Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Automated Audio Captioning

and generation of captions for sound events in movies.

Διπλωματικη Εργασια

του

ΚΟΥΖΕΛΗ ΘΟΔΩΡΗ

Επιβλέποντες: Αλέξανδρος Ποταμιάνος
Αναπληρωτής Καθηγητής

Αθανάσιος Κατσαμάνης
Κύριος Ερευνητής Ι.Ε.Λ, Ε.Κ. ΑΘΗΝΑ

Αθήνα, Ιούλιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Automated Audio Captioning

and generation of captions for sound events in movies.

Διπλωματικη Εργασια

του

ΚΟΥΖΕΛΗ ΘΟΔΩΡΗ

Επιβλέποντες: Αλέξανδρος Ποταμιάνος, Αναπληρωτής Καθηγητής
Αθανάσιος Κατσαμάνης, Κύριος Ερευνητής Ι.Ε.Λ, Ε.Κ. ΑΘΗΝΑ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουίου 2022.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Αλέξανδρος Ποταμιάνος	Στέφανος Κόλλιας	Κωνσταντίνος Τζαφέστας
Αναπληρωτής Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Αθήνα, Ιούλιος 2022

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Θοδωρής Κουζέλης

Αθήνα, Ιούλιος 2022

Περίληψη

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη της Αυτόματης Περιγραφής Ήχου, δηλαδή της περιγραφή ηχητικών συμβάντων με χρήση φυσικής γλώσσας. Η Αυτόματης Περιγραφής Ήχου είναι ένα πολύτροπικο πρόβλημα μετάφρασης που συνδυάζει την Επεξεργασία Ηχητικού Σήματος και την Επεξεργασία Φυσικής Γλώσσας. Η Περιγραφή Ήχου αναγνωρίζει τα ηχητικά συμβάντα και τις χωροχρονικές τους σχέσεις σε ένα ηχητικό σήμα και τα περιγράφει σε φυσική γλώσσα. Είναι ένα πρόσφατο και αρκετά ανεξερεύνητο πρόβλημα, το οποίο έχει μεγάλες δυνατότητες για πρακτικές εφαρμογές.

Σε αυτή την εργασία, μοντελοποιούμε την παραγωγή περιγραφών για ένα δοθέν ηχητικό σήμα σαν ένα ακολουθιακό πρόβλημα, χρησιμοποιώντας αρχιτεκτονικές με μηχανισμούς προσοχής. Δείχνουμε ότι πρόσφατες στρατηγικές από τον συγγενικό ερευνητικό πεδίο της ταξινόμησης ήχων, μας επιτρέπουν να μειώσουμε δραματικά την πολυπλοκότητα του μοντέλου μας χωρίς να επηρεάζουν την αποδοτικότητα του. Σκοπεύοντας να παράγουμε εύγλωττες και ποικιλόμορφες περιγραφές ερευνούμε αλγόριθμους αποκωδικοποίησης που ελαχιστοποιούν το αντιστάθμισμα μεταξύ της σημασιολογικής ακρίβειας και της ποικιλομορφίας των περιγραφών.

Προτείνουμε μια πραγματική εφαρμογή για την Αυτόματη Περιγραφή Ήχου. Ένα νέο πρόβλημα, όπου δοθέντος του ηχητικού σήματος μιας ταινίας, το σύστημά έχει σκοπό να παράγει περιγραφές για βασικά ηχητικά συμβάντα. Ουσιαστικά, το πρόβλημα που προτείνουμε είναι η αυτόματη παραγωγή υποτίτλων για κωφά και βαρήκοα άτομα. Το σύστημά που προτείνουμε ανιχνεύει τα τμήματα των ηχητικών συμβάντων χρησιμοποιώντας ένα προ-εκπαιδευμένο μοντέλο ταξινόμησης ήχων και παράγει περιγραφές χρησιμοποιώντας το μοντέλο Αυτόματης Περιγραφής Ήχου που έχουμε εκπαιδεύσει. Για να βελτιώσουμε την απόδοσή του μοντέλου μας δημιουργούμε ένα εξειδικευμένο σύνολο δεδομένων από ταινίες και υπότιτλους για κωφά και βαρήκοα άτομα. Επιπλέον, ενσωματώνουμε την κειμενική πληροφορία του μοντέλου ταξινόμησης στην παραγωγή των περιγραφών, δημιουργώντας ένα μοντέλο για περιγραφή ήχων οδηγούμενη από κείμενο. Τέλος προτείνουμε μια μετρική για την αξιολόγηση των αποτελεσμάτων μας.

Λέξεις Κλειδιά

Αυτόματης Περιγραφής Ήχου, βαθιά μάθηση, επεξεργασία φυσικής γλώσσας, μηχανισμοί προσοχής, ταινίες.

Abstract

The purpose of this dissertation is to study Automated Audio Captioning. The aim of this task is to describe the content of an audio clip using natural language. It is a cross-modal translation task at the intersection of audio signal processing and natural language processing. Audio Captioning focuses on the audio events and their spaciotemporal relationships in an audio clip and expresses them in natural language. It is a recent and rather unexplored task, that has a great potential for practical applications.

In this work, we model caption generation for a given audio clip as a sequence-to-sequence task, using a Transformer architecture. We show that using recent strategies from the related field of Audio Tagging, allows us to significantly reduce the complexity of our model without affecting performance. In order to generate rich and varied descriptions we investigate decoding algorithms that minimize the trade-off between, semantic fidelity and diversity in captions.

As a real world application of Automated Audio Captioning, we propose a novel task, where given the audio of movie a system aims to generate captions of salient sound events. Essentially, the aim of our proposed task is to automatically generate Subtitles for Deaf and Hard of Hearing (SDH). Our proposed system detects the segments of sound events using a pre-trained tagging model and generates a textual description using our model for Audio Captioning. To improve the performance of our audio captioning model we create a task specific dataset using SDH subtitles and movies. Furthermore, we integrate the textual information of the tagging model into caption generation by building a model for text guided audio captioning. Finally, we propose an novel metric to evaluate our results.

Keywords

Deep learning, Automated Audio Captioning, Natural Language Processing, Transformers, Movies

Ευχαριστίες

Θα ήθελα καταρχάς να ευχαριστήσω τον καθηγητή κ. Αλέξανδρο Ποταμιάνο για την επίβλεψη αυτής της διπλωματικής εργασίας. Οι διαλέξεις του ήταν ένα από τα βασικά ερεθίσματα που με ώθησαν να θέλω να ασχοληθώ με τη Μηχανική Μάθηση και την Επεξεργασία Φυσικής Γλωσσάς.

Θα ήθελα επίσης να ευχαριστήσω βαθιά τον συν-επιβλέποντα της διπλωματικής μου εργασίας, Αθανάσιο Κατσμάνη για πολύτιμη βοήθεια του, τα σχόλια και τις παρατηρήσεις του, καθώς και για τον χρόνο που αφιέρωσε στις συζητήσεις και την συνεργασία μας. Χωρίς την καθοδήγηση του η εκπόνηση αυτής της εργασίας δεν θα ήταν δυνατή.

Πολύ σημαντικές ήταν επίσης οι συζητήσεις με του υποψήφιους διδάκτορες Γρήγορη Μπάστα, Γιώργο Παρασκευόπουλο και Ευθύμη Γεωργίου με τους οποίου είχα την τύχη να συνεργαστώ ως υπότροφος του Ινστιτούτου Επεξεργασίας του Λόγου του Ερευνητικού Κέντρου Αθηνά.

Τέλος θα ήθελα να ευχαριστήσω φίλες και φίλους, αγαπημένες και αγαπημένους που με ανέχτηκαν και με στήριξαν αυτή την δύσκολη χρονιά.

Αθήνα, Ιούλιος 2022

Θοδωρής Κουζέλης

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο εργαστήριο Επεξεργασίας Φωνής και Φυσικής Γλώσσας (NTUA Speech And Language Processing Group) της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου στην Αθήνα, κατά το έτος 2021-2022.

Πίνακας εξωφύλλου: Andrei Tarkovsky, *Stalker*, 1979

Time past and time future
Allow but a little consciousness.
To be conscious is not to be in time
But only in time can the moment in the rose-garden,
The moment in the arbour where the rain beat,
The moment in the draughty church at smokefall
Be remembered; involved with past and future.
Only through time time is conquered.

Burnt Norton - T.S Eliot

Κεφάλαιο 0

Εκτεταμένη Ελληνική Περίληψη

0.1 Εισαγωγή

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη της Αυτόματης Περιγραφής Ήχου, δηλαδή της περιγραφής ηχητικών συμβάντων με χρήση φυσικής γλώσσας. Η Αυτόματη Περιγραφή Ήχου είναι ένα πολύτροπικο πρόβλημα μετάφρασης που συνδυάζει την Επεξεργαία Ηχητικού Σήματος και την Επεξεργασία Φυσικής Γλώσσας. Η Περιγραφή Ήχου εστιάζει σε ηχητικά συμβάντα και τις χωροχρονικές τους σχέσεις σε ένα ηχητικό σήμα και τις εκφράζει σε φυσική γλώσσα. Είναι ένα πρόσφατο και αρκετά ανεξερεύνητο πρόβλημα, το οποίο έχει μεγάλες δυνατότητες για πρακτικές εφαρμογές, όπως τη αυτόματη παραγωγή υποτίτλων για κωφά και βαρήκοα άτομα σε ταινίες.

Προτείνουμε μια εφαρμογή για την Αυτόματη Περιγραφή Ήχου. Ένα νέο πρόβλημά, όπου δοθέντος του ηχητικού σήματος μιας ταινίας, ένα σύστημά έχει σκοπό να παράγει περιγραφές για ηχητικά συμβάντα. Ουσιαστικά, το προβλήματος που προτείνουμε είναι η αυτόματη παράγωγή υποτίτλων για κωφά και βαρήκοα άτομα. Το σύστημά που προτείνουμε ανιχνεύει τα τμήματα των ηχητικών συμβάντων χρησιμοποιώντας ένα προ-εκπαιδευμένο μοντέλο ταξινόμησης ήχων και παράγει περιγραφές χρησιμοποιώντας το μοντέλο Αυτόματης Περιγραφής Ήχου που έχουμε εκπαιδεύσει. Για να βελτιώσουμε την απόδοση του μοντέλου μας δημιουργούμε ένα εξειδικευμένο σύνολο δεδομένων από ταινίες και υπότιτλους για κωφά και βαρήκοα άτομα. Επιπλέον, ενσωματώνουμε την κειμενική πληροφορία του μοντέλου ταξινόμησης στην παραγωγή των περιγραφών, δημιουργώντας ένα μοντέλο για περιγραφή ήχων οδηγούμενη από κείμενο. Τέλος προτείνουμε μια μετρική για την αξιολόγηση των αποτελεσμάτων μας.

Οι σημαντικότερες συνεισφορές μας στο σχετικό ερευνητικό πεδίο είναι οι εξής:

- Προτείνουμε μια νέα αρχιτεκτονική για την Αυτόματη Περιγραφή Ήχου που έχει σκοπό να είναι λειτουργική όταν λίγοι υπολογιστικοί πόροι είναι διαθέσιμοι.

- Ερευνούμε το αντιστάθμισμα μεταξύ της σημασιολογικής ακρίβειας και της ποικιλομορφίας των περιγραφών.

- Προτείνουμε μια εφαρμογή της Αυτόματης Περιγραφής Ήχου για της αυτόματης παραγωγή υποτίτλων για κωφά και βαρήκοα άτομα σε ταινίες.

- Για να πετύχουμε αυτό το σκοπό συλλέγουμε ένα σύνολο δεδομένων από ταινίες και υπότιτλους στα οποία εκπαιδεύουμε το μοντέλο μας.

0.2 Θεωρητικό Υπόβαθρο

Ο Transformer [92] είναι μια αρχιτεκτονική βαθιάς μηχανικής μάθησης που εισήχθη για να λύσει το πρόβλημα της αυτόματης μετάφρασης. Είναι εμπνευσμένος από την επιτυχία του μηχανισμού προσοχής και είναι σε θέση να επεξεργάζεται δεδομένα παράλληλα. Αυτό τον καθιστά πολύ γρήγορο σε κατανεμημένα περιβάλλοντά, ενώ ξεπερνά σε επίδοση τις αναδρομικές αρχιτεκτονικές. Πρόκειται για μια αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή. Τόσο ο κωδικοποιητής όσο και ο αποκωδικοποιητής αποτελούνται από πολλά στοιβαγμένα επίπεδα. Κάθε επίπεδο του κωδικοποιητή αποτελείται από υπο-επίπεδα αυτοπροσοχής και Feed Forward δικτύων. Ο αποκωδικοποιητής έχει την ίδια δομή, με την προσθήκη ενός υπο-επιπέδου δια-προσοχής. Το υπο-επίπεδο αυτοπροσοχής δημιουργεί μια αναπαράσταση της ακολουθίας με βάση τα συμφραζόμενα, αναλύοντας την εξάρτηση μεταξύ των μερών της. Το υπο-επίπεδο δια-προσοχής είναι υπεύθυνο για την ανάλυση της εξάρτησης μεταξύ των ακολουθιών εισόδου και εξόδου. Το αποτέλεσμα του τελευταίου επιπέδου του αποκωδικοποιητή μετατρέπεται τελικά σε πιθανότητες συμβόλων, χρησιμοποιώντας έναν γραμμικό μετασχηματισμό και μια συνάρτηση softmax.

Ο μηχανισμός προσοχής είναι το βασικότερο κομμάτι και δίνεται από τον εξής τύπο :

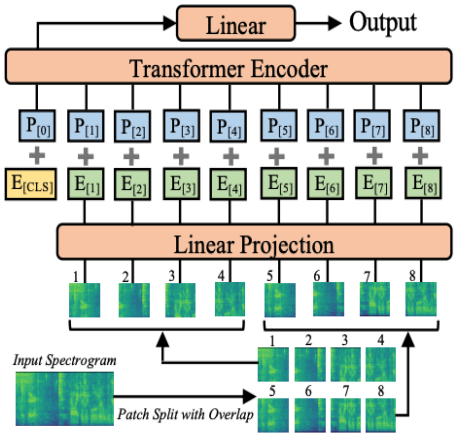
$$Attention(Q, V, K) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{1}$$

όπου d_k είναι η διάσταση του πίνακα K, οι πίνακες K και V αναφέρονται στον κωδικοποιητή ενώ ο πίνακας Q στον αποκωδικοποιητή, στην περίπτωση του μηχανισμου δια-προσοχής, ενώ κατά το μηχανισμό αυτοπροσοχής οι πίνακες αφορούν το ίδιο μέρος του δικτύου.

Πρόσφατα, Transformer αρχιτεκτονικές έχουν κυριαρχήσει στα πεδία της Επεξεργασίας Εικόνας και της Επεξεργασίας Ηχητικού Σήματος. Το μοντέλο ViT [16] που είναι βασισμένο πάνω στην Transformer αρχιτεκτονική, ξεπέρασε σε απόδοση τα συνελικτικά δίκτυά στη ταξινόμηση εικόνων. Επίσης πρόσφατα ένα μοντέλο, το Audio Spectrogram Transformer [25], με αρκετά παρόμοια αρχιτεκτονική με αυτή του ViT, έδωσε τα καλύτερα αποτελέσματά στην ταξινόμηση ήχων επί του συνόλου δεδομένων AudioSet [23]. Το μοντέλο αυτό δέχεται σαν είσοδο ένα σπεκτρόγραμμα το οποίο αντιμετωπίζει σαν μονοκάναλη εικόνα. Τα μοντέλα αυτά ακολουθούν την παρακάτω διαδικασία προκειμένου να μετατρέψουν την εικόνα / σπεκτρόγραμμα της εισόδου σε μια ακολουθία από tokens:

- Η εικόνα χωρίζεται σε επιφανειακά τμήματα.
- Τα επιφανειακά τμήματα μετασχηματίζονται σε μονοδιάστατα διανύσματα.
- Παράγονται διανύσματα χαμηλότερης διάστασης μέσω ενός γραμμικού δικτύου.
- Προσθέτονται embeddings που κωδικοποιούν την θέση του κάθε διανύσματος.

Το βασικότερο πρόβλημα του AST είναι ότι η εξαγωγή των επιφανειακών τμημάτων από το σπεκτρόγραμμα δημιουργεί μια μεγάλη σε μήκος ακολουθία εισόδου. Ένα βασικό μειονέκτημά των Transformers είναι η χρονική και χωρική τους πολυπλοκότητά αυξάνεται τετραγωνικά ως προς την είσοδο.



Σχήμα 1: *Η εξαγωγή της ακολουθίας εισόδου από το σπεκτρόγραμμα στο μοντέλο AST.*

Αυτό το τετραγωνικό bottleneck δημιουργείται κατά τον υπολογισμό του μηχανισμού προσοχής. Ο πολλαπλασιασμός των μητρώων $Q \cdot K^T$ έχει πολυπλοκότητα $O(L^2)$ όπου L το μήκος της εισόδου.

Το μοντέλο PaSST [48] χρησιμοποιεί μια απλή και αποτελεσματική μέθοδο για να αντιμετωπίσει το πρόβλημα που δημιουργεί το τετραγωνικό bottleneck. Η μέθοδος ονομάζεται Patchout, και μειώνει δραστικά τον χρόνο εκπαίδευσής, ενώ ταυτόχρονα λειτουργεί σαν ομαλοποιητής βοηθώντας το μοντέλο να γενικεύσει καλύτερα. Η βασική ιδέα του Patchout είναι, να διαγράφουμε κάποια κομμάτια της εξαγόμενης ακολουθίας εισόδου, ενθαρρύνοντας το μοντέλο να κάνει μια πρόβλεψή, χρησιμοποιώντας μια ημιτελή ακολουθία.

0.3 Αυτόματη Περιγραφή Ήχου

Η Αυτόματη Περιγραφή Ήχου έχει ως στόχο να αναγνωρίσει τα ηχητικά συμβάντα καθώς και τις χώρο-χρονικές του σχέσεις σε ένα αρχείο ήχου και να τα περιγράψει με φυσική γλώσσα. Είναι ένα αρκετά ανεξερεύνητο πρόβλημα και τράβηξε την προσοχή των ερευνητών όταν σύνολα δεδομένων για επιβλεπόμενη μάθηση έγιναν διαθέσιμα.

Το μεγαλύτερο σύνολο δεδομένων για Αυτόματη Περιγραφή Ήχου και αυτό που χρησιμοποιούμε σε αυτήν την εργασία είναι το AudioCaps [40]. Όλα τα αρχεία ήχου που περιέχει είναι μεγέθους 10 δευτερολέπτων και έχουν εξαχθεί από το AudioSet [23]. Περιέχει συνολικά 51 χιλιάδες δείγματα χωρισμένα σε σύνολα εκπαίδευσης, επιβεβαίωσης και δοκιμής. Κάθε αρχείο ήχου στο σύνολο εκπαίδευσης συνοδεύεται από μία περιγραφή, ενώ τα αρχεία ήχου στα σύνολα επιβεβαίωσης και δοκιμής από πέντε περιγραφές.

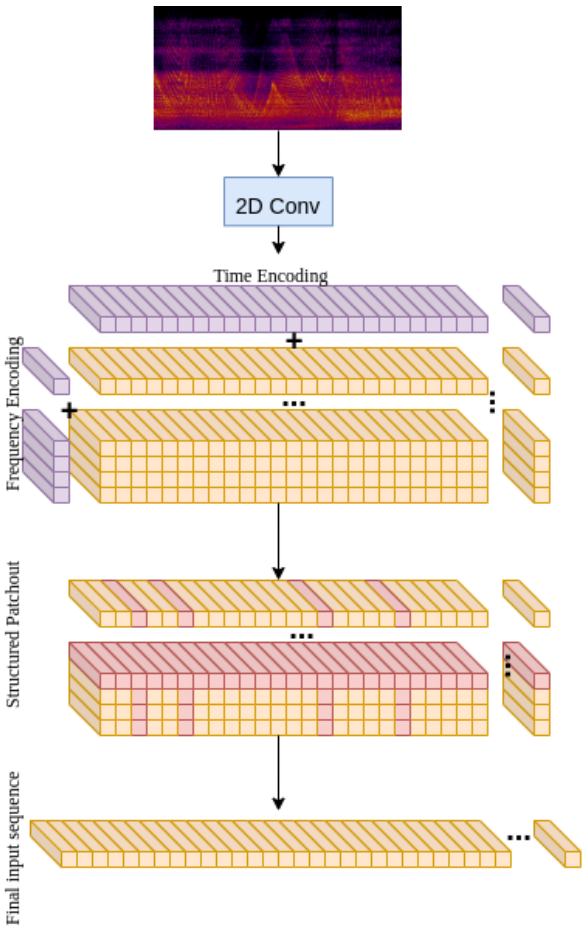
Τα μοντέλα Αυτόματης Περιγραφής Ήχου ακολουθούν μια Sequence-to-Sequence μοντελοποίηση προκειμένου να μετατρέψουν μια ακολουθία ήχου σε ακολουθία λέξεων. Ένας κωδικοποιητής χρησιμοποιείται για να μετατρέψει την ακολουθία εισόδου σε μια αναπα-

ράσταση που περιέχει την βασικότερη πληροφορία της εισόδου και ένας αποκωδικοποιητής παίρνει αυτήν την αναπαράστασή και παράγει τις περιγραφές.

Στη βιβλιογραφία χρησιμοποιούνται κυρίως συνελικτικά δίκτυα (CNN's) για να κωδικοποιήσουν το ηχητικό σήμα και ακολουθιακά δίκτυά (RNN's) σαν αποκωδικοποιητές. Για να λύσουν το πρόβλημα της ανεπάρκειας δεδομένων των σύνολων δεδομένων για Αυτόματη Περιγραφή Ήχου χρησιμοποιούνται συχνά προ-εκπαιδευμένα μοντέλα.

Η αρχιτεκτονική που προτείνουμε είναι μια Transformer αρχιτεκτονική η οποία χρησιμοποιεί ένα προ-εκπαιδευμένο μοντέλο σαν κωδικοποιητή. Επιπλέον για να μειώσουμε την πολυπλοκότητα του μοντέλου μας εφαρμόζουμε την μέθοδο Patchout.

Αρχικά εξάγουμε ένα σπεκτρόγραμμα από το αρχείο ήχου και το περνάμε από ένα δισδιάστατο συνελικτικό δίκτυο. Η έξοδος του δικτύου είναι ένα feature map $Q_{map} \in \mathbb{R}^{d \times F \times T}$. Έπειτα προσθέτουμε δυο διανύσματα στο feature map τα οποία κωδικοποιούν τις θέσεις των επιφανειακών τμημάτων. Εφαρμόζουμε την μέθοδο Patchout όπου διαγράφουμε τυχαία έναν αριθμό από συχνοτικές και χρονικές λωρίδες από το feature map. Τέλος μετασχηματίζουμε το feature map σε ένα μονοδιάστατο διάνυσμα και το περνάμε στον Transformer κωδικοποιητή.



Σχήμα 2: *Εξαγωγή της εισόδου ακολουθίας εισόδου από το σπεκτρόγραμμα.*

Αρχικοποιούμε τα βάρη του κωδικοποιητή με αυτά ενός προ-εκπαιδευμένου PaSST[48] μοντέλου ώστε αυξήσουμε την αποδοτικότητά του μοντέλου. Προκειμένου να αυξήσουμε

περαιτέρω την ομαλοποίηση χρησιμοποιούμε GELU [33] συναρτήσεις ενεργοποίησης τόσο στον κωδικοποιήτη όσο και στον αποκωδικοποιητή.

Για να αντιμετωπίσουμε την έλλειψή δεδομένων του συνόλου δεδομένων εκπαίδευσης χρησιμοποιούμε δύο τεχνικής επαύξησης δεδομένων. Η πρώτη τεχνική ονομάζεται Specaugment [76] και έχει χρησιμοποιηθεί αρχικά στην Αναγνώριση Φωνής με μεγάλη επιτυχία. Η δεύτερη ονομάζεται Mixup. Με την μέθοδο αυτή δημιουργούμε τεχνητά δεδομένα που είναι αποτέλεσμα ενός γραμμικού συνδυασμού δεδομένων από το σύνολο δεδομένων εκπαίδευσης. Η εφαρμογή του Mixup στην Αυτόματη Περιγραφή Ήχου δεν είναι τετριμμένη καθώς δεν πρόκειται για ένα πρόβλημα ταξινόμησης. Έτσι δοκιμάζουμε να συνδυάσουμε τις περιγραφές στο επίπεδο των embeddings.

Η αποτίμηση των αποτελεσμάτων της Αυτόματης Περιγραφής Ήχου γίνεται στην βιβλιογραφία με χρήση μετρικών για Μηχανική Μετάφραση και Περιγραφής Εικόνων. BLEUn, ROUGEl και METEOR είναι μετρικές Μηχανικής Μετάφρασης. Η BLEUn είναι ένα τροποποιημένο precision, που υπολογίζεται ως σταθμισμένος γεωμετρικός μέσος όρος σε διαφορετικά -grams. Η ROUGEl υπολογίζει τα F-measure μετρώντας το μήκος της μεγαλύτερης κοινής υποακολουθίας. Η METEOR αξιολογεί μια περιγραφή υπολογίζοντας έναν αρμονικό μέσο όρο του precision και του recall με βάση σαφείς αντιστοιχίσεις λέξης με λέξη μεταξύ της παραγώμενης περιγραφής και των πραγματικών περιγραφών. Η CIDEr υπολογίζει την ομοιότητα του συνημιτόνου μεταξύ n-grams τα οποία έχουν σταθμιστεί ως προς τη συχνότητα αντίστροφης συχνότητας εγγράφου (TF-IDF). Η SPICE δημιουργεί γραφήματα σκηνής για περιγραφές και υπολογίζει το F-measure με βάση τις πλειάδες στα γραφήματα της σκηνής. SPIDEr είναι ο μέσος όρος SPICE και CIDEr και επιλέγεται ως η επίσημη μετρική κατάταξης στην πρόκληση DCASE [1]. Η η μετρική SPICE εξασφαλίζει περιγραφές που είναι σημασιολογικά πιστές στο ηχητικό περιεχόμενο, ενώ η βαθμολογία CIDEr διασφαλίζει ότι οι περιγραφές είναι συντακτικά ορθές.

Τα αποτελέσματα των πειραμάτων μας δείχνουν ότι το μονελο μας πετυχαίνει ανταγωνιστικά απολεσματα σε σχέση με τα state-of-the-art μοντέλα ενώ ταυτόχρονα χρησιμοποιώντας την μέθοδο Patchout μειώνουμε δραστικά τον χρόνο εκπαίδευσης. Ολα μας τα πειράματα διεξάχθηκαν σε μία NVIDIA GeForce RTX 2080. Στους παρακάτω πίνακες συγκρίνουμε κάποια από τα μοντέλα που εκπαιδεύσαμε με τρία state-of-the-art μοντέλα, και με το ένα από αυτά το οποίο είναι και αυτό βασισμένο σε Transformer αρχιτεκτονική, συγκρίνουμε την απόδοση, την χρονική και χωρική τους πολυπλοκότητα.

Model	Quality Metrics								
	BLUE-1	BLEU-2	BLEU-3	BLEU-4	METEOR	$ROUGE_L$	CIDEr	SPICE	SPIDEr
CNN+Transformer	0.64	0.47	0.34	0.23	0.22	0.46	0.69	0.16	0.42
ACT	0.64	0.48	0.35	0.25	0.22	0.46	0.67	0.16	0.42
BART + YAMNet + PANNs	**0.69**	**0.52**	**0.38**	**0.26**	**0.24**	**0.49**	**0.75**	**0.17**	**0.46**
captionPaSST	0.24	0.14	0.09	0.05	0.13	0.27	0.62	0.17	0.40
captionPaSST + SpecAugment	0.67	**0.50**	0.35	0.24	**0.23**	**0.48**	0.66	0.16	0.41
captionPaSST + SpecAugment+ Mixup	**0.68**	0.50	0.37	**0.26**	0.23	0.48	0.71	**0.17**	0.44

Model	Mem (MiB)	Samples/Sec	Epochs	SPIDEr
ACT	6419	78	30	0.42
CaptionPaSST	4231	120	15	0.44

[1] https://dcase.community/

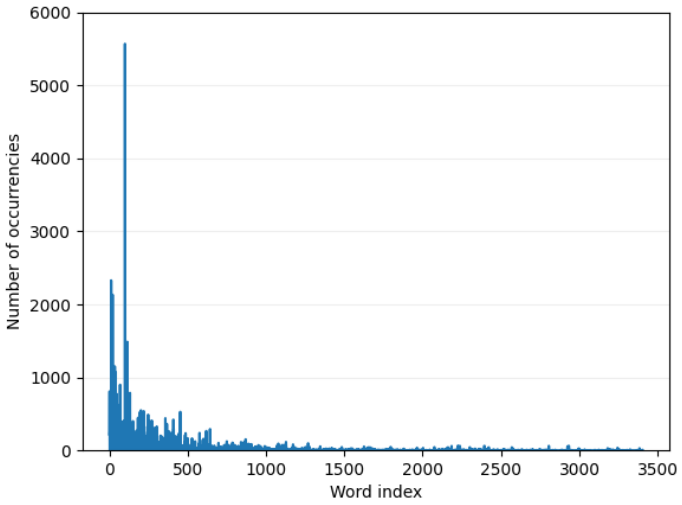
0.4 Συλλογή Δεδομένων από Ταινίες

Για να δημιουργήσουμε ένα μοντέλο για περιγραφή ήχων σε ταινίες κατασκευάσαμε ένα σύνολο δεδομένων χρησιμοποιώντας ταινίες και υπότιτλους για κωφά και βαρήκοα άτομα. Οι υπότιτλοι για κωφά και βαρήκοα άτομα περιέχουν περιγραφές για ηχητικά γεγονότα και την μουσική.

Για την κατασκευή του συνόλου δεδομένων συλλέξαμε περίπου πεντακόσιες (500) ταινίες και επεισόδιά από σειρές και τους αντίστοιχους υπότιτλους για κωφά και βαρήκοα άτομα. Μπορούμε να εξάγουμε τους υπότιτλους που περιγράφουν τα ηχητικά συμβάντα καθώς εμφανίζονται πάντα εντός παρενθέσεων ή αγκυλών [] ή (). Εφόσον γνωρίζουμε χάρη στα αρχεία υποτίτλων τους χρόνους της αρχής και του τέλος κάθε ηχητικού συμβάντος μπορούμε να εξαγάγουμε ζευγάρια ήχων και περιγραφών. Αρχικά εξαγάγουμε 56935 τέτοια ζευγάρια.

Κατά της διάρκεια της επεξεργασίας αφαιρούμε όλα τα ζευγάρια που σχετίζονται με την αναγνώριση του εκάστοτε ομιλητή. Για να το πετύχουμε αυτό αφαιρούμε όλους τους υπότιτλους που περιέχουν κύρια ονόματα χρησιμοποιώντας ένα προ-εκπαιδευμένο BERT μοντέλο.

Το τελικό σύνολο δεδομένων το οποίο ονομάζουμε MovieCaps περιέχει 45432 ζευγάρια ήχων και υποτίτλων. Το λεξιλόγιο του συνολικού κειμένου των περιγραφών περιέχει 4305 λέξεις. Κάποιες περιγραφές εμφανίζονται πολύ συχνά στο σύνολο δεδομένων όπως: Music Playing, ενώ κάποιες άλλες μόνο μία φορά: Device slurping. Η κατανομή των συχνοτήτων εμφάνισης των λέξεων φαίνεται στο παρακάτω γράφημα.



Σχήμα 3: *Η κατανομή συχνοτήτων των λέξεων του λεξιλογίου.*

Σπάμε το σύνολο δεδομένων σε τρία υποσύνολα με ποσοστά 80 - 10 - 10 %. Για να πετύχουμε ένα ισότιμο σπάσιμο στις κατανομές των συχνοτήτων των λέξεων στα υποσύνολα εκπαίδευσής και επιβεβαίωσης και δοκιμής εφαρμόζουμε μια μέθοδο διαστρωμάτωσης.

Σε σύγκριση με άλλα σύνολα δεδομένων για Αυτόματη Περιγραφή Ήχου το MovieCaps, οι περιγραφές είναι πολύ πιο σύντομες και ακριβείς. Στον παρακάτω πίνακα φαίνεται η σύγκρισή του MovieCaps με τρία άλλα σύνολα δεδομένα σε κάποια βασικά τους χαρακτηρι-

στικά.

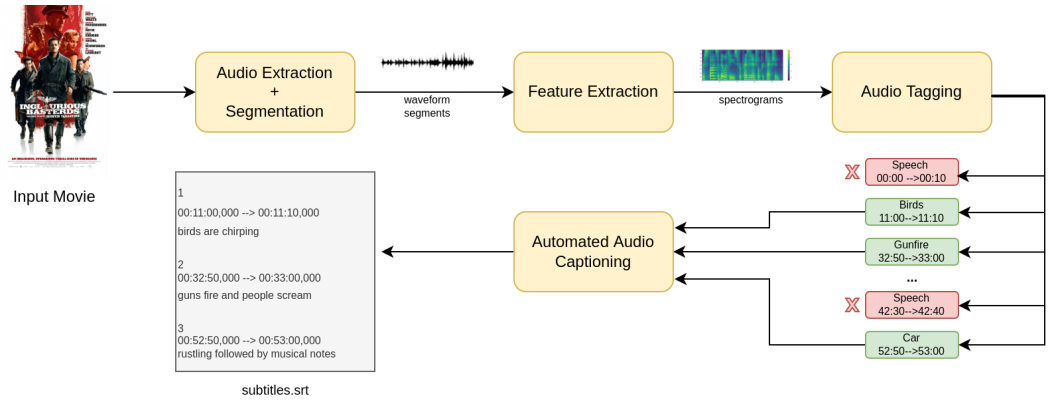
Dataset	Audio Clips	Vocab Size	Unique Captions	Caption Lenght (std)
AudioCaps	57188	5218	52198	9.17 (4.27)
Clotho	5929	4373	29611	11.34 (2.78)
MACS	3930	2775	16262	9.46 (3.89)
MovieCaps	**45432**	**4002**	**8685**	**1.86 (1.12)**

0.5 Αυτόματη Περιγραφή Ήχων σε Ταινίες

Προτείνουμε μια εφαρμογή της Αυτόματης Περιγραφής Ήχου κατασκευάζοντας ένα σύστημά για την περιγραφή ηχητικών συμβάντων σε ταινίες. Σκοπός του συστήματος είναι δοθέντος του ηχητικού σήματος μιας ταινίας, να παράγει υπότιτλους για βαρήκοα και κωφά άτομα. Με μια γενικότερη ματιά το πρόβλημα που προτείνουμε είναι ένας συνδυασμός της Αυτόματης Περιγραφής Ήχου και της Αναγνώρισης Ηχητικών Συμβάντων (Sound Event Detection) καθώς στοχεύει να εντοπίσει τα χρονικά διαστήματα των ήχων και να τους περιγράψει με φυσική γλώσσα.

Η baseline μέθοδος που ακολουθούμε για την επίλυση αυτού του προβλήματος χωρίζεται σε δύο υπό-συστήματά. Ένα για την αναγνώριση των ηχητικών συμβάντων και ένα για την αυτόματη περιγραφή. Τα βήματα που εκτελούνται είναι τα εξής:

- Αρχικά, εξάγουμε το ηχητικό σήμα από μία ταινία και το χωρίζουμε σε τμήματα 5-10 δευτερολέπτων.
- Εξάγουμε σπεκτρογράμματα από κάθε ηχητικό τμήμα.
- Κάθε ένα από αυτά τα σπεκτρογράμματα προωθούνται σε ένα προ-εκπαιδευμένο μοντέλο ταξινόμησης.
- Αν το label που αποδόθηκε από το μοντέλο ταξινόμησης στο ηχητικό τμήμα, δεν αφορά ομιλία, τότε το ηχητικό αυτό τμήμα προωθείται σε ένα μοντέλο Αυτόματης Περιγραφής Ήχου.
- Η περιγραφή αντιστοιχίζεται χρονικά με το ηχητικό τμήμα και παράγεται ένα αρχείο υποτίτλων.



Σχήμα 4: *Το σύστημα αυτόματης περιγραφής ήχων σε ταινίες.*

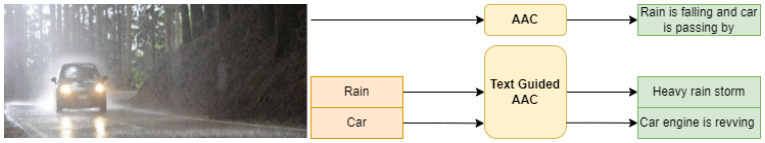
Σκοπεύοντας να περιγράφουμε μόνο τα sailient ηχητικά τμήματα εφαρμόζουμε ένα κατώφλι στην πιθανότητα του πιο πιθανού label. Έτσι αν σε ένα ηχητικό τμήμα δεν είναι κυρίαρχο ένα ηχητικό γεγονός, δεν το περνάμε στο μοντέλο Ηχητικής Περιγραφής.

Τα βασικά **μειονεκτήματά** της μεθόδου είναι τα εξής:

- Δεν βρίσκουμε την έναρξη και το τέλος των ηχητικών συμβάντων καθώς σπάμε το αρχικό ηχητικό σήμα σε τμήματα σταθερού μεγέθους.
- Η πληροφορία που εξάγουμε από το μοντέλο ταξινόμησης, δηλαδή τα labels, χάνεται.
- Υπάρχει ασυμβατότητα ανάμεσά στα δεδομένα εκπαίδευσής του μοντέλου Ηχητικών Περιγραφών, δηλαδή του AudioCaps, και των δεδομένων που καλείτε να αναγνωρίσει, δηλαδή ήχους και περιγραφές ταινιών.

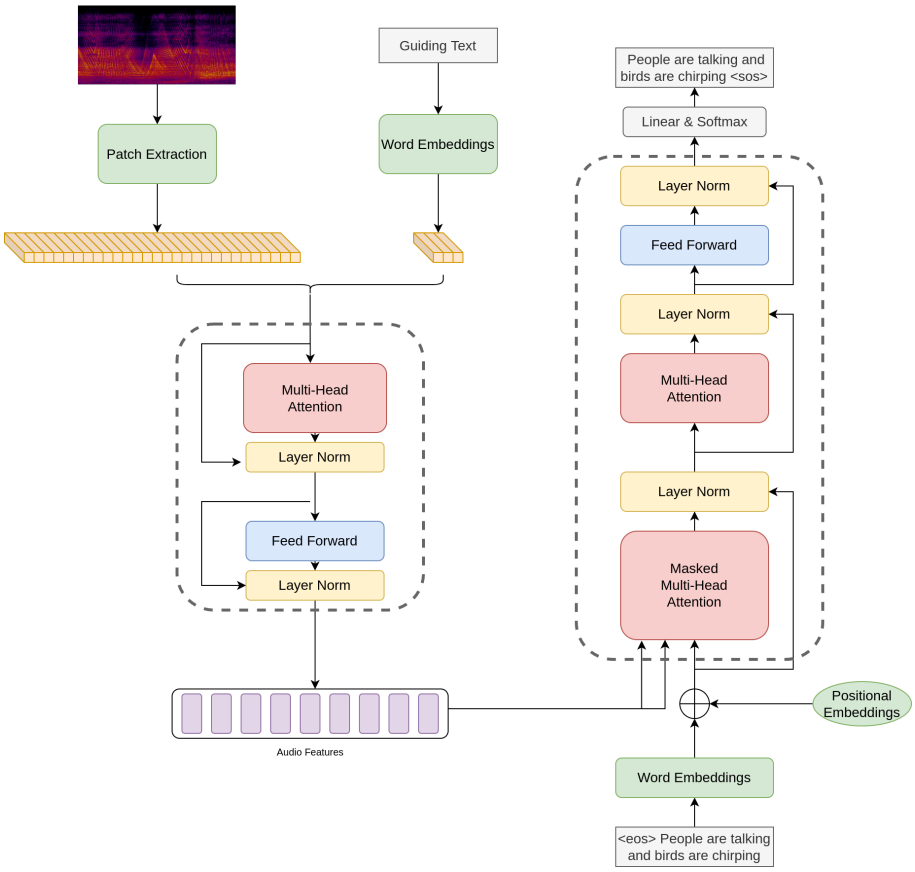
Στην παρούσα εργασία προσπαθούμε να αντιμετωπίσουμε τα 2 τελευταία μειονεκτήματα που μόλις αναφέραμε.

Προκειμένου να εκμεταλλευτούμε την πληροφορία των labels, κατασκευάζουμε ένα μοντέλο το οποίο δέχεται ως είσοδο ένα αρχείο ήχου και ένα σύντομο κείμενο καθοδήγησης, και έχει σκοπό να παράξει μια περιγραφή που εστιάζει στα ηχητικά συμβάντα που περιγράφονται στο κείμενο καθοδήγησης.



Σχήμα 5: *Αυτοματή Περιγραφή Ήχου καθοδηγούμενη από κείμενο.*

Το μοντέλο που προτείνουμε δέχεται ως επιπλέον είσοδο στον Transformer κωδικοποιήτη, μία ακολουθία από tokens τα οποία έχουν εξαχθεί από το κείμενο εισόδου.



Σχήμα 6: *Το κείμενο εισόδου γίνεται embedded στο χώρο εισόδου και συνδέεται συριακά με την ακολουθία που έχει εξαχθεί απο το σπεκτρόγραμμα*

Προκειμένου να λύσουμε την αναντιστοιχία ανάμεσα στα δεδομένα εκπαίδευσης και στις ταινίες, εκπαιδεύουμε το μοντέλο μας στο σύνολο δεδομένων MovieCaps που κατασκευάσαμε. Εκπαιδεύουμε ένα μοντέλο στο MovieCaps και κάνουμε fine-tune ένα μοντέλο προ-εκπαιδευμένο στο AudioCaps. Το δεύτερο παρουσιάζει και τα καλύτερα αποτελέσματα.

Τέλος προτείνουμε μια μετρική αξιολόγησης για να αποτιμήσουμε την απόδοση του συστήματος που κατασκευάσαμε. Η μετρική είναι εμπνευσμένη από το Diarization Error Rate και ορίζεται ως εξής:

$$ER = \frac{D + I - O \cdot SPIDEr}{N} \tag{2}$$

όπου:

- Το Deletion **D** ορίζετε ως ο χρόνος των τμημάτων όπου το σύστημα παράγει περιγραφή χωρίς να υπάρχει στους πραγματικούς SDH υπότιτλους.
- Το Insertion **I** ορίζετε ως ο χρόνος των τμημάτων που το σύστημα παράγει περιγραφή χωρίς να υπάρχει στους πραγματικούς SDH υπότιτλους.
- Το Overlap **O** ορίζετε ως ο χρόνος των τμημάτων που τόσο το σύστημα όσο και οι πραγματικοί υπότιτλοι περιγράφουν.

Στόχος της μετρικής είναι να αξιολογήσει ταυτόχρονα την δυνατότητα του συστήματος να εντοπίσει σωστά τα ηχητικά συμβάντα, καθώς και την ποιότητά των παραγομένων περιγραφών

σε σχέση με τις αληθινές περιγραφές. Έχουμε στη διάθεση μας άλλες 50 ταινίες που δεν συμπεριλάβαμε στο σύνολο MovieCaps, με σκοπό να αξιολογήσουμε το σύστημα μας σε αυτές. Τα αποτελέσματα για τέσσερα μοντέλα Περιγραφής Ήχου φαίνονται στον παρακάτω πίνακα:

AAC Model	**I**	**D**	**C**	**SPIDEr**	**ER**
Zero-shot	0.73	0.83	0.011	0.07	1.548
MovieCaps (from scratch)	0.73	0.83	0.020	0.12	1.539
Guiding Text	0.73	0.83	0.025	0.15	1.534
MovieCaps (AudioCaps pretrained)	0.73	0.83	0.035	0.21	**1.524**

Στις στήλες I, D, C παρουσιάζουμε το ποσοστό των Insertions, Deletions και Correct ως προς το N δηλαδή τον συνολικό χρόνο των πραγματικών τμημάτων περιγραφής.

0.6 Μελλοντικές Κατευθύνσεις

Μία από τις κύριες προκλήσεις της Αυτόματης Περιγραφής Ήχου είναι η έλλειψη επαρκών δεδομένων. Συνολικά, τα ελεύθερα διαθέσιμα σύνολα δεδομένων για την Αυτόματη Περιγραφή Ήχου περιέχουν λιγότερα από 60 χιλιάδες δείγματα. Ως εκ τούτου, μια κατεύθυνση για την μελλοντική έρευνα, μπορεί να είναι η συλλογή δεδομένων με αδύναμη επισήμανση από υπότιτλους και ταινίες SDH ή από διαθέσιμες διαδικτυακές πηγές. Περισσότερα διαθέσιμα δεδομένα θα μπορούσαν να επιτρέψουν στα μοντέλα να μάθουν πιο ισχυρή αναπαράσταση ήχου-κειμένου, όπως το CLIP στην Όραση υπολογιστών.

Επιπλέον, η Αυτόματης Περιγραφής Ήχου μπορεί ενδεχομένως να συνδεθεί με άλλα πολυτροπικά προβλήματα ήχου-γλώσσας όπως η Ηχητική Απάντηση Ερωτήσεων (Audio Question Answering) και Αυτόματη Περιγραφή Ήχου καθοδηγούμενη από κείμενο.

Στην εργασία μας πειραματιστήκαμε με ένα μοντέλο Αυτόματης Περιγραφής Ήχου καθοδηγούμενης από κείμενο για να βελτιώσουμε την απόδοση του συστήματος μας για τη δημιουργία υποτίτλων ηχητικών συμβάντων σε ταινίες. Πιστεύουμε ότι αυτό το πρόβλημα αξίζει περαιτέρω εξερεύνηση. Μια εφαρμογή της Αυτόματης Περιγραφής Ήχου καθοδηγούμενης από κείμενο θα μπορούσε να δημιουργήσει εργαλεία προσβασιμότητας για χρήστες με προβλήματα ακοής, οι οποίοι μπορούν να επιλέξουν ένα κείμενο καθοδήγησης που παράγεται από έναν ανιχνευτή ήχου, για να λάβουν έπειτα μια καθοδηγούμενη περιγραφή των γύρω ήχων τους.

Όπως έχουμε ήδη συζητήσει, το κύριο μειονέκτημα της προτεινόμενης προσέγγισής μας είναι η εφαρμογή μιας αυθαίρετης τμηματοποίησης σταθερού μήκους στον ήχο εισόδου αντί για μία άμεση ανίχνευση των χρόνων έναρξης και τέλους των συμβάντων ήχου. Στο μέλλον, στοχεύουμε να χρησιμοποιήσουμε τις χρονικές πληροφορίες των υποτίτλων για κωφά και βαρήκοα άτομα (SDH) και να εκπαιδεύσουμε ένα μοντέλο ανίχνευσης συμβάντων ήχου.

Για να εκπαιδεύσουμε ένα μοντέλο ανίχνευσης συμβάντων ήχου για τους σκοπούς μας, πρέπει να δημιουργήσουμε ένα σύνολο δεδομένων με ισχυρές ετικέτες χρησιμοποιώντας

ταινίες και υπότιτλους SDH. Για να γίνει αυτό, κάθε περιγραφή πρέπει να αντικατασταθεί με μια ετικέτα. Μια προσέγγιση που μπορούμε να ακολουθήσουμε, είναι να δημιουργήσουμε N κλάσεις όπως *Speech*, *Rain*, *Music* κ.λπ. και να αντικαταστήσουμε κάθε λεξάντα με την πιο παρόμοια σημασιολογικά ετικέτα, χρησιμοποιώντας BERT embeddings. Επιπλέον, θα διεξαγάγουμε μια έρευνα προκειμένου να διερευνήσουμε σε ποιο βαθμό η προτεινόμενη μετρική μας συσχετίζεται με τους ανθρώπινους αξιολογητές.

Chapter 1

Introduction

In this introductory chapter we aim to present the tasks that our dissertation aims to address and the broader field related to them. Firstly we discuss how linguistics, Natural Language Processing and Artificial Intelligence are intertwined in our view. We present our motivation and our main research topic. Lastly we present our main contributions in the field.

1.1 AI and Language

Language is on each own a very complex system. According to Descartes, language is a power only we humans possess, a power that sets us apart, in a qualitative, unbridgeable way from everything else there is, notably from animals and machines. in 1950, Alan Turing wrote a paper describing a test for a “thinking” machine. He argued that if a machine could have a conversation through the use of a teleprinter, and it imitated a human without noticeable differences then the machine could be considered capable of thinking.

Many different paradigms have been proposed in the field of linguistics to approach a broader understanding of language. In the early 1900s, a Swiss linguistics professor named Ferdinand de Saussure aimed to attack the concept of language as a product of human speech, describing languages as "systems of difference". He argued that words are just an acoustic image unhinged on them self of any particularly meaning. In this paradigm we can think of language as an arbitrary system that exists somewhat independent of speakers and can be analyzed independent of who speaks. De Saussure’s famous distinction between signifier and signified doesn’t mean that there is no relation between them but that their relation obeys a law that is independent from the subject. In his view language is governed by an inner combinatorial logic.

Recent progress in Artificial Intelligence and Natural Language Processing has produced models that perform surprisingly well at generating text, textual descriptions of images, answering questions, summarizing large documents, etc. We dont believe that models such us GPT-3 bridge the unbridgeable gap described by Descartes, but we view such models in continuity with the structuralist paradigm: language comes into view as a logical system to which the subject (the speaker) is incidental.

1.2 Motivation

Nonetheless, we are fascinated by AI systems. Their capability of generating natural language, shifts our understanding of reality and how we experience the world. Besides language there other modalities that play a crucial role to the human understanding of the world such images and sounds. We are interested of the ways these modalities are intertwined with language and how machines are able to mutually process and understand them. In this dissertation we research a mutli-modal task called Automated Audio Captioning. It can be viewed as a cross-modal translation task that aims to generate natural language descriptions of sound and audio events. It is a task that has received increasing attention in the recent years, but is still largely unexplored, compared to other multimodal task such as Image Captioning.

We believe that Automated Audio Captioning has a great potential in practical applications such as assisting deaf and hard of hearing people. Since a very close person to the writer of this dissertation suffers from hearing loss, we are motivated to research ways in which deaf and hard of hearing people can enjoy an equal access to the vastly growing and complex technologies and media that surround us.

1.3 Contributions

Our main contributions are the following:

- We propose a novel architecture for Automated Audio Captioning utilizing state-of-the-art strategies to reduce the computational complexity of our model.
- We investigate the quality vs diversity trade of language generation in our proposed model.
- We propose an application of Automated Audio Captioning in generating textual descriptions of sound events in Movies. This novel task is a fusion of Sound Event Detection and Automated Audio Captioning.
- To archive this goal we build a dataset from sound events in movies and their descriptions found in SDH (Subtitles for Deaf and Hard of Hearing).
- We propose a novel evaluation metric in order to evaluate the performance of our system.

1.4 Thesis outline

The rest of this dissertation is organized in the following way:

In **Chapter 2** we present a machine learning background and the theoretical foundations that are related to our work. We discuss older and today's deep learning strategies

for NLP and Audio Signal Processing tasks. Finally we explain in detail how Transformer based models work and how they are employed in Audio related tasks.

In **Chapter 3** we present our proposed model for Automated Audio Captioning (AAC). First we present the task and discuss the related work that has been recently done to tackle the challenges it presents. We discuss an error analysis we conducted on a state-of-the-art model and investigate several decoding algorithms that can be used to solve the issues exposed by the analysis. We present our proposed method for AAC, our experiments and our results on a freely available dataset for AAC, AudioCaps [40].

In **Chapter 4** we present the AAC dataset we created MovieCaps. We build this dataset in order to train a model for automated captioning of sound events in movies. We present how we collected the data, the pre-processing procedure we followed and a splitting methodology to create stratified training and testing splits.

In **Chapter 5** we present our proposed application of AAC, building a pipeline that given the audio of a movie, detects salient sound events and captions them. We address to this novel task as Sound Event Detection and Captioning (SEDC). We also propose a metric in order to evaluate our task. Finally we present our experiments and our results.

Chapter 6 is the last chapter of this dissertation. We make some concluding remarks drawn from our work and we propose future work that can be done to improve current results.

Chapter 2

Background

As mentioned in the Introduction, in this chapter we present our theoretical background, regarding Machine Learning (ML) literature and especially the Deep Learning (DL) subfield. We discuss Deep Learning approaches to Natural Language Processing and Audio Signal Processing and focus on models such as the Transformer [92] that is the backbone of our proposed approach, and its usage in sequence-to-sequence modelling. We also present in detail the PaSST tranformer model [48] which we use as the encoder for our AAC model captionPaSST.

2.1 Deep Learning Introduction

In recent years, machine learning has become more and more popular in research and has been incorporated in a large number of applications, including image classification, video recommendation, social network analysis, text mining, machine translation and so forth. Among the different ML algorithms, deep learning is very commonly employed in these applications. The continuing appearance of novel studies in the fields of deep learning is due to both the unpredictable growth in the ability to obtain data and the progress made in the hardware technologies, e.g. High Performance Computing.

Using conventional machine learning techniques requires several sequential steps, specifically pre-processing, feature extraction, wise feature selection, learning, and classification. Furthermore, feature selection has a great impact on the performance of machine learning techniques. Biased feature selection may lead to incorrect discrimination between classes. Conversely, deep learning has the ability to automate the learning of feature sets for several tasks, unlike conventional machine learning methods [52].

2.1.1 Feed Forward Neural Networks

A feed-forward neural network (FFNN) is an artificial neural network where connections between the nodes do not form a cycle. The information feed to the network moves in only one direction, forward, from the input nodes, through the hidden nodes to the output nodes. Each node of an FFNN usually computes the weighted sum of its inputs, possibly followed by a non-linear activation function 2.1.2.

The simplest kind of neural network is a single-layer perceptron network, which con-

sists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold the neuron fires and takes the activated value, otherwise it takes the deactivated value.

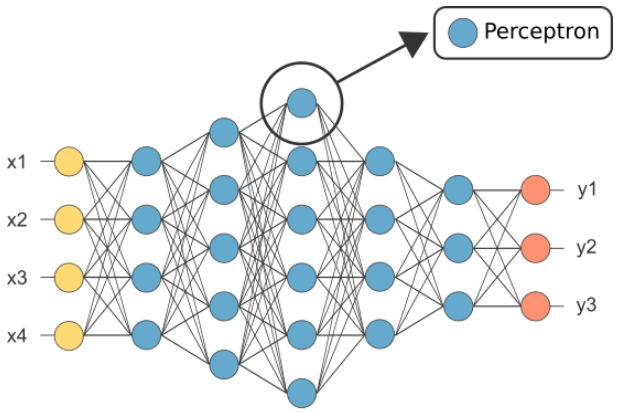


Figure 2.1: *(Source Medium)*

A multi-layer perceptron network consists of multiple layers of computational units, interconnected in a feed-forward way. Each neuron in one layer is connected to the neurons of the subsequent layer.

Even a FFNN has a very strong theoretical base. As stated by the Universal Approximation Theorem, even a FFNN with a single hidden layer can approximate arbitrarily closely any continuous function that maps intervals of real numbers to some output interval of real numbers.

2.1.2 Activation Functions

An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network. If a neuron has n inputs $x_1 ... x_n$ then the output of a neuron is

$$y = g(w_1 x_1 + ... w_n x_n + b) \tag{2.1}$$

where g is an activation function. Sometimes the activation function is called a "transfer function" [31]. If the output range of the activation function is limited, then it may be called a "squashing function." Many activation functions are nonlinear and may be referred to as the "nonlinearity" in the layer or the network design.

Sigmoid

The sigmoid function is differentiable, defined for all real input values and has a non-negative derivative at each point. It takes a real number and outputs a real number bounded in the range [0, 1]. It is defined as:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \tag{2.2}$$

In practice, the sigmoid is rarely used nowadays because of two drawbacks. Sigmoid, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small. Gradients of neural networks are found using an algorithm called backpropagation 2.2.2. Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one. Using the chain rule, the derivatives of each layer are multiplied down the network to compute the derivatives of the initial layers.

However, when many stacked hidden layers use an activation like the sigmoid function, small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers. This is known as the Vanishing Gradient Problem.

RELU

Over the last years, many different activation functions have been proposed, most of which inspired by the success obtained by ReLU [24], and therefore based on a similar shape, with small variations, compared to the original function. This kind of activation functions is the now standard in current neural network architectures, overcoming other such classic functions as sigmoid used in the past.

This function, defined as:

$$ReLU(a) = max(0, a) \tag{2.3}$$

The most significant positive features of this function is that: (1) It mitigates the vanishing gradient problem being not bounded in at least one direction. (2) It facilitates sparse coding as the percentage of neurons that are really active at the same time is usually low. The benefits of sparsity as described in [24] and can be resumed in a better dimensionality of the representation and a more significant invariance to small changes of the input data.

GELU

As mentioned above, activations such as ReLU enable network become faster and better convergence than that in Sigmoid. However, these cannot cover Dropout regularization (discussed in Section 2.3.2) that randomly multiplies a few activation functions by 0 at certain nodes in layers. With inspiration from an idea of fusing an activation function with the regularization capabilities of Dropout, GELU activation function is developed [33]. Gelu is defined as:

$$GELU(x) = xP(X < x) = x\Phi(x) \tag{2.4}$$

where $\Phi(x)$ the standard Gaussian cumulative distribution function. The GELU nonlinearity weights inputs by their percentile, rather than gates inputs by their sign as in RELU [33]. Consequently the GELU can be thought of as a smoother ReLU. GELUs

have been very successfully and are used in many state of the art architectures including GPT-3 [7] and Bert[15].

2.2 Training

Artificial Neural Networks are trained by an optimization method, that aims to select a set of model parameters that minimizes the prediction error. Thus the error or how well the model fits to the training data has to be quantified. The the models parameters are optimized to minimize this quantified error. In this section we present an overview of standard optimization and training stradegies.

2.2.1 Loss Functions

In the context of an optimization algorithm, the function used to evaluate a candidate solution (i.e. a set of model parameters) is referred to as the loss function. The cost or loss function has an important job in that it must faithfully distill all aspects of the model down into a single number i.e. *the loss*, in such a way that improvements in that number are a sign of a better model. Thus if we denote a model as f with parameters ∂, a data sample x_i and its coressponding label y_i in the trainig set $\mathcal{D}$ and a loss function $\mathcal{L}$ the loss is denoted as $\mathcal{L}(\partial; f(x_i, y_i))$. The total loss over our training set is denoted as:

$$J(\partial) = \frac{1}{N} \sum_{i=1}^{D} \mathcal{L}(\partial; f(x_i, y_i)) \tag{2.5}$$

where D is the cardinality of $\mathcal{D}$.

The goal is now to determine the optimal parameters $\hat{\partial}$ that minimize the total loss $J(\partial)$:

$$\hat{\partial} = argmin_{\partial} J(\partial) \tag{2.6}$$

Cross-Entropy Loss

For binary classification i.e. for models that predict an output with probability $p \in [0, 1]$ the standard loss function is binary cross-entropy defined as:

$$J(\partial) = \frac{1}{N} \sum_{i \in \mathcal{D}} \mathcal{L}_{\mathcal{BCE}}(\partial; f(x_i, y_i)) \tag{2.7}$$

where $\mathcal{L}_{\mathcal{BCE}}(\partial; f(x_i, y_i)) = -\sum_{i=1}^{2} y_i \log(\hat{y}_i)$ with $\hat{y}_i = f(\partial; x_i)$

The general formula for multi-label classification is calculating the loss among two probability vectors p and q i.e. the vector of the predicted (empirical distribution) and of the ground truth distribution. The cross-entropy loss, for C classes, is given as:

$$\mathcal{L}_{\mathcal{CE}}(\partial; f(x_i, y_i)) = -\sum_{i=1}^{C} y_{i,c} \log(\hat{y_{i,c}}) \tag{2.8}$$

2.2.2 Backpropagation

Backpropagation is the backbone used when training a neural network. Backpropagation computes the gradient of the loss function with respect to the weights (the parameters) of the network for a single sample in the training set. Using the chain rule, it computes the gradients one layer at a time. To avoid redundant calculations of intermediate terms backpropagation starts from the last layer caching the intermediate, as in a dynamic programming algorithm.

2.2.3 Optimization and Gradient Descent

Gradient descent is a way to minimize a loss function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ with respect to the parameters. The learning rate η determines the size of the steps the algorithm takes to reach a (local) minimum. In other words, the model follows the direction of the slope of the surface created by the objective function downhill until it reaches a valley as depicted in Figure 2.2.

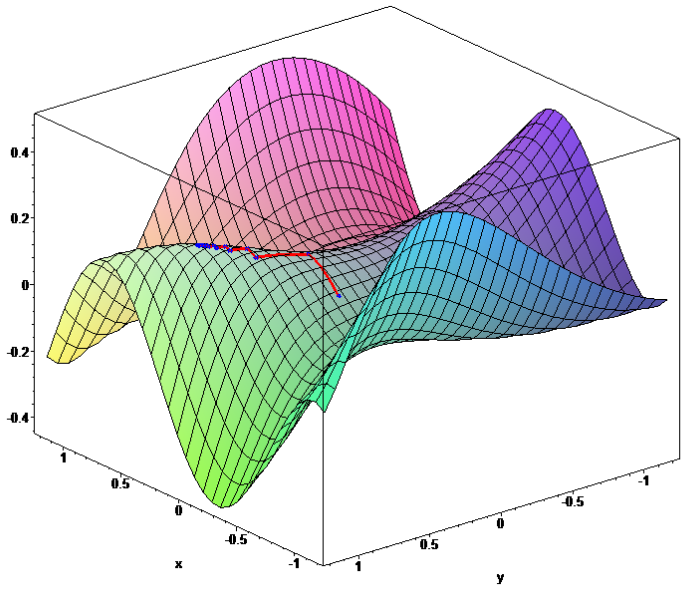


Figure 2.2: *(Source Wikipedia)*

There are three variants of gradient descent, which differ on the amount of data used to compute the gradient of the loss function. Depending on the amount of data, there is a trade-off between the accuracy of the update and the time it takes to perform it.

Batch gradient descent

Batch gradient descent, computes the gradient of the cost function with respect to the parameters θ for the entire training dataset:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.9}$$

While the gradient descent algorithm is very simple to implement, it is not guaranteed to converge at a global minimum. It is possible to get stuck to a local minimum if the learning rate is not properly chosen. Moreover, when training on large datasets, computing at each iteration the loss over the entire dataset may be computationally expensive and inefficient.

Stochastic Gradient Descent (SGD)

The stochastic gradient descent algorithm [6] is a variant of gradient descent. In contrast to gradient descent it computes the gradient of the loss function over a subset of samples, and not for the whole dataset. Thus, SGD makes an estimation of the gradient using a sample, instead of computing the true gradient using all samples. For each training example x_i and its corresponding label y_i a parameter ubdate is performed as:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x_i, y_i) \tag{2.10}$$

While batch gradient descent converges to the minimum of the basin that the parameters are placed in, SGD enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD may keep overshooting. However, it has been shown that when learning rate is slowly decreased, SGD shows the same convergence behaviour as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively [82].

Mini-batch Gradient Descent

Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x_{i:n}, y_{i:n}) \tag{2.11}$$

where the mini-batch is denoted as $x_{i:n}, y_{i:n}$. This way, it reduces the variance of the parameter updates, which can lead to more stable convergence. It also makes use of highly optimized matrix optimizations deep learning libraries that make the computation of the gradient with respect to a mini-batch very efficient.

Learning Rate

The size of the steps the optimization algorithm takes plays very important role in training the model hyperparameter and effectively controls the speed at which the model learns. If the step or learning rate is too large, we may overshoot the minima and bounce back and forth on the loss surface or the model can become unstable and diverge. If it is too small, it may take too long to reach a minimum, or even worse the algorithm might get stuck at a suboptimal local minimum. Finding an optimal learning rate is difficult and many times in practice it comes down to experimenting with the models hyperparameters.

Specifying a learning rate for each epoch i.e. scheduling can mitigate to some degree

these problems. There are two types of methods for scheduling global learning rates: the **decay**, and the **cyclical** one. The most preferred method is the learning rate annealing that is scheduled to gradually decay the learning rate during the training process. A relatively large step-size is preferred at the initial stages of training in order to obtain a better generalization effect [56]. Cyclical method [26], in which a learning rate period that consists of an upper and a lower bound is repeated during epochs. The observation behind the cyclic method is that increasing the learning rate in the optimization process may have a negative effect, but can result in a better generalization of the trained model.

Learning rate warmup, is a recent approach that uses a relatively small step size at the beginning of the training. Then the learning rate is increased linearly or non-linearly to a specific value in the first few epochs, and then gradually shrinks to zero. The observations behind warmup are that: the model parameters are initialized using a random distribution, and thus, the initial model is far from the ideal one; thus, an large learning rate causes the model to recede from a local minimus. Also training a initial model carefully in the first few epochs may enable the application of a larger learning rate in the middle stage of the training, resulting in a better regularization.

2.3 Data Augmentation and Regularization Strategies

When training data are not sufficient to train large and complex models, it is possible to increase the effective size of existing dataset through the process of data augmentation, which has contributed to significantly improving the performance of deep networks in many domains.

If the distribution to be learned has transformation invariance properties generating addition data with with transformations can help the over all performance and generalization of the model [84]. For example in image tasks, traditional augmentation techniques include horizontal image flips, cropping, translations, and rotation. Recently, more complex augmentation techniques such as elastic distortion, tilting, and theme adjustments have been developed. The effectiveness of the newly developed techniques in preventing overfitting and improving accuracy [73].

Another challeng of contemporary machine learining is overfitting. Overfitting occures when a model fails to capture the underlying structure of the data being unable to perform on unseen samples. Overfitting, can also be mitigated with Regularization strategies. Any arrangement in the model architecture, learning process or inference that is used to compensate overfitting and the shortage of training data is called regularization [68].

2.3.1 Weight Decay

Weigth Decay [50], one of the most original and trivial methods of regularization is to constrain the capacity of the model by adding some penalty function to the original objective function $\mathcal{L}$ making a new objective function $\mathcal{L}_d$:

$$\mathcal{L}_d(\partial; x, y) = \mathcal{L}(\partial; x, y) + a\Omega(\partial) \tag{2.12}$$

$\Omega(\partial)$ is the regularization term that is controlled by the parameter a. The regularizer, Ω, here consists of the norms of the model trainable parameters. To simplify ∂ is assumed to be the network weights.

$$\mathcal{L}_d(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\| \tag{2.13}$$

For $\lambda = 0$, the original loss function is recovered. For $\lambda > 0$, we restrict the size $\|\mathbf{w}\|$. When we the derivative of a quadratic function is calculated, the 2 and 1/2 cancel out, ensuring that the expression for the update looks nice and simple this is why we $\frac{\lambda}{2}$ is used. The reason the squared norm is used and not the standard norm (i.e., the Euclidean distance) is for computational convenience. By squaring the L_2 norm, the square root is removed, leaving the sum of squares of each component of the weight vector. This makes the derivative of the penalty easy to compute: the sum of derivatives equals the derivative of the sum.

2.3.2 Dropout

Deep neural nets with lots of learnable parameters can be very powerful. However, as discussed above overfitting is a severe problem in such networks. Training large neural networks is also an expensive process, making it difficult to deal with overfitting by combining the predictions of multiple trained neural nets at test time. Dropout [35] is a technique for addressing this problem. The key idea is that at each training stage, individual nodes are either kept with probability p or dropped out of the net with probability 1-p. In this case, incoming and outgoing edges to a dropped-out node are also removed. In this way, a large model that overfits easily is considered as the base model, and repeatedly smaller sub-models are sampled and trained. Since all the sub-models share their parameters with the base model, this process trains the large model simultaneously, which is ultimately used at test time.

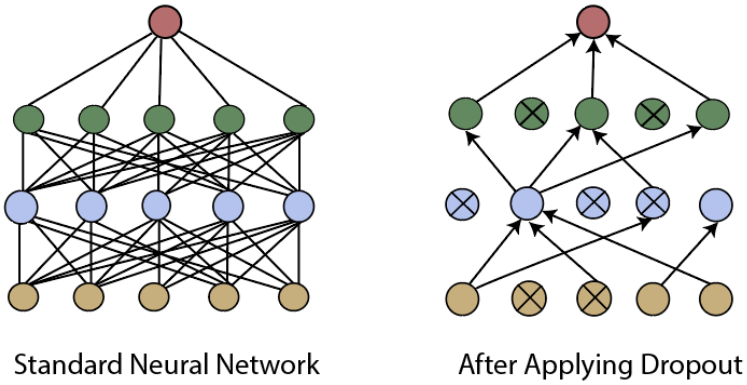


Figure 2.3: *(Source Dropout)*

2.3.3 Label Smoothing

Takeuchi et al. [87] introduced a mechanism called Label Smoothing Regularization (LSR) in order to encourage the model to be less confident. LSR regularizes the model

and makes it more adaptable. Assuming a model that uses a cross-entropy cost function, LSR is implemented in the followng way: For the i-th training example x^i, the model computes the probability of each label $k \in 1, ..., K$. Such a model aims to minimize the Cross-Entropy cost function:

$$CE(p, q|x^i) = -\sum_{k=1}^{K} log(p(k)|x^i)q(k|x^i) \tag{2.14}$$

considering a distribution over labels u(k), independent of any training example x^i, and a smoothing parameter e. For a training example with ground-truth label y, the label distribution $q(k|x^i) = \delta_{k,y}$ is replaced with:

$$q^s(k|x^i) = (1 - e)\delta_{k,y} + eu(k) \tag{2.15}$$

This can be seen as the distribution of the label k obtained as a result of doing the following experiment: first, set it to the ground-truth label k=y; then, with probability e, replace k with a sample drawn from the distribution u(k). As a simple choice, the uniform distribution u(k)=1/K can be used, so that:

$$q^s(k|x^i) = (1 - e)\delta_{k,y} + \frac{e}{K} \tag{2.16}$$

This change in ground-truth label distribution is label-smoothing regularization.

2.3.4 Normalization

Training a deep neural network is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. The problem that arises is called internal covariate shift and as a result it requires lower learning rates, making training slower, and careful parameter initialization, making it harder to train models with saturating non-linearities. A way to address this problem is to normalize layer inputs. [38]

Batch Normalization

In [38], normalization is designed as part of the model architecture and is performed for each training mini-batch. The goal of batch normalization is to achieve a stable distribution of activation values throughout training, and it is applied before the nonlinearity, since this is where matching the first and the second moments is more likely to result in a stable distribution. Batch normalization is employed so the resulting networks can be trained with saturating nonlinearities without further regularization arrangements. In this way, the network is more tolerant of increased training rates that yield a substantial speedup in training.

At first, mini-batch mean and variance are computed, respectively. Then, all members of the batch are normalized using mini-batch mean and variance. The parameter ϵ is a small value and is added for numerical stability. Note that simply normalizing each input

µ 2.1: *Batch Normalization*

Input values of x in a mini-batch $\mathcal{B} = \{x^1, ..., x^m\}$
Output $\{y^1, ...y^m\}$ = **BN**$\{x^1, ..., x^m\}$
Learnable parameters: γ, β
$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x^i$
$\sigma_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m}(x^1 - \mu_{\mathcal{B}})^2$
for i = 1...m **do**
$\hat{x}^i = \frac{x^i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}} + \epsilon}}$
$\hat{y}^i = \gamma\hat{x}^i + \beta$
end for

of a layer may change what the layer can represent. To address this, a transformation is added that is able to represent the identity mapping. For each layer input, a scaling γ and a shifting β parameter is defined and learned along with other learnable parameters of the network ϵ.

Layer Normalization

Layer normalization [2] normalizes the inputs across the features, while batch normalization normalizes the input features across the batch dimension. In batch normalization, the statistics are computed across the batch and are the same for each example in the batch. In contrast, in layer normalization, the statistics are computed across each feature and are independent of other examples. This means that layer normalization is not a simple re-parameterization of the network. Also, unlike batch normalization, the layer normalization reduces the training time by computing the mean and variance used for normalization from all of the summed inputs to the neurons in a layer on a single training case. Layer normalization is widely used, since it stabilizes the hidden state dynamics in recurrent networks, reduces the training time and generalizes well.

2.3.5 SpecAugment

SpecAugment is a simple yet effective augmentation strategy that was first introduced for Automatic Speech Recognition (ASR) and is know wildly used in many audio tasks. Augmentation of audio training data is normally applied to the waveform audio before it is converted into the spectrogram, such that after every iteration, new spectrograms must be generated. SpecAugment's approach is augmenting the spectrogram itself, rather than the waveform data. Since the augmentation is applied directly to the input features of the network, it can be run online during training without significantly impacting training speed.

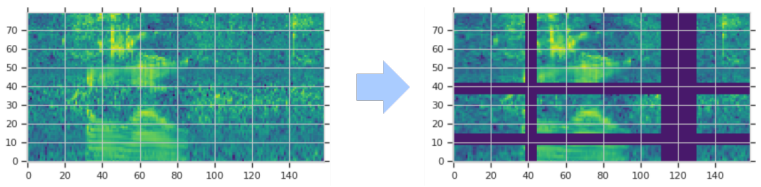


Figure 2.4: *(The masked portion of the spectrogram is displayed in purple for emphasis. SpecAugment)*

SpecAugment modifies the spectrogram by (1) warping it in the time direction, (2) masking blocks of consecutive frequency channels, and (3) masking blocks of utterances in time. These augmentations have been chosen to help the network to be robust against deformations in the time direction, partial loss of frequency information and partial loss of small segments of speech of the input [76].

2.3.6 Mixup

Mixup [104] is a also a simple but powerful data augmentation / regularization technique. In essence, mixup trains a neural network on convex combinations of pairs of examples and their labels by constructing virtual training examples:

$$x_{mix} = \lambda x_i + (1 - \lambda)x_j. \tag{2.17}$$

$$y_{mix} = \lambda y_i + (1 - \lambda)y_j. \tag{2.18}$$

where x_i, x_j are raw input vectors and y_i, y_j are one-hot label encodings. (x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$ Therefore, Mixup extends the training distribution by incorporating the prior knowledge, that linear interpolations of feature vectors should lead to linear interpolations of the associated targets [104]. Here is how the output of the MixUp augmentation can look like when implemented on images:

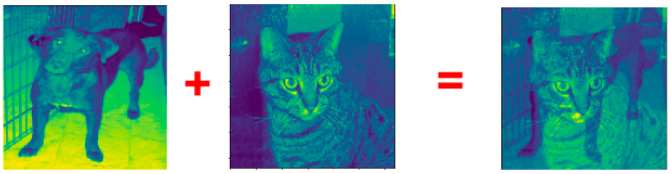


Figure 2.5: *Blended together dog and cat images with* $\lambda = 0.3$ *Source: Mixup*

2.4 Today's Deep Learning

In this section, we briefly discuss common architectural blocks in state-of-the-art models for Audio Signal Processing and Natural Language Processing. We will discuss the Transformer model in a different section since it is the backbone of our proposed architecture for Audio Captioning.

2.4.1 Convolutional Neural Networks (CNN)

In Feed Forward Networks each neuron in one layer is connected to all neurons in the next layer. The strong connectivity of FFNNs has two downsides: (i) they are prone to overfitting, (ii) they are computationally intensive. In 2012, Alex et al. [49] achieved the best classification result at that time using deep CNN in the ImageNet Large Scale Visual Recognition Challenge (LSVRC), which attracted researchers' attention and greatly promoted the development of modern CNNs.

A Convolutional Neural Network [51] is a kind of feedforward neural network that is able to extract features from data with convolution structures. Different from the traditional feature extraction methods, CNN does not need to extract features manually. For example in traditional Computer Vision high and low level features such as blobs, corners, etc, are extracted and then feed into a neural network. A deep CNN will learn these feature extractors as it trains. Compared with FFNN, CNN possesses many advantages:

- Local connections. Each neuron is no longer connected to all neurons of the previous layer, but only to a small number of neurons, which is effective in reducing parameters and speeding up convergence.
- Weight sharing. A group of connections can share the same weights, which reduces parameters further.
- Down-sampling dimensionality reduction.

In order to build a CNN model, four components are typically needed. (1) A pivotal step for the convolution kernel. (2) Setting a convolution kernel with a certain size, will result to information loss in the border regions. Hence, padding is introduced. (3) For the sake of controlling the density of convolving, stride is employed. (4) After convolution, feature maps consist of a large number of features that is prone to overfitting. As a result, pooling (i.e. down-sampling) is proposed to obviate redundancy.

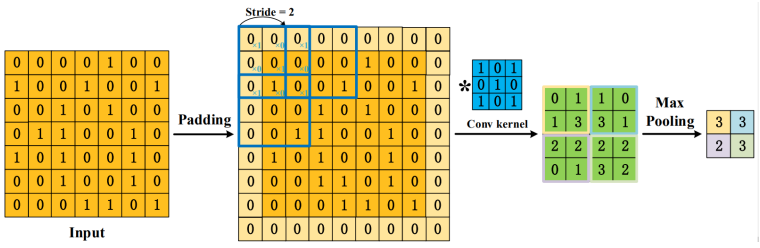


Figure 2.6: *Procedure of a two-dimensional CNN [57]*

2.4.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a common solution for modeling time-series sequences such as speech signals, language texts, video frames, weather data, and so on. While Convolutional Neural Networks (CNNs) are most suitable to capture spatial information, they lack the ability to capture the temporal dimension.

Formulated mathematicly at time step t, input features x_t and previous time hidden features h_{t-1} are learned by corresponding weight matrices W_{hx} and U_{hh} with f_a, f_b nonlinear function mapping. The training procedure maps the temporal complex relationships that exist between the input features x_t and the output label y_t associated with weight matrix W_{yh}.

$$h_t = f_a[U_{hh}ht - 1 + W_{hx}x_t + b_h] \tag{2.19}$$

$$y_t = f_b[W_{yh}h_t + b_y] \tag{2.20}$$

The capability of RNNs to handle context makes them very theoretically appealing. However, in practice RNNs fail to perform well for large contexts.

2.4.3 Sequnce to Sequce Modeling

Sequence to Sequence (often abbreviated to seq2seq) models is a special class of neural network architectures that is typically used to solve NLP tasks like Machine Translation, Question Answering, Text Summarization, etc. It is the standard methodology when the task in hand when dealing with sequential input. Seq2seq is based on an Encoder-Decoder framework. The encoder encodes the input sequence into an abstract representation, while the decoder takes this representation as input to generate the final output.

2.4.4 The Attention Mechanism

There are two well known challenges with this traditional encoder-decoder framework. As discused above, the encoder has to compress all the input information into a single fixed length vector that is passed to the decoder. Using a single fixed length vector to compress long and detailed input sequences may lead to loss of information[11]. Intuitively, in sequence-to-sequence tasks, each output token is expected to be more influenced by some specific parts of the input sequence. However,the decoder lacks any mechanism to selectively focus on relevant input tokens while generating each output token. Thus the attention mechanism aims to mitigate these problems by allowing the decoder to attend to the entire input sequence. The central idea is to introduce attention weights a over the input sequence to prioritize the set of positions where relevant information is present for generating the next output token. A seq-to-seq task with an encoder decoder architecture with and without an attention block:

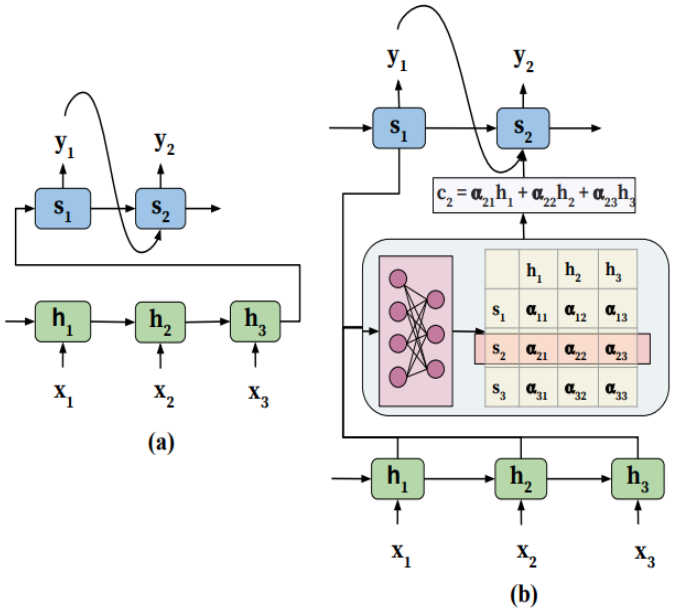


Figure 2.7: *Encoder-decoder architecture: (a) traditional (b) with attention model Source: [28]*

The attention block in the architecture is responsible for automatically learning the attention weights a_{ij} , which capture the relevance between h_i (the encoder hidden state) and s_{j-1} (the decoder hidden state). Note that the query state s_{j-1} is hidden state of the decoder just before emitting s_j and y_j . These attention weights are then used for building a context vector c, which is passed as an input to the decoder. At each decoding position j, the context vector c_j is a weighted sum of all hidden states of the encoder and their corresponding attention weights, i.e.

$$c_j \sum_{i=1}^{T} a_{ij} h_i \tag{2.21}$$

In the traditional framework c = h_T i.e. the last hidden layer of the decoder whereas in the attention framework c is infused with a combination of the encoder hidden states. Note that the attention weights a are the parameters of a learnable feed forward layer. This function is called the **alignment function** as it scores how relevant is the encoder hidden state h_i for the decoder hidden state s_{j-1}.

Generalized Attention Model

The attention model shown in Figure 2.7 can also be seen as a mapping of sequence of keys K to an attention distribution a according to query q where keys are encoder hidden states h_i and query is the single decoder hidden state s_{j-1}. Here the attention distribution a_{ij} emphasizes the keys which are relevant for the main task with respect to the query q. Then e = a(K, q) and a = p(e). Hence a generalized attention model A works with a set of

key-value pairs (K,V) and query q such that:

$$A(q, K, V) = \sum_i p(a(k_i, q) \cdot v_i \tag{2.22}$$

The alignment function (denoted by a) and distribution function (denoted by p) determine how keys and query are combined to produce attention weights. We discuss some of the commonly used alignment functions and distribution functions in the literature and present them in Table 2.1

Alignment Function	**Equation**	**References**
similarity	$a(k_i) = sim(k_i q)$	[27]
dot product	$a(k_i) = q^T k_i$	[1]
scaled dot product	$a(k_i) = fracq^T k_i \sqrt{d}$	[92]
general	$a(k_i) = q^T W k_i$	[1]
activated general	$a(k_i) = act(q^T W k_i + b)$	[58]
generalized kernel	$a(k_i) = \varphi(kq^T)\varphi(K_i)$	[12]
location-based	$a(k_i) = a(q)$	[1]

Table 2.1: *Different alignment functions*

The function that maps the alignment function scores to attention weights is called **Distribution function**. The most commonly used distribution functions are **logistic**, **sigmoid** and **softmax**. These functions ensure that attention weights are constrained in [0,1] and sum to 1. Such weights can thus be interpreted as probabilities that an element is relevant and the model attends to it more to generate the output compared to other elements. We have discussed some of those functions in a previous section 2.1.2.

2.5 Transformer

Recurrent architectures rely on sequential processing of input at the encoding step that results in computational inefficiency, as the processing cannot be parallelized [92]. The Transformer relies only on self attention mechanism to capture global dependencies between input and output. Authors demonstrated that Transformer architecture achieved significant parallel processing, shorter training time and higher accuracy for Machine Translation without any recurrent component [28].

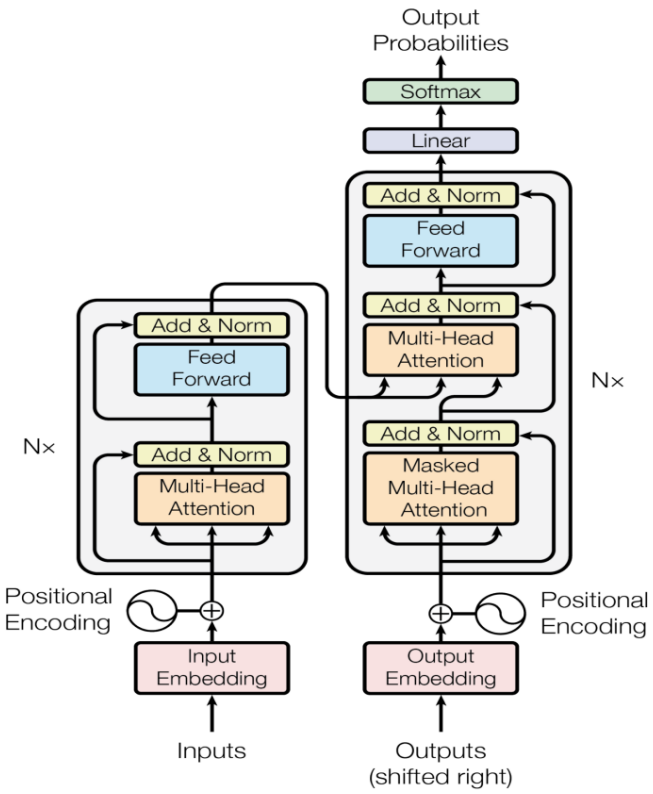


Figure 2.8: *Architecture of the Transformer model. Source [92]*

The Transformer architecture is shown in Figure 2.8. Authors propose a novel scaled dot product alignment function for self attention mechanism shown in Table 2.1. The encoder and the decoder consist of N stacked layers, where the output of each layer is the input to the next. The layers are identical, with different weights. Each encoder layer consists of a self-attention and a fully connected feedforward sublayer interleaved by a residual connection and layer normalization. The output of the top encoder is then transformed into a set of attention vectors K and V and feed to each decoder layer to a Cross-Attention module which helps the decoder focus on appropriate places in the input sequence. The second difference between the encoder and the decoder is that the decoders self attention is only allowed to attend to earlier positions in the output sequence. This is done by masking future positions before the softmax step in the self-attention calculation. The idea behind this masking is that while decoding a sequence, for example in MT, the model must only be aware and attend to previously decoded words. Thus in the framework proposed predicting the current token doesn't depend in feature tokens.

Scaled Dot-Product Attention

For each input $\mathbf{x}$ the self-attentions mechanism creates a Query vector Q, a Key vector K, and a Value vector V. These vectors are created by multiplying the input embedding by three learnable matrices $W_Q, W_K, and W_V$. In the case of cross-attention the Q vector is created by multiplying the output of the encoder z with W_Q. Secondly the dot product of the Key vector and the Query vector Q are scaled and passed trough a softmax function.

The scaling factor $\sqrt{d_k}$ avoids too large inputs to the softmax, which lead to extremely small gradients. The output of the softmax function is then multiplied to the Value vector:

$$Attention(Q, V, K) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{2.23}$$

The same procedure is shown in the figure below:

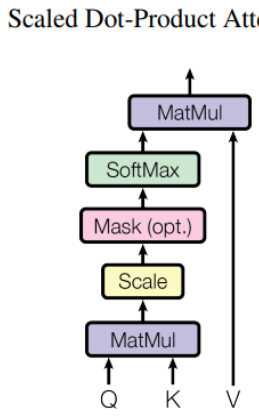


Figure 2.9: *Source [92]*

2.5.1 Multi-Head Attention

The Multi-head attention mechanism extends the scaled-dot-product attention mechanism. Instead of performing a single attention function with keys, values and queries, it linearly projects the queries, keys and values h times, with different learned linear projections. In this way the models ability to focus on different positions is expanded. The multi-head attention also gives the attention layer multiple "representation subspaces"[92]. Given the weight matrices $W_i^Q \in \mathbb{R}^{d\times d_q}, W_i^K \in \mathbb{R}^{d\times d_k}, W_i^V \in \mathbb{R}^{d\times d_v} and W_i^O \in \mathbb{R}^{d\times d_v}$ where d is the dimension of the models hidden layer d_k, d_q, d_v are learnable linear projections the multi-head attention is denoted as:

$$MultiHead = Concat(head_1, ..., head_h)W^O \tag{2.24}$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.25}$$

Or as shown in the figure below:

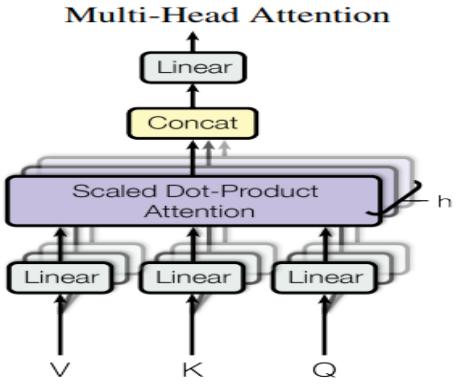


Figure 2.10: *Source [92]*

2.5.2 Positional Embeddings

Position and order of words are the essential parts of any language. They define the grammar and thus the actual semantics of a sentence. Even in other fields such us audio, temporal information is crucial. Recurrent Neural Networks (RNNs) inherently take the order of word into account; They parse a sentence word by word in a sequential manner. This will integrate the words' order in the backbone of RNNs. But the Transformer architecture ditched the recurrence mechanism in favor of multi-head self-attention mechanism. Avoiding the RNNs' method of recurrence will result in massive speed-up in the training time. And theoretically, it can capture longer dependencies in a sequence of tokens. The model itself doesn't have any sense of position/order for each word. Without the positional encoddings the input sequence of tokens would just be like a *bag of tokens*.

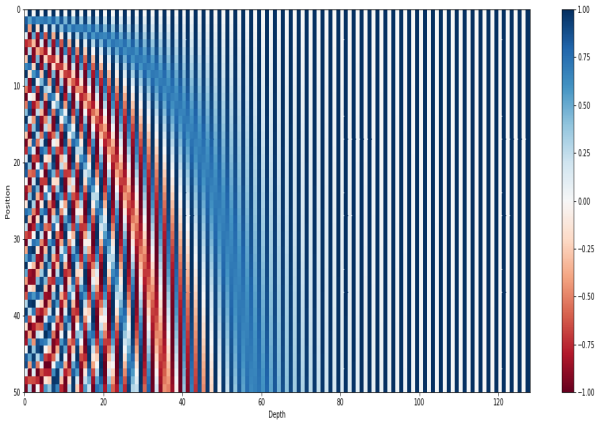


Figure 2.11: *The 128-dimensional positional encoding for a sentence with the maximum length of 50. Each row represents the embedding vector p_t*

The positional encoding proposed in [92] mitigate this issue by adding a vector to each input sequencing which in a way imprints positional information in the input token's representation. Let t be the desired position in an input sentence/sequence $p_t \in \mathbb{R}^d$ be its

corresponding encoding, and d be the dimension of the encoding. Then $f : \mathbb{N} \rightarrow \mathbb{R}^d$ will be the function that produces the output vector p_t and it is defined as follows:

$$p_t^i = f(t)^i = \begin{cases} sin(w_k t), & \text{if} i = 2k \\ cos(w_k t), & \text{if} i = 2k + 1 \end{cases}$$

where $w_k = 10000^{-2k/d}$. As it can be derived from the function definition, the frequencies are decreasing along the vector dimension. Thus it forms a geometric progression from 2π to $10000 \cdot 2\pi$ on the wavelengths. One can also imagine the positional embedding p_t as a vector containing pairs of sines and cosines for each frequency:

$$p_t = \begin{bmatrix} sin(w_k t) \\ cos(w_k t) \\ \vdots \\ sin(w_{d/2} t) \\ cos(w_{d/2} t) \end{bmatrix} \tag{2.26}$$

2.6 Transformer Models for Vision and Audio

Convolutional neural networks (CNNs) [51] have been widely used to learn representations from raw spectrograms for end-to-end modeling, as the inductive biases inherent to CNNs such as spatial locality and translation equivariance are believed to be helpful. But recent works in vision tasks have showed that purely attention-based models outperform the CNN-based approaches [16] [90] [103]. Recently Audio Spectrogram Transformer (AST)[25] was proposed, a convolution-free transformer based model for Audio Event Detection and Tagging, trained upon the Vision Transformer (ViT)[16] model for audio classification.

2.6.1 Vision Transoformer - ViT

We briefly discuss Vision Transormer's methodology and architecture as it is the backbone model for both AST and PaSST models 2.7. Vision Transformer (ViT) achieves remarkable results compared to convolutional neural networks (CNN) while obtaining fewer computational resources for pre-training. In comparison to convolutional neural networks (CNN), Vision Transformer (ViT) show a generally weaker inductive bias resulting in increased reliance on model regularization or data augmentation when training on smaller datasets. An overview of the methods pipeline can be the following:

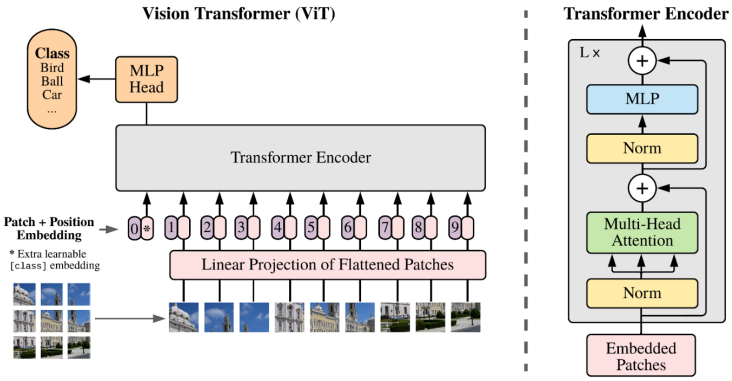


Figure 2.12: *Model overview*

1. Split an image into patches.
2. Flatten the patches.
3. Produce lower-dimensional linear embeddings from the flattened patches.
4. Add positional embeddings.
5. Feed the sequence as an input to a standard transformer encoder.

We will discuss 1,2 and 3 in the following section. Many positional embedding schemes were investigated in ViT: (1) no positional information (i.e. *bag of patches*), (2) 1-dimensional positional embeddings, (2) 2-dimensional positional embeddings and relative positional embeddings i.e. considering the relative distance between patches. The authors claim that while adding position embeddings was crucial to the models performance the differences between the different embedding schemes were minor.

2.6.2 Audio Spectrogram Transformer

AST proposes a ViT-like architecture that views spectrograms as single channel images. ViT is not only the backbone architecture for AST, the model is trained upon an ImageNet [14] pretrained ViT model slightly modified. Since the available data on audio are less than those for vision tasks [25], AST applies a cross-modality transfer learning scheme, since images and audio spectrograms have similar formats. It is hard to interprete how pretraining a model on ImageNet improves its performance on audio data. An explanation might be that even for human level interpretation the audio events in a spectrogram can be identified just by looking at the raw spectrogram. For example phonologists and other experts having specialized knowledge of how phonemes are imprinted in the spectrogram domain can "read" spectrograms of recorded speech[1]. Transfer learning from vision tasks to audio tasks has been previously studied in [30], [29], [74], but only for CNN-based models, where ImageNet-pretrained CNN weights are used as initial CNN weights for audio classification training.

[1]https://home.cc.umanitoba.ca/ robh/howto.html

While ViT and AST have similar architectures (e.g., both use a standard Transformer, same patch size, same embedding size), they are not same. First, the input of ViT is a 3-channel image while the input to the AST is a single-channel spectrogram, thus the weights corresponding to each of the three input channels of the ViT patch embedding layer are used as the weights of the AST patch embedding layer [25]. Secondly because the patches extracted in ViT are 24×24 and the ones extracts from a spectrogram are 12×100 to be able to use the pretrained position embedding the width positional embedding are interpolated and the height positional embeddings are pruned.

AST outperforms state-of-the-art systems on AudioSet [23] and other datasets. Also, AST naturally supports variable-length inputs and can be applied to different tasks without any change of architecture.

2.7 PaSST

The main drawback of previous related methods such as AST [25] is that the patch extraction results in a large sequence of input tokens. As we have discussed abovem, the transformer's quadratic attention bottleneck results to models with high computational complexity. Reducing the quadratic complexity has been the target of several approaches in natural language processing, the idea being to restrict each input token to attend only to a pre-selected subset of input tokens [95][86][41] . PaSST introduces a simple yet effective time complexity reduction for training a transformer model on spectrograms called Patchout.

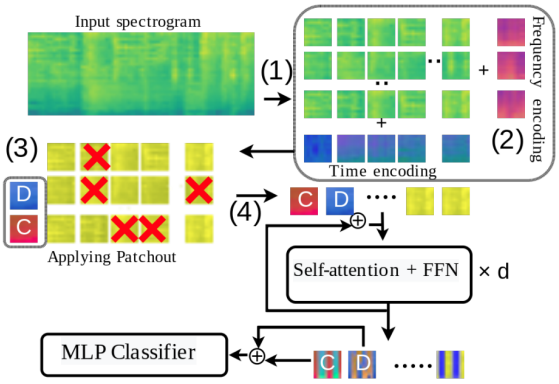


Figure 2.13: *The The Patchout faSt Spectrogram Transformer (PaSST) architecture*

2.7.1 Complexity Analysis

Multi-head attention layers [92] rely on computing a distance between each pair of positions in the input sequence (in the form of an attention matrix), therefore having a complexity of $O(n^2)$ where n is the input sequence length.

The output of one head can be summarized as follows:

$$Attention(Q, K, V) = softmax\frac{QK^T}{\sqrt{d}}V \tag{2.27}$$

$$SelfAttention = Attention(XW_Q, XW_K XW_V) \tag{2.28}$$

Here $Q \in \mathbb{R}^{n\times d}$ and $K \in \mathbb{R}^{n\times d}$ hence the matrix product QK^T has a cost $O(d \times n^2)$ which is quadratic in the length of the input sequence n. The rest of the operations in the network have a linear complexity relationship with n [92] [16]. In short, reducing the sequence length would have a large impact on the computational complexity of these models.

2.7.2 Patchout

The main idea in *Patchout* is to drop some of the extracted patches and encourage the transformer to make a prediction with an incomplete sequence. Similar approaches have been adopted in vision tasks, in [69] the authors investigate vision-transformer's robustness against input perturbations by removing some of the patches in inference. In PaSST the patches are removed from the training input sequence whereas during inference the whole input sequence is passed to the Transformer. The authors investigate two patchout methodologies:

Unstructured Patchout

Unstructured Patchout is the basic form of Patchout, where a number of patches are randomly excluded from the patch grid, regardless of their position.

Structured Patchout

In Structured Patchout a number of frequency bins and time frames are selected and the whole column/row of extracted patches is removed. The authors mention that this method is inspired by SpecAugment [76].

2.7.3 An in depth look on the model architecture

In this subsection we aim to present a detailed explanation of the model's architecture, and discuss its similarities and dissimilarities with ViT [16] and AST [25].

Patch extraction and linear projection

The patch extraction and the linear projection in PaSST are the same with ViT. The input spectrogram $x \in \mathbb{R}^{H\times W}$ is reshaped into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N\times P^2}$ where H and W are the number of frequency bins and time frames and (P, P) is the resolution of each patch and N = H/P^2 is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. A similar but not convolution free approach is to pass the input spectrogram as a single channel image through a 2D Convolutional layer with output layer size equal to the hidden dimension

of the transformer model d, kernel size equal to P and the desired stride S. Thus an input spectrogram viewed as single channel image $x_{in} \in \mathbb{R}^{1 \times H_{in} \times W_{in}}$ is passed through the convolutional layer and produces the feature map: $x_p = Conv2D(x_{in})$ with $x_p \in \mathbb{R}^{d \times H_p \times W_p}$ and $H_p = \frac{H_{in}-P}{S} + 1$, $W_p = \frac{W_{in}-P}{S} + 1$. Finally x_p is flattened to $x_f \in \ \times \mathbb{L}$ where $L = H_p W_p$.

Frequency and time positional encoding

Apart from the patchout mechanism the other difference of PaSST from AST [25] is the way they address the need for positional encoding [48]. In PaSST the positional encoding for the time and frequency dimensions are disentangled, and as a result, we have two positional encodings: one representing the frequency, and one for time. Whereas AST follows a similar approach to ViT employing grid positional encoding. Before flattening the extracted feature map x_p two distinct trainable parameters $f_{pos} \in \mathbb{R}^{d \times H_p \times 1}$ and $t_{pos} \in \mathbb{R}^{d \times 1 \times W_p}$ are added resulting to $x_p^{pos} = X_p + f_{pos} + t_{pos}$.

After the positional encoding is added a classification and a distillation token are added as in [90] [25]. We discuss the classification and a distillation tokens and the position encoding aplied in AST and ViT in section 2.6.

2.7.4 Pretraining and depth reduction

PaSST is also pretrained on ImageNet [14]. As showed in [25] pre-trained models on Imagenet significantly improves their performance on Audioset [23]. To truncate the Vit and Deit models pretrained on ImageNet they remove every other attention bloch. This allows the model to benefit from the pretraining, compared to removing consecutive blocks, since the residual activations will have a less sudden change [48]. The final model has d = 7 self-attention blocks and 50M parameters compared to 87M in the full model.

Chapter 3

Automated Audio Captioning

In this Chapter we analyze the task of Automated Audio Captioning. We discuss the recent related work on the task and specially the work that mostly influenced our approach. We will present an error analysis on the results on a state-of-the-art model on Automated Audio Captioning. Based on our observations we will present challenges that emerge in language generation models regarding diversity and quality. We introduce our approach, the training environment and the hyperparameters. We present the standardized evaluation methodology for the task and evaluate our experiments.

3.1 Automated Audio Captioning

Automated audio captioning (AAC) is an intermodal translation task, where the system receives as an input an audio signal and outputs a textual description of the contents of the audio signal (i.e. outputs a caption). AAC is not speech-to-text, as the caption does not transcribe speech. In a nutshell, an AAC method learns to identify the high-level, humanly recognized information in the input audio, and expresses this information with text. [91] AAC methods can model concepts (e.g. "muffled sound"), physical properties of objects and environment (e.g. "the sound of a big car", "people talking in a small and empty room"), and high level knowledge ("a clock rings three times").

Audio captioning has practical potential for various applications such as helping the hearing-impaired to understand environmental sounds. In addition, audio captioning can be used for multimedia retrieval , which can be applied in areas including education, film production, and web searching [43]. Unlike image and video captioning, which have been widely studied for almost a decade, audio captioning is a relatively new task that has been studied since 2017 [17].

3.2 Related Work

In the past three years, this field has received increasing attention due to freely available datasets released and being held as a task in DCASE Challenges in 2020 and 2021 [1]. A number of papers have been published and the encoder-decoder 2.4.3 framework has been adopted as a standard recipe for solving this cross-modal translation task. Using a

[1] https://dcase.community/

Seq2seq setting, the encoder extracts audio features from the input audio clips, and the decoder generates captions based on the extracted audio features as shown in the figure below:

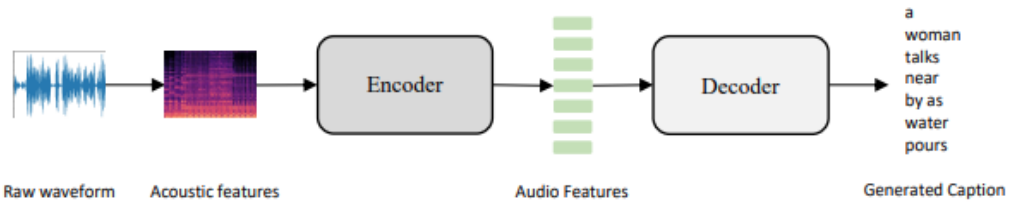


Figure 3.1: *Seq2Seq modeling for AAC. Source([64])*

In the recent literature Recurrent Neural Networks (RNNs) [83], Convolutional Neural Networks (CNNs) [51] and Transformers [92] have been proposed as audio encoders. For the decoder, RNNs and Transformers are usually employed, inspired by works in NLP.

3.2.1 Acoustic encoding

Analyzing the content of an audio clip largely depends on obtaining an effective feature representation for it, which is the aim of the encoder in an AAC system. Current popular approaches for acoustic encoding consist of two steps, first extracting acoustic features, and then passing them into an encoder to obtain compact audio features. In this section, we first discuss popular acoustic features used in literature, then audio encoding approaches, focusing on these based on deep neural networks.

Acoustic features

Time-frequency representations, such as spectrograms, are widely used as the acoustic features. The spectrogram is a 2-D representation whose horizontal axis is time and vertical axis is frequency, the value at each point of the spectrogram represents the energy at a specific time and frequency. Inspired by the selectivity of human auditory system to different frequencies, the frequency axis of a spectrogram may be converted to different scales, resulting in representations such as mel-spectrogram, log melspectrogram. All the recent works we are aware of use spectrograms as their input features.

RNNs

RNNs are designed to process sequential data with variable lengths [12]. Audio is a time series signal, therefore RNNs initial works adopted RNN's. In a simple recipe, a RNN is used to model temporal relationships between acoustic features, and the hidden states of the last layer of the RNN is regarded as the audio feature sequence. In [17] the authors utilized a three-layered bi-directional gated recurrent unit (GRU) [13] as the encoder. In [98] and [96] a uni-directional single layer GRUs where used whereas Ikawa et. al. [37] used a single-layered bi-directional long-short term memory (LSTM).

In [71] Nguyen et. al. proposed a temporal sub-sampling method to sub-sample the audio features between the RNN layers, and showed that the temporal sub-sampling of audio features could benefit audio captioning methods. They argue that the sub-sampling is crucial since the audio features are generally much longer than the audio captions. The main drawback of the RNNs is that they may not be able to effectively model long-range time dependencies.

CNNs

In recent years, CNNs are adapted to audio-related tasks and show powerful ability in extracting robust audio patterns [46], [34]. CNNs treat the spectrograms as 1-channel images, and model local dependencies within the spectrograms. Many work directly employ AudioSet [23] pre-trained CNN models as the audio encoder. VGG-like CNNs [10] and ResNets [101], [79] are popular choices as these networks perform well on audio related tasks such as audio tagging and sound event detection. Eren et al. [20] used WavegramLogmel-CNN adapted from PANNs [46], which takes raw waveform for 1-D convolution and spectrogram for 2-D convolution and combine the outputs of 1-D convolutional layers and 2-D convolutional layers in deep layers. Tran et al. [88] also proposed to utilise 1-D and 2-D convolutions for extracting temporal and time-frequency information, however, they only used spectrogram as input and reshape it for 1-D convolution. To obtain global audio features, some methods use a global pooling after the last convolutional block to summarize feature maps into a vector of fixed size [20] Others keep the time axis to get temporal features and feed them to an attention module in order to attend to the informative features when performing language decoding.

Transformer

Mei et. al. [61] are the first to propose a convolutional free transformer encoder-decoder architecture. To tackle the data scarcity problem of audio captioning they pretrain the encoder to to AudioSet [23]. Their encoder model is a ViT model [**?**] pretrained on ImageNet [14]. In [4] the authors propose extracting the audio features from an Audio Spectrogram Transformer (AST)[25] model before passing them to a bi-directional LSTM.

3.2.2 Text decoding

The aim of the language decoder is to generate a caption given audio features from the encoder. All existing work we are aware of adopt an auto-regressive model, where each predicted word is conditioned on previous predictions. In addition to the main decoder block, there is often a word embedding layer before the main decoder block, which embeds each input word into a fixed-dimension vector. In this section, we first introduce popular word embeddings and then discuss main text decoding approaches.

Word Embeddings

In natural language processing (NLP), word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.

Pre-trained word embeddings are trained using neural networks with large corpus, and could capture semantic information, that is, semantically similar words are close to each other in the embedding space [67]. Word2Vec [67], GloVe [78] and fastText [5] are widely used in existing audio captioning works [45] [101], [40], [99], [61].

RNNs

Since text is also sequential RNN decoders are a usual choice for many recent works in AAC. The audio feature sequence from the audio encoder is first aggregated to get a global feature representation and then passed to the RNN decoder for caption generation. Drossos et al. [17] proposed a 2-layer GRU as the decoder in the initial work.

Due to the limited availability of data, many subsequent works have adopted single-layer RNNs, either GRU or LSTM [40], [37], [71], [100], [8]. The main differences of these works is how the decoder interacts with the endoers output. In a simple recipe, a global audio feature representation is obtained by applying mean pooling on the audio feature sequence extracted by the encoder, which is then used as the initial hidden state of the RNN decoder [98], [96] or is injected to the RNN decoder at each time step [71], [20]. These methods do not consider the relationship between the audio and text features. This issue is mitigated with the Attention mechanism as in [17]. At each timestep the decoder attends to the hole input sequence in order to predict the next token.

Transformer

Transformer-based decoders show state of-the-art performance in audio captioning, and they are also more computationally efficient compared with RNN-based decoders. The vanilla Transformer decoder is widely used [10], [61], [62], [9]. Xiao et al. [97] introduced an attention-free Transformer decoder that could capture local information within audio features.

3.2.3 Objectives

Supervised training with a cross-entropy (CE) loss is a standard recipe for training an audio captioning model. Reinforcement learning is introduced to directly optimize evaluation metrics. Lastly there are works that experiments with Contrastive learning i.e. a machine learning technique used to learn the general features of a dataset without labels by teaching the model which data points are similar or different.

Cross-Entropy Loss

The cross-entropy loss with maximum likelihood estimation (MLE) is widely used for training audio captioning models. During training, this approach adopts a 'teacher-forcing' strategy [80]. That is, the objective of training is to minimize the negative log-likelihood of current ground truth word given previous ground truth words at each time step, which can be formulated as follows:

$$\mathcal{L}_{CE}(\partial) = -\frac{1}{T}\sum_{t=1}^{T} \log p(y_t|y_{1:t-1}, \partial)$$

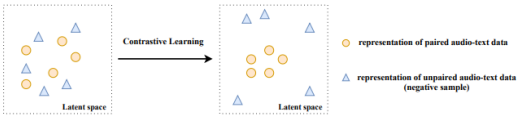
where y_t is the ground-truth word at time step t. Most works in AAC use CE as their objective function. The main drawback with this approach is that each word is conditioned to the previously generated words thus being prone to error accumulation. Also the output tends to be generic and repetitive as we discuss in section 3.4.

Reinforcement Learning

Xu et al. [99] and Mei et al. [61] employ a reinforcement learning approach and directly optimize the non-differentiable evaluation metrics. In a reinforcement learning setting, the captioning model is regarded as an agent and a policy is determined by the model's parameters. The agent executes an action at each time step to sample a word according to the policy. Once a sentence is generated, the agent receives a reward of the generated sentence. The goal of training is to optimize the model to maximize the expected reward that could be any evaluation metrics. The metric that the authors chose to optimize in [99] is CIDEr, because it is the fastest to compute. Although RL improves the evaluation metrics it appears to lead to captions syntactically incorrect, introduces some repetitive words and generates incomplete captions. After optimizing with RL Mei et al. [61] reported that most captions appended a phrase "in the background" which was not in their ground truth captions. Namely 756 of the predicted captions contained the phrase when only 302 audio clips contained the phrase in the ground-truth captions. The authors conclude that this may indicate that the existing evaluation metrics used in the captioning system may not fully reflect the quality of captions and human judgment.

Contrustive Learning and Multi-Task Learning

In [55] a contrastive learning framework is proposed. Mismatching audio text pairs are constructed and are used as negative samples. That is done by pairing audio with unrelated captions from the ground truths. A contrasting task is designed to to maximize the difference between the representations of the true audio text pairs and negative-false pairs that are constructed.

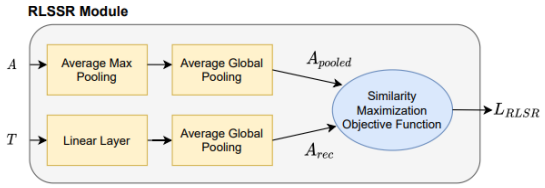


Since the last vector of the Transformer decoder's output has been able to attend the context of all tokens, it used as input to a binary classifier f to predict whether that audio-text input is a negative sample. The loss function $\mathcal{L}_{CL}$ of the contrastive learning module is a binary cross entropy loss. The final training objective is defined as:

$$\mathcal{L} = (1 - y)\mathcal{L}_{CE} + \mathcal{L}_{CL}$$

In this way when an negative sample is passed through the training procedure the gradient from the CE is not utilized.

In [44] a novel objective, Reconstruction Latent Space Similarity Regularization (RLSSR) is used to regularize the training of model. They argue that because Audio Captioning is an abstract task compared to other audio-text tasks like ASR cross attention mechanism is not suffisient to bridge the two modalities. RLSSR aims to regularize the training of the decoder by minimizing the euclidean distance between the latent output of the encoder and the decoder.



The similarity function is the L1 norm and the final loss function is a sum of the cross-entropy loss and the the loss from the RLSSR module.

3.3 Datasets for Audio Captioning

3.3.1 AudioCaps

AudioCaps [40] is the largest audio captioning dataset so far. All the audio clips are 10- seconds long and are sourced from AudioSet, a largescale audio event dataset [1]. The audio clips are selected by following some selection qualifications that ensure the chosen audio clips are balanced with respect to audio tags and diverse in terms of content. The audio clips are annotated by crowdworkers through Amazon Mechanical Turk (AMT), annotators are provided with an audio clip with corresponding word hints and video hints, and are required to write a natural language description with provided information.

The official release of AudioCaps contains 51k audio clips and is divided into a training set, a validation set and a test set. Each audio clip in the training set contains one corresponding human-annotated caption while those in validation set and test set contain five corresponding captions. As audio clips from AudioSet are extracted from YouTube videos, it is worth noting that some audio clips might be no longer downloadable, thus the number of downloadable audio clips might be varied from the official release of AudioCaps.

3.3.2 Clotho

Clotho [18] is a freely available audio captioning dataset , with 4981 audio samples and 24 905 captions. All audio samples are from Freesound platform [22], and are of duration from 15 to 30 seconds. Each audio sample has five captions of eight to 20 words length. Contrary to other audio captioning datasets [40] during annotation no other information but the audio signal was available to the annotators, e.g. video or word tags. The vocabulary size of the captions is 4367.

3.4 Diversity, Repetition and Decoding Strategies

In practice, given an audio clip, people may focus on different parts and tend to describe the content from various aspects using distinct words, phrases and grammars. As a result, the generated captions may be of great diversity. For example, in the audio captioning datasets [18], each audio clip comes with several diverse, human annotated captions. Unlike accuracy, diversity is usually neglected [65] but is an important property that an audio captioning system should possess. In this section we will discuss , why the models that are used to decode text tend to be repetitive, dull and lacking diversity, and propose some decoding strategies i.e., the set of decision rules used to decode strings from the model, to battle these issues.

3.4.1 Error Analysis of a SOTA model

When we began investigating AAC, we conducted an error analysis on a (at the time) state-of-the-model for AAC, Wavetranformer [91]. WaveTransformer appears to be very repetitive. As it is illustrated in the dataframe below some predicted captions are repeated again and again for different input sounds. The evaluation set of Clotho dataset has 1045 audio-caption pairs but the models unique predicted sentences that caption the audio events are 503.

Figure 3.2: *The number of occurrences of the most common captions generated by the model.*

	captions	occurrences
0	birds are chirping and singing in the background	43
1	a person is flipping through the pages of a book	42
2	a machine is running while people are talking ...	34
3	the wind is blowing while the wind is blowing	30
4	a group of people are talking to each other	22
...	...	...
499	people are talking while cars pass by	1
500	a door is opened and closed	1
501	a machine is running at a constant rate of speed	1
502	the loud beast snarls as time goes on	1
503	a loud mechanical whirring resonates incessant...	1

The model is also suffering from repetitiveness in the sentence level. Many of the generated captions contain the same word or n-grams. The model for example gives the following prediction for many audio clips: "The wind is blowing while the wind is blowing". Such word repetition is very rear in human generated text and it can be both syntactically

and semantically wrong. We observe that in 95 predicted captions there is an occurrence of a repeated word sequence. That is almost the 10% of the evaluation dataset.

Motivated by this error analysis we will briefly discuss the recent literature on the issues of natural language (de)-genration [36], and the solution proposed to mitigate them.

3.4.2 Likelyhood Trap

WaveTransformer uses Beam Search algorithm (Section 3.4.5) for decoding with a beam width of 2. Beam Search algorithm has recently been associated with this kind of problem. In the extreme, beam search approximates finding the single most likely generation $x^* = \arg\max \log p_{model}(x)$ and is the approach principally adopted in most language generation tasks. As it is observed in [36] decoding strategies that optimize for output with high probability, such as beam search, lead to text that can be bland, incoherent, or gets stuck in repetitive loops. The assumption of a monotonically positive relationship between sequence likelihood and sequence quality breaks down at the extremes [106]. As it has been observed by translation and image captioning communities increasing beam search width hearts the quality of the generated text after a certain point [85] [42] [94]. We will refer to this problem as *the likelihood trap*.

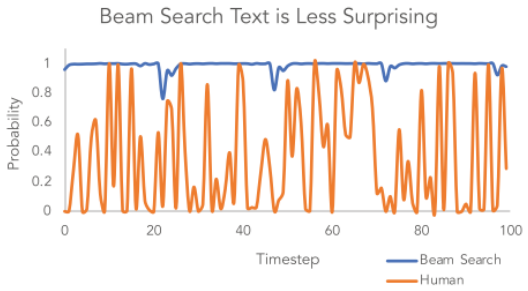


Figure 3.3: *Likelihoods of words chosen by a Beam Search and Human over time. Source([36])*

It clear from the figure above the human generated text is more diverse and unpredictable. We would like our models outputs to be more diverse while maintaining their quality. In this manner decoding has to be viewed as a multi-objective optimization problem aiming to simultaneously maximize both response quality and diversity.[106]. To address these problems we will explore other decoding techniques that try to minimize the lack of diversity and repetitiveness of vanilla beam search.

3.4.3 Decoding Strategies

While conditional language models have greatly improved in their ability to output high-quality natural language, many NLP applications benefit from being able to generate a diverse set of candidate sequences. Diverse decoding strategies aim to cover as much of the space of high-quality outputs as possible, leading to improvements for tasks that re-

rank and combine candidate outputs. Standard decoding methods, such as beam search, optimize for generating high likelihood sequences rather than diverse ones, though recent work has focused on increasing diversity in these methods [39].

3.4.4 Greedy

The simplest approach to decoding a likely sequence is to greedily select a word at each timestep:

$$\hat{y}_t = argmax_{y_t} P(y_t | y < t, \mathbf{x})$$

However, because this deterministic approach typically yields repetitive and short output sequences, and does not permit generating multiple samples, it is rarely used in language modeling.

3.4.5 Beam Search

Beam search approximates finding the most likely sequence by performing breadth-first search over a restricted search space. At every step of decoding, the method keeps track of b partial hypotheses. The next set of partial hypotheses are chosen by expanding every path from the existing set of b hypotheses, and then choosing the b with the highest scores. Most commonly, the log-likelihood of the partial sequence is used as the scoring function.

3.4.6 Pure Random Sampling

In this approach we randomly sample from the model's distribution at every timestep. Often, a temperature parameter T is added to control the entropy of the distribution before sampling. Before the logits are passed to the softmax function they are divided with T. Choosing a temperature greater than one causes outputs to look increasingly more random, while bringing the temperature less than zero causes sequences to increasingly resemble greedy sampling.

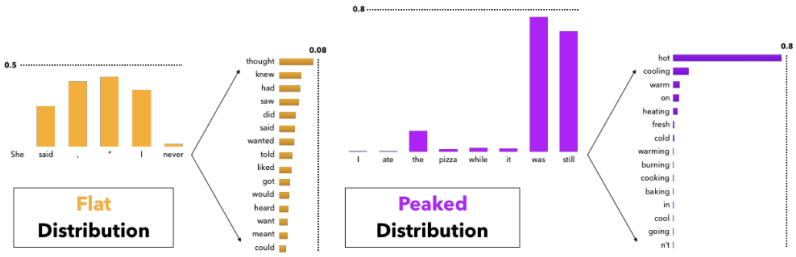
$$P(y_t = w_i | y_{<t}, x) = \frac{exp(z_{t,i}/T)}{\sum_{j=1}^{V} exp(z_{t,i}/T)}$$

Random sampling often results in text that is incoherent and almost unrelated to the context. This is because of the "unreliable tail"[36] of the distribution. Because at each stem the algorithm randomly samples from the distribution it is likely that at some steps it with produce a token that is very unlikely and completely out of context. This can mislead the text generation as every output depends from the previews. The algorithms below try to tackle this problem.

3.4.7 Top-k Sampling

To avoid the "unreliable tail"[36], Top k [21] sampling sorts the distribution by probability and prunes the probabilities below the k'th token. This technique addresses the problem of pure random sampling but in some cases, there really are many words we could sample from reasonably (Flat Distribution), and in some cases there aren't (Peaked Distribution).

Figure 3.4: *Flat vs Peaked Distribution*



3.4.8 Nucleus Sampling (Top-p) [36]

If we only keen the top-k tokens from a flat distribution we lose a large fragment of the probability mass. To avoid this issue Nucleus Samping is proposed [36]. The key idea is to use the shape of the probability distribution to determine the set of tokens to be sampled from. Given a distribution $P(x|x_{<i})$, we define its top-p vocabulary V(p) V as the smallest set such that:

$$\sum_{x \in V^{(p)}} P(x|x_{<i}) \geq p$$

For the new distribution we keep only the tokens in V(p) which we scale by the sum of their probabilities$\sum_{x \in V^{(p)}} P(x|x_{<i})$. The key difference with top-k sampling is that the size of the sampling set will adjust dynamically based on the shape of the probability distribution at each time step rather that being a hyper-parameter we hard-code before we start decoding.

Both Top-k and Nucleus sampling can be combined with temperature e.g. dividing logits by $T \in (0, 1)$ before passing them through the softmax. Low temperature sampling e.g. skewing the distribution towards high probability events is used to partially alleviate the issues of top-k sampling discussed above [21]. But analysis shows that lowering the temperature hurts diversity as it makes tokens with lower probability even more unlikely to occur [32].

3.4.9 Clustered Beam Search

Tam et. al. [72] proposed a clustering-based beam search method to help condense and remove meaningless responses from chatbots. Specifically, at each decoding step

t, this method initially considers the top $2 \cdot b$ candidates. From there, each candidate sequence is embedded3, and the embed- dings are clustered into c clusters using K-means. Finally, we take the top $\frac{b}{c}$ candidates from each cluster. Note that in the case any clusters have size less than $\frac{b}{c}$, we then include the highest-ranked candidates not found after clustering.

When we experiment with clustered beam search we remove all hypothesis with repeated bi-grams. In this way we force the decoding process to exclude any caption which contains a repeated n-gram.

3.5 Evaluation Metrics

In this section we present the standard evaluation metrics used to evaluate the quality of captions generated by AAC models. Many of these metrics were introduces and are also used to evaluated other task such as Machine Translation, Summarization, and Image Captioning.,

3.5.1 BLEU

BLEU (BiLingual Evaluation Understudy) is an algorithm for evaluating the quality of text in many NLP tasks and was originaly used to evaluate machine-translation [75]. BLEU uses a modified form of precision to compare a candidate text against multiple reference texts. The metric calculates the precision for n-grams. To calculate precision, the matching words in the actual sentence and the predicted sentence is calculated. BLEU does not consider the context of the word in the sentence. The metric range is between [0,1]. If the actual sentence and the predicted sentence are totally the same, then the score is 1. BLEU-1 (B-1) represents 1-gram, whereas BLEU-4 (B-4) represents 4-grams.

3.5.2 METEOR

METEOR [3] unlike BLEU incorporates both precision and recall in the evaluation score. The algorithm has two stages. First given a ground truth and a predicted sentence METEOR creates an alignment between them i.e. a mapping between unigrams, such that every unigram in each string maps to zero or one unigram in the other string. If there are two alignments with the same number of mappings, the alignment is chosen with the fewest crosses, that is, with fewer intersections of two mappings.

Then METEOR calculates unigram recall and unigram precision together and takes a harmonic mean score. Finally the harmonic mean score is multiplied with a penalty calculated as follows: First fewest possible number of *chunks* is calculated such that the unigrams in each *chunk* are in adjacent positions in the system translation, and are also mapped to unigrams that are in adjacent positions in the reference translation. Penalty is then computed by the following.

$$Penalty = 0.5 * \frac{\#chunks}{\#unigrams\_matched}$$

The penalty increases as the number of chunks increases to a maximum of 0.5. As the number of chunks goes to 1, penalty decreases, and its lower bound is decided by the number of unigrams matched. Finally, the METEOR Score for the given alignment is computed as follows:

$$Score = Fmean * (1 - Penalty)$$

3.5.3 ROUGE$_L$

ROUGE-L measures the longest common subsequence (LCS) between the ground truth and reference sentence. The idea is that a longer shared sequence would indicate more similarity between the two sequences. Then recall and precision calculations are applied as follows:

$$R_{LCS} = \frac{LCS(X,Y)}{m}$$
$$P_{LCS} = \frac{LCS(X,Y)}{n}$$

Where m is the length of the reference sentence and n is the length of the ground truth sentence. Then finally $ROUGE_L$ is calculated as:

$$F_{ROUGE_L} = \frac{(1+b^2)R_{LCS}P_{LCS}}{R_{LCS} + b^2 P_{LCS}}$$

Where $b = P_{LCS}/R_{LCS}$. ROUGE-L is 1 when the sentences are the same, while ROUGE-L is zero when LCS(X,Y) = 0, i.e. there is no common sub-sequence in the sentences.

3.5.4 CIDEr

Consensus-based Image Description Evaluation (CIDEr) is a new paradigm for evaluation of image captions that is based on human consensus [93]. It aims to capture sentence similarity, grammaticality, importance, saliency and accuracy. To evaluate how well a generated caption c_i matches the consensus of a set of captions $S_i = s_{i1}, ..., sim$, all words are first mapped to their stem forms and each caption is represented using the set of n-grams ω_k, that are present in it. Then, a Term Frequency Inverse Document Frequency (TF-IDF) weighting is performed for each n-gram to encode how often ngrams in the generated caption are present in the reference ones, and how often n-grams not present in the reference captions are not in the generated captions. Additionally, frequent n-grams are given low weight. The TF-IDF $g_k(s_{ij})$ for each n-gram ω_k is:

$$g_k(s_{i,j}) = \frac{h_k(s_{ij})}{\sum_{\omega_l \in \Omega} h_l(s_{ij})} \log\left(\frac{|S|}{\sum_{S_p \in S} min(1, \sum_q h_k(s_{qp}))}\right)$$

where h_k (c) is the frequency that an n-gram k occurs in the caption c, Ω is the vocabulary of n-grams and S is the set of all samples. The $CIDEr_n$ score for n-length n-grams is the average cosine similarity between the generated caption and the reference

captions and is given by:

$$CIDEr_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{\mathbf{g}^n(c_i) \cdot \mathbf{g}^n(s_{ij})}{||\mathbf{g}^n(c_i)||||\mathbf{g}^n(s_{ij})||}$$

where $\mathbf{g}^n(c_i)$ is a vector with elements $g_k(c_i)$ that correspond to all n-length n-grams. The final CIDEr score combines the scores of variable length n-grams as follows:

$$CIDEr(c_i, S_i) = \sum_{n=1}^{N} w_n CIDEr_n(c_i, S_i)$$

3.5.5 SPICE

All the evaluation metrics mentioned above are primarily sensitive to n-gram overlap. However, n-gram overlap is neither necessary nor sufficient for two sentences to convey the same meaning. If we take for consideration the following example we observe that it produces a high similarity score to all of the above metrics:

- A dog is standing on top of a chair
- A woman is standing on top of a field

These two sentences describe very different events but would get a fairly good similarity score with the all of the above metrics due to the phrase *is standing on top of a* which is common in both sentences.

Spice address this issue with the following procedure. At first, the generated caption c and the reference captions $S = s_1, ..., s_m$ are transformed to the scene graphs $G(c)$ and $G(S)$ respectively, where $G(S)$ is the union of scene graphs $G(s_i)$ for $s_i \in S$. The semantic relations in a scene graph are considered to be a conjuction of logical propositions or tuples and the function T returns these tuples from a scene graph as:

$$T(G(c)) \triangleq O(c) \cup E(c) \cup K(c)$$

where O(c) is the set of object mentions in c, $E(c)$ is the set of hyper-edges that represent relations between objects and $K(c)$ is the set of attributes associated with objects. The precision P, recall R and SPICE score are defined as:

$$P(c, S) = \frac{|T(G(c)) \bigotimes T(G(S))|}{|T(G(c))|}$$
$$R(c, S) = \frac{|T(G(c)) \bigotimes T(G(S))|}{|T(G(S))|}$$
$$SPICE(c, S) = F_1(c, S) = \frac{2 \cdot P(c, S) \cdot R(c, S)}{P(c, S) + R(c, S)}$$

where the binary matching operator $\bigotimes$ returns the matching tuples in two scene graphs. Since SPICE is an F-score, it is easily interpretable and its range is between 0 and 1. Moreover, it can be applied equally to both small and large datasets.

3.5.6 SPIDEr

SPIDEr [54] is used as the official ranking metric in the automatic audio captioning task in DCASE Challenge. SPIDEr is the average of SPICE and CIDEr. The SPICE score ensures captions are semantically faithful to the content, while CIDEr score ensures captions are syntactically fluent.

3.6 Our proposed model: CaptionPaSST

In this section we present the model architecture that we use in our experiments. The proposed method for the AAC task is a transformer model we call **CaptionPaSST**, which is based on the traditional sequence-to-sequence architecture and is (almost) convolution-free. The model takes the log mel-spectrogram of an audio clip as input and outputs the posterior probabilities of the predicted words. In order to train such a model with limited resources we experiment with the The Patchout faSt Spectrogram Transformer (PaSST) [48] model as the encoder in order to reduce the complexity.

3.6.1 Architecture

The model we present here is trained to implement the following mapping where $\mathbf{A}$ is an audio clip and $\mathbf{c}$ is a caption:

$$f: \mathbf{A} \rightarrow \mathbf{c} \tag{3.1}$$

It is composed of an encoder with parameters ∂_e and a decoder (right on Figure 3.5) with parameters ∂_d and it models the conditional probability distribution:

$$p_{\partial_e \partial_d}(\mathbf{c}|\mathbf{A}) \tag{3.2}$$

The encoder part encodes the input sequence of patches $P_{1:k}$ to a new sequence $\mathbf{X}_{1:n}$, thus defining the mapping:

$$f: \mathbf{P}_{1:k} \rightarrow \mathbf{X}_{1:k} \tag{3.3}$$

where $X_{1:h}$ are the audio features. The audio features function as compressed representation of the input audio. We can rewrite Equation 3.2, to depend only on the decoder part and the above contextualized sequence, as follows:

$$p_{\partial_d}(\mathbf{c}|\mathbf{X}_{1:k}) = \prod_{c_i \in \mathbf{c}} p_{\partial_d}(c_i|c_{<i}, \mathbf{X}_{1:k}) \tag{3.4}$$

Note that the audio feature vector is adjusted through a fully connected layer and a non-linearity to match the hidden dimension of the decoder.

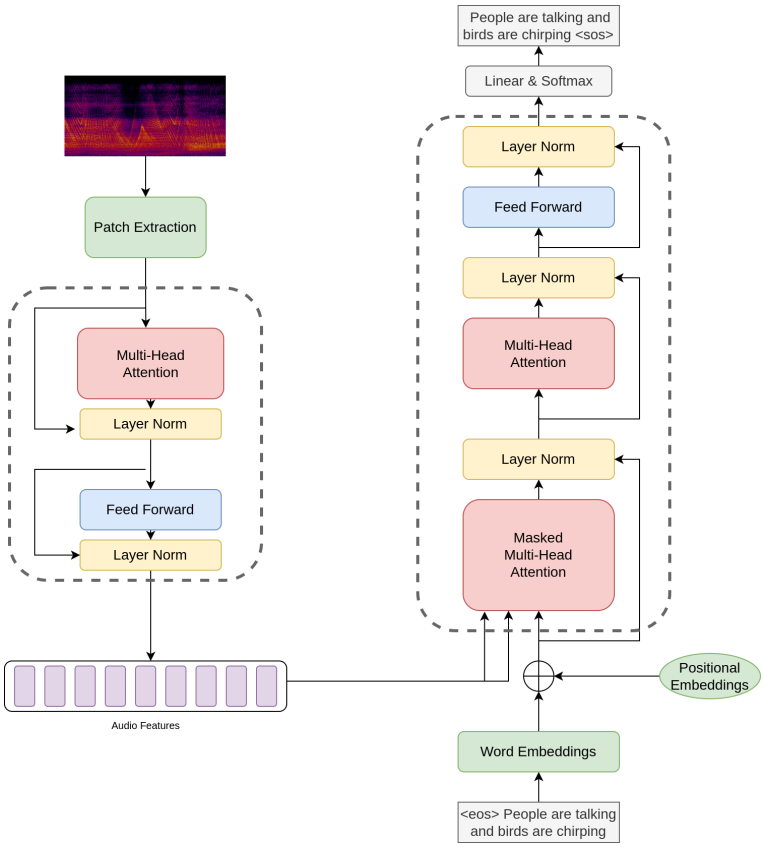


Figure 3.5: *Our proposed transformer encoder-decoder architecture. The patch extraction is shown in Figure 3.6 for clarity.*

Encoder

Let $x \in \mathbb{R}^{T \times F}$ denote the log mel-spectrogram of an audio clip, where T is the number of time frames and F is the number of mel bins. The spectrogram is passed through a 2-dimentional convolutional layer and a feature map is produced:

$$x_{map} = Conv2D(x) \tag{3.5}$$

The kernel of the convolution layer is 16 and can be interpreted as the size of the patches P. We experiment with strides S of size 10, 12, 14, and 16. A stride of size 16 means no-overlap. So the smaller the stride the higher the computational complexity of the model. The output dimension of the convolutional layer or the featuremap's depth is 768 which is the hidden dimension d_{hid} size of the transformer encoder. Thus $x_{map} \in \mathbb{R}^{d_{hid} \times F_{map} \times T_{map}}$, where $F_{map} = \frac{F-P}{S} + 1, T_{map} = \frac{T-P}{S} + 1$.

Positional embeddings are added to this stage to encode the positional information of the patches.

$$x_{map} = x_{map} + t_{pos} + f_{pos} \tag{3.6}$$

where, $t_{pos} \in \mathbb{R}^{d_{hid} \times 1 \times T_{map}}$ and $f_{pos} \in \mathbb{R}^{d_{hid} \times F_{map} \times 1}$.

After *Patchout* the the feautermap is reduces to:

$$x_{map}^{p} \in \mathbb{R}^{d_{hid} \times (F_{map} - P_F) \times (T_{map} - P_T)}$$

. We discuss Structured Patchout in section 2.7.2. Note that the positional information is injected to each patch of the featuremap before Patchout, so it is maintained. The featuremap is then flattened to $x_{fl} \in \mathbb{R}^{d_{hid} \times d_{fl}}$ where $d_{fl} = (F_{map} - P_F) \cdot (T_{map} - P_T)$. Lastly, in line with Deit [90] a global learnable class token and another special embedding for distillation are added:

$$X_{in} = x_{fl} + x_{cls} + x_{dis} \tag{3.7}$$

Finally X_{in} is passed to the transformer encoder. The overall procedure can be summed up in the following:

$$X_{in} = Flatten(Patchout(Conv2d(x) + t_{pos} + f_{pos})) + x_{cls} + x_{dis} \tag{3.8}$$

The encoder consists of N_e stacked identical layers. Each layer contains two sub-layers, a multi-head self-attention layer and a position-wise fully-connected feed-forward layer. In the self attention sub-layer, the input is first transformed into query Q, key K and value V through matrix multiplication with three learnable matrices W_Q, W_K and $W_V \in \mathbb{R}^{d \times d_k}$ where d_k is the dimension of each attention head. Then the scaled dot-product attention is computed as

$$Attention(Q, V, K) = softmax(\frac{QK^\top}{\sqrt{d_k}})V \tag{3.9}$$

Each self-attention layer contains h attention heads which extends the model's ability to attend to different positions and creates multiple representation subspaces [92]. The outputs of heads are then aggregated through a linear transformation matrix $W_O \in \mathbb{R}^{(h \times d_k) \times d_k}$ which can be formulated as:

$$MultiHead = Concat(head_1, ..., head_h)W^O \tag{3.10}$$

$$where head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{3.11}$$

The feed-forward network contains two linear layers with GELU [33] activation function and dropout applied between them. Layer normalization is applied before each sub-layer and a residual connection is employed around each of them, such that the output of each sub-layer is given by:

$$X_{out} = X_{in} + Sublayer(LayerNorm(X_{in})) \tag{3.12}$$

In order to make use of pre-trained models, the encoder architecture is the same as PaSST containing 12 encoder blocks and 12 heads with an embedding dimension of 768.

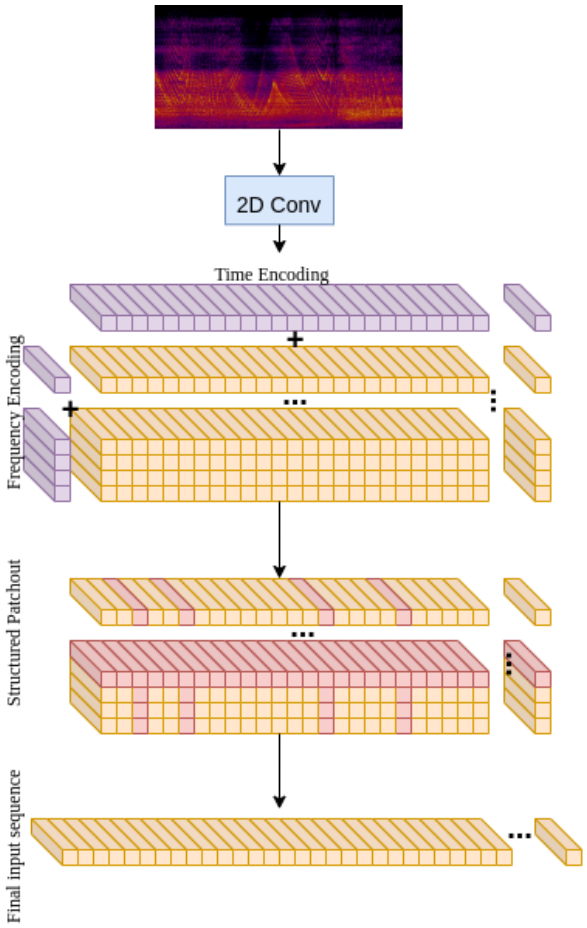


Figure 3.6: *Patch Extraction: First a featuremap is extracted from the spectrogram. Time end frequency positional encoding is added. Structured Patchout is applied and the remaining featuremap in flattened and feed to the transformer encoder.*

Decoder

The decoder contains three parts: a word embedding layer, a Transformer decoder block, and a linear layer. Each input word is embedded through the word embedding layer into a fixed dimension word vector and then fed into the Transformer decoder block. The word vectors are pre-trained by a Word2Vec model on all caption corpus [67]. The Transformer decoder consists of N_d identical stacked layers. There are two main differences compared to the captionPaSST encoder block. First, the first self-attention sub-layer in the decoder is a masked self-attention because the caption generating process is causal and auto-regressive. Second, there is a new cross multi-head attention sub-layer between self-attention sub-layer and feed-forward sublayer, which allows every position in the decoder to attend over all positions in the audio features extracted by the encoder [92]. The output of the decoder module is fed through a final linear layer with a softmax activation function to output a probability distribution over the vocabulary.

3.6.2 Objective

The training objective of the model is to minimize the crossentropy (CE) loss:

$$\mathcal{L}_{CE}(\partial) = -\frac{1}{T}\sum_{t=1}^{T} \log p(y_t|y_{<t}, X, \partial)$$

where c_t is the ground-truth word at time step t and ∂ are the model parameters. The "Teacher forcing" strategy is used during training, i.e. each word to be predicted is conditioned on previous groundtruth words. To avoid overfitting we also experiment with Label Smoothing [87].

3.6.3 Transfer Learning

We apply a transfer learning by utilizing variations of PaSST model that have been made publicly available by the authors [2]. We present and discuss the variations of PaSST model in section 2.7 and present our experiments in section 3.8. We also present the impact of transfer learning on our models performance.

These PaSST models are trained on AudioSet and are pre-trained on ImageNet [14]. The authors report that ImageNet pre-training improves the models accuracy. This has also been reported by other works [25].

We also note that pretraining is helpful for positional encoding. The PaSST encoder uses two vectors to specify the position of each patch to the initial input spectrogram. These positional embeddings are adapted from the pretrained position embeddings of a ViT [16] model trained on ImageNet. Specifically the 2-dimentional positional embeddings of ViT are averaged in the time and frequency dimensions respectively to create to individual vectors.

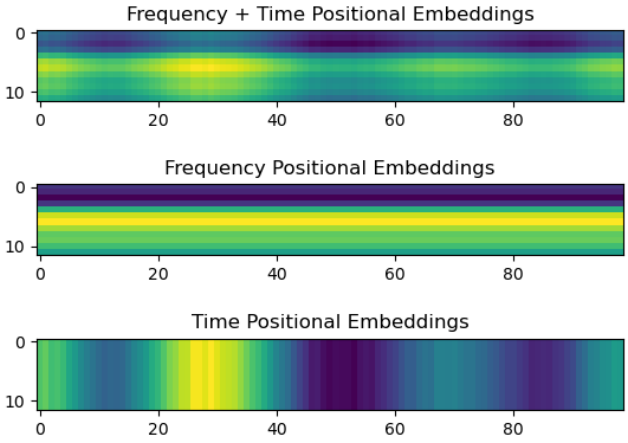


Figure 3.7: *Time, frequency and overall positional embeddings after AudioSet pretraining. To obtain these position embeddings we take the mean over the hidden dimension of the featuremap*

[2] https://github.com/kkoutini/PaSST/releases/

3.7 Feature Extraction and Augmentation

We experiment with two feature extraction schemes. In the first the features are standard log mel-spectrograms extracted using a 1024-points Hanning window with 50% overlap and 128 mel bins are used as the input features. The log Mel energy features denoted as $x = [x_1, ..., x_T], x_t \in \mathbb{R}^{128}$, where x_t is a vector that contains 128 features of the audio clip and T is the number of audio frames. But the transformer models are very prone to overfitting, therefore some data augmentation plays an essential role in the training process [90]. Therefor in the second scheme we apply the following augmentation techniques.

Mix-Up

Mix-up works by linearly combining two input samples and their targets. It was shown to be an effective augmentation method that is simple but can have a great impact on performance. We use Mix-up [104] since it has been shown to improve performance [25], [47]. In the input domain we mix input spectrograms using the standard approach.

$$x = \lambda x_i + (1 - \lambda) x_j \tag{3.13}$$

where, λ $Beta(a, a)$ and a = 0.3 and x_i, x_j are two samples from our training dataset.

Mixing the target sentence is not trivial since Mixup generally requires a single label output and can only be used in classification tasks. We experiment with to methods: **Captions mixing:** Suppose we have y_i and y_j captions corresponding to x_i, x_j. Thus a simple way to implement caption mixing is concatenation:

$$y = Concat(y_i, y_j) \tag{3.14}$$

Embeddings Mixing: In this method the embedded vector of the input caption:

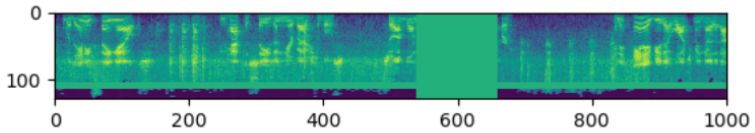
$$y^e = \lambda y_i^e + (1 - \lambda) y_j^e \tag{3.15}$$

The second method appears to be effective and boost the performance of the model even though the caption label with which the cross entropy loss is calculated are not mixed. The first worsens the results of the model.

SpecAugment

We use SpecAugment [76] by masking up to 48 frequency bins and 192 time frames similar to [25]. We present a detailed description of SpecAugment in Section 2.3.5. The following spectrogram is produced by performing SpecAugment on an audio clip of AudioCaps dataset [40]:

Figure 3.8: *The results of frequency and time masking with SpecAugment [76]*



We will discuss the impact of the augmentation strategies in Section 3.8

3.8 Experiments and Results

In this section we discuss some experimental details and training hyperparameters we used, as well as the impact of the various pretrained PaSST encoders. Using the best model with regards to the quality metrics i.e *BLEU*, *METEOR*, $ROUGE_L$, *SPICE*, *CIDEr* and *SPIDEr* we discuss the impact of the decoding strategy on the diversity of the generated captions. All our experiments where conducted on a single NVIDIA GeForce RTX 2080 Ti.

3.8.1 Training Hyperparameters

Training Hyperparameters	
Batch size	32
Mini-Batch size	8
Gradient Clipping	2
Optimizer	Adam
Learning Rate	10^{-4}
Weight Decay	10^{-4}
Linear Warmup steps	5
Epochs	20

Table 3.1: *Training Hyperparameters for all experiments*

In Table 3.1, the reader can see the configuration we used for the training of our models. We manage to use a batch size 4 times the mini-batch size by performing gradient accumulation (updating only every 4 steps. We use gradient clipping (introduced in [77] and analyzed thoroughly in [105]). Gradient clipping solves the exploding gradient problem and smoothens the gradient landscape, by restricting the gradient norm: if $\|\mathbf{g}\| > \upsilon$ then $\mathbf{g} \leftarrow \frac{\mathbf{g}^{\upsilon}}{\|\mathbf{g}\|}$ where υ is a norm threshold, 2 here. The learning rate is linearly increased to 10^4 in the first five epochs using warm-up, which is then multiplied by 0.1 every 5 epochs.

In order to conduct our experiments fast and more efficiently we use early stopping policy with a patience of 5 epochs. If SPIDEr [54] on the validation set hasn't been unproved beyond a threashold $\tau = 0.05$ in the last 5 epochs we stop training.

3.8.2 Model Hyperparameters

In this section we present and discuss the hyperparameters that where mostly constant throughout our experiments. There are some exeptions and we will present discuss them in the next sections where we present the results for each experiment.

Hyperparameters	PaSST Encoder	Decoder
hidden dimension	768	512
depth	12	6
heads	12	8
attention dropout	0	0.2
feed-forward dropout	0	0.2
feed-forward dimension	3072	2048
non-linear activation	GELU	GELU

Table 3.2: *Model Hyperparameters for Encoder and the Decoder*

Dropout is 0 for the encoder since Patchout 2.7.2 has a similar impact on the models performance, i.e. encouraging the model to perform, using an incomplete sequence thus preventing overfitting. Overfitting is also mitigated by the use of GELUs [33] as the activation function both on the encoder and the decoder.

3.8.3 Evaluation on quality metrics

It is evident by the results presented in the table below that using an encoder pretrained on Audioset [23] largely affects the performance of the model. This has been observed by all of the resent studies on AAC.

Model Hyperparameters						Quality Metrics								
Pretrained Encoder	Patch size	Frequency stride	Time Stide	SpecAugment	Mixup	BLUE-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr	SPICE	SPIDEr
No	16	10	10	No	No	0.60	0.42	0.28	0.18	0.18	0.42	0.44	0.13	0.28
Yes	16	12	12	No	No	0.24	0.15	0.09	0.05	0.13	0.28	0.59	0.17	0.38
Yes	16	10	10	No	No	0.24	0.14	0.09	0.05	0.13	0.27	0.62	0.17	0.40
Yes	16	14	14	No	No	0.63	0.46	0.32	0.22	0.21	0.46	0.61	0.15	0.38
Yes	16	16	16	No	No	0.63	0.45	0.31	0.21	0.20	0.44	0.54	0.14	0.34
Yes	16	10	10	Yes	No	0.67	**0.50**	0.35	0.24	**0.23**	**0.48**	0.66	0.16	0.41
Yes	16	10	10	Yes	Yes	**0.68**	**0.50**	**0.37**	**0.26**	**0.23**	**0.48**	**0.71**	**0.17**	**0.44**

Table 3.3: *Experiment results on AudioCaps Test set*

Since patch size is constant throughout all our experiments, a stride of the 16 means no overlap between the patches. Frequency stride and time stride are the strides in the y-axis and x-axis of the spectrogram. Stride influences the results of the model and we the best performance when using a stride of 10 thus resulting to a 6 point overlap. The smaller the stride the higher the computation complexity of the model, since a higher stride will result in a longer input to the Transformer. Through our experiments we find that a stride of 10 gives the best results. We also observe that both SpecAugment and MixUp improve the performance of our model.

Experimenting with structured patchout 2.7.2 we find that a frequency patchout of 4 and a time patch out of 40 archives the best results. As mentioned earlier patchout behaves in a similar way to dropout thus a very aggressive patchout will be detrimental to

the models performance. A small patchout will result to overfitting to the training data. The experiments are shown in the Table below:

Model Hyperparameters	Quality Metrics								
Frequency/ Time S-Patchout	BLUE-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE_L	CIDEr	SPICE	SPIDEr
3/30	0.67	0.45	0.31	0.21	0.21	0.44	0.66	**0.19**	0.42
4/40	**0.68**	**0.50**	**0.37**	**0.26**	**0.23**	**0.48**	**0.71**	0.17	**0.44**
5/50	0.65	0.44	0.30	0.20	0.21	0.42	0.67	0.17	0.42

Table 3.4: *Experiments with different values of Frequency and Time structured patchout*

We also experiment with a larger decoder, in order to fit our model to a single GPU we use a smaller mini-batch of size 6.

Making the decoder larger worsen the results of the model, we suspect this is due to the limited data in AudioCaps dataset, as large transformer models need a lot of data to train. Training with a smaller learning rate larger mini-batch size and and for more epochs would probably help train captionPaSST with a large decoder, but due to the limit resources available to us such experiments where very difficult to conduct.

3.8.4 Investigating the quality diversity trade off.

In this section we present the results of the various decoding algorithms we presented in section 3.4. As we have discussed already, the literature in the fields of language generation suggests that there is a "likelihood trap"[106], i.e. the counter-intuitive observation that high likelihood sequences are often surprisingly low quality. Also decoding methods such as beam search have been found to lead to degeneration — output text that is bland, incoherent, or gets stuck in repetitive loops. Moreover beam search is ill-suited to generating a set of diverse candidate sequences. This is because candidates outputted from a large-scale beam search often only differ by punctuation and minor morphological variations [39].

To measure the diversity across the generated candidate sequences for each decoding algorithm, we report **Dist-k** and **Ent-k**. Dist-k is calculated by dividing the total number of distinct k-grams by the total number of produced tokens in all of the candidate responses for a prompt [53]. We report Dist-1 and Dist-2 averaged over the prompts in the test set. A limitation of Dist-k is that all k-grams that appear at least once are weighted the same, ignoring the fact that infrequent k-grams contribute more to diversity than frequent ones[39]. Zhang et. al. [106] instead propose an entropy metric, Ent-k, defined as:

$$Ent = -\frac{1}{\sum_w F(w)} \sum_{w \in V} \log \frac{F(w)}{\sum_w F(w)} \tag{3.16}$$

Where V is the set of all k-grams and $F(w)$ is the frequency of k-gram w

In Table 3.5 we report the evaluation of **captionPaSST** in both quality and diversity metrics with various decoding algorithms. As we expected algorithms that sample from the distribution perform better on diversity metrics. On the other hand beam search and greedy decoding perform better on quality metrics. Temperature sampling achieves

the best performance on diversity metrics and the worst scores on the quality metrics, whereas greedy search from which we get the best results on SPIDEr metric performs the worst in diversity metrics.

In our experiments we find that the decoding algorithm that minimizes the quality-diversity trade of is **Clustered Beam Search**. As discussed in Section 3.4 this variation of beam search clusters similar hypotheses together increasing the probability of a more diverse and less likely sequence to be selected. It also discards any candidate hypothesis with repeating n-grams preventing degenerate predictions.

Decoding Algorithm	BLUE-1	BLUE-2	BLUE-3	BLUE-4	METEOR	ROUGE-L	CIDEr	SPICE	SPIDEr	Dist-1	Dist-2	Ent-1	Ent-2
Temperature Sampling	0.57	0.39	0.26	0.17	0.20	0.41	0.51	0.14	0.32	**0.10**	**0.34**	**4.99**	**6.97**
Top_k	0.54	0.45	0.31	0.20	0.22	0.45	0.58	0.15	0.37	0.08	0.26	4.73	6.58
Top_p	0.60	0.42	0.28	0.18	0.21	0.43	0.53	0.14	0.34	0.08	0.30	4.8	6.7
Greedy	**0.68**	**0.50**	**0.37**	**0.26**	**0.23**	**0.48**	**0.71**	0.17	**0.44**	0.06	0.19	4.50	6.15
Beam Search (size 2)	0.61	0.46	0.33	0.23	0.21	0.45	0.67	0.17	0.42	0.07	0.19	4.57	6.03
Beam Search (size 3)	0.59	0.44	0.31	0.21	0.20	0.45	0.63	0.15	0.39	0.07	0.18	4.58	6.01
Beam Search (size 4)	0.59	0.44	0.31	0.22	0.20	0.44	0.63	0.15	0.39	0.07	0.18	4.59	6.00
Beam Search (size 5)	0.59	0.43	0.31	0.21	0.20	0.44	0.61	0.14	0.38	0.07	0.18	4.59	6.00
Clustered Beam Search	0.16	0.10	0.06	0.04	0.12	0.26	0.64	**0.19**	0.41	0.07	0.22	4.70	6.22

Table 3.5: *Comparison of decoding algorithms on quality and diversity metrics*

3.8.5 Comparing captionPaSST with SOTA

Our method achieves results comparable with the current state-of-the-art approaches on AudioCaps [40]. Our method outperforms Audio Captioning Transformer (ACT) [63] which is the only convectional-free transformer encoder-decoder architecture for AAC that we are aware of. Also compared to ACT our model is faster, requires less memory to train and converges faster. We compare our model to ACT on a NVIDIA GeForce RTX 2080.

Model	Quality Metrics								
	BLUE-1	BLEU-2	BLEU-3	BLEU-4	METEOR	$ROUGE_L$	CIDEr	SPICE	SPIDEr
CNN+Transformer	0.64	0.47	0.34	0.23	0.22	0.46	0.69	0.16	0.42
ACT	0.64	0.48	0.35	0.25	0.22	0.46	0.67	0.16	0.42
BART + YAMNet + PANNs	**0.69**	**0.52**	**0.38**	**0.26**	**0.24**	**0.49**	**0.75**	**0.17**	**0.46**
captionPaSST	0.24	0.14	0.09	0.05	0.13	0.27	0.62	0.17	0.40
captionPaSST + SpecAugment	0.67	**0.50**	0.35	0.24	**0.23**	**0.48**	0.66	0.16	0.41
captionPaSST + SpecAugment+ Mixup	**0.68**	0.50	0.37	**0.26**	0.23	0.48	0.71	**0.17**	0.44

Table 3.6: *Comparing with state-of-the-art on AudioCaps.*

Model	Mem (MiB)	Samples/Sec	Epochs	SPIDEr
ACT	6419	78	30	0.42
CaptionPaSST	4231	120	15	0.44

Table 3.7: *Comparison of GPU memory, spectrograms per second, training epochs and SPIDEr metric between Audio Captioning Transformer (ACT) and CaptionPaSST.*

Chapter 4

MovieCaps Dataset

The dataset used to training an audio captioning method defines to a great extent what the method can learn. So in order to create a model that can efficiently identify and describe the audio events in a movie we have created a domain specific dataset with which we can train an audio captioning model to learn to caption audio events that are present in movies.

We present **MovieCaps**, an audio captioning dataset for sound events in movies. The dataset is created from SDH subtitles available online. Subtitles for the deaf and hard of hearing (SDH) are subtitles that combine the information of both captions and subtitles. They can be in the source language of the video, as they include important non-dialogue audio like sound effects and speaker identification.

4.1 Data collection and processing

We have collected 73 Movies 360 episodes of various TV Series with SDH subtitles. These subtitles contain audio event description and speaker identification information. We are able to identify these subtitles because they appear in brackets or parentheses. From these Movies/Series and their corresponding subtitles we extract audio-text pairs $\mathbb{D}_{init} = \{a^i, C^i\}_i^{N_{init}}$ with $N_{init} = 56935$. We extract the audio using the onset and offset times of each subtitle. From these initial samples, we first omit the ones that contain speaker identification and diarization information in the folloing way:

- We omit any samples with numbers appearing in the captions, as we observe that usually the these captions perform speaker diarization (eg. man 1, man 2...).
- We omit any sample with Names appearing in the captions. To achieve this we use a pretrained Bert based Name Entity Recognition (NER) model. [1]
- We omit any sample with the words *all*, *both* which in the context of SDH captions inform the viewer for the number of the people talking

The result of this process is $\mathbb{D} = \{a^i, C^i\}_i^N$ with $N = 45432$. The final processing step is to remove the word *continues* from the captions it appears. This word is very informative for the viewer but after the samples are extracted from the movie to create a dataset

[1] https://huggingface.co/dslim/bert-base-NER

the temporal information is lost and the word adds no information to the captions. The captions in $\mathbb{D}$ are turned in to lower case letters and punctuation is removed.

4.2 Dataset Characteristics

This prepossessing results in a 3405 word vocabulary. As we observe in the figure below the dataset contains many unique words e.g. words that appear only in one sample and few words with very high frequency. This will make a fair split of the dataset more difficult as we will discuss in the Section below.

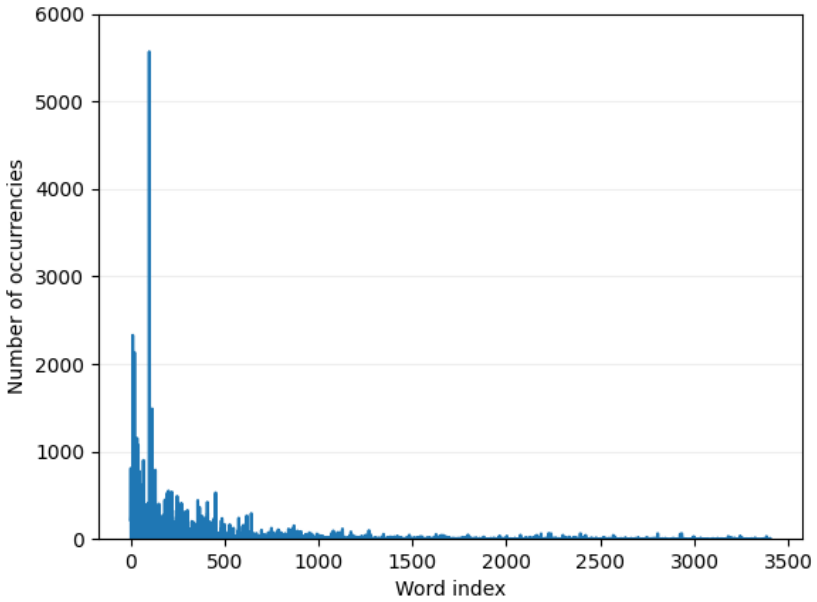


Figure 4.1: *Number of occurrences of every word in the vocabulary*

The number of words that occur only in one caption is 1114 and the average number of occurrences is 23. The 5 most frequent words are:

- *music* with 5568 occurrences.
- *sighs* with 2328 occurrences.
- *playing* with 2275 occurrences.
- *chuckles* with 2132 occurrences.
- *door* with 1487 occurrences.

The frequency distribution of captions is similar. Some captions (such as *music is playing*) are very frequent while others (such as *device slurping*) occur only once.

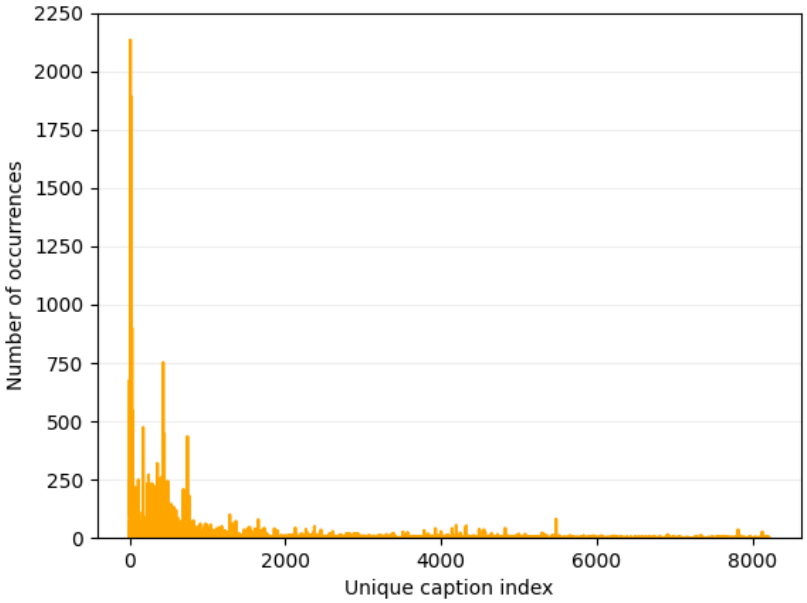


Figure 4.2: *Number of occurrences of every caption in the MovieCaps*

The characteristic that differentiates MovieCaps from other audio captioning datasets is that the average caption length is much smaller. This is because audio descriptions in SDH subtitles are meant to be small and information dense because the viewer has to multi task in order to read them i.g read and watch at the same time. The distribution of the caption length is shown in the figure below:

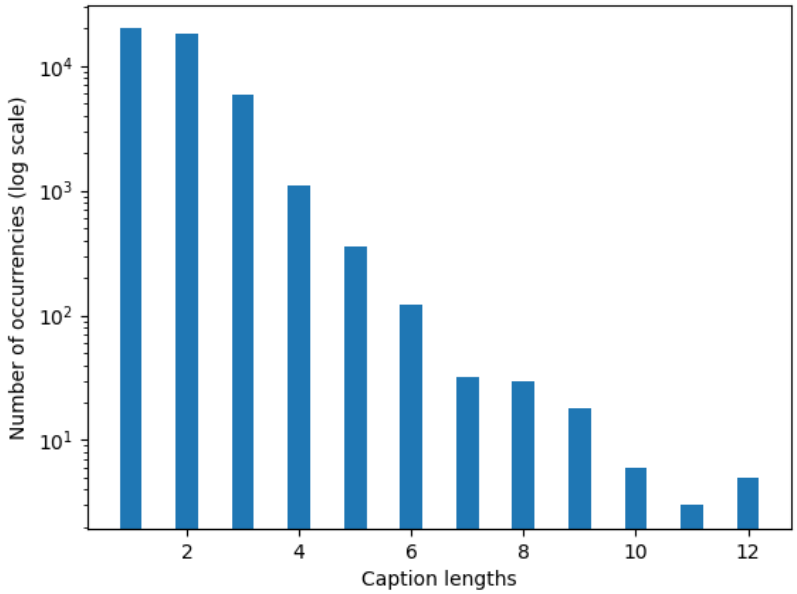


Figure 4.3: *The distribution of caption lengths in logarithmic scale*

Resent work [60] in the evaluation of the existing audio captioning datasets suggests

that one simple measure that represents the variety in vocabulary, or lexical diversity, is the type-token ratio (TTR). **TTR** is often used in measuring language acquisition in infants or learners of a second language, to assess if the learner uses the same words over and over, or uses a variety of different words to communicate [102]. We will report TTR calculated in three different versions as in [60]: (1) without any processing of the text; (2) with lemmatization; and (3) with lemmatization and removal of stopwords.

S	L	MovieCaps
-	-	4.2 %
-	✓	3.7 %
✓	-	4.3 %
✓	✓	3.8 %

Table 4.1: *Lexical diversity of captions. S: removal of stopwords; L: lemmatization*

4.2.1 Music related audio-caption pairs

Moviecaps dataset has many music audio-caption pairs. Music captioning could be viewed as separate task, since describing music is different than identifying and describing the audio events present in an audio clip. A music caption could describe the genre, the intensity and even the emotion that the music evokes to the listener. Music captioning could have several useful applications, such as producing descriptions for items in large music catalogers or vast collections of amateur and user-generated content and as automatically generating descriptions of music in films and videos for deaf and hard-of-hearing people. Resent work [59] has been done for music captioning and audio retrieval for music clips with text quires, but unfortunately the dataset used is this work is private.

MovieCaps has 5839 music related audio-caption pairs in total and 986 unique music related captions. We define a caption as music related if it contains one of the words: *music*, *singing*, *guitar*, *drums*, *piano*, *song*, *singing*. These captions usually describe the *genre* of the music, the *language* of the vocalizations. Some examples are the following:

- irish music playing
- upbeat electronic music
- angsty punk music
- upbeat french pop music playing over radio

Data Splitting

In the most part, we follow the splitting recipe used in [18]. We split $\mathbb{D}$ in three non-overlapping splits of 80% – 10% – 10% , termed as train, validation, and testing, respectively. If a word is not appearing in the training split, then the evaluation procedure suffers by having to evaluate on words not known during train. For this reason every word in the captions of $\mathbb{D}$ must appears at the training split and at least in one of the other two splits.

To archive this we split $\mathbb{D}$, 10000 times in sets of splits of 80% – 20%, where 80% corresponds to the training split. We reject the sets of splits that have at least one word appearing only in one of the splits. Ideally, the words with $f_w = 2$ should appear only once in the training split. The other appearance of this word should be in the validation or testing splits. This will prevent having unused words in the training (i.e. words appearing only in the training split) or unknown words in the evaluation/testing process (i.e. words not appearing in the training split). The words with $f_w \geq 3$ should appear $f_w^{train} = \lfloor 0.8 f_w \rfloor$ times in the training split, where 0.8 is the percentage of data in the development split and $\lfloor ... \rfloor$ is the floor function. We calculate the frequency of words in the training split, f_w^{train}, and observe that it is impossible to satisfy the $f_w^t = f_w^{train}$ for the words with $f_w \geq 3$. Therefore, we adopted a tolerance δ_w to the f_w^{train} used as $f_w^{train} \pm \delta_w$:

$$\delta_w = \begin{cases} 1 & \text{iff } f_w \in [4,6], \\ 2 & \text{iff } f_w \in [7,16], \\ 4 & \text{iff } f_w \in [17,20], \\ 8 & \text{iff } f_w \in [29,50], \\ \lfloor 0.2 f_w \rfloor & \text{otherwise} \end{cases}$$

The tolerance means, for example, that we can tolerate a word appearing a total of 3 times in the whole MovieCaps dataset $\mathbb{D}$, to appear 2 times in the training split (appearing 0 times in training split results in the rejection of the split set). This will result to this word appearing in either validation or testing split. To pick the best set of split, we count the amount of words that have a frequency:

- $f_w^t < 2$ iff $f_w^t = 3$
- $f_w^t \notin [\, f_w^{train} - \delta_w, f_w^{train} + \delta_w \,]$

We score, in an ascending fashion, the sets of splits according to that amount of words and we pick the one with the lowest score. This is the final MovieCaps dataset. The figure below displays a histogram of the percentage of words f_w^{split}/f_w in the three different splits.

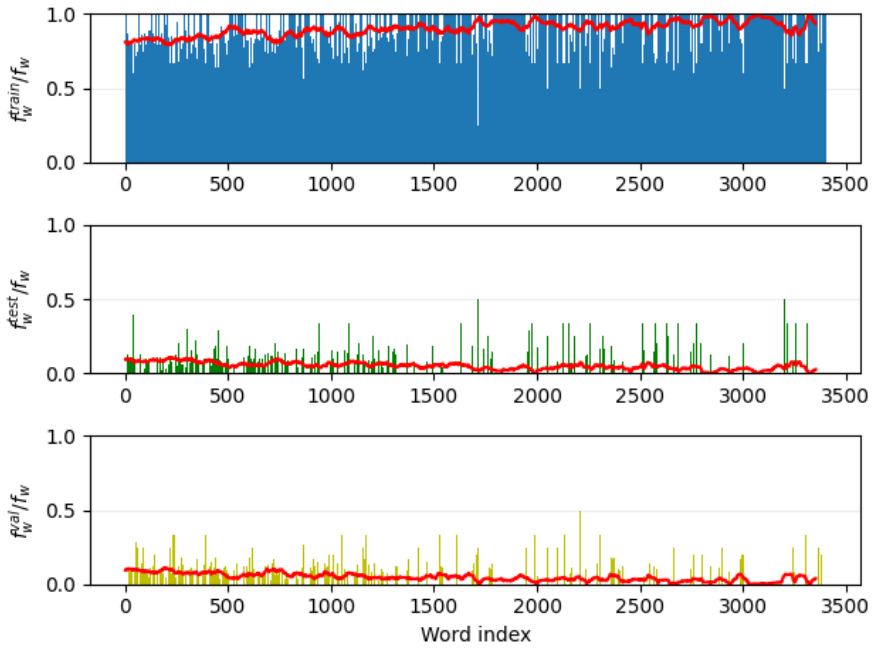


Figure 4.4: *Percentage of words in the three different splits. The red line is a moving average*

Comparison with other Datasets

The number of audio captioning datasets and related work in audio captioning is very small in comparison to the vast amounts of data and related scientific literature available for image and video captioning.

AudioCaps, is a collection of sentence-long descriptions for a subset of AudioSet [23], focused on the audio input. The video was provided to be played if necessary, and the AudioSet tags were presented to the annotator as hints. Clotho was also collected using MTurk using a three-step framework composed of captioning, grammar correction, and rating of the captions [18]. It contains five captions per clip, for audio clips 15 to 30 seconds long that were collected from Freesound [22].

Dataset	Audio Clips	Vocab Size	Unique Captions	Caption Lenght (std)
AudioCaps	57188	5218	52198	9.17 (4.27)
Clotho	5929	4373	29611	11.34 (2.78)
MACS	3930	2775	16262	9.46 (3.89)
MovieCaps	**45432**	**4002**	**8685**	**1.86 (1.12)**

Table 4.2: *Statistics of the MovieCaps and other Audio Captioning datasets.*

MACS [60] is a new audio captioning dataset focuced at minimizing the bias of the audio annotators. A number of 3930 audio-caption pairs, each file being 10-seconds long. The 133 annotators, students taking an audio signal processing course, were randomly assigned a maximum of 131 files each. Annotators were assigned into 30 groups, aiming

that each group will provide annotations to the same set of files.

The main differences of MovieCaps with the rest of audio captioning datasets are the audio and caption lengths. The audio events present in most of the audio clips in MovieCaps are spontaneous.

S	L	AudioCaps	Clotho	MACS	MovieCaps
-	-	1.09 %	1.30 %	1.80 %	**4.2 %**
-	✓	0.79%	0.91 %	1.38 %	**3.7 %**
✓	✓	1.27%	1.66 %	2.17 %	**3.8 %**

Table 4.3: *Compearing Lexical diversity of captions in audio captioning datasets. S: removal of stopwords; L: lemmatization*

Chapter 5

Generating Audio Captiong for Sound Events in Movies

In this Chapter we present our method for automatic audio captioning of sound effects in movies. We could also describe our tasks goal as to automatically generate the audio captions found in SDH (subtitles for deaf and hard of hearing). SDH communicate all audio information, including sound effects, speaker IDs, and non-speech elements such as music. We will only focus on sound events. The overall task is different from audio captioning because a system is called to both identify which segments contain non-speech audio events (detection subtask) and to described them in natural language (captioning subtask).

5.1 Introduction

5.1.1 Subtitles for the Deaf and Hard of Hearing

In 1927, the first sound film was screened: The Jazz Singer. Since then the accessibility of Deaf and hard of hearing audiences has been a challenge yet to be fully overcome. In 1950, the American Schools for the Deaf found a subtitling solution that consisted in superimposing subtitles upon existing print without having to cut and insert intertitles. The subtitles would appear at the bottom of the screen without interrupting the film [19].

SDH include important non-dialogue audio like sound effects and speaker identification, information about the music or the lyrics of a song. SDH subtitles may make note of ambient noises that help create the movies setting. Speakers in the movie may react to off-screen noises; when SDH subtitles point that out, viewers who are deaf or hard of hearing can better understand what is happening in the movie.

5.1.2 Task overview

Our proposed task is a fusion of Automated Audio Captioning (AAC) and Sound Event Detection (SED). SED involves locating and classifying sounds in audio, estimating onset and offset for distinct sound event instances and providing a label for each. Our task involves automatically identifying the onset and offset times of non-speech audio events

in the audio of a movie and providing a textual description. Thus we will refer to this task as Sound Event Detection and Captioning for movies (SEDC).

5.2 Baseline Approach

In this section we present the baseline approach of our system. First the audio signal is retrieved from the video file using the FFMPEG software. Then the audio is segmented to 5-10 second segments and spectrograms of each audio segment is passed to an Audio Tagging model. The Audio Tagging model used for this task is a pretrained PaSST [48]. The PaSST model receives as input a spectrogram and outputs the probabilities for each of the 527 audio classes of AudioSet. The output of the tagging model is shown Figure 5.1.

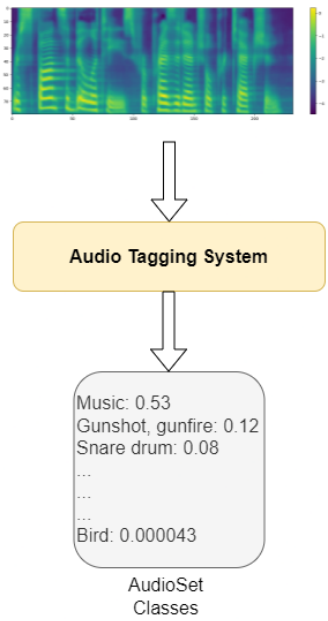


Figure 5.1: *The audio tagging system used for our task outputs a probability over the 527 classes of AudioSet.*

The most probable output label, is selected from PaSST's output distribution, as the tag of the segment. Essentially we use the PaSST model as a speech - non-speech classifier. There are 7 classes in AudioSet that involve speech: *Speech*, *Child speech, kid speaking*, *Female speech, woman speaking*, *Male speech, man speaking*, *Hubbub, speech noise, speech babble*, *Speech synthesizer*. The segments that are classified in one of those classes will not be captioned. In the captioning phase, the extracted spectrograms of the non-speech segments are passed to the AAC model to produce captions. These captions are then aligned to their segment's onset and offset times and a subtitles (SRT) file is created. Figure 5.2 shows the pipeline we propose.

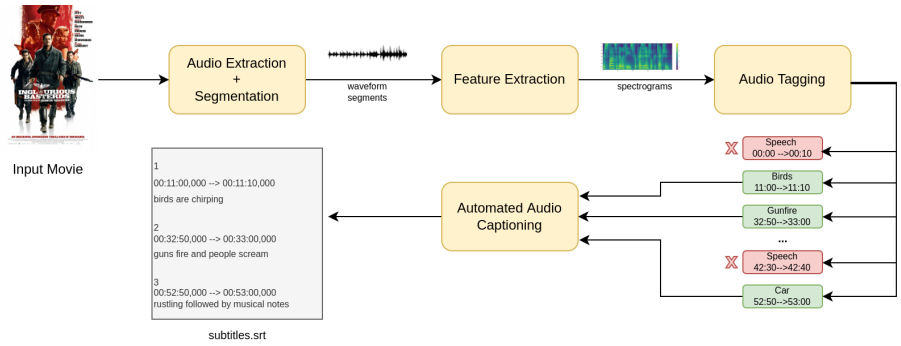


Figure 5.2: *The pipeline of our proposed system: First the audio is extracted from the input movies and is segmented in 5-10 second segments. Spectrograms extracted from each segments are feed to an audio tagging model, to distinguish speech from non-speech segments. Finaly the spectrograms of non-speech segments are feed to an AAC model.*

The first problem of the baseline approach is the selection methodology. As mentioned above the selection of a segment to be captioned is performed depending on the most probable tag that the PaSST model assigned. There are two issues that arise with this approach. First, selecting too many regions to be captioned thus resulting in redundant captions where the audio events described are not salient. Second, missing segments (eg classifying them as Speech or Music) where important but spontaneous audio events (gunshot, lightning) occur but speech is the dominant audio event in the segment.

We observe that always choosing the most probable output from the PaSST model is problematic because the output distribution is often flat, that is, all the output probabilities are relatively similar. So to address this case where the second most probable audio event (that might be worth captioning) is almost as probable as the first, we consider a threshold that the first probability has to break.

We also observe that by changing the length of the audio segments we get different results. By making audio segments smaller the probability to lose a spontaneous but important audio event is smaller. We consider the threshold discussed above and length of the audio segments as hyperparameters.

5.3 Leave no information behind

One major drawback of the baseline approach is that the information obtained by the detection step, is lost and not incorporated in the captioning step. In this section we will present a method that makes it possible to pass this information i.e. the output label of the tagging model as input to the audio captioning model *guiding* the models output. We will refer to this task as Keyword Guided Audio Captioning. We will first describe and present the task then describe the recipe we applied in order to obtain a dataset for keyword guided audio captioning and finally discuss how we integrated the new model in the pipeline described in the previous section.

5.3.1 Keyword Guided Audio Captioning

A major challenge in audio captioning is the word selection indeterminacy. Furthermore, captions of an audio clip can vary depending on the which audio information will be described. A caption may focus only on the most salient audio event, or describe multiple events, including information about the backround noises etc. Realworld audio may contain a varying number of audio events/concepts that may be of interest to the caption consumer, and therefore the optimal description depends on the degree to which the caption covers what the user is interested in at any given moment.

Guided Audio Captioning is a task where the guiding text is provided to the model as an additional input to control the concepts that an audio caption should focus on.

Moreover in a setting where a keyword or tag that classifies the audio is known, the keyword guided caption generation reduces the task complexity by guiding the model to generate a caption based on the given keyword.

5.3.2 Proposed Model

Given an audio and guiding text $K = \{t_1, ..., t_{Tk}\}$ with T tokens, the goal of Guided Audio Captioning is to generate an audio caption $c = \{c_1, ..., C_{Tc}\}$ with tokens, such that c focuses on the concepts provided in K. In our proposed architecture the input guiding text is first tokenized and then the tokens are embedded in the input dimension using a Word2Vec [67] embeddings. Then the embedded sequence is concatenated with the extracted patch sequence to form the final input to the transformer encoder. In order to train a keyword guided audio captioning model we need a dataset with (**A**, **K** , **c**) tuples to search for the optimal model parameters ∂^* by maximizing the likelihood of the ground truth caption:

$$\partial^* = argmax_\partial \sum_{A,K,c} \log p(\mathbf{c}|A, K; \partial)$$

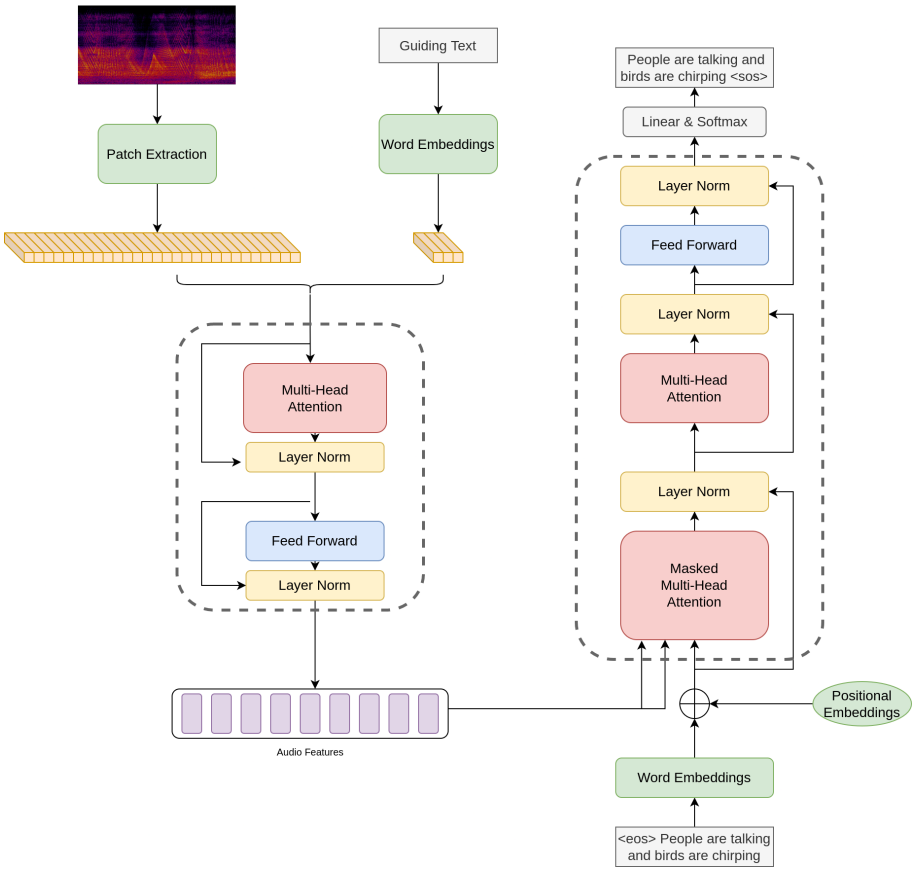


Figure 5.3: *Our proposed architecture for text guided audio captioning: The guiding text is embedded in the input space and concatenated with the extracted patches.*

5.3.3 Building a dataset for text guided audio captioning from AudioCaps

In order to build a dataset for keyword guided audio captioning we introduce a keyword for each audio-caption pair in AudioCaps to function as the input guiding text. The keywords used are the 527 classes of AudioSet $K = \{k_1, k_2, ...k_{527}\}$. To select a keyword k_i for an audio-caption pair (**A**,**c**) in AudioCaps we use a pretrained BERT model. Using BERT we embed each AudioSet class $K_{embed} = \{k_1^e, k_2^e, ..., k_{527}^e\}$. For each caption c_i for $i \in \{1, ...N_{train}\}$ of the audio-caption pairs in the Audiocaps train split, we extract its BERT embedding and find the embedded keyword $k_j^e \in K_{embed}$ that maximizes the cosine similarity. For test and validation splits we extract embeddings from *caption_1*. In this manner we construct tuples (**A**, **c**, **K**) of audio, caption and keyword pairs. This procedure is shown in Algorithm 5.1.

µ 5.1: *Create Caption-Keyword pairs*

$K_{embed} = \{k_1^e, k_2^e, ..., k_{527}^e\}$
$C_{embed} = \{c_1^e, c_2^e, ..., c_N^e\}$
for $c_i \in C_{split}$ **do**
  $m = argmax_{K_{embed}}(S_c(c_i^e, k_j^e)) = \frac{c_i^e \cdot k_j^e}{\|c_i^e\| \|k_j^e\|}$
  $(c_i, k_i) \leftarrow (c_i, k_m)$
end for

Some examples of the caption - keywords pairs constructed with the method above are the following:

Caption	Guiding Keywords	Similarity Score
A gunshot is followed by a click and clack and then a second gunshot	Gunshot, gunfire	0.73
A woman talking followed by a young girl talking while an infant cries	Baby cry, infant cry	0.73
Police car siren starts with two horn blasts then becomes a high pitched wail	Vehicle horn, car horn, honking	0.79
A man speaking while crinkling paper followed by plastic creaking then a toilet flushing	Male speech, man speaking	0.63
A mid-size motor vehicle engine accelerates and is accompanied by hissing and spinning tires, then it decelerates and an adult male begins to speak	Engine knocking	0.62

Table 5.1: *Example of caption - keywords pairs and their cosine similarity created by Algorithm 5.1*

5.3.4 Evaluation of text guided AAC

Evaluation of text guided AAC is not trivial. An evaluation scheme should be able to account whether the generated captions where influenced from the input guiding text. In [70] the authors investigate text guided image captioning and they evaluate their methods with human evaluators. Since we investigate text guided AAC in order to incorporate it to our detection and caption pipeline we don't need to evaluate its ability to produce captioning relevant to the guiding text. What we aim in approach is to generate better captions thus we will conduct the evaluating with regards to the standard quality metrics.

Model	**Quality Metrics**								
	BLUE-1	**BLEU-2**	**BLEU-3**	**BLEU-4**	**METEOR**	**ROUGE_L**	**CIDEr**	**SPICE**	**SPIDEr**
CaptionPaSST	0.68	0.50	0.37	0.26	0.23	0.48	0.71	0.17	0.44
CaptionPaSST + Guiding Text	0.70	0.62	0.41	0.29	0.27	0.52	0.89	0.23	0.56

Table 5.2: *Experiment results on AudioCaps Test set with and without text guidance*

Model	T ∈ *cap*	
	= %	≈ %
captionPaSSTtag	15.02	73.52

Table 5.3: *Percentage cases where the input guiding text T appears verbatim (=) or at least one word from the guiding test appears in the generated caption ().*

It is evident from Tabele 5.2 that the guiding text greatly improves the quality of the generated captions. Also as we observe from Table 5.3 the model behaves as we expected since most of the generated captions contain at least on word from the input text.

5.4 Training a task specific AAC model in MovieCaps

In this section we present how we trained captionPaSST on MovieCaps the task specific dataset we presented in the previous chapter. We created MovieCaps primarily motivated by the fact the sound events found in movies are most of the times different than the audio events in the existing audio captioning datasets. Also the captions are drastically different. Audio captions in movies are sort in length and not very detailed whereas captions in AudioCaps [40] can get up to 22 words long. We apply two training schemes.

A model trained only on MovieCaps and a transfer learning scheme were we pretrain a model on AudioCaps [40] and fine-tune it on MovieCaps with a smaller learning rate. Both models are evaluated on MovieCaps test set. As expected the transfer learning method yields the best results. We also present the results of a zero-shot evaluation on MovieCaps of our model trained only on AudioCaps.

Scheme	**Quality Metrics**								
	BLUE-1	**BLEU-2**	**BLEU-3**	**BLEU-4**	**METEOR**	**ROUGE_L**	**CIDEr**	**SPICE**	**SPIDEr**
Zero-shot	0.02	0.0	0.0	0.0	0.0	0.2	0.3	0.00	0.02
MovieCaps	0.03	0.03	0.01	**0.08**	0.01	0.08	0.38	0.01	0.19
AudioCaps + MovieCaps	**0.04**	**0.03**	**0.03**	0.0	**0.02**	**0.10**	**0.5**	**0.01**	**0.25**

Table 5.4: *Evaluation of different models on MovieCaps*

As it was to be expected the transfer learning method outperforms the other two. We now have a decent model trained on data similar to the target domain of our task i.e. movies.

5.5 Evaluation Metrics

To evaluate the automated movie captioning pipeline we propose a new evaluation metric. As we have already discussed SEDC is composed of two subtasks. A detection subtask were a system aims to automatically identify which audio segments are worth captioning, that is, which audio events provide information that deaf and hard of hearing people would miss if it weren't for the captions. This subtask can be formulated as monophonic sound event detection (SED) problem where there is only one audio event to detect. The second subtask is automatically captioning the selected audio segments. In order to evaluated our proposed system we will experiment with a fusion of SED and AAC evaluation metrics.

5.5.1 Sound Event Detection Evaluation Metrics

Sound event detection involves locating and classifying sounds in audio, estimating onset and offset for distinct sound event instances and providing a textual descriptor for each [66]. The task is referred as polyphonic or monophonic depending on whether the sound events in the audio are overlapping. SED systems are conventionally evaluated with an event error rate (ER) or an F1-score [66].

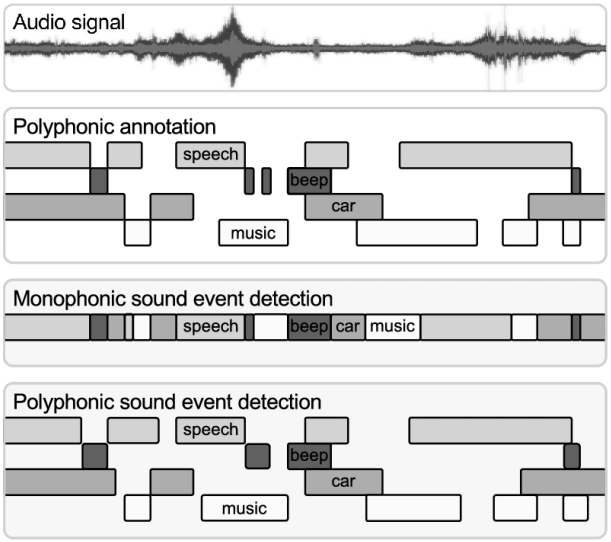


Figure 5.4: *Illustration of the output of monophonic and polyphonic sound event detection systems, compared to the polyphonic annotation. Source([66])*

The detection subtask of SEDC can be viewed as monophonic SED task with a single audio event as shown in the figure below:

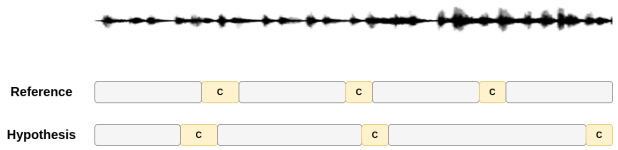


Figure 5.5: *Detection subtask as a monophonic SED task with a single audio event C*

The comparison between the system output and reference can be done in fixed length intervals or at event-instance level. This results in two different ways of measuring performance: segment-based metrics and event-based metrics [66]. We will only present segment-based metrics because in our experiments we observe that evaluating our pipeline with segment-based metrics results to a more robust evaluation scheme.

Segment-Based Metrics

Segment-based metrics compare system output and reference in short time segments. Active/inactive state for each event class is determined in a fixed length interval that represents a segment. Based on the activity, the following intermediate statistics are defined:

- true positive: the reference and system output both indicate an event to be active in that segment
- false positive: the reference indicates an event to be inactive in that segment, but the system output indicates it as active
- false negative: the reference indicates an event to be active in that segment, but the system output indicates it as inactive

Total counts of true positives, false positives and false negatives are denoted as TP, FP and FN, respectively.

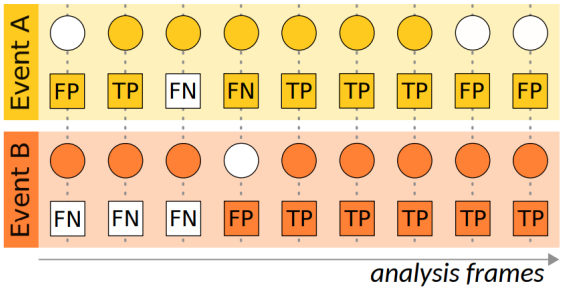


Figure 5.6: *Segment-Based intermediate statistics Source([66])*

Precision, Recall and F-Score

Precision (P) and Recall (R) were introduced in [81] for information retrieval purposes, together with the F-score derived as a measure of effectiveness of retrieval. They are defined as:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN} \tag{5.1}$$

F-score is calculated based on P and R:

$$F = \frac{2 \cdot P \cdot R}{P + R} \tag{5.2}$$

Error Rate

Error rate measures the amount of errors in terms of insertions (I), deletions (D) and substitutions (S). In segment-based error rate we count I, D, and S in every segment. I,D and S for polyphonic SED are defined as follows [66]:

- Insestion: $max(0, FP(k) - FN(k))$
- Deletion: $max(0, FN(k) - FP(k))$
- Substitution: $min(FN(k), FP(k))$

The number of Substitutions is the number of segments where the correct event was not the output of the system but something else was. Insertions are equal equal to false positives after we have counts Substitutions. Respectively Deletions are equal to false negatives after we have counter Substitutions. Total error rate is calculated as follows:

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} S(k)}{\sum_{k=1}^{K} N(k)} \tag{5.3}$$

with N(k) being the number of sound events marked as active in the reference in segment k and K being the total number of segments. The above metric is very similar to DER (Diarization Error Rate), and DER has also been used to evaluate SED systems [89]

5.5.2 A metric for evaluation of Sound Event Detection and Captioning (SEDC)

In this section we will propose an evaluation metric that jointly evaluates the detection and the captioning subtasks of our system, thus evaluating the whole movie captioning pipeline. As we have already discussed our task is different from SED since we aim to generate descriptions for the sound events present in audio and not just label. Thus our task would degenerate to monophonic SED if the captions of the sound events in SDH subtitles where selected from a predefined set of tags or sound events. In the case of SEDC, the descriptions of the same or similar sound events can drastically vary. For example the sound of a thunder storm can be captioned in SDH subtitles as: *Loud thunderstorm*, *Thunderstorm in the backround* etc. We could define our task as a monophonic SED task if we used a different class for every caption in the ground truth and the hypothesis combined. But in that case a *True Posistive* would only occur when the hypothesis predicted an exact same caption with the reference. With the proposed metrics we aim to jointly evaluate if the system identifies which segments to caption correctly and the similarity of the generated captions with the ground truth. So based on the metrics for monophonic SED we propose a variation Error Rate.

Error Rate for SEDC

We redefine the Error Rate to account for caption similarity.

$$ER_{sedc} = \frac{\sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k) - SPIDEr \cdot \sum_{k=1}^{K} O(k)}{\sum_{k=1}^{K} N(k)} \tag{5.4}$$

Substitutions and *Insertion* are defined as in Equation 5.3. *Ovelap* denoted as O in the equation above is the same as TP since we are we are dealing with a monophonic detection. Essentially the number of *Overlaps*, O is the number of segments where both the system and the and the reference predict as caption segments. *SPIDEr* is calculated on the captions of the overlapping segments and it functions as a penalty to Overlap. We choose *SPIDEr* as our the evaluator of the semantic fidelity of captions since it ensures faithfulness to the content, and syntactical fluency.

We can re-write ER_{sedc} in terms of , Deletion and Insertion and *Correct* (C) percentages.

$$C = \frac{SPIDEr \cdot \sum_{k=1}^{K} O(k)}{\sum_{k=1}^{K} N(k)} \tag{5.5}$$

$$I = \frac{\sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)} \tag{5.6}$$

$$D = \frac{\sum_{k=1}^{K} D(k)}{\sum_{k=1}^{K} N(k)} \tag{5.7}$$

$$ER_{sedc} = I + D - C \tag{5.8}$$

Note the this metric can get greater that 1, since *Insertions* can be arbitrarily many.

5.6 Experiments and Results

In this section we report the experiments we conducted for our proposed pipeline for sound event detection and captioning of audio events in movies. We conduct this evaluation on the evaluation metrics we proposed in Section 5.5.2. Our experiments concern our pipelines hyper-parameters such as the length of the audio segments in detection phase, and the AAC model we use in the captioning phase .i.e (1) a model trained only on AudioCaps [40] (zero-shot), (2) a text-guided model trained on the variant of AudioCaps we build. (3) a model trained on MovieCaps and (4) an AudioCaps pretrained model fine-tuned on MovieCaps.

5.6.1 Evaluation of SEDC for movies

For this evaluation we have reserved 50 movies and their corresponding SDH subtitles, that are not part of MovieCaps dataset. We preprocess the SDH subtitles in same way as we did for MovieCaps, excluding captions that concern speaker identification. Both the output of our system and the reference subtitles have the same format:

```
onset time,          offset time,         caption
00:01:12,            00:01:15,            loud explosion
00:05:13,            00:05:15,            pop music playing
...
...
...
01:31:12,            01:31:17,            birds chirping
```

From our experiments we observe that the tagging model in detection phase assigns the tag *Music* very often resulting in a high insertion error. We suspect this happens because generally in movies and series music is a background audio event that can be present in the same time as other spontaneous audio events. To tackle this issue we apply a threshold demanding a high level of certainty from the tagging model in order consider a segments tag as *Music* and thus feed it to the captioning model. Throughout all our experiments we caption a segment which is tagged as *Music* if the output probability of the tag was grater that 0.8. For the remaining non-speech tags the threshold is 0.6

In Table 5.5 we present the evaluation of our pipeline with four audio captioning models. In this experiment we used a segment length of 5 seconds. We observe that up to a point decreasing the segment lengths increases the overlapping segments and decreased the overall Error Rate. Unfortunately both PaSST [48] that we utilise as a tagging model for detection and our AAC models, are trained in 10 second audios and straggle to perform with audio clips smaller than 5 seconds long.

AAC Model	**I**	**D**	**C**	**SPIDEr**	**ER**
Zero-shot	0.73	0.83	0.011	0.07	1.548
MovieCaps (from scratch)	0.73	0.83	0.020	0.12	1.539
Guiding Text	0.73	0.83	0.025	0.15	1.534
MovieCaps (AudioCaps pretrained)	0.73	0.83	0.035	0.21	**1.524**

Table 5.5: *Evaluation of the proposed pipeline with four AAC models on our proposed metric. The segments in this experiment are 5 second long.*

Indeed as we expected from the evaluation we conducted on MovieCaps (Section 5.4) the AAC model pre-trained on AudioCaps and fine-tuned on MovieCaps yields the best results. In the following experiment we apply a 7 second segmentation:

AAC Model	**I**	**D**	**C**	**SPIDEr**	**ER**
Zero-shot	0.92	0.71	0.026	0.09	1.603
MovieCaps (from scratch)	0.92	0.71	0.037	0.13	1.593
Guiding Text	0.92	0.71	0.043	0.15	1.587
MovieCaps (AudioCaps pretrained)	0.92	0.71	0.035	0.24	**1.595**

Table 5.6: *Evaluation of the proposed pipeline with four AAC models on our proposed metric. The segments in this experiment are 7 second long.*

The longer audio segments reduce the deletions but increase the insertions. Also the caption quality is slightly better probably because our models where trained in 10 second audio clips and thus perform better with longer audio sequences.

It is evident from the evaluation as shown in the Tables above that insertion and deletion rates of our pipeline are high. This is a result of the detection strategy of our pipeline. In our approach we segment the input audio into fixed length segments and feed them to a tagging model to distinguish between speech and non-speech segments. With this strategy our method is unable to identify the onset and offset segments of the audio events that are naturally variable in length. Moreover, another drawback of our proposed method is that apart from applying a probability threshold to some frequent audio events such as *Music*, it cannot distinguish between salient and insignificant audio events. We will further discuss the limitations of our approach and feature work that could be done to address them in Section 6.2.

Chapter 6

Conclusions and Future Work

In this final chapter we conclude our thesis. We discuss the challenges we tried to address with our approach and present some ideas for feature work that could further improve Automated Audio Captioning and its application on captioning sound events in movies.

6.1 Conclusions

In this work we research a largely unexplored task, that of generating textual descriptions of audio events. This complex task involves identifying audio events present and their spaciotemporal relationships and providing a meaningful description in natural language. As a practical application to Automated Audio Captioning we present a pipeline that given the audio signal of a movie identifies salient audio events and captions them. The aim of this application is to automatically produce subtitles for deaf and hard of hearing.

The main challenge in Automated Audio Captioning is the lack of sufficient data. AudioCaps [40] the largest freely available AAC dataset contains around 46k samples that have proven insufficient to train large and complex models like transformers. To solve the data scarcity problem many recent approaches utilize large models pretrained on AudioSet [23] such as PANNs [46]. This direction has proven to be effective but comes with cost of having to fine-tune very large models with more than 100 million parameters. Another challenge is that other AAC datsets such as Clotho contain audio clips that are up to 30 seconds, making the training of a transformer model difficult due to the quadratic bottleneck.

To address those issues we proposed a transformer encoder-decoder architecture that uses *Patchout* [48], an effective method to significantly reduce the complexity of transformer model by training it on an incomplete input sequence. After conducting an error analysis of a SOTA model for AAC we observed that it is highly repetitive. In order to produce more diverse captions and balance the quality diversity trade-off we experiment with various decoding algorithms. We finally propose an effective way of using Mixup for AAC.

We proposed an application of AAC in movies that involves a novel task that we call Sound Event Detection and Caption (SEDC). In this application we where confronted with

the following challenges. Our AAC model trained on AudioCaps did not perform well in the sound events of movies, and its output captions where too long and informative whereas captions in SDH subtitles are short and concise. To solve this mismatch we created a dataset from movies and SDH subtitles, MovieCaps.

To improve our baseline approach on Sound Event Detection and Captioning in movies we trained a model on MovieCaps from scratch, fine-tuned a model pretrained on AudioCaps, and proposed a AAC model that utilizes the tags extracted from the detection phase as guiding text.

Finally to evaluate our results we proposed an evaluation metric that allows for combined evaluation of SED and AAC.

6.2 Feature Work

6.2.1 On Automated Audio Captioning

One of the main challenges in Automated Audio Captioning is the lack of sufficient data. Altogether, the freely available datasets for AAC contain less then 60k samples. Thus a direction for feature work, can be to collect weakly labeled data from SDH subtitles and movies, or from online available sources. More available data could enable models to learn more robust audio-text representation, such as CLIP in computer vision.

Additionally, Automated Audio Captioning, can be potentially linked with other audio-language multi-modal tasks, audio question answering, text-based audio generation and text guided audio captioning.

In our work we experimented with a text guided audio captioning model to improve the performance of our pipeline for detection and captioning of sound events in movies. We believe that this task is worth further exploration. As a real world application text guided audio captioning could enable accessibility tools for hearing impaired users, who can select a guiding text produced from an upstream audio detector to receive a guided description of their surrounding sounds.

6.2.2 On Sound Event Detection and Captioning

As we have already discussed the main drawback of our proposed approach is applying an arbitrary, fixed length segmentation on the input audio and not directly detecting the onset and offset times of the audio events. In feature work, we aim to utilize the ground truth temporal information of SDH subtitles and train a Sound Event Detection model.

In order to train a Sound Event Detection model for our purposes, we need to build a dataset with strong labels using movies and SDH subtitles. To do so, each caption has to be replaced with a label. One approach we can follow, is to create N classes such as *Speech*, *Rain*, *Music* etc and replace each caption with the most semantically similar label, using a BERT embeddings.

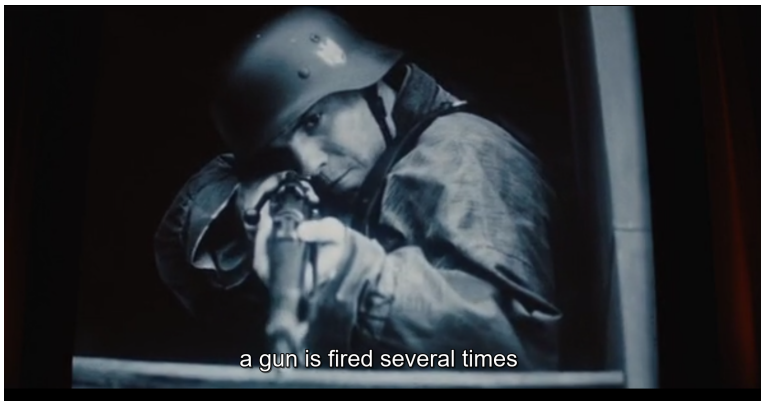
Additionally, we will conduct a survey in order to investigate to what degree our proposed metric correlates with human evaluators.

Παραρτήματα

Appendix A

Examples of the Generated Movie Captions

In this appendix we present some examples of captions generated by our system for the movie Inglourious Basterds. These captions were generated using the AAC model trained on AudioCaps.




clinking of metal


No, it's not controversial at all.
a man and woman speak while music plays

Βιβλιογραφία

[1]

[2] Jimmy Lei Ba, Jamie Ryan Kiros και Geoffrey E. Hinton. *Layer Normalization*, 2016.

[3] Satanjeev Banerjee και Alon Lavie. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, σελίδες 65–72, Ann Arbor, Michigan, 2005. Association for Computational Linguistics.

[4] Swapnil Bhosale, Rupayan Chakraborty και Sunil Kumar Kopparapu. *Automatic Audio Captioning using Attention weighted Event based Embeddings*. *CoRR*, αβσ/2201.12352, 2022.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin και Tomás Mikolov. *Enriching Word Vectors with Subword Information*. *CoRR*, αβσ/1607.04606, 2016.

[6] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*. *Proceedings of COMPSTAT'2010*Yves Lechevallier και Gilbert Saporta, επιμελητές, σελίδες 177–186, Heidelberg, 2010. Physica-Verlag HD.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder κ.ά. . *Language Models are Few-Shot Learners*, 2020.

[8] Emre Cakır, Konstantinos Drossos και Tuomas Virtanen. *Multi-task regularization based on infrequent classes for audio captioning*. *arXiv preprint arXiv:2007.04660*, 2020.

[9] Chen Chen, Nana Hou, Yuchen Hu κ.ά. . *Interactive Audio-text Representation for Automated Audio Captioning with Contrastive Learning*. *arXiv preprint arXiv:2203.15526*, 2022.

[10] Kun Chen, Yusong Wu, Ziyue Wang κ.ά. . *Audio captioning based on transformer and pretrained cnn*. *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop*, σελίδες 21–25, 2020.

[11] KyungHyun Cho, Bartvan Merrienboer, Dzmitry Bahdanau και Yoshua Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. *CoRR*, αβσ/1409.1259, 2014.

[12] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan κ.ά. . *Rethinking Attention with Performers*. *CoRR*, αβσ/2009.14794, 2020.

[13] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho και Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. *CoRR*, αβσ/1412.3555, 2014.

[14] Jia Deng, Wei Dong, Richard Socher κ.ά. . *Imagenet: A large-scale hierarchical image database*. *2009 IEEE conference on computer vision and pattern recognition*, σελίδες 248–255. Ieee, 2009.

[15] Jacob Devlin, Ming Wei Chang, Kenton Lee και Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov κ.ά. . *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. *CoRR*, αβσ/2010.11929, 2020.

[17] Konstantinos Drossos, Sharath Adavanne και Tuomas Virtanen. *Automated audio captioning with recurrent neural networks*. *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, σελίδες 374–378. IEEE, 2017.

[18] Konstantinos Drossos, Samuel Lipping και Tuomas Virtanen. *Clotho: An Audio Captioning Dataset*. *CoRR*, αβσ/1910.09387, 2019.

[19] Boatner Edmund. *Captioned films for the deaf*. *American Annals of the Deaf 96*, σελίδες 346–352, 2020.

[20] Ayşegül Özkaya Eren και Mustafa Sert. *Audio Captioning with Composition of Acoustic and Semantic Information*. *arXiv preprint arXiv:2105.06355*, 2021.

[21] Angela Fan, Mike Lewis και Yann Dauphin. *Hierarchical Neural Story Generation*.

[22] Frederic Font, Gerard Roma και Xavier Serra. *Freesound Technical Demo*. *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, σελίδα 411–412, New York, NY, USA, 2013. Association for Computing Machinery.

[23] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman κ.ά. . *Audio Set: An ontology and human-labeled dataset for audio events*. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, σελίδες 776–780, 2017.

[24] Xavier Glorot, Antoine Bordes και Yoshua Bengio. *Deep Sparse Rectifier Neural Networks*. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*Geoffrey Gordon, David Dunson και Miroslav Dudík, επιμελητές, τόμος 15 στο *Proceedings of Machine Learning Research*, σελίδες 315–323, Fort Lauderdale, FL, USA, 2011. PMLR.

[25] Yuan Gong, Yu-An Chung και James R. Glass. *AST: Audio Spectrogram Transformer*. *CoRR*, αβσ/2104.01778, 2021.

[26] Priya Goyal, Piotr Dollár, Ross Girshick κ.ά. . *Accurate, large minibatch sgd: Training imagenet in 1 hour*. *arXiv preprint arXiv:1706.02677*, 2017.

[27] Alex Graves, Greg Wayne και Ivo Danihelka. *Neural Turing Machines*. *CoRR*, αβσ/1410.5401, 2014.

[28] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu κ.ά. . *Attention Mechanisms in Computer Vision: A Survey*. *CoRR*, αβσ/2111.07624, 2021.

[29] Andrey Guzhov, Federico Raue, Jörn Hees και Andreas Dengel. *ESResNet: Environmental Sound Classification Based on Visual Domain Models*. *CoRR*, αβσ/2004.07301, 2020.

[30] Grzegorz Gwardys και Daniel Michał Grzywczak. *Deep image features in music information retrieval*. *International Journal of Electronics and Telecommunications*, 60(4):321–326, 2014.

[31] Martin T. Hagan, Howard B. Demuth και Mark Beale. *Neural Network Design*. PWS Publishing Co., USA, 1997.

[32] Tatsunori B Hashimoto, Hugh Zhang και Percy Liang. *Unifying Human and Statistical Evaluation for Natural Language Generation*.

[33] Dan Hendrycks και Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*, 2020.

[34] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis κ.ά. . *CNN Architectures for Large-Scale Audio Classification*. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[35] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky κ.ά. . *Improving neural networks by preventing co-adaptation of feature detectors*, 2012.

[36] Ari Holtzman, Jan Buys, Li Du κ.ά. . *THE CURIOUS CASE OF NEURAL TEXT DeGENERATION*.

[37] Shota Ikawa και Kunio Kashino. *Neural Audio Captioning Based on Conditional Sequence-to-Sequence Model*. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York University, 2019.

[38] Sergey Ioffe και Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015.

[39] Daphne Ippolito, Reno Kriz, João Sedoc κ.ά. . *Comparison of Diverse Decoding Methods from Conditional Language Models*. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, σελίδες 3752–3762, Florence, Italy, 2019. Association for Computational Linguistics.

[40] Chris Dongjoo Kim, Byeongchang Kim, Hyunmin Lee και Gunhee Kim. *AudioCaps: Generating Captions for Audios in The Wild*. *Proceedings of the 2019 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), σελίδες 119–132, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[41] Nikita Kitaev, Lukasz Kaiser και Anselm Levskaya. *Reformer: The Efficient Transformer*. *CoRR*, αβσ/2001.04451, 2020.

[42] Philipp Koehn και Rebecca Knowles. *Six Challenges for Neural Machine Translation*. *CoRR*, αβσ/1706.03872, 2017.

[43] A. S. Koepke, A. M. Oncescu, J. Henriques και Albanie S. Akata, Z. *Audio Retrieval with Natural Language Queries: A Benchmark Study*. *IEEE Transactions on Multimedia*, 2022.

[44] Andrew Koh, Xue Fuzhao και Chng Eng Siong. *AUTOMATED AUDIO CAPTIONING USING TRANSFER LEARNING AND RECONSTRUCTION LATENT SPACE SIMILARITY REGULARIZATION*.

[45] Yuma Koizumi, Ryo Masumura, Kyosuke Nishida κ.ά. . *A Transformer-based Audio Captioning Model with Keyword Estimation*, 2020.

[46] Qiuqiang Kong, Yin Cao, Turab Iqbal κ.ά. . *PANNs: Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition*. *CoRR*, αβσ/1912.10211, 2019.

[47] Khaled Koutini, Hamid Eghbal-zadeh και Gerhard Widmer. *Receptive Field Regularization Techniques for Audio Classification and Tagging with Deep Convolutional Neural Networks*. *CoRR*, αβσ/2105.12395, 2021.

[48] Khaled Koutini, Jan Schlüter, Hamid Eghbal-zadeh και Gerhard Widmer. *Efficient Training of Audio Transformers with Patchout*. *CoRR*, αβσ/2110.05069, 2021.

[49] Alex Krizhevsky, Ilya Sutskever και Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Commun. ACM*, 60(6):84–90, 2017.

[50] Anders Krogh και John Hertz. *A Simple Weight Decay Can Improve Generalization*. *Advances in Neural Information Processing Systems*J. Moody, S. Hanson και R.P. Lippmann, επιμελητές, τόμος 4. Morgan-Kaufmann, 1991.

[51] Yann LeCun και Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, σελίδα 255–258. MIT Press, Cambridge, MA, USA, 1998.

[52] Yann LeCun, Yoshua Bengio και Geoffrey Hinton. *Deep learning*. *nature*, 521(7553):436–444, 2015.

[53] Jiwei Li, Michel Galley, Chris Brockett κ.ά. . *A Diversity-Promoting Objective Function for Neural Conversation Models*.

[54] Siqi Liu, Zhenhai Zhu, Ning Ye κ.ά. . *Optimization of image description metrics using policy gradient methods*. *CoRR*, αβσ/1612.00370, 2016.

[55] X. Liu, Q. Huang, X. Mei κ.ά. . *CL4AC: A Contrastive Loss For Audio Captioning. Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE) 2021 Workshop*, 2021.

[56] Yuanzhi Li, Colin Wei και Tengyu Ma. *Towards explaining the regularization effect of initial large learning rate in training neural networks. Advances in Neural Information Processing Systems*, 32, 2019.

[57] Zewen Li, Wenjie Yang, Shouheng Peng και Fan Liu. *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. CoRR*, αβσ/2004.02806, 2020.

[58] Dehong Ma, Sujian Li, Xiaodong Zhang και Houfeng Wang. *Interactive Attention Networks for Aspect-Level Sentiment Classification. CoRR*, αβσ/1709.00893, 2017.

[59] Ilaria Manco, Emmanouil Benetos, Elio Quinton και Gyorgy Fazekas. *MusCaps: Generating Captions for Music Audio. 2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[60] Irene Martin Morato και Annamaria Mesaros. *Diversity and bias in audio captioning datasets. Proceedings of the 6th Workshop on Detection and Classication of Acoustic Scenes and Events (DCASE 2021)*Frederic Font, Annamaria Mesaros, Daniel P.W. Ellis κ.ά. , επιμελητές, σελίδες 90–94, 2021. Δετεστιον ανδ ῞λασσιςατιον οφ Αςουστις Σςενες ανδ Εεντς · ὄνφερενςε δατε: 15-11-2021 Τηρουγη 19-11-2021.

[61] Xinhao Mei, Qiushi Huang, Xubo Liu κ.ά. . *Detection and Classification of Acoustic Scenes and Events 2021 AN ENCODER-DECODER BASED AUDIO CAPTIONING SYSTEM WITH TRANSFER AND REINFORCEMENT LEARNING*.

[62] Xinhao Mei, Xubo Liu, Qiushi Huang κ.ά. . *Audio captioning transformer. arXiv preprint arXiv:2107.09817*, 2021.

[63] Xinhao Mei, Xubo Liu, Qiushi Huang κ.ά. . *Audio Captioning Transformer*, 2021.

[64] Xinhao Mei, Xubo Liu, Mark D Plumbley και Wenwu Wang. *Automated Audio Captioning: an Overview of Recent Progress and New Challenges. arXiv preprint arXiv:2205.05949*, 2022.

[65] Xinhao Mei, Xubo Liu, Jianyuan Sun κ.ά. . *Diverse Audio Captioning via Adversarial Training*, 2021.

[66] Annamaria Mesaros, Toni Heittola και Tuomas Virtanen. *Metrics for Polyphonic Sound Event Detection. Applied Sciences*, 6(6), 2016.

[67] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space. CoRR*, αβσ/1301.3781, 2013.

[68] Reza Moradi, Reza Berangi και Behrouz Minaei. *A survey of regularization strategies for deep models*. τόμος 53, σελίδες 3947–3986, 2020.

[69] Muzammal Naseer, Kanchana Ranasinghe, Salman H. Khan κ.ά. . *Intriguing Properties of Vision Transformers*. *CoRR*, αβσ/2105.10497, 2021.

[70] Edwin G. Ng, Bo Pang, Piyush Sharma και Radu Soricut. *Understanding Guided Image Captioning Performance across Domains*. *CoRR*, αβσ/2012.02339, 2020.

[71] Khoa Nguyen, Konstantinos Drossos και Tuomas Virtanen. *Temporal sub-sampling of audio feature sequences for automated audio captioning*. *arXiv preprint arXiv:2007.02676*, 2020.

[72] Yik Cheung Tam Jiachen Ding Cheng Niu και Jie Zhou. *Cluster-based beam search for pointergenerator chatbot grounded by knowledge*.

[73] Sarah O'Gara και Kevin McGuinness. *Comparing data augmentation strategies for deep image classification*. 2019.

[74] Kamalesh Palanisamy, Dipika Singhania και Angela Yao. *Rethinking CNN Models for Audio Classification*. *CoRR*, αβσ/2007.11154, 2020.

[75] Kishore Papineni, Salim Roukos, Todd Ward και Wei Jing Zhu. *Bleu: a Method for Automatic Evaluation of Machine Translation*. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, σελίδες 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

[76] Daniel S. Park, William Chan, Yu Zhang κ.ά. . *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*. *Interspeech 2019*. ISCA, 2019.

[77] Razvan Pascanu, Tomas Mikolov και Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*, 2013.

[78] Jeffrey Pennington, Richard Socher και Christopher Manning. *GloVe: Global Vectors for Word Representation*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

[79] Sergi Perez-Castanos, Javier Naranjo-Alcazar, Pedro Zuccarello και Maximo Cobos. *Listen carefully and tell: an audio captioning system based on residual learning and gammatone audio representation*. *arXiv preprint arXiv:2006.15406*, 2020.

[80] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh κ.ά. . *Self-critical Sequence Training for Image Captioning*. *CoRR*, αβσ/1612.00563, 2016.

[81] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2νδη έκδοση, 1979.

[82] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. *CoRR*, αβσ/1609.04747, 2016.

[83] David E. Rumelhart, Geoffrey E. Hinton και Ronald J. Williams. *Learning Representations by Back-propagating Errors*. *Nature*, 323(6088):533–536, 1986.

[84] P.Y. Simard, D. Steinkraus και J.C. Platt. *Best practices for convolutional neural networks applied to visual document analysis. Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, σελίδες 958–963, 2003.

[85] Felix Stahlberg και Bill Byrne. *On NMT Search Errors and Model Errors: Cat Got Your Tongue? Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, σελίδες 3356–3362, Hong Kong, China, 2019. Association for Computational Linguistics.

[86] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski και Armand Joulin. *Adaptive Attention Span in Transformers. CoRR*, αβσ/1905.07799, 2019.

[87] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe κ.ά. . *Rethinking the Inception Architecture for Computer Vision. CoRR*, αβσ/1512.00567, 2015.

[88] Daiki Takeuchi, Yuma Koizumi, Yasunori Ohishi κ.ά. . *Effects of word-frequency based pre-and post-processings for audio captioning. arXiv preprint arXiv:2009.11436*, 2020.

[89] Andrey Temko, Climent Nadeu, Dušan Macho κ.ά. . *Acoustic Event Detection and Classification*, σελίδες 61–73. Springer London, London, 2009.

[90] Hugo Touvron, Matthieu Cord, Matthijs Douze κ.ά. . *Training data-efficient image transformers & distillation through attention. CoRR*, αβσ/2012.12877, 2020.

[91] An Tran, Konstantinos Drossos και Tuomas Virtanen. *WaveTransformer: A Novel Architecture for Audio Captioning Based on Learning Temporal and Time-Frequency Information. CoRR*, αβσ/2010.11098, 2020.

[92] Ashish Vaswani, Noam Shazeer, Niki Parmar κ.ά. . *Attention Is All You Need*, 2017.

[93] Ramakrishna Vedantam, C. Lawrence Zitnick και Devi Parikh. *CIDEr: Consensus-based Image Description Evaluation. CoRR*, αβσ/1411.5726, 2014.

[94] Oriol Vinyals, Alexander Toshev, Samy Bengio και Dumitru Erhan. *Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge. CoRR*, αβσ/1609.06647, 2016.

[95] Zhiguo Wang, Patrick Ng, Xiaofei Ma κ.ά. . *Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. CoRR*, αβσ/1908.08167, 2019.

[96] Mengyue Wu, Heinrich Dinkel και Kai Yu. *Audio Caption: Listen and Tell. ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, σελίδες 830–834, 2019.

[97] Feiyang Xiao, Jian Guan, Qiaoxi Zhu κ.ά. . *Local Information Assisted Attention-free Decoder for Audio Captioning. CoRR*, αβσ/2201.03217, 2022.

[98] Xuenan Xu, Heinrich Dinkel, Mengyue Wu και Kai Yu. *What does a Car-ssette tape tell? CoRR*, αβσ/1905.13448, 2019.

[99] Xuenan Xu, Heinrich Dinkel, Mengyue Wu και Kai Yu. *A crnn-gru based reinforcement learning approach to audio captioning. Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, σελίδες 225–229, 2020.

[100] Zhongjie Ye, Helin Wang, Dongchao Yang και Yuexian Zou. *Detection and Classification of Acoustic Scenes and Events 2021 IMPROVING THE PERFORMANCE OF AUTOMATED AUDIO CAPTIONING VIA INTEGRATING THE ACOUSTIC AND TEXTUAL INFORMATION Technical Report*.

[101] Zhongjie Ye, Helin Wang, Dongchao Yang και Yuexian Zou. *Improving the Performance of Automated Audio Captioning via Integrating the Acoustic and Semantic Information. CoRR*, αβσ/2110.06100, 2021.

[102] G. Youmans. *Measuring lexical style and competence: The type-token vocabulary curve*.

[103] Li Yuan, Yunpeng Chen, Tao Wang κ.ά. . *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet. CoRR*, αβσ/2101.11986, 2021.

[104] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin και David Lopez-Paz. *mixup: Beyond Empirical Risk Minimization. CoRR*, αβσ/1710.09412, 2017.

[105] Jingzhao Zhang, Tianxing He, Suvrit Sra και Ali Jadbabaie. *Why gradient clipping accelerates training: A theoretical justification for adaptivity*, 2020.

[106] Yizhe Zhang, Michel Galley, Jianfeng Gao κ.ά. . *Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization*.