
Dense(1)	513
	20.949.729 (Συνολικά)

Πίνακας 4.5: Δομή του ισχυρότερου discriminator.

Σε ένα επόμενο βήμα, με χρήση του νέου, βελτιωμένου discriminator, δοκιμάστηκε η αύξηση του μεγέθους του latent size, δηλαδή του διανύσματος θορύβου που ο generator δέχεται ως είσοδο, από 128 σε 512 και μετά σε 256, με τη λογική ότι ένας generator με μεγαλύτερη χωρητικότητα θα μπορέσει να αποδώσει ακόμα καλύτερα τις λεπτομέρειες των εικόνων. Τα αποτελέσματα που προέκυψαν όμως ήταν τα αντίθετα, αφού και στις δύο περιπτώσεις το δίκτυο οδηγήθηκε σε κατάρρευση. Εκ των υστέρων βέβαια η κατάρρευση είναι ίσως η μόνη λύση που βγάζει νόημα, αφού μία αύξηση της ισχύος της αρχιτεκτονικής κατά 15%-20% που είναι μονομερής απλά διευκολύνει το δυνατό μισό να επισκιάσει το άλλο.

Στη συνέχεια δοκιμάστηκε η χρήση της ELU αντί για τη LeakyReLU ως συνάρτηση ενεργοποίησης στα δύο δίκτυα, ωστόσο και στην περίπτωση αυτή η εκπαίδευση απέτυχε παταγωδώς. Αργότερα, ο λόγος βρέθηκε ότι ήταν bug στην υλοποίηση της ELU στο TensorFlow.

Ακόμη, στα αρχικά τουλάχιστον πειράματα δεν έγινε χρήση του layer normalization όπως προτείνεται στο αρχικό άρθρο διότι δεν υπήρχε έτοιμη, επίσημη υλοποίηση στο Keras. Αρχικά χρησιμοποιήθηκε υλοποίηση από το διαδίκτυο, η οποία όμως αποδείχθηκε λάθος, οπότε τελικά γράφτηκε κώδικας από το μηδέν που υλοποιεί το ζητούμενο layer. Δυστυχώς, χρονικοί περιορισμοί μας απέτρεψαν από το να συμπεριλάβουμε τα αποτελέσματα αυτά στην εργασία.

Τέλος, δοκιμάστηκε η χρήση των Auxiliary Classifier GANs (ACGAN) [77] στο MNIST, αλλά και πάλι λόγω πίεσης χρόνου δεν δοκιμάστηκε κάτι περισσότερο ουσιώδες. Τα ACGANs μαζί με το διάνυσμα θορύβου δέχονται ως είσοδο και την επιθυμητή κλάση του τεχνητού δεδομένου που θα προκύψει. Όλα τα ημιτελή πειράματα που περιγράφηκαν παραπάνω θα πραγματοποιηθούν στο εγγύς μέλλον.

Το λογικό ερώτημα που τέθηκε στη συνέχεια ήταν το ποσοστό τεχνητών/αληθινών εικόνων που θα χρησιμοποιούνταν για την εκπαίδευση των δικτύων. Για λόγους πειραματικής πληρότητας επιλέχθηκαν οκτώ ποσοστά: 25%, 50%, 75%, 100%, 125%, 150%, 175 % και 200%. Έτσι, δημιουργήθηκαν 50.000 τεχνητές εικόνες ανά κλάση, αφού τα αληθινά δεδομένα ήταν από πριν ισορροπημένα, με περίπου 25.000 εικόνες ανά κλάση. Η επιλογή του ζητούμενου κάθε φορά ποσοστού έγινε ως εξής: Με χρήση της συνάρτησης listdir της Python αποθηκεύτηκαν σε μία λίστα όλα τα ονόματα των εικόνων. Στη συνέχεια η λίστα ανακατευόταν τυχαία (shuffle), και αντιγράφονταν στους αντίστοιχους φακέλους οι εικόνες των οποίων τα ονόματα ήταν στο πρώτο 12.5%, 25% κ.λ.π. της λίστας. Η διαδικασία αυτή επαναλήφθηκε οκτώ φορές ανά κλάση, ενώ τα ονόματα των επιλεγμένων εικόνων αποθηκεύτηκαν σε αντίστοιχα .txt αρχεία.

Τέλος, υπολογίστηκαν τα στατιστικά χαρακτηριστικά (μέση τιμή και τυπική απόκλιση) των αναμεμιγμένων datasets, τα οποία παρουσιάζονται στον Πίνακα 4.6 (στην πρώτη στήλη αναφέρονται τα χαρακτηριστικά του αρχικού training set, που δεν περιέχει δηλαδή καθόλου τεχνητές εικόνες):

	0%	25%	50%	75%	100%	125%	150%	175%	200%
mean	35.3174	35.2401	35.1995	35.1210	35.1177	35.0893	35.0730	35.0500	35.0347
std	44.9201	44.8261	44.7831	44.7086	44.6856	44.6691	44.6399	44.6261	44.6097

Πίνακας 4.6: Στατιστικά χαρακτηριστικά των αναμεμιγμένων training sets.

4.3: Αρχιτεκτονικές Ταξινόμησης

Όσον αφορά το κομμάτι της ταξινόμησης, χρησιμοποιήθηκαν ResNet δίκτυα των 18 layers σύμφωνα με τον Πίνακα 2.2, καθώς συνδύαζαν μεγάλη ισχύ με σχετικά μικρό αριθμό παραμέτρων. Τα δίκτυα εκπαιδεύτηκαν εξ' αρχής στο ζητούμενο task, δε χρησιμοποιήθηκαν δηλαδή έτοιμα βάρη, ενώ παράλληλα γράφτηκε από την αρχή ο απαιτούμενος κώδικας. Τα δεδομένα προτού προωθηθούν στο δίκτυο κανονικοποιούνταν με βάση τα αντίστοιχα στατιστικά χαρακτηριστικά ώστε

να έχουν μηδενική μέση τιμή και μοναδιαία τυπική απόκλιση. Όσον αφορά την ποικιλία των αρχιτεκτονικών ταξινόμησης, τα συνελικτικά layers έμεναν ίδια παντού, ενώ δοκιμάστηκαν:

- απλό softmax layer στο τέλος.
- fully connected layers των 100 και 200 νευρώνων.
- χρήση 0%, 25% και 50% dropout μεταξύ των μη συνελικτικών layers.

Εφαρμόστηκε τυπικό data augmentation ως εξής: Δινόταν ως είσοδος μία υπερπαράμετρος που ονομάστηκε aug_prob (στην εργασία δοκιμάστηκε 25% και 50%). Αρχικά, η εικόνα καθρεφτιζόταν (flip) ως προς τον κάθετο άξονα (δεξία – αριστερά) με πιθανότητα aug_prob και στη συνέχεια, πάλι με πιθανότητα aug_prob, άλλαζε η φωτεινότητά της στο [0.9, 1.1] της αρχικής της αξίας. Τέλος, με πιθανότητα aug_prob η εικόνα περνούσε από μία σειρά μετασχηματισμών που άλλαζαν το μέγεθός της και στους δύο άξονες στο [0.9, 1.1] του αρχικού της μεγέθους, τη μετακινούσαν κατά [-5%, +5%] στους δύο άξονες και την περίστρεφαν κατά [-5,5] μοίρες.

Το flipping της εικόνας είναι επιτρεπτό αφού οι συνέπειες της νόσου του Alzheimer εμφανίζονται και στις δύο πλευρές του εγκεφάλου, δε θα επηρεαζόταν δηλαδή αρνητικά η επίδοση του δικτύου λόγω εκπαίδευσης σε "λάθος" ή "αφύσικα" δεδομένα. Η αλλαγή της φωτεινότητας χρησιμοποιήθηκε ώστε να εξαλειφθούν τυχόν διαφορές στις εικόνες που οφείλονται στη χρήση διαφορετικών ιατρικών μηχανημάτων για τη λήψη των MRI δεδομένων. Τέλος, σκοπός των τελικών μετασχηματισμών είναι η καταπολέμηση τυχόν λαθών κατά τη λήψη των δεδομένων, όπως π.χ. κακοκεντραρισμένες εικόνες ή ασθενείς που δεν κοιτούσαν ακριβώς ευθεία.

Εύκολα μπορεί να παρατηρήσει κανείς ότι τα παραπάνω όρια είναι αρκετά συντηρητικά, κάτι που δικαιολογείται από τη φύση της εφαρμογής, αφού αν ο ασθενής βρισκόταν υπό γωνία μεγαλύτερη των 5 περίπου μοιρών ή βρισκόταν σε απόσταση μεγαλύτερη των μερικών εκατοστών από το κέντρο, κατά πάσα πιθανότητα ο χειριστής του μηχανήματος θα επαναλάμβανε τη διαδικασία. Οι κλασικές τεχνικές εμπλουτισμού δεδομένων λοιπόν δεν μπορούν να προσφέρουν πολλά, γι' αυτό και στην παρούσα εργασία αναζητήθηκαν άλλες, πιο προχωρημένες εναλλακτικές που παρακάμπτουν κάποιους από τους παραπάνω περιορισμούς.

Τα δίκτυα εκπαιδεύονταν για 100 εποχές με χρήση του Adam optimizer, με τις προτεινόμενες παραμέτρους, ενώ τα βάρη της τελευταίας εποχής αποθηκεύονταν για την περίπτωση που η εκπαίδευση θα συνεχιζόταν. Κάποιες αρχιτεκτονικές εκπαιδεύτηκαν για άλλες 100 εποχές, καθώς εκ των υστέρων οι γραφικές παραστάσεις τους έδειχναν ότι υπήρχαν ακόμα περιθώρια ουσιαστικής μάθησης. Οι αρχιτεκτονικές αυτές χρησιμοποιούσαν dropout με ποσοστό απενεργοποίησης 0.25, κλασικό augmentation (η διαδικασία περιγράφεται στην επόμενη παράγραφο) με πιθανότητα 0.25 και τέλος εκπαιδεύονταν με τεχνητά δεδομένα σε ποσοστά 75% έως 150%.

Καθ' όλη τη διάρκεια της εκπαίδευσης αποθηκεύονταν τα βάρη που έδιναν καλύτερα αποτελέσματα ως προς κάποια μετρική του validation set. Ως μετρική επιλογής αρχικά είχε οριστεί η validation loss, αποθηκευόταν δηλαδή το μοντέλο με τη χαμηλότερη τιμή. Ωστόσο, κατά πάσα πιθανότητα κάποιο bug στο Keras ή στο TensorFlow την οδηγούσε στο να αυξάνεται, και μάλιστα κατά τη διάρκεια της εκπαίδευσης μοντέλων που, σύμφωνα με την αύξηση του validation accuracy, εκπαιδεύονταν κανονικά και συνέκλιναν πέρα από κάθε αμφιβολία. Η αύξηση του validation loss δυστυχώς είχε ως αποτέλεσμα τα βάρη που αποθηκεύονταν να είναι ουσιαστικά άχρηστα, αφού ήταν αυτά του τέλους της πρώτης εποχής.

Σκοπός ήταν η αξιολόγηση των καλύτερων "εκδοχών" των μοντέλων στο test set, ένα ακόμα άγνωστο σύνολο ατόμων. Εκτός από την ακρίβεια (true positives + true negatives) θα μετρούνταν και το precision (true positives / all positives), το recall (true posotives / (true positives + false negatives)) και το F1-measure (2*precision*recall/precision + recall). Όταν ανακαλύφθηκε αυτό το λάθος είχαν ήδη προσομοιωθεί σχεδόν 60 διαφορετικές αρχιτεκτονικές. Ως λύση της τελευταίας στιγμής εκτελέστηκαν κάποια πειράματα που θεωρήθηκαν αντιπροσωπευτικά και υποσχόμενα (μεταξύ των δικτύων αυτών ήταν τα μοντέλα που εκπαιδεύτηκαν για άλλες 100 φορές) και εν

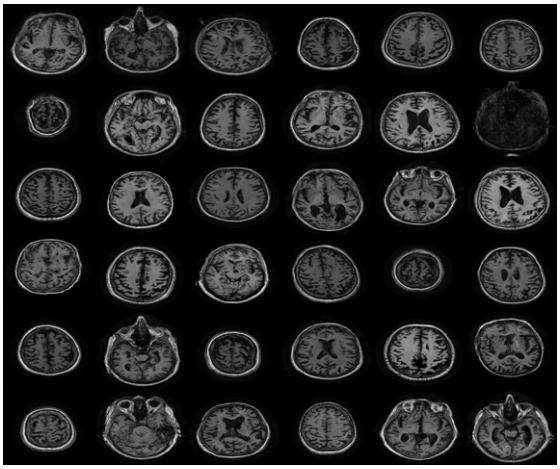
τέλει λήφθηκαν μετρήσεις στο test set, ωστόσο αυστηροί χρονικοί περιορισμοί εμπόδισαν την πληρότητα αυτού του κομματιού του πειραματικού μέρους.

Κεφάλαιο 5: Πειραματικά Αποτελέσματα

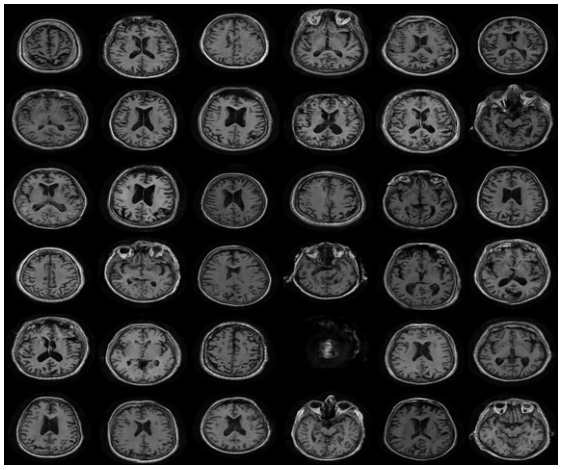
Όλος ο κώδικας που γράφτηκε στα πλαίσια της εργασίας θα ανέβει στο μέλλον στη διεύθυνση https://github.com/filippos1994/diploma_thesis. Παρακάτω ακολουθούν τα πειραματικά αποτελέσματα στις δύο κύριες περιοχές της εργασίας: στη δημιουργία των τεχνητών εικόνων και στη χρήση τους στην ταξινόμηση.

5.1: Δημιουργία Τεχνητών Εικόνων

5.1.1: Απλός discriminator, latent size = 128 (Επιτυχία, Εκπαίδευση ResNet)



Σχήμα 5.1: Ενδεικτικές τεχνητές εικόνες του AD GAN.

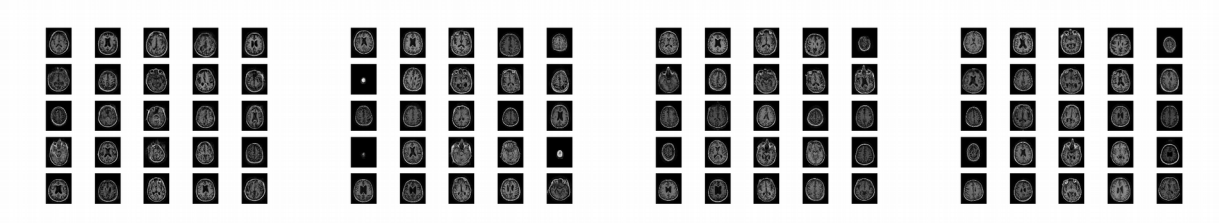


Σχήμα 5.2: Ενδεικτικές τεχνητές εικόνες του Normal GAN.

Στα Σχήματα 5.1 και 5.2 φαίνονται μερικές τυχαία επιλεγμένες εικόνες που παράχθηκαν από τα δύο GAN και χρησιμοποιήθηκαν στην εκπαίδευση των ResNet. Στο σημείο αυτό μπορούμε με σιγουριά να πούμε ότι η εκπαίδευση των GAN έχει πετύχει: Εύκολα βλέπουμε ότι στο σημείο αυτό τα δύο δίκτυα έχουν συλλάβει πολύ καλά τα χαρακτηριστικά των αληθινών κατανομών: Στο επίπεδο των γενικότερων, αφηρημένων χαρακτηριστικών σχεδόν όλες οι εικόνες που επιλέχθηκαν όχι μόνο δεν έχουν κάποιο προφανές, κραυγαλέο σημείο που να προδίδει την "τεχνητότητά" τους, όπως π.χ. τμήμα του κόκαλου ξαφνικά να εξαφανίζεται ή σιγά σιγά να θολώνει, αλλά έχουν τα σωστά χρώματα στα σωστά σημεία κ.λ.π. Παράλληλα, έχουν αποδοθεί καλά και οι μικρές λεπτομέρειες, π.χ. μύτη, μάτια.

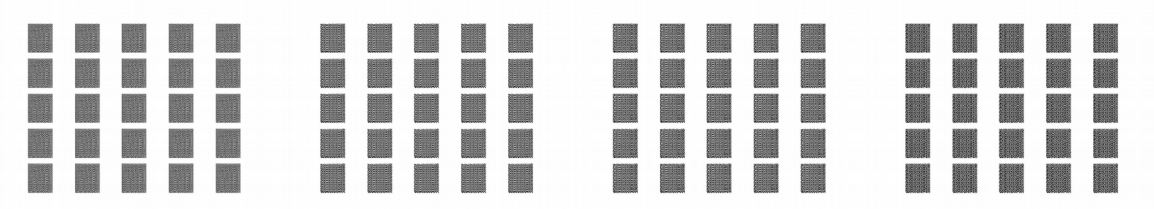
Παρ' όλα αυτά, το πιο σημαντικό και ελπιδοφόρο από άποψη ταξινόμησης αποτέλεσμα είναι το γεγονός ότι οι εικόνες που δημιουργούνται έχουν υιοθετήσει τα χαρακτηριστικά που διαχωρίζουν τη μία κλάση από την άλλη (έντονος εκφυλισμός του κέντρου του εγκεφάλου στις μεσαίες τομές, εκφυλισμός των πλάγιων περιοχών στο ύψος των ματιών για τους ασθενείς, έλλειψη των χαρακτηριστικών αυτών για τους υγιείς), και μάλιστα χωρίς να έχει η μία κατανομή επιπλέον πληροφορίες για την ύπαρξη της άλλης, όπως θα συνέβαινε π.χ. σε ένα ACGAN. Ο discriminator δηλαδή, παρά τον όχι και τόσο καλό σχεδιασμό του, "κατάλαβε" τη σημασία τους και έδωσε σημαντικό ποσοστό από την περιορισμένη του συνελικτική χωρητικότητα για τον εντοπισμό των χαρακτηριστικών αυτών ως απαραίτητα στοιχεία της κατανομής, αντί να το σπαταλήσει π.χ. για πιο ρεαλιστικές μύτες.

5.1.2: Βελτιωμένος discriminator, latent size = 128 (Επιτυχία)



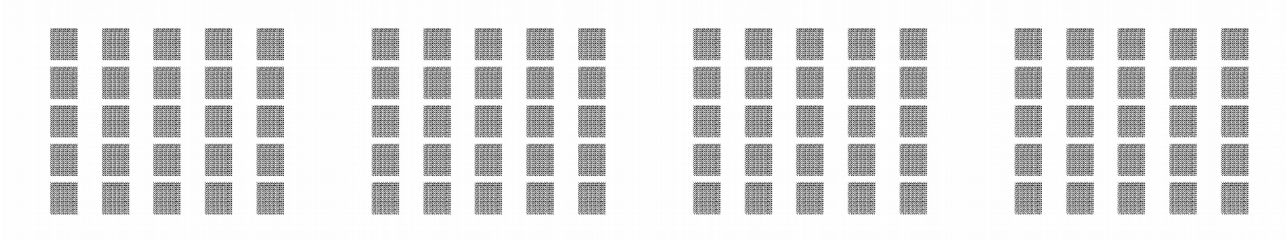
Σχήμα 5.3: Τεχνητές εικόνες με την ίδια είσοδο υπό τον βελτιωμένο discriminator στις εποχές 10, 50, 150 και 350.

5.1.3: Απλός discriminator, latent size = 128, ELU (Κατάρρευση)



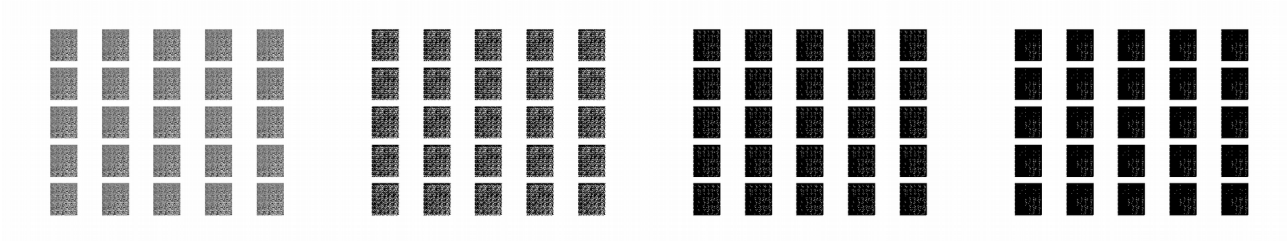
Σχήμα 5.4: Κατάρρευση στις εποχές 10, 50, 100 και 500.

5.1.4: Βελτιωμένος discriminator, latent size = 128, ELU (Κατάρρευση)



Σχήμα 5.5: Κατάρρευση στις εποχές 10, 50, 100 και 200.

5.1.5: Βελτιωμένος discriminator, latent size = 512, ELU (Κατάρρευση)



Σχήμα 5.6: Κατάρρευση στις εποχές 10, 100, 300 και 500.

5.1.6: Βελτιωμένος discriminator, latent size = 256, ELU, Layer Norm (Κατάρρευση)

Σχήμα 5.7: Κατάρρευση στις εποχές 10, 50, 100 και 150.

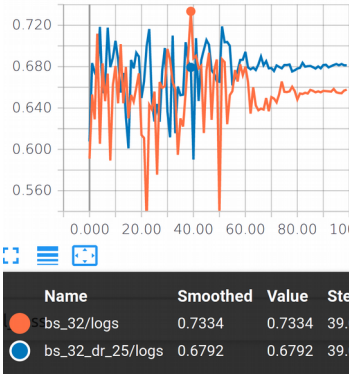
5.2: Ταξινόμηση

Η ενότητα αυτή αποτελείται από δύο τμήματα: Στο πρώτο παρουσιάζονται κάποιες ενδεικτικές γραφικές παραστάσεις του validation accuracy κάποιων αρχιτεκτονικών μέσα από το πρόγραμμα οπτικοποίησης TensorBoard, ενώ στο δεύτερο παρουσιάζονται κάποια διαγράμματα της επίδοσης διάφορων αρχιτεκτονικών στο test set.

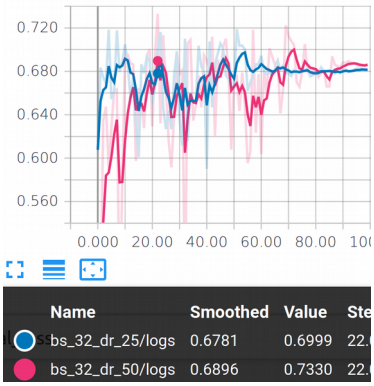
5.2.1: Γραφικές Παραστάσεις από το TensorBoard

Αρχικά εξετάζονται αρχιτεκτονικές που εκπαιδεύονται μόνο με αληθινά δεδομένα, αρχικά χωρίς, και στη συνέχεια με την εφαρμογή κλασικού εμπλουτισμού. Συγκεκριμένα, συγκρίνονται ως προς την ύπαρξη ποσοστού dropout, καθώς και την ύπαρξη (και ατίστοιχα τον αριθμό) ή όχι fully connected layer μετά το τελευταίο συνελικτικό. Στη συνέχεια, γίνονται συγκρίσεις ανάμεσα στις αρχιτεκτονικές αυτές και πανομοιότυπες, όπου εφαρμόζεται κλασικός εμπλουτισμός. Τέλος, η καλύτερη "κλασική" αρχιτεκτονική θα συγκριθεί με τις GAN αρχιτεκτονικές. Λόγω της φύσης του εργαλείου οπτικοποίησης, κάθε φορά συγκρίνονται δύο αρχιτεκτονικές, ενώ από το σημείο αυτό για λόγους συντομίας ως GAN αρχιτεκτονική θα θεωρούμε απλά ένα ResNet-18 που εκπαιδεύτηκε σε "μεικτά" δεδομένα.

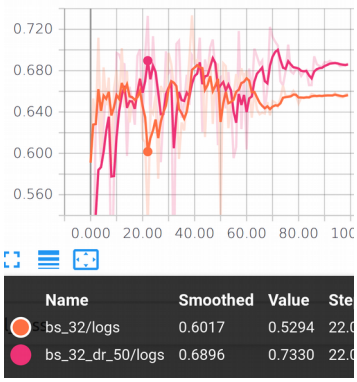
5.2.1.1.: Dropout χωρίς χρήση εμπλουτισμού



Σχήμα 5.8: Dropout 0% και 25%



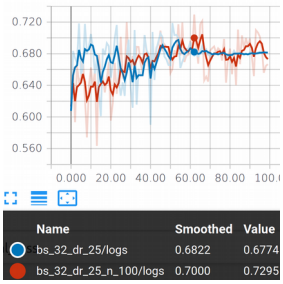
Σχήμα 5.9: Dropout 25% και 50%



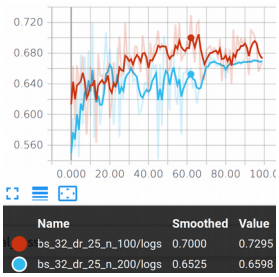
Σχήμα 5.10: Dropout 0% και 50%

Στα τρία παραπάνω Σχήματα εξετάζεται η επίδραση του dropout. Στο πρώτο δεν έχει εφαρμοστεί εξομάλυνση, ώστε να φαίνεται καλύτερα η επίδοση της αρχιτεκτονικής που δεν το χρησιμοποιεί, καθώς είναι η καλύτερη αρχιτεκτονική (validation accuracy 73.34%), οπότε με αυτή θα συγκριθούν παρακάτω τα GAN. Γενικά, παρατηρούμε ότι η χρήση 25% dropout, αν και δεν αυξάνει την κορυφαία επίδοση του δικτύου, οδηγεί σε κατά μέσο όρο καλύτερη επίδοση (Σχ. 5.8). Αντίθετα, η χρήση 50% dropout έχει ως αποτέλεσμα το δίκτυο να δυσκολεύεται να σταθεροποιηθεί, παρ' όλο που στην εποχή 22 η επίδοσή του είναι στιγμιαία καλύτερη (Σχ. 5.9). Σε κάθε περίπτωση τα δίκτυα έχουν σταθεροποιηθεί μέχρι την εποχή 80, με αυτά που χρησιμοποιούν dropout να είναι κατά 2-3% καλύτερα.

5.2.1.2: Αριθμός νευρώνων χωρίς χρήση εμπλουτισμού



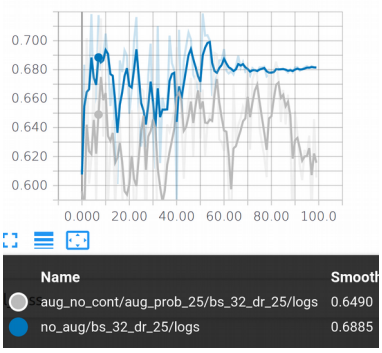
Σχήμα 5.11: Μηδέν και εκατό νευρώνες.



Σχήμα 5.12: Εκατό και διακόσιοι νευρώνες

Στα παραπάνω Σχήματα φαίνεται η επίδραση της προσθήκης fully connected layer στο τέλος της αρχιτεκτονικής. Στο πρώτο Σχήμα βλέπουμε ότι στην αρχή το δίκτυο των εκατό νευρώνων δυσκολεύεται να εκπαιδευτεί, λόγω των επιπλέον παραμέτρων, ενώ στο δεύτερο φαίνεται εμφανώς ότι η προσθήκη διακοσίων νευρώνων είναι μία κακή επιλογή. Συμπεραίνουμε λοιπόν ότι η προσθήκη επιπλέον νευρώνων δεν φαίνεται να οδηγεί σε κάποια αισθητή βελτίωση της επίδοσης, γι'αυτό και στα υπόλοιπα πειράματα λαμβάνονται υπόψη μόνο αρχιτεκτονικές που συνδέουν τα συνελικτικά τους χαρακτηριστικά απ'ευθείας με ένα softmax layer.

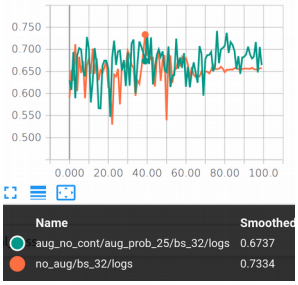
5.2.1.3: Χρήση κλασικού εμπλουτισμού



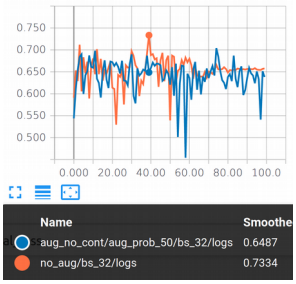
Σχήμα 5.13: 25% dropout, 0% και 25% πιθανότητα εμπλουτισμού.

Σχήμα 5.14: 25% dropout, 0% και 50% πιθανότητα εμπλουτισμού.

Στα παραπάνω Σχήματα φαίνεται η επίδραση της χρήσης κλασικών τεχνικών εμπλουτισμού. Το αξιοσημείωτο είναι ότι και στις δύο περιπτώσεις η χρήση τους όχι μόνο δε δείχνει να βοηθά το δίκτυο, αλλά αντίθετα το καθιστά πολύ πιο ευάλωτο σε διακυμάνσεις της επίδοσής του. Παρόμοια συμπεριφορά παρατηρείται και για το μοντέλο που δε χρησιμοποιεί dropout.



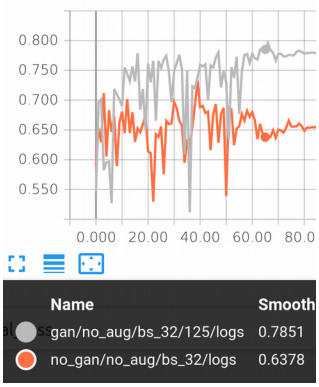
Σχήμα 5.15: 0% dropout, 0% και 25% πιθανότητα εμπλουτισμού.



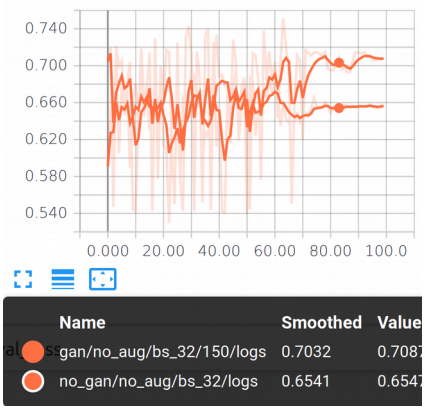
Σχήμα 5.16: 0% dropout, 0% και 50% πιθανότητα εμπλουτισμού.

5.2.1.4: Χρήση GAN χωρίς εμπλουτισμό

Η επίδοση των GAN αρχιτεκτονικών ήταν παρόμοια με των κλασικών για μικρά ποσοστά (25%, 50%) τεχνητών εικόνων, καθώς όμως το ποσοστό μεγάλωνε υπήρχε σημαντική βελτίωση (75%, 100%, 125%), με την καλύτερη επίδοση να είναι στο 125%, και στη συνέχεια η επίδοση πάλι επέφτε, με τη χειρότερη αρχιτεκτονικη να είναι στο 150%.



Σχήμα 5.17: 125% GAN (καλύτερο) 0% εμπλουτισμός και η καλύτερη κλασική



Σχήμα 5.18: 150% GAN (χειρότερο) 0% εμπλουτισμός και η καλύτερη κλασική.

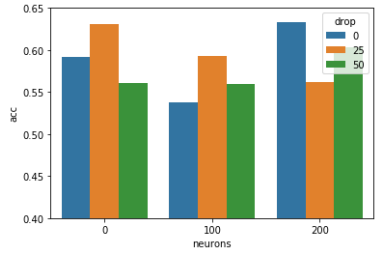
Η υπεροχή των GAN αρχιτεκτονικών είναι εμφανής, αφού στην καλύτερη περίπτωση υπάρχει διαφορά στην καλύτερη ακρίβεια της τάξης του 5%, και αισθητά καλύτερη γενική συμπεριφορά κατά τη διάρκεια της εκπαίδευσης, ενώ στη χειρότερη περίπτωση η καλύτερη ακρίβεια της 150% GAN είναι 75.15%, ενώ εμφανίζει πάλι πολύ καλύτερη πειραματική συμπεριφορά.

5.2.1.5: Χρήση GAN με dropout χωρίς εμπλουτισμό

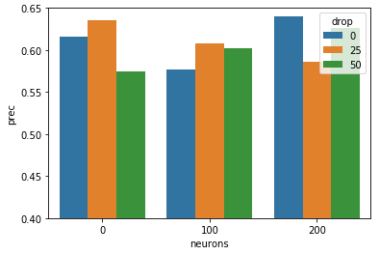
Σχήμα 5.19: Συνδυασμός GAN με εμπλουτισμό

Στο παραπάνω σχήμα φαίνεται ότι ο συνδυασμός των δύο τεχνικών, μαζί με χρήση dropout οδηγεί σε ακόμα μεγαλύτερη βελτίωση, με την καλύτερη GAN αρχιτεκτονική (100%) να έχει 10% καλύτερη ακρίβεια από την αντίστοιχη κλασική.

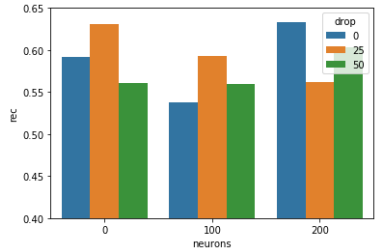
5.2.2: Μετρικές στο test set



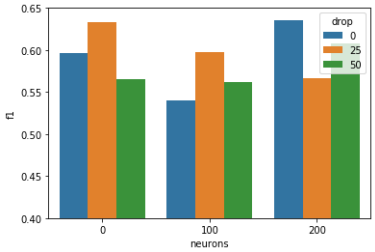
Σχήμα 5.20: Test set accuracy χωρίς χρήση εμπλουτισμού.



Σχήμα 5.21: Test set precision χωρίς χρήση εμπλουτισμού.

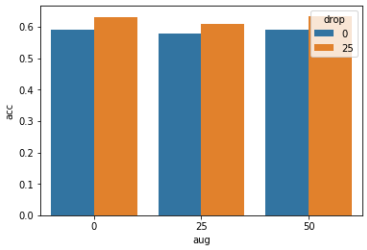


Σχήμα 5.22: Test set recall χωρίς χρήση εμπλουτισμού.



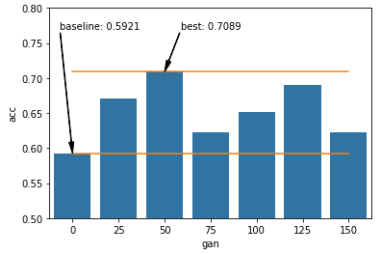
Σχήμα 5.23: Test set f1-score χωρίς χρήση εμπλουτισμού.

Στα τέσσερα παραπάνω σχήματα φαίνονται οι επιδόσεις στο test set των αρχιτεκτονικών που εκπαιδεύτηκαν από τις αληθινές μόνο εικόνες, χωρίς τη χρήση κλασικού εμπλουτισμού. Λόγω της ισορροπίας μεταξύ των κλάσεων οι μετρικές είναι σχεδόν πανομοιότυπες, γι'αυτό και δε χρησιμοποιήθηκαν πουθενά αλλού στην εργασία, με τις καλύτερες αρχιτεκτονικές να είναι αυτή που έχει ένα 25% dropout layer μεταξύ του τελευταίου συνελικτικού και του softmax, καθώς και αυτή που διαθέτει ένα fully connected layer των διακοσίων νευρώνων, χωρίς τη χρήση dropout.

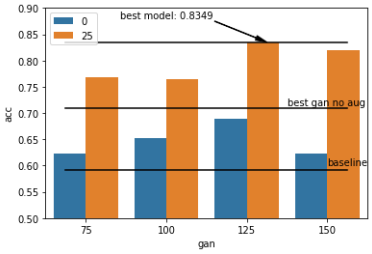


Σχήμα 5.25: Test set accuracy ως προς την πιθανότητα εμπλουτισμου.

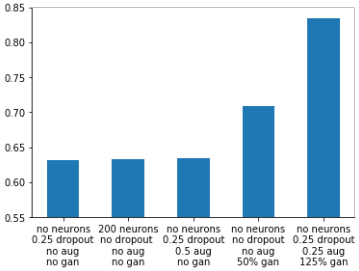
Στο παραπάνω σχήμα φαίνεται η ακρίβεια των δύο πιο συχνά χρησιμοποιημένων αρχιτεκτονικών ως προς την πιθανότητα εμπλουτισμού. Αξίζει να παρατηρηθεί ότι στην περίπτωση της 25% πιθανότητας, όχι μόνο δεν υπάρχει αύξηση, αλλά αντίθετα υπάρχει μείωση της απόδοσης των αρχιτεκτονικών.



Σχήμα 5.26: Σύγκριση της καλύτερης "κλασικής" αρχιτεκτονικής με τις GAN.



Σχήμα 5.27: Test set accuracy των GAN αρχιτεκτονικών με και χωρίς κλασικό εμπλουτισμό.



Σχήμα 5.28: Test set accuracy των καλυτερων αρχιτεκτονικων.

Στο Σχήμα 5.26 φαίνεται ξεκάθαρα η υπεροχή των GAN αρχιτεκτονικών, αφού ακόμα και αυτή με τη χειρότερη επίδοση είναι καλύτερη από την “κλασική” κατά 2 με 3%, ενώ στο Σχήμα 5.27 φαίνεται η επίδραση του κλασικού εμπλουτισμού στις GAN αρχιτεκτονικές. Συγκεκριμένα, ενώ στις κλασικές αρχιτεκτονικές η επίδρασή του ήταν στην καλύτερη περίπτωση μηδαμινή και στη χειρότερη επιβλαβής, βλέπουμε ότι οι GAN αρχιτεκτονικές κυριολεκτικά απογειώνονται, με την μικρότερη αύξηση να είναι της τάξεως του 10%, και πιο σημαντικά, τη μεγαλύτερη, στο 150% να πλησιάζει το 20%, κάτι που πιθανώς οφείλεται στην ανάγκη της ποσότητας των δεδομένων να περάσει ένα όριο, ώστε να μπορέσει να φανεί χρήσιμη. Τέλος, στο Σχήμα 5.28 συγκρίνονται οι καλύτερες αρχιτεκτονικές στις τέσσερις γενικές κατηγορίες εμπλουτισμού (όχι κλασικός και όχι GAN, όπου υπάρχει ισοπαλία, κλασικός και όχι GAN, όχι κλασικός και GAN, κλασικός και GAN), όπου διαπιστώνεται και εδώ πέρα από κάθε αμφιβολία η υπεροχή του συνδυασμού των δύο τεχνικών.

Βιβλιογραφία

[1] Krizhensky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, pp. 1097-1105. arXiv preprint arxiv:1511.07571.

[2] Johnson, J., Karpathy, A., Fei-Fei, L. (2015). DenseCap: Fully Convolutional Localization Networks for Dense Captioning.

[3] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis , D. (2016). Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), pp. 484-489.

[4] OpenAI. (2017). https://blog.openai.com/dota-2/

[5] Rashidi, M., Wolkow, R. A. (2018). Autonomous Scanning Probe Microscopy in Situ Tip Conditioning through Machine Learning. ACS Nano, 12(6), pp. 5185-5189.

[6] Turing, A. M. (1950). Computing Machinery and Intelligence. Mind.

[7] Rosenblatt, F. (1957). The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

[8] Papert, S., Minsky, M. L. (1988). Perceptrons: an introduction to computational geometry. Cambridge, Massachusetts: MIT Press.

[9] Cook, S. A. (1971).The Complexity of Theorem-Proving Procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing.

[10] Karp, R. M. (1972). Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum, pp. 85–103.

[11] Lucas, J. (1961). Minds, Machines and Gödel. Philosophy, vol. 36, pp. 112-127.

[12] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning internal representations by error-propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1 (6088), pp. 318-362.

[13] Cortes, C., Vapnik, V. N. (1995). Support-vector networks. Machine Learning 20(3), pp. 273–297.

[14] Ho, T. K. (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995, pp. 278–282.

[15] Hochreiter, S., Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation 9(8), pp. 1735-1780.

[16] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278-2324.

[17] Mitchell, T. (1997). Machine Learning, McGraw Hill, p.2.

[18] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. Neural computation, vol 14 (8), pp. 1771-1800.

[19] Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. arXiv preprint 1406.2572.

[20] Ruder, S. (2016). http://ruder.io/optimizing-gradient-descent/

[21] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. Neural Networks : The Official Journal of the International Neural Network Society, 12(1), pp. 145–151.

[22] Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543– 547.

[23] Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, pp. 2121–2159.

[24] Hinton, G. E. (2013). https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[25] Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701.

[26] Kingma, D. P., Ba, J. L. (2014). Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[27] Nair, V., Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th ICML, pp. 807-814.

[28] Hubel, D. H., Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology, 160, pp. 106-154.

[29] Hubel, D. H., Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology, 195, pp. 215-243.

[30] Karpathy, A., Fei-Fei, L. (2015). CS231n Convolutional Neural Networks for Visual Recognition, Stanford. http://cs231n.github.io/convolutional-networks/

[31] Maas, A. L., Hannun, A. Y., Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the ICML, 30(1), pp. 3-12.

[32] He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.Proceedings of the International Conference on Computer Vision (ICCV).

[33] Clevert, D.-A., Unterthiner, T., Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289.

[34] Springenberg, J. T., Dosovitskiy, A., Brox, T., Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

[35] Glorot, X., Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th ICAIS, pp. 249-256.

[36] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, vol 15 (1) pp. 1929-1958.

[37] Simonyan, K., Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv preprint arXiv:1409.1556.

[38] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. Proceedings of the IEEE Conference on Pattern Recognition and Computer Vision (CVPR), pp. 1-9.

[39] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv preprint arXiv:1602.07261.

[40] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[41] Ioffe, S., Szegedy, C. (2015). Batch Normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[42] Zagoruyko, S., Komodakis, N. (2016). Wide Residual Networks. arXiv preprint arXiv:1605.07146.

[43] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K. (2017).Aggregated residual transformations for deep neural networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[44] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. IEEE Conference on Pattern Recognition and Computer Vision (CVPR), pp. 4700-4708.

[45] Hinton, G. E., Sejnowski, T. J. (1983). Analyzing Cooperative Computation. Proceedings of the 5th Annual Congress of the Cognitive Science Society.

[46] Hinton, G. E., Sejnowski, T. J. (1983). Optimal Perceptual Inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 448–453, IEEE Computer Society.

[47] Hopfield, J. J. (1982). Neural Networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA, 79(8), pp. 2554–2558.

[48] Smolensky, P. (1986). “Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory”. In Rumelhart, D. E., McLelland, J. L. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations. MIT Press. pp. 194–281.

[49] Hinton, G. E. (2009). Deep belief networks. Scholarpedia 4(5), pp. 5947.

[50] Hinton, G. E., Ostindero, S., Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), pp. 1527-1554.

[51] Hinton, G. E., Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science 313 (5786) pp. 504-507.

[52] Ballard, D. (1987). Modular Learning in Neural Networks. AAAI.

[53] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks. Advances in neural information processing systems, pp. 153-160.

[54] LeCun, Y. (1987). Modeles connexionnistes de l'apprentissage. Université Pierre et Marie Curie.

[55] Vincent, P., Larochelle, H., Lajoie, I., Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. 25th ICML, pp 1096-1103.

[56] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11(12), pp. 3371-3408.

[57] Ng, A.Y. (2011). Sparse autoencoder. CS294A Lecture notes.

[58] Kingma, D. P., Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[59] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680.

[60] Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H. (2016). Generative Adversarial Text to Image Synthesis. arXiv preprint arXiv: 1605.05396.

[61] Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K., Zhavoronkov, A. (2017). The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology, Oncotarget vol. 8 pp. 10883-10890.

[62] Isola, P., Zhu, J.-Y., Zhou, T., Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. CVPR.

[63] Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. Proceedings of the International Conference on Medical image computing and computer-assisted intervention, pp. 234-241.

[64] Zhu, J.-Y., Park, T.,Isola, P., Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV.

[65] Antipov, G., Baccouche, M., Dugelay, J.-L. (2017). Face ageing with conditional generative adversarial networks. arXiv preprint arXiv: 1702.01983.

[66] Creswell, A., Bharath, A. A. (2016). Creatism: A deep-learning photographer capable of creating professional work. ArXiv preprint arXiv: 1607.02748.

[67] Zhu, J.-Y., Krähenbühl, P., Shechtman, E., Efros, A. A. (2016). Generative Visual Manipulation on the Natural Image Manifold. ECCV.

[68] Radford, A., Metz, L., Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434

[69] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. (2016). Improved techniques for Training GANs. Advances in Neural Information Processing Systems, pp. 2234-2242

[70] Arjovsky, M., Bottou, L. (2017). Towards principled methods for training generative adversarial networks. ICLR.

[71] Villani, C. (2009). Optimal Transport: Old and New. Grundlehren der mathematischen Wissenschaften. Springer, Berlin.

[72] Arjovsky, M., Chintala, S., Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875

[73] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A. (2017). Improved Training of Wasserstein Gans. arXiv preprint arXiv:1704.00028.

[74] Ba, J. L., Kiros, J. R., Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

[75] Petersen, R. C., Aisen, P. S., Beckett, L. A., Donohue, M. C., Gamst, A. C., Harvey, D. J., Jack, Jr, C. R., Jagust, W. J, Shaw, L. M., Toga, A. W., Trojanowski, J. Q., Weiner, M. W.(2010). Alzheimer's Disease Neuroimaging Initiative (ADNI). Neurology, 74(3), pp. 201-209.

[76] LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient BackProp. Neural Networks: Tricks of the Trade, 1524, pp. 9–50.

[77] Odena, A., Olah, C., Shlens, J. Conditional Image Synthesis With Auxiliary Classifier GANs. (2016). arXiv preprint arXiv:1610.09585.