National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

MULTI-STAGE UNSUPERVISED DOMAIN ADAPTATION FOR AUTOMATIC SPEECH RECOGNITION

Diploma Thesis

of

DIMITRIOS DAMIANOS

Supervisor: Alexandros Potamianos
Associate Professor NTUA

Athens, October 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

MULTI-STAGE UNSUPERVISED DOMAIN ADAPTATION FOR AUTOMATIC SPEECH RECOGNITION

Diploma Thesis

of

DIMITRIOS DAMIANOS

Supervisor: Alexandros Potamianos
Associate Professor NTUA

Approved by the examination committee on 18th October 2024.

(Signature)	*(Signature)*	*(Signature)*
Alexandros Potamianos	Athanasios Rodogiannis	Athanasios Voulodimos
Associate Professor NTUA	Associate Professor NTUA	Assistant Professor NTUA

Athens, October 2024

National Technical University of Athens
School of Electrical and Computer Engineering
Department of Signals, Control and Robotics

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signatures)

.....
Dimitrios Damianos

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών

18th October 2024

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως στόχο τη μελέτη της μη εποπτευόμενης προσαρμογής πεδίου (unsupervised domain adaptation) για την Αυτόματη Αναγνώριση Ομιλίας (Automatic Speech Recognition). Στο πλαίσιο της μη εποπτευόμενης προσαρμογής πεδίου, εργαζόμαστε με δύο διακριτές κατανομές δεδομένων, το πεδίο-πηγή και τον πεδίο-στόχο. Ενώ και τα δύο πεδία διαθέτουν δεδομένα εισόδου, οι αντίστοιχες ετικέτες είναι προσβάσιμες μόνο στο πεδίο-πηγή. Στόχος είναι η ανάπτυξη ενός μοντέλου που να μπορεί να εφαρμοστεί αποτελεσματικά στο πεδίο-στόχο, χρησιμοποιώντας τα δεδομένα και των δύο διαθέσιμων πεδίων. Στην παρούσα διατριβή συζητάμε τις βασικές αρχές της μηχανικής μάθησης και τις προκλήσεις που σχετίζονται με την αναγνώριση ομιλίας, καλύπτοντας τόσο τις παραδοσιακές όσο και τις σύγχρονες προσεγγίσεις. Στη συνέχεια, εξετάζουμε τη βιβλιογραφία σχετικά με τις μεθόδους προσαρμογής πεδίου, κατηγοριοποιώντας αυτές τις προσεγγίσεις σε τρεις κύριες ομάδες, συμπεριλαμβανομένων των τεχνικών ημι-εποπτευόμενης μάθησης (semi-supervised learning) και αυτο-επίβλεψης (self-supervision).

Στην παρούσα εργασία, διερευνούμε τις δυνατότητες του Meta PL - μιας τεχνικής προσαργμογής πεδίου που έχει εφαρμοστεί στην αναγνώριση εικόνας - στην Αυτόματη Αναγνώριση Φωνής. Επιπλέον, εισάγουμε μια μεθοδολογία δύο σταδίων που συνδυάζει στρατηγικές αυτο-επίβλεψης με τεχνικές ημι-εποπτευόμενης μάθησης, η οποία έχει σχεδιαστεί για να ενισχύσει τη γενίκευση των μοντέλων Αυτόματης Αναγνώρισης Φωνής σε γλώσσες με λίγα διαθέσιμα δεδομένα, όπως η ελληνική, καθώς και σε δεδομένα με ετικέτες χαμηλής ποιότητας. Τα πειράματα μας δείχνουν ότι το Meta PL μπορεί να εφαρμοστεί επιτυχώς σε εφαρμογές Αυτόματης Αναγνώρισης Φωνής, προσφέροντας αποτελέσματα ανταγωνιστικά με προηγούμενες μεθόδους, καθώς οδηγεί σε σχετική βελτίωση της μετρικής WER κατά 4%. Επιπλέον, δείχνουμε ότι η μέθοδός μας υπερέχει σημαντικά άλλων επιλεγμένων προσεγγίσεων, προσφέροντας μια πιο αποτελεσματική λύση στο πρόβλημα προσαρμογής πεδίου στην Αυτόματη Αναγνώριση Φωνής, καθώς προσφέρει βελτίωση της μετρικής WER της τάξης του 7%. Τέλος, εξετάζουμε τους περιορισμούς σχετικά με την ενχωμάτωση της αυτόεπιβλεπόμενη μάθησης με την ημιαυτόεπιβλεπόμενη εκπαίδευση στο πλαίσιο του Meta PL και καταλήγουμε στο συμπέρασμα ότι οι αυτοεπιβλεπόμενες τεχνικές πρέπει να εφαρμόζονται ξεχωριστά από την ημιαυτόεπιβλεπόμενη μάθηση.

Λέξεις Κλειδία

Μη-εποπτευόμενη Προσαρμογή Πεδίου, Αυτόματη Αναγνώριση Φωνής, Αυτό-επίβλεψη, Ημί-επίβλεψη, Ψευδοσήμανση

Abstract

The purpose of this diploma thesis is to study unsupervised domain adaptation for Automatic Speech Recognition. In the context of unsupervised domain adaptation, we work with two distinct data distributions, the source domain and the target domain. While both domains have available input data, corresponding labels are only accessible in the source domain. The goal is to develop a model that can be effectively applied to the target domain, leveraging both the available labeled and unlabeled data. In this dissertation, we discuss the fundamentals of machine learning and the challenges associated with speech recognition, covering both traditional and modern approaches. We then review the literature on domain adaptation methods, categorizing these approaches into three major groups, including semi-supervised learning and self-supervision techniques.

In the present work, we explore the capabilities of the Meta PL domain adaptation framework - previously applied to image recognition task- for Automatic Speech Recognition. Additionally, we introduce Multi-Stage Domain Adaptation, a two-stage domain adaptation method that combines self-supervised strategies with semi-supervised techniques. Multi-Stage Domain Adaptation is designed to enhance the robustness and generalization of Automatic Speech Recognition models in the context of low-resource languages, such as Greek, and weakly supervised data where labeled data is scarce or noisy. Our extensive experiments show that Meta PL can be effectively applied to Automatic Speech Recognition tasks, resulting in an average WER improvement of 4%. Additionally, we demonstrate that Multi-Stage Domain Adaptation outperforms our baselines WER by 7% on average, providing a more robust solution for domain adaptation in Automatic Speech Recognition, especially in underrepresented linguistic settings. Finally, we examine the limitations of integrating self-supervised tasks with semi-supervised training within the Meta PL framework and conclude that self-supervised tasks should be applied separately from semi-supervised training.

Keywords

Unsupervised Domain Adaptation, Automatic Speech Recognition, Self-supervision, Semi-supervision, Pseudo-labeling

Dedicated to the memory of my grandfather Konstantinos and grandmother Konstantina

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Αλέξανδρο Ποταμιάνο, για την πολύτιμη συνεργασία και καθοδήγηση καθ' όλη τη διάρκεια της διπλωματικής μου εργασίας.

Επίσης, ευχαριστώ θερμά τον υποψήφιο διδάκτορα και συνεργάτη μου, Γιώργο Παρασκευόπουλο, για την άριστη συνεργασία, τις συμβουλές και την καθοδήγησή του. Η συμβολή του υπήρξε καθοριστική για την ολοκλήρωση αυτής της εργασίας, ενώ αποτέλεσε για μένα πρότυπο ερευνητή και μηχανικού.

Παράλληλα, θα ήθελα να ευχαριστήσω τους φίλους μου και τη Λίντα. Με την ολοκλήρωση αυτής της εργασίας κλείνει ένας κύκλος σπουδών πέντε ετών, κατά τη διάρκεια του οποίου η φιλία, η συντροφικότητα και η καθημερινή στήριξή τους έκαναν τις ατελείωτες ώρες μελέτης και άγχους πιο υποφερτές. Θερμές ευχαριστίες και στον οικογενειακό μας φίλο Βασίλη για την έμπνευση και τη στήριξή του.

Τέλος, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου, Γιώργο και Σωτηρία. Οι συμβουλές τους, η αδιάκοπη στήριξη και τα εφόδια που μου παρείχαν είναι ο λόγος που αυτή η εργασία φέρει την υπογραφή μου.

Αυτή η διπλωματική εργασία αφιερώνεται στη μνήμη του παππού μου Κωνσταντίνου και της γιαγιάς μου Κωνσταντίνας.

Αθήνα, Οκτώβριος 2024

Δημήτριος Δαμιανός

Table of Contents

List of Figures

List of Tables

Chapter 0

Ελληνική περίληψη

0.1 Εισαγωγή

Η Αυτόματη Αναγνώριση Ομιλίας (Automatic Speech Recognition) αποτελεί έναν από τους βασικούς τομείς όπου οι μοντέλα μηχανικής και βαθιάς μάθησης έχουν εφαρμοστεί επιτυχώς, επιτρέποντας μια σειρά καθημερινών εμπορικών εφαρμογών όπως οι φωνητικοί βοηθοί και τα συστήματα απομαγνητοφώνησης. Ωστόσο, η απόδοση αυτών των συστημάτων μπορεί να επιδεινωθεί απότομα όταν τα δεδομένα στο πεδίο εφαρμογής διαφέρουν σημαντικά από τα δεδομένα εκπαίδευσης. Ως εκ τούτου, είναι αναγκαίο να χρησιμοποιηθούν τεχνικές προσαρμογής πεδίου (domain adaptation), ωστε να διατηρηθεί η απόδοση των συστημάτων αυτών.

Οι μέθοδοι Μη-εποπτευόμενης Προρμογής Πεδίου (Unsupervised Domain Adaptation) έχουν ιδιαίτερο ενδιαφέρον, καθώς δεν εξαρτώνται από την ακριβή και χρονοβόρα αναγνώριση δεδομένων ανά συγκεκριμένο πεδίο. Αντίθετα, εστιάζουν στη χρήση μεγάλων ποσοτήτων μη επισημασμένων δεδομένων (unlabeled data) για να βοηθήσουν τα μοντέλα να προσαρμοστούν σε νέες, άγνωστες περιοχές. Οι πιο κοινές προσεγγίσεις για την μη-εποπτευόμενη προσαρμογή είναι η αυτοεποπτεία και η ημιεποπτευόμενη μάθηση. Πρόσφατες έρευνες [5, 8] έχουν δείξει ότι αυτές οι τεχνικές οδηγούν σε σημαντικές βελτιώσεις στην απόδοση σε μια σειρά προβλημάτων, συμπεριλαμβανομένης της αναγνώρισης ομιλίας και εικόνας. Ένα βασικό ερώτημα που προκύπτει είναι αν αυτές οι μέθοδοι μπορούν να συνδυαστούν αποτελεσματικά για να ενισχύσουν περαιτέρω την προσαρμογή και να επιτύχουν ακόμη καλύτερα αποτελέσματα.

Ο κύριος ερευνητικός στόχος αυτού του έργου είναι η ανάπτυξη μιας μεθόδου μη εποπτευόμενης προσαρμογής τομέα για προβλήματα αναγνώρισης ομιλίας στο πλαίσιο γλωσσών με περιορισμένους πόρους και δεδομένων με κακής ποιότητας επισημάνσης (weakly labeled data). Αυτή η μέθοδος συνδυάζει τεχνικές αυτοεποπτείας που έχουν ήδη εφαρμοστεί επιτυχώς στην αναγνώριση φωνής, με ημιεποπτευμένες μεθόδους που έχουν χρησιμοποιηθεί κυρίως στην αναγνώριση εικόνας, προσαρμόζοντάς τις στην αναγνώρισης φωνής.

Οι κύριες συνεισφορές μας είναι: (1) δείχνουμε ότι μια τεχνική ημι-επίβλεψης που μέχρι στιγμής έχει εφαρμοστεί σε προβήματα αναγνώρισης εικόνας, μπορεί επιτυχός να εφαρμοστεί σε προβλήματα αναγνώρισης φωνής, (2) προτείνουμε μια νέα μέθοδος μη εποπτευόμενης προσαρμογής, η οποία συνδυάζει την προ-αναφερθείσα τεχνική ημι-επίβλεψης με στρατηγικές αυτοεπίβλεψης, και (3) εξετάζουμε τους περιορισμούς αυτής της τεχνικής ημιεπίβλεψης όταν συνδυάζεται με μεθόδους αυτοεπίβλεψης. Η ερευνά μας επικεντρώνεται σε γλώσσες με λίγα δεδομένα, όπως τα Ελληνικά, και σε δεδομένα με επισημάνσης κακής ποιότητας. Η προσέγγιση μας πετυχαίνει

να βελτιώσει αποτελεσματικά την προσαρμογή ακόμα και σε αυτά τα γλωσσικά περιβάλλοντα.

0.2 Αυτόματη Αναγνώριση Φωνής

Το πρόβλημα αναγνώρισης φωνής [9, 10] έγκειται στην ανάπτυξη ενός συστήματος που μετατρέπει το σήμα φωνής σε κείμενο, το οποίο αποτελείται από δύο στάδια: Στο πρώτο στάδιο γίνεται η επεξεργασία του φωνητικού σήματος ώστε να παραχθεί μια ακολουθία χαρακτηριστικών όπως οι Συντελεστές Mel-Frequency Cepstral (Mel-Frequency Cepstral Coefficients) [10]. Στη συνέχεια ένας γλωσσικός αποκωδικοποιήτης ερμηνεύει αυτά τα χαρακτηριστικά και τα αποκωδικοποιεί σε μια ακολουθία χαρακτήρων ή λέξεων, παράγοντας το κείμενο που προκύπτει.

Η Αυτόματη Αναγνώριση Ομιλίας αντιμετωπίζεται ως πρόβλημα ακολουθία-σε-ακολουθία, καθώς περιλαμβάνει την αντιστοίχιση μιας ακολουθίας επεξεργασμένων χαρακτηριστικών εισόδου σε μια αντίστοιχη αλληλουχία λέξεων. Παρακάτω, εξερευνούμε συμβατικές μεθόδους που έχουν σχεδιαστεί για να ολοκληρώσουν αυτήν την εργασία.

0.2.1 Συμβατικές μέθοδοι

Στις συμβατικές μεθόδους, η αναγνώριση φωνής αντιμετωπίζεται σαν στατιστικό πρόβλημα [10], όπου ο στόχος είναι να βρεθεί η πιο πιθανή ακολουθία λέξεων/φωνημάτων W δεδομένης μια ακολουθίας χαρακτηριστικών X. Με την βοήθεια του κανόνα του Bayes, αναζητούμε την ακολουθία $\hat{W}$ η οποία μεγιστοποιεί την πιθανότητα:

$$\hat{W} = \arg\max_{W} P(W|X) = arg\max_{W} P(X|W)P(W) \tag{1}$$

Όπου ο όρος $P(X)$ αγνοείται καθώς είναι ανεξάρτητος του W. Η πιθανότητα $P(X|W)$ παράγεται από ένα ακουστικό μοντέλο, ενώ ο όρος $P(W)$ από ένα γλωσσικό μοντέλο.

Κρυφά Μοντέλα Μαρκόβ

Τα Κρυφά Μοντέλα Μαρκόβ (Hidden Markov Models) [11, 10] χρησιμοποιούνται κατά κόρων για να μοντελοποιήσουν ακολουθίες γεγονότων που μεταβάλλονται στο χρόνο, και στην αναγνώριση φωνής χρησιμοποιούνται σαν ακουστικά μοντέλα. Αποτελούνται από μια σειρά καταστάσεων s_i , που αντιπροσωπεύουν τις διαφορετικές λέξεις ή φωνήματα, όπου κάθε κατάσταση μπορεί να 'επιστρέψει' κάποια παρατήρηση o_t, η οποία μοντελοποιεί το αναγνωρισμένο φώνημα ή λέξη. Έχουν δύο βασικές παραμέτρους, τις πιθανότητες μετάβασης μεταξύ καταστάσεων s_i και τις πιθανότητες για τις παρατηρήσεις o_t. Βασίζονται στην Μαρκοβιανή Ιδιότητα, και για την εκπαίδευση τους χρησιμοποιούν τον αλγόριθμο Baum-Welch ή Forward-Backward. Μετά την εκπαίδευση, χρησιμοποείται ο αλγόριθμος Viterbi για να καθοριστεί ποια ακολουθία λέξεων ή φωνημάτων έχει αναγνωριστεί.

HMM-GMM

Σημαντική παράμετρος των Κρυφών Μοντέλων Μαρκόβ είναι η μοντελοποίηση των πιθανοτήτων των παρατηρήσεων o_t, καθώς αυτές ορίζουν την αναγνώριση κάθε φωνήματος με δεδομένη φωνή στην είσοδο. Σύμφωνα με το μοντελό HMM-GMM, το Κρυφό Μοντέλο Μαρκόβ -

Μοντέλο Μίξης Γκαουσιανών (Hidden Markov Model - Gaussian Mixture Model) [11], αυτή η πιθανότητα μοντελοποιείται σαν 'μίξη' διαφορετικών Γκαουσιανών κατανομών. Οι παράμετροι αυτής της μίξης κατανομών εκπαιδεύονται με την βοήθεια αλγρίθμων όπως τον Expectation Maximization [11]. Χάρη στην χρήση της μίξης Γκαουσιανών κατανομών, και των Κρυφών Μοντέλων Μαρκόβ, διαμορφώνεται το συνολικό μοντέλο HMM-GMM.

N-grams

Η πιο απλή προσέγγιση για την δημιουργία γλωσσικού μοντέλου είναι να υπολογίσουμε πόσο συχνά εμφανίζεται ένας όρος W στα διαθέσιμα δεδομένα, σε ένα παράθυρο $N-1$ άλλων όρων. Ακριβώς αυτό υλοποιεί το N-gram [10], καθώς υπολογίζει την πιθανότητα $P(W)$ με βάση την εμφάση του όρου W σε σχέση με τους $N-1$ προηγούμενους όρους.

0.2.2 Σύγχρονες μέθοδοι

DNN-HMM

Τα DNN-HMM, τα Βαθιά Νευρωνικά Δίκτυα - Κρυφά Μοντέλα Μαρκόβ (Deep Neural Networks - Hidden Markov Models) [12] χρησιμοποιούν βαθιές αρχιτεκτονικές νευρωνικών δικτύων για να μοντελοποιήσουν την πιθανότητα o_t των Κρυφών Μοντέλων Μαρκόβ, καθώς η προσέγγιση της μίξης Γκαουσιανών κατανομών δεν ανταποκρίνεται στην φύση της ανθρώπινης ομιλίας. Παρόλο την καλύτερη απόδοση τους από τα HMM-GMM μοντέλα, αντικαθιστούνται πλέον από μοντέλα άκρης-σε-άκρη (end-to-end) τα οποία βρίσκουν απευθείας την σύνδεση μεταξύ ομιλίας και κείμενου, χωρίς την ανάγκη προ-ευθυγράμμησης (pre-alignment) μεταξύ κείμένου και ηχητικών χαρακτηριστικών.

Ζητήματα των τεχνικών άκρη-σε-άκρη

Το κύριο πρόβλημα των τεχνικών άκρη-σε-άκρη είναι η ευθυγράμμηση μεταξύ των ακολουθιών είσόδου και εξόδου, καθώς πολλές φορές των ηχητικό σήμα μπορεί να είναι αρκετά μεγαλύτερο του αντίστοιχου κειμένου. Για την αντιμετώπιση αυτού του θέματος, έχουν προταθεί η συνάρτηση εκπαίδευση Συνδεσιονιστικής Χρονικής Ταξινόμησης (Connectionist Temporal Classification) [2] και η αρχιτεκτονική Μετατροπέα Επαναλαμβανόμενου Νευρωνικού Δικτύου (Recurrent Neural Network Transducer) [13].

Συνδεσιονιστική Χρονική Ταξινόμηση

Η Συνδεσιονιστική Χρονική Ταξινόμηση (Connectionist Temporal Classification) [2] είναι μια συνάρτηση κόστους η οποία επιτρέπει σε διάφορες αρχιτεκτονικές βαθιάς μάθησης να λύσουν με επιτυχία το πρόβλημα της αναγνώρισης φωνής. Λύνει το ζήτημα της ευθυγράμμισης (alignment) εισόδου-εξόσου εισάγοντας ένα νέο όρο επισήμανσης, την κενή επισήμανση.

Η κενή επισήμανση χρησιμοποιείται για να πετύχει οποιαδήτε ευθυγράμμιση α μεταξύ είσοδο x και έξοδο y. Παραδείγματος χάρη, αν θεωρήσουμε ώς κενή επισήμανση τον όρο '-', τότε έχουμε τις ευθυγραμμίσης $(a,-,b,-,-,c),(-,a,-,b,c,-)$ για την ακολουθία (a,b,c).

Σκοπός είναι να την ευθυγράμμιση α που μεγιστοποιεί την πιθανότητα $P(y|x)$. Αρχικά πρέπει να ορίσουμε την πιθανότητα κάθε ευθυγράμμισης α, $P(a|x) = \prod_{t=1}^{T} P(a_t, t|x)$, όπου a_t είναι η

παραγόμενη ετικέτα για το βήμα t. Ορίζουμε την πιθανότητα $P(y|x)$ ως $P(y|x) = \sum_{a \in B} P(a|x)$, όπου B είναι το σύνολο όλων των δυνατών ευθυγραμμίσεων μεταξύ της εξόδου y και της εισόδου x. Η ιδέα πίσω από την αναζήτηση όλων των δυνατών ευθυγραμμίσεων είναι ότι, επειδή δεν γνωρίζουμε πού ακριβώς πρέπει να εμφανίσουμε τις αντίστοιχες ετικέτες, αθροίζουμε όλα τα μέρη όπου θα μπορούσαν να εμφανιστούν. Δεδομένης μια ακολουθίας-στόχος y^*, η συνάρτηση κόστους που προκύπτει είναι $L_{CTC} = -logP(y^*|x)$. Ο βασικός περιορισμός της συνάρτησης κόστους Συνδεσιονιστικής Χρονικής Ταξινόμησης είναι ότι υποθέτει ανεξαρτησία μεταξύ των αναγνωρισμένων λέξεων, συνεπώς δεν μπορεί να αποτυπώσει καλά γλωσσικές αλληλεξαρτήσεις, ενώ υποθέτει ότι το μήκος της εξόδου είναι το πολύ ίσο με την έξοδο, δηλαδή απαιτεί να ισχύει $|y| \leq |x|$. Για να αντιμετωπιστούν αυτοί οι περιορισμοί, αναπτύχθηκε η αρχιτεκτονική Μετατροπέα Επαναλαμβανόμενου Νευρωνικού Δικτύου.

Μετατροπέας Επαναλαμβανόμενου Νευρωνικού Δικτύου

Ο Μετατροπέας Επαναλαμβανόμενου Νευρωνικού Δικτύου (Recurrent Neural Network Transducer) [13] αντιμετωπίζει τους προηγούμενους περιορισμούς προσφέροντας έναν πιο αποδοτικό τρόπο μοντελοποίησης των εξαρτήσεων μεταξύ εισόδου και εξόδου. Χρησιμοποιεί δύο δίκτυα: το δίκτυο κειμένου (transcription network) που επεξεργάζεται τις ακουστικές εισόδους και το δίκτυο πρόβλεψης (prediction network) που επεξεργάζεται τις εξόδους, ώστε να προβλέψει πιθανότητες για κάθε συμβολοσειρά. Εισάγει το ειδικό σύμβολο Ø που χρησιμοποιείται όπως η κένη επισήμανση για τη στοίχιση εισόδου x και εξόδου y.

Για τον υπολογισμό της πιθανότητας $P(y|x)$ χρησιμοποιούνται τα γλωσσικα και ακουστικά χαρακτηριστικα που προκύπτουν από τα δύο υποδίκτυα. Το μοντέλο παράγει όλες τις πιθανές ευθυγραμμίσεις μεταξύ εισόδου και εξόδου, και χρησιμοποιεί έναν αλγόριθμο forward-backward για να υπολογίσει την πιθανότητα μιας δεδομένης εξόδου. Οι μεταβλητές forward και backward υπολογίζονται επαναληπτικά, ενώ η συνολική πιθανότητα προκύπτει από την πιθανότητα στο τελικό βήμα. Κατά την εκπαίδευση, στόχος είναι η μεγιστοποίηση της πιθανότητας της επιθυμητής εξόδου $P(y^*|x)$ και η απώλεια (loss) ορίζεται από τη συνάρτηση log-loss, $L_{RNNT} = -lnP(y^*|X)$.

Χάρη στα δύο υποδίκτυα, αυτή η προσέγγιση μπορεί να καλύψει καλύτερα τις συσχετίσεις μεταξύ εισόδου-εξόδου και εξόδου-εξόδου, προσφέροντας μια πιο ισχυρή μοντελοποίηση του προβλήματος αναγνώρισης φωνής.

Wav2Vec2.0

Μια άλλη προσέγγιση για τη βελτίωση της αναγνώρισης φωνής είναι η εξαγωγή πιο ισχυρών χαρακτηριστικών από το ηχητικό σήμα, η οποία βελτιώνει την ποιότητα της αναγνώρισης και ευθυγράμμισης στις μεθόδους που αναφέρθηκαν προηγουμένως. Η καλύτερη εξαγωγή χαρακτηριστικών επιτρέπει στα μοντέλα να συλλάβουν περισσότερες πληροφορίες από το σήμα της ομιλίας, με αποτέλεσμα την καλύτερη αναγνώριση. Το Wav2Vec 2.0 [3] είναι μια αρχιτεκτονική αυτοεπιβλεπόμενης μάθησης που αναπτύχθηκε για την αυτόματη αναγνώριση ομιλίας, σχεδιασμένη να ανακαλύπτει και να εξάγει αυτόματα χαρακτηριστικά υψηλού επιπέδο από δεδομένα ομιλίας. Η βασική ιδέα πίσω από το Wav2Vec 2.0 είναι να αναπαράγει την προσέγγιση του Μασκαρισμένου Γλωσσικού Μοντέλου (Masked Language Modeling) [14] από την Επεξ-

εργασία Φυσικής Γλώσσας (Natural Language Processing), για δεδομένα ομιλίας. Το μοντέλο εκπαιδεύεται να προβλέπει ποια ψευδο-φωνήματα αντιστοιχούν στα τμήματα που έχουν "καλυφθεί", επιτρέποντας έτσι την αυτοεπιβλεπόμενη εκμάθηση νοημάτων από τον ήχο.

0.3 Μη-εποπτευόμενη Προσαρμογή Πεδίου

0.3.1 Ορισμός προβλήματος

Το πρόβλημα της Μη Εποπτευόμενης Προσαρμογής Τομέα (Unsupervised Domain Adaptation) μπορεί να οριστεί ως εξής: Δεδομένες δύο διαφορετικές κατανομές, την κατανομή πηγής $S(x, y)$ και την κατανομή στόχου $T(x, y)$, όπου $x \in X$ είναι διανύσματα χαρακτηριστικών που ανήκουν σε έναν πραγματικό χώρο X και $y \in Y$ είναι ετικέτες από ένα πεπερασμένο σύνολο Y, ο στόχος είναι η εκπαίδευση ενός μοντέλου που να αντιστοιχεί τα διανύσματα χαρακτηριστικών της κατανομής στόχου x_T στις αντίστοιχες ετικέτες τους y_T. Κατά την εκπαίδευση, έχουμε πρόσβαση μόνο σε δείγματα με ετικέτες από την κατανομή πηγής $S(x, y)$, ενώ οι ετικέτες της κατανομής στόχου $T(x, y)$ απουσιάζουν. Η πρόβλημα είναι η ανάπτυξη ενός μοντέλου που μπορεί να γεφυρώσει το χάσμα μεταξύ των δύο κατανομών χρησιμοποιώντας τα δεδομένα με ετικέτες της πηγής για να προβλέψει ετικέτες για τα δεδομένα της κατανομής στόχου, παρά την απουσία ετικετών για την τελευταία.

0.3.2 Τεχνικές Δασκάλου-Μαθητή

Η εκπαίδευση με τη μέθοδο δασκάλου-μαθητή (Teacher-Student) ή η χρήση ψευδοετικετών (pseudo-labeling) είναι η πιο κοινή προσέγγιση για την Μη Εποπτευόμενη Προσαρμογή Τομέα και μία από τις πρώτες μορφές ημι-εποπτευόμενης μάθησης. Στη μέθοδο αυτή, το ήδη εκπαιδευμένο μοντέλο δάσκαλος στο πεδίο της πηγής, παράγει ψευδοετικέτες για το πεδίο στόχο, τις οποίες το μοντέλο-μαθητής χρησιμοποιεί για εποπτευόμενη εκπαίδευση. Δεδομένου ότι το μοντέλο-δάσκαλος έχει εκπαιδευτεί μόνο στο πεδίο-πηγή, προκύπτει το ζήτημα ο μαθητής να εκπαιδεύεται σε λανθασμένες ετικέτες. Για να αντιμετωπιστεί αυτό το ζήτημα, έχουν προταθεί τεχνικές φιλτραρίσματος, οι οποίες απορίπτουν δεδομένα που τυχόν να οδηγήσουν σε λανθασμένες ετικέτες. Στο [15, 16] έχει προταθεί ένα Μοντέλο Εκτίμησης Εμπιστοσύνης (Confidence Estimation Module), το οποίο χρησιμοποιείται για το φιλτράρισμα αναξιόπιστων δεδομένων. Στο [17] χρησιμοποιείται μια συνάρτηση κόστους πολλαπλών εργασιών (multi-task objective loss) η οποία χρησιμοποιεί σφάλμα εμπιστοσύνης. Τέλος, στο [18] έχουμε την Εκπαίδευση Θορυβόμενου Μαθητή (Noisy Student Training), όπου το μοντέλο-μαθητής εκπαιδεύεται σε δεδομένα με προσθήκη θορύβου, μέσα από τεχνικές όπως η SpecAugment [19], που στοχεύει στην εκμάθηση πιο ισχυρών αναπαραστάσεων. Μια άλλη στρατηγική για την αντιμετώπιση λανθασμένων ετικετών περιλαμβάνει την ενημέρωση των παραμέτρων του δασκάλου για τη βελτίωση της ποιότητας των ψευδοετικετών. Στο [4], ο δάσκαλος ενημερώνεται κάθε Δ βημάτα ως ο Εκθετικός Κινητός Μέσος Όρος (Exponential Moving Average) των παραμέτρων του μαθητή. Στο [5], ο δάσκαλος αξιοποιεί την απόδοση του μαθητή στο πεδίο-πηγή ως ανάδραση, βελτιστοποιώντας τις παραμέτρους του με βάση αυτή την ανάδραση για να παράγει καλύτερες ψευδοετικέτες, γεγονός που με τη σειρά του βελτιώνει την απόδοση του μαθητή και στα δύο πεδία.

0.3.3 Τεχνικές αυτοεπίβλεψης

Μια άλλη κυρίαρχη μέθοδος είναι η αυτοεπιβλεπόμενη μάθηση (self-supervision). Αυτή η προσέγγιση εφαρμόστηκε αρχικά σε εργασίες Επεξεργασίας Φυσικής Γλώσσας και έχει αποδειχθεί αποτελεσματική και απλή τεχνική προεκπαίδευσης που διευκολύνει την προσαρμογή σε νέους τομείς. Η βασική ιδέα είναι η αξιοποίηση της αυτοεπιβλεπόμενης συνάρτησης σφάλματος που χρησιμοποιείται κατά την αρχική προεκπαίδευση του μοντέλου. Αυτό μπορεί να επιτευχθεί είτε μέσω της Συνεχούς Προεκπαίδευσης (Continual Pre-Training) είτε μέσω δημιουργίας μια συνάρτησης σφάλματος πολλαπλών εργασιών (multi-task loss) για την εκπαίδευση του μοντέλου. Η Συνεχής Προεκπαίδευση έχει μελετηθεί ιδιαίτερα στο πλαίσιο της προσαρμογής πεδίου για την Αυτόματη Αναγνώριση Ομιλίας Για παράδειγμα, στο [20] τονίζεται η σημασία της αξιοποίησης μη επισημασμένων δεδομένων εντός του πεδίου-πηγή, ενώ στο [21] δείχνεται ότι ο συνδυασμός Συνεχής Προεκπαίδευσης με στρατηγικές ψευδοετικετών μπορεί να μειώσει σημαντικά το ποσοστό σφάλματος στο πεδίο στόχο. Στα [6] και [8], η αυτοεπιβλεπόμενη μάθηση χρησιμοποιείται για τη δημιουργία μιας συνάρτησης σφάλματος πολλαπλών εργασιών (multi-task loss) για την εκπαίδευση του μοντέλου, με έμφαση στη διατήρηση της προσαρμογής κατά τη λεπτομερή προσαρμογή (fine-tuning) και την αποφυγή της τροπικής κατάρρευσης (mode collapse).

0.3.4 Τεχνικές εκπαίδευσης ανταγωνιστικών πεδίων

Ο κύριος στόχος της Εκπαίδευσης Ανταγωνιστικών Πεδίων (Domain Adversarial Training) είναι να εκπαιδεύσει ένα μοντέλο που μαθαίνει βαθιά χαρακτηριστικά ικανά να λύσουν το πρόβλημα στο πεδίο-πηγή (στην περίπτωση μας το πρόβλημα θα ήταν η αναγνώρισης φωνής), ενώ δεν επηρεάζεται από τις διαφορές μεταξύ των πεδίων. Στο [7] προτείνονται δύο μέθοδοι για την αντιμετώπιση της προσαρμογής από γλώσσες υψηλών πόρων σε γλώσσες χαμηλών πόρων, οι οποίες μοιράζονται έναν κοινό ακουστικό χώρο, τα οποία είναι τα Χίντι και τα Σανσκριτικά.

0.4 Προτεινόμενη Μεθοδολογία

Η προσέγγισή μας απεικονίζεται στο Σχήμα 1 και αποτελείται από δύο στάδια προσαρμογής: ένα στάδιο αυτοεπιβλεπόμενης μάθησης, ακολουθούμενο από ένα στάδιο ημι-επιβλεπόμενης μάθησης, το οποίο αναφέρουμε συνολικά ως Πολυ-Σταδιακή Προσαρμογή Πεδίου (Multi-Stage Domain Adaptation- MSDA).

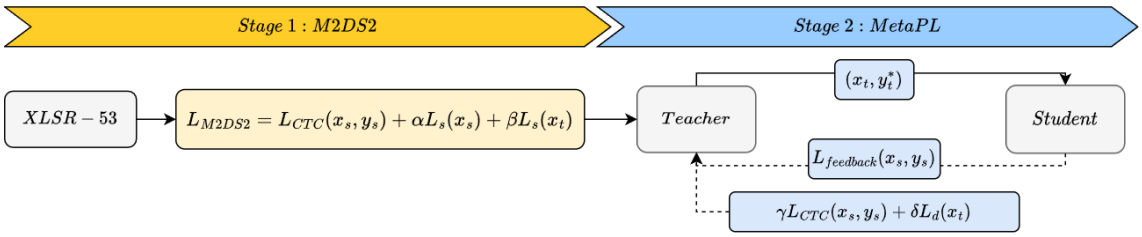


Figure 1. Προσαρμογή τομέα πολλαπλών σταδίων *(MSDA)*. Στο Στάδιο *1*, χρησιμοποιούμε την αυτο-επίβλεψη για να προσαρμόσουμε το προεκπαιδευμένο μοντέλο μας με πεδίο-πηγή (x_s, y_s) και πεδίο-στόχο (x_t). Στο Στάδιο 2, εφαρμόζουμε μια βελτιωμένη έκδοση της *Meta PL* για περαιτέρω βελτίωση της συνολικής προσαρμογής. Σε αυτό το στάδιο, ο δάσκαλος παρέχει ψευδο-ετικέτες y_t^* για την εκπαίδευση του μοντέλου-μαθητή.

Στάδιο **1:**
Σε αυτό το στάδιο, ακολουθούμε τη μέθοδο που προτάθηκε στο [8] για να αναπτύξουμε ένα μοντέλο δασκάλου που εκπαιδεύεται και στους δύο τομείς με αυτοεπιβλεπόμενο τρόπο, χρησιμοποιώντας την συνάρτηση κόστους:

$$L_{M2DS2} = L_{CTC}(x_s, y_s) + \alpha L_s(x_s) + \beta L_s(x_s) \tag{2}$$

Εδώ, το L_{CTC} είναι η συνάρτηση κόστους Συνδεσιονιστικής Χρονικής Ταξινόμησης, που εφαρμόζεται στο πεδίο-πηγή (x_s, y_s). Οι όροι $L_s(x_s)$ και $L_s(x_t)$ είναι οι απώλειες αυτοεπιβλεπόμενης μάθησης από το [8], που εφαρμόζονται στα δεδομένα ομιλίας της πηγής και του στόχου αντίστοιχα, όπως περιγράφεται στο [3]. Αυτές οι απώλειες ενθαρρύνουν το μοντέλο να μάθει ανθεκτικές, ανεξάρτητες από το πεδίο αναπαραστάσεις, επιτυγχάνοντας επιτυχώς ένα αρχικό επίπεδο προσαρμογής. Οι παράμετροι α και β είναι συντελεστές-βάρη.

Στάδιο **2:**
Σε αυτό το στάδιο, ακολουθούμε το πλαίσιο που προτάθηκε στο [5], χρησιμοποιώντας το δάσκαλο από το Στάδιο 1 για να δημιουργήσουμε ψευδο-ετικέτες για τον τομέα στόχο. Αυτές οι ψευδο-ετικέτες χρησιμοποιούνται στη συνέχεια για την εκπαίδευση του μοντέλου μαθητή με χρήση της συνάρτησης απώλειας L_{CTC}. Επιπλέον, ο δάσκαλος ενημερώνεται σύμφωνα με την ακόλουθη συνάρτηση σφάλματος:

$$L_T = L_{feedback}(x_s, y_s) + \gamma L_{CTC}(x_s, y_s) + \delta L_d(x_t) \tag{3}$$

Η $L_{feedback}$ αναπαριστά την απώλεια L_{CTC} του μαθητή στον τομέα πηγής, η οποία παρέχει μια ανάδραση στον δάσκαλο και τον καθοδηγεί να παράγει καλύτερες ψευδο-ετικέτες. Το L_{CTC} είναι η απώλεια Συνδεσιονιστικής Χρονικής Ταξινόμησης του δασκάλου στον τομέα πηγής, ενώ το L_d είναι η απώλεια διαφοροποίησης (diversity loss) που περιγράφεται στο [3]. Οι παράμετροι γ και δ είναι συντελεστές-βάρη.

Στο Στάδιο 1, εφαρμόζουμε το SpecAugment [19] στα ηχητικά δεδομένα και των δύο τομέων για να εκπαιδεύσουμε έναν πιο ανθεκτικό δάσκαλο. Ωστόσο, στο Στάδιο 2, το SpecAugment χρησιμοποιείται μόνο στις εισόδους του μαθητή, καθώς ο πρόσθετος θόρυβος στην είσοδο του δασκάλου οδηγεί στη δημιουργία ψευδο-ετικετών χαμηλότερης ποιότητας.

Χρησιμοποιούμε την εκπαίδευση M2DS2 στο πρώτο στάδιο επειδή, όπως περιγράφεται στο [8], η αυτοεπιβλεπόμενη εκπαίδευση πολλαπλών πεδίων διευκολύνει την προσαρμογή τομέα ενώ αποτρέπει την κατάρρευση λειτουργίας (mode collapse). Αυτή η προσέγγιση διευκολύνει τη δημιουργία ψευδο-ετικετών υψηλότερης ποιότητας στο δεύτερο στάδιο. Επιπλέον, η ενσωμάτωση του L_d στο πεδίο στόχο είναι ουσιώδης για την αποφυγή τροπικής κατάρρευσης κατά το 2ο στάδιο, ενώ το L_{CTC} διασφαλίζει ότι το μοντέλο του δασκάλου διατηρεί την προσαρμογή του στον τομέα πηγής παρά τις αλλαγές που προκύπτουν από το $L_{feedback}$.

0.5 Πειράματα

0.5.1 Προ-εκπαιδευμένο μοντέλο

Για το βασικό μας μοντέλο, χρησιμοποιήσαμε το XLSR-53 [22], ένα μοντέλο ομιλίας που έχει εκπαιδευτεί εκ των προτέρων με βάση την αρχιτεκτονική Wav2Vec 2.0 [3]. Το XLSR-53 ξεχωρίζει λόγω της εκτενούς εκπαίδευσής του σε ένα ποικιλόμορφο σύνολο δεδομένων, που περιλαμβάνει 56.000 ώρες ομιλίας σε 53 γλώσσες, κάτι που του επιτρέπει να έχει ισχυρη κατανόηση διαφόρων γλωσσικών και ακουστικών χαρακτηριστικών.

0.5.2 Δεδομένα

Το σύνολο δεδομένων που επιλέξαμε είναι το Greek Podcast Corpus (GPC) [23], το οποίο περιλαμβάνει 3124 ώρες ήχου από ελληνικά podcasts. Το σύνολο δεδομένων είναι οργανωμένο σε 16 διαφορετικές κατηγορίες, συμπεριλαμβανομένων των TrueCrime, News, Art και άλλων. Για να διευκολυνθούν διαφορετικές κλίμακες πειραμάτων, το σύνολο δεδομένων έχει χωριστεί σε μικρότερα υποσύνολα: GPC-50, GPC-20 και GPC-10, που περιέχουν 50, 20 και 10 ώρες ήχου ανά κατηγορία, αντίστοιχα. Οι ετικέτες για αυτά τα υποσύνολα δημιουργήθηκαν αυτόματα χρησιμοποιώντας την πλατφόρμα WhisperX [24], η οποία παρέχει ετικέτες χαμηλής ποιότητας (weakly-supervised labels). Στα πειράματά μας, επικεντρωθήκαμε στις παρακάτω κατηγορίες του GPC-20 ως διακριτές περιοχές:

1. **TrueCrime:** Podcasts που εξερευνούν αληθινές ιστορίες εγκλημάτων και αστυνομικές έρευνες.
2. **Education:** Podcasts που καλύπτουν θέματα όπως η ιστορία, η κοινωνιολογία και άλλες εκπαιδευτικές θεματολογίες.
3. **Business:** Podcasts επικεντρωμένα στη διοίκηση επιχειρήσεων, τις νεοφυείς εταιρείες και τις επενδύσεις.
4. **Comedy:** Podcasts αφιερωμένα στην stand-up κωμωδία και τις κωμικές παραστάσεις.

Χρησιμοποιώντας το whisper-large-v2 [23], οι συγγραφείς αποκτούν ποσοστά WER 8%, 5%, 15% και 28% για τους τομείς TrueCrime, Education, Business και Comedy αντίστοιχα, υποδεικνύοντας τις διαφορές μεταξύ των τομέων. Στον Πίνακα 1 εξετάζουμε την ομοιότητα μεταξύ αυτών των τομέων, χρησιμοποιώντας το μέτρο ATDS [17]. Αυτό το μέτρο, που αναπτύχθηκε αρχικά για συγκρίσεις μεταξύ γλωσσών, κλίνει λογικά προς το 1 στην περίπτωση μονογλωσσικών δεδομένων, όπως αναμενόταν.

Ομοιότητα	TrueCrime	Education	Business	Comedy
TrueCrime	-	0.93937	0.91852	0.89848
Education	0.93937	-	0.96935	0.94209
Business	0.91852	0.96935	-	0.96998
Comedy	0.89848	0.94209	0.96998	-

Table 1. Το μέτρο *ATDS* μεταξύ των πεδίων.

0.5.3 Συγκρινόμενες Μέθοδοι

Αξιολογούμε την αποτελεσματικότητα της Πολυ-Σταδιακής Προσαρμογής Πεδίου χρησιμοποιώντας τη μετρική του Ποσοστού Λάθους Λέξεων (Word Error Ratio - WER). Η προσέγγισή μας συγκρίνεται με τις εξής τέσσερις μεθόδους:

1. **Finetuned:** Επιτηρούμενη βελτιστοποίηση του XLSR-53 στον τομέα πηγής, σύμφωνα με την 2 με $\alpha = \beta = 0$.

2. **M2DS2:** Εκπαίδευση με M2DS2 χρησιμοποιώντας και τους δύο τομείς (πηγής και στόχου), σύμφωνα με την 2 με $\alpha = \beta = 0.01$.

3. **Finetuned - Meta PL:** Εφαρμογή του πλαισίου Meta PL στο βελτιστοποιημένο μοντέλο στον τομέα στόχο, σύμφωνα με την 3 με $\gamma = \delta = 0$, δηλαδή, χρησιμοποιώντας μόνο την ανατροφοδότηση του μαθητή.

4. **M2DS2 - Meta PL:** Εφαρμογή του πλαισίου Meta PL στα εκπαιδευμένα μοντέλα M2DS2, σύμφωνα με την 3 με $\gamma = \delta = 0$.

Για την εκπαίδευση MSDA, ορίζουμε $\alpha = \beta = 0.01$ στην 2 και$\gamma = \delta = 0.001$στην 3. Επιπλέον, πειραματιζόμαστε με διάφορες τιμές των γ και δ, προκειμένου να εξετάσουμε τις επιδράσεις τους στην απόδοση του μοντέλου. Συγκεκριμένα, διεξάγουμε δύο πειράματα, κάθε ένα από τα οποία εστιάζει σε έναν από τους συντελεστές. Σε κάθε πείραμα, ένας συντελεστής είναι σταθερός στο0.001 ενώ ο άλλος μεταβάλλεται μεταξύ 1 και 0.00001.

0.5.4 Αποτελέσματα

Στον Πίνακα 2 συγκρίνουμε τη MSDA με τις προαναφερθείσες μεθόδους σε δώδεκα σενάρια προσαρμογής.

Αρχικά, εστιάζουμε στην απόδοση του μοντέλου Finetuned-Meta PL. Παρατηρούμε ότι η εφαρμογή του Meta PL μειώνει το WER του μοντέλου Finetuned κατά 2%-5% σε όλες τις δοκιμές και, επιπλέον, υπερβαίνει την μέθοδο M2DS2 σε κάθε σενάριο. Αυτό δείχνει ότι το πλαίσιο Meta PL μπορεί να εφαρμοστεί αποτελεσματικά στην αναγνώριση φωνής. Στη συνέχεια, αξιολογούμε την απόδοση του μοντέλου M2DS2-Meta PL. Ενώ το Meta PL έχει ως αποτέλεσμα τη μείωση του WER και την βελτίωση της απόδοσης σε σύγκριση με το αρχικό μοντέλο M2DS2, αποδίδει χειρότερα συγκριτικά με την βελτίωση που παρείχε το μοντέλο Finetuned-Meta PL. Αποδίδουμε την ελλειπή επίδοση στην πιθανή κατάρρευση λειτουργίας του δασκάλου κατά την εκπαίδευση Meta PL.

Πεδία		Finetuned	Finetuned Meta PL	M2DS2	M2DS2 Meta PL	MSDA
πηγή	στόχος					
TrueCrime	Education	48.8	43.47	45.32	42.35	**37.98**
	Business	62.7	57.16	58.26	56.27	**50.3**
	Comedy	61.9	57.56	68.77	57.1	**49.21**
Education	TrueCrime	37.28	**33.3**	43.23	42.07	39.89
	Business	50.93	48.25	56.01	55.17	**47.03**
	Comedy	52.54	**50.7**	71.14	58.06	56.16
Business	TrueCrime	41.07	**38.5**	52.2	43.02	41.01
	Education	42.38	**38.9**	55.32	48.27	42.4
	Comedy	51.93	53.5	66.98	64.3	**50.02**
Comedy	TrueCrime	43.32	40.01	50.27	42.67	**39.08**
	Education	42.78	40.8	46.5	39.71	**38.77**
	Business	51.41	49.17	59.52	50.12	**47.3**
Μέσο WER		48.92	45.94	56.13	49.93	**44.93**

Table 2. Αποτελέσματα απόδοσης στα ζεύγη πεδίων *GPC-20*, χρησιμοποιώντας τη μέτρηση *WER*. Παρατηρούμε ότι ακόμη και όταν το μοντέλο δάσκαλος *(M2DS2)* αποδίδει κακά στον τομέα στόχο, το μοντέλο που έχει εκπαιδευτεί με *MSDA* μειώνει αποτελεσματικά το *WER* κατά *4%* έως *14%*

Τέλος, εξετάζουμε το μοντέλο MSDA. Παρατηρούμε ότι το μοντέλο MSDA υπερβαίνει τα προηγούμενα μοντέλα σε πολλές περιπτώσεις, επιτυγχάνοντας μείωση WER έως 10%-12% σε ορισμένες από αυτές. Το μοντέλο Finetuned-Meta PL ξεπερνά μόνο τη MSDA στις περιπτώσεις όπου η εκπαίδευση M2DS2 αποτυγχάνει να επιτύχει οποιαδήποτε προσαρμογή και οδηγεί σε χειρότερο WER από το μοντέλο Finetuned. Ακόμα και σε αυτές τις περιπτώσεις, το μοντέλο MSDA καταφέρνει να μειώσει το WER του μοντέλου M2DS2 κατά 5%-14%, υποδεικνύοντας την ικανότητά του να παράγει σημαντικά πιο ισχυρούς μαθητές ακόμα και σε περιπτώσεις όπου το μοντέλο-δάσκαλος αποτυγχάνει. Αποδίδουμε την αδυναμία του M2DS2 να επιτύχει προσαρμογή στην κακή ποιότητα των δεδομένων μας, καθώς και στην περιορισμένη ρύθμιση των παραμέτρων α και β.

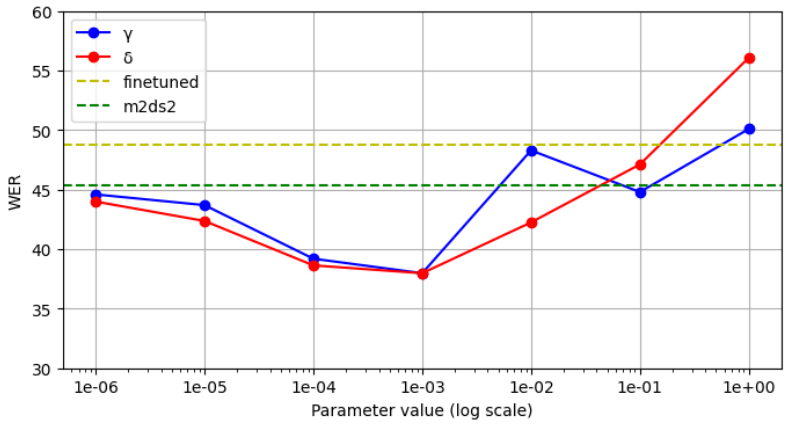


Figure 2. Σύγκριση *WER* μεταξύ διαφορετικών ρυθμίσεων των γ *(μπλε)* και δ *(κόκκινο)*, των βασικών μοντέλων *finetuned (κίτρινο)* και *M2DS2 (πράσινο)*, που αξιολογούνται στην προσαρμογή τομέα πηγής-στόχου *TrueCrime - Education*

Στο Σχήμα 2 εξετάζουμε τις επιδράσεις των μεταβλητών τιμών γ και δ στην απόδοση του μοντέλου στο σενάριο προσαρμογής τομέα από TrueCrime σε Education. Το σχήμα απεικονίζει πώς οι διαφορετικοί συνδυασμοί αυτών των υπερπαραμέτρων επηρεάζουν την ικανότητα του μοντέλου να γενικεύει από τον τομέα TrueCrime (πηγή) στον τομέα Education (στόχος). Παρατηρούμε ότι για $\gamma, \delta > 10^{-3}$, η MSDA αποτυγχάνει να ξεπεράσει την απόδοση των μεθόδων Finetuned και M2DS2. Υποθέτουμε ότι για $\gamma > 10^{-3}$, το μοντέλο δάσκαλος αρχίζει να υπερπροσαρμόζεται στον τομέα πηγής, οδηγώντας σε κακή παραγωγή ψευδοετικετών. Από την άλλη πλευρά, όταν $\delta > 10^{-3}$, η επιρροή του L_d γίνεται αρκετά σημαντική ώστε να υπονομεύσει οποιαδήποτε προσαρμογή επιτεύχθηκε κατά τη διάρκεια του Σταδίου 1. Επιπλέον, όταν $\gamma, \delta < 10^{-4}$, η MSDA οδηγεί σε κακή προσαρμογή σε σύγκριση με την περίπτωση όπου $\gamma, \delta \in [10^{-4}, 10^{-3}]$. Πιστεύουμε ότι αυτό οφείλεται στην πιθανή κατάρρευση λειτουργίας, που οφείλεται στην ανατροφοδότηση του μαθητή. Σε αυτές τις περιπτώσεις, η επίδραση του L_{CTC} και του L_d κατά την εκπαίδευση μειώνεται, αποτυγχάνοντας να διατηρήσει την προσαρμογή που επιτεύχθηκε στη Στάδιο 1, οδηγώντας το μοντέλο-δάσκαλος σε κατάρευση λειτουργίας.

0.6 Προηγούμενες προσεγγίσεις

Κατά την διάρκεια της ανάπτυξης της μεθόδους μας πειραματιστκαμε και με άλλες προσεγγίσεις και συνδυασμούς της μεθόδου Meta PL και τεχνικών αυτοεπίβλεψης. Σε αυτές τις προσεγγίσεις χρησιμοποιήσαμε δασκάλους που έχουν εκπαιδευτεί με Finetuned και M2DS2,ενώ η αξιολόγηση έγινε στην προσαρμογή πεδίου πηγής-στόχου TrueCrime - Education. Η μέθοδος SpecAugment εφαρμόστηκε μόνο στα δεδομένα εισόδου του μαθητή για να αποτραπεί η δημιουργία ετικετών χαμηλής ποιότητας από το μοντέλο του δασκάλου.

Αυτοεποπτεία και στα δύο πεδία

Η προσέγγιση αυτή περιλαμβάνει την εκπαίδευση του μαθητή με τη χρήση της συνάρτησης Συνδεσιονιστικής Χρονικής Ταξινόμησης ενώ το μοντέλο του δασκάλου εκπαιδεύεται με το σφάλμα-ανάδραση του μαθητή, καθώς και με το σφάλμα αυτοεπίβλεψης από το [3] και στα δύο πεδία.

Η ιδέα πίσω από αυτή την προσέγγιση ήταν ότι η συνεχής προσαρμογή του δασκάλου μέσω της αυτοεπίβλεψης, σε συνδυασμό με την ανάδραση του μαθητή, θα οδηγήσει στην παραγωγή καλύτερων ψευδο-ετικέτων, και άρα στην περαιτέρω προσαρμογή του μαθητή.

Είδος δασκάλου	Πεδία		πριν **Meta PL**		μετά **Meta PL**	
	πηγή	στόχος	πηγή	στόχος	πηγή	στόχος
Finetuned	TrueCrime	Education	37.83	48.8	43.01	97
M2DS2	TrueCrime	Education	36.9	45.32	41.6	95.83

Table 3. Αποτελέσματα προσαρμογής χρησιμοποιώντας αυτοεποπτεία και στα δύο πεδία κατά τη διάρκεια της εκπαίδευσης του δασκάλου.

Τα αποτελέσματα προσαρμογής παρουσιάζονται στον Πίνακα 3. Παρατηρούμε ότι το μοντέλο του μαθητή αποτυγχάνει να προσαρμοστεί εντελώς, καθώς το WER στους τομείς στόχου είναι κοντά στο 100. Πιστεύουμε ότι αυτό το αποτέλεσμα οφείλεται στην αυτοεποπτευόμενη

απώλεια, η οποία πιθανώς προκάλεσε στον δάσκαλο να δημιουργήσει ψευδο-ετικέτες κακής ποιότητας. Αυτές οι ψευδο-ετικέτες, σε συνδυασμό με την αντίστοιχη ανατροφοδότηση από τον μαθητή, φαίνεται να δημιουργούν έναν φαύλο κύκλο εκπαίδευσης, εμποδίζοντας το μοντέλο του μαθητή να συγκλίνει και να προσαρμοστεί.

Αυτοεποπτεία του μαθητή
Σε αυτή την προσέγγιση, το μοντέλο του μαθητή εκπαιδεύεται χρησιμοποιώντας ταυτόχρονα την συνάρτηση Συνδεσιονιστικής Χρονικής Ταξινόμησης καθώς και το σφάλμα αυτοεπίβλεψης . Το μοντέλο του δασκάλου εκπαιδεύτηκε χρησιμοποιώντας μόνο την απώλεια ανατροφοδότησης του μαθητή.

Η βασική ιδέα πίσω από αυτή την προσέγγιση ήταν ότι η ταυτόχρονη προσαρμογή του μαθητή στο πεδίο-στόχος, που καθοδηγείται από ημι-εποπτευόμενη και αυτοεποπτευόμενη εκπαίδευση, θα οδηγούσε σε συνολική επιτυχημένη προσαρμογή. Τα αποτελέσματα παρουσιάζονται στον Πίνακα 4.

Τύπος δασκάλου	Πεδία		πριν **Meta PL**		μετά **Meta PL**	
	πηγή	στόχος	πηγή	στόχος	πηγή	**target**
Finetuned	TrueCrime	Education	37.83	48.8	35.2	58
M2DS2	TrueCrime	Education	36.9	45.32	33.4	55.1

Table 4. Αποτελέσματα προσαρμογής χρησιμοποιώντας αυτοεποπτεία στη στοχοθετημένη περιοχή κατά την εκπαίδευση του μαθητή

Παρατηρούμε ότι το μοντέλο του μαθητή δυσκολεύεται να προσαρμοστεί αποτελεσματικά, αποδίδοντας χειρότερα στο πεδίο-στόχος από ότι το μοντέλο του δασκάλου. Πιστεύουμε ότι αυτό το αποτέλεσμα οφείλεται στην αυτοεποπτευόμενη εκπαίδεσυη, η οποία πιθανώς προκάλεσε το μοντέλο του μαθητή να παρέχει «παραπλανητική» ανατροφοδότηση στο μοντέλο του δασκάλου. Αυτό, με τη σειρά του, μπορεί να έχει οδηγήσει στην παραγωγή ψευδο-ετικετών χαμηλής ποιότητας, εμποδίζοντας την ημι-εποπτευόμενη εκπαίδευση να επιτύχει περαιτέρω προσαρμογή.

0.7 Συμπεράσματα και Μελλοντικές Προεκτάσεις

Αυτή η εργασία επικεντρώνεται στην εφαρμογή μεθόδων εκπαίδευσης δασκάλου-μαθητή που έχουν χρησιμοποιηθεί προηγουμένως στην αναγνώριση εικόνας και εξετάζει πώς αυτές οι μέθοδοι μπορούν να ενσωματωθούν επιτυχώς με στρατηγικές αυτοεποπτείας. Εξετάζουμε την εφαρμογή του πλαισίου Meta PL στις εργασίες ASR και διαπιστώνουμε ότι το Meta PL, ως μέθοδος, είναι απλή και εύκολα υλοποιήσιμη για την UDA, επιτυγχάνοντας σημαντικά αποτελέσματα προσαρμογής. Επίσης, ερευνάμε την ενσωμάτωσή των αυτοεποπτευόμενων στόχων στο πλαίσιο του Meta PL, προκειμένου να αναπτύξουμε μια στρατηγική που θα προσφέρει ακόμα καλύτερα αποτελέσματα προσαρμογής. Με βάση τα πειράματα, καταλήγουμε ότι για να ενσωματωθεί με επιτυχία το πλαίσιο Meta PL με αυτοεποπτευόμενους στόχους, είναι απαραίτητο η ανατροφοδότηση του μαθητή να μην επηρεάζεται από άλλες απώλειες. Προτείνουμε

μια νέα μέθοδο προσαρμογής, την Πολυ-Σταδιακή Προσαρμογή Πεδίου (MSDA), η οποία συνδυάζει επιτυχώς τεχνικές αυτοεποπτείας και ημιεποπτείας. Η μέθοδός μας επιτυγχάνει σημαντικά καλύτερα αποτελέσματα προσαρμογής σε σχέση με τις μεθόδους Meta PL και M2DS2. Στις περιπτώσεις όπου η MSDA δεν επιτυγχάνει προσαρμογή σε σύγκριση με το επιλεγμένο μοντέλο, αναγνωρίζουμε την κακή αρχική προσαρμογή του δασκάλου κατά τη διάρκεια της M2DS2 ως παράγοντα συμβολής. Ωστόσο, είμαστε αισιόδοξοι ότι με προσεκτική ρύθμιση υπερπαραμέτρων και βελτίωση της ποιότητας των δεδομένων, η MSDA έχει τη δυνατότητα να ξεπεράσει τους περιορισμούς που επιβάλλει η αποτυχία προσαρμογής του M2DS2.

Μελλοντικά, θα μπορούσαμε να εξετάσουμε την εφαρμογή της μεθόδου μας σε άλλες σχετικές με την ομιλία εργασίες, όπως η αναγνώριση ομιλητών και συναισθημάτων, οι οποίες διαφέρουν από την Αυτόματη Αναγνώριση Ομιλίας καθώς είναι προβλήματα κατηγοριοποίησης. Παράλληλα, θα μπορούσαμε να εξετάσουμε την εφαρμοσημότητα της μέθοδού μας σε διάφορους τομείς της μηχανικής μάθησης, όπως την Επεξεργασίας Φυσικής Γλώσσας και την Επεξεργασίας Εικόνας.

Chapter 1

Introduction

1.1 Motivation

Automatic Speech Recognition (ASR) has been one of the key areas where machine learning and deep learning models have been successfully applied, enabling a range of everyday commercial applications such as voice assistants and dictation systems. However, the performance of these systems can rapidly deteriorate when the data in the deployment domain differs significantly from the training data. These domain differences can arise due to various factors, including environmental noise, recording conditions, or shifts in the speaker's vocabulary and accent. As a result, domain adaptation techniques are crucial for maintaining system performance when operating on out-of-domain data.

Unsupervised Domain Adaptation (UDA) methods are particularly interesting because they do not depend on the expensive and time-consuming annotation of domain-specific data; instead, they focus on using the large amounts of unlabeled data to help models adapt to new, unseen domains. In the context of ASR, UDA techniques have been employed to improve robustness across various recording conditions, such as environmental noise and reverberation. Additionally, UDA has been applied to cross-lingual and multilingual adaptation tasks, boosting performance in low-resource languages [3] and different dialects [4].

The most common approaches for UDA are self-supervision and semi-supervised leaning. Recent research has demonstrated that these techniques lead to substantial performance gains across a range of tasks, including speech and image recognition. A key question that arises is whether these methods can be effectively combined to further enhance adaptation and achieve even better results.

1.2 Research Objective & Contribution

The primary research objective of this work is to develop an unsupervised domain adaptation (UDA) framework for speech recognition tasks in the context of low-resource language and weakly supervised data. This framework combines self-supervised learning techniques that have already been applied successfully in ASR tasks, with semi-supervised methods that have primarily been used in image recognition, adapting them for speech-based applications.

Our main contributions are:

1. We show that a semi-supervised framework, previously used in image recognition tasks, is an effective adaptation method for Automatic Speech Recognition.

2. We introduce Multi-Stage Domain Adaptation (MSDA), a novel two-step domain adaptation method that integrates self-supervised learning with semi-supervised strategies.

3. We explore the limitations regarding the integration of self-supervised tasks and semi-supervised training in the context of said semi-supervised framework.

We explore the potential of the Meta PL framework for ASR adaptation tasks. Meta PL is a semi-supervised approach that consists of a teacher-student model structure, where the teacher provides pseudo-labels for the student to train on using unlabeled data. What makes this approach particularly interesting is that the student model generates a feedback loss for the teacher, guiding it to improve the quality of its pseudo-labels. Since Meta PL has primarily been applied to classification tasks like image recognition, it is especially intriguing to investigate whether it can be effectively adapted for sequence-to-sequence tasks, such as ASR.

Additionally, we propose a novel two-stage domain adaptation approach, called Multi-Stage Domain Adaptation (MSDA). Our approach combines the semi-supervised approach introduced in Meta PL, with self-supervised strategies used in recent studies. Our results show that MSDA significantly enhances model performance in the target domain, outperforming baseline models, particularly in scenarios involving low-resource languages and weakly supervised data.

Finally, we examine the limitations associated with integrating self-supervised tasks into the Meta PL framework. Our findings indicate that the feedback-based training process of Meta PL is highly sensitive to the introduction of additional tasks. Therefore, any attempts at integration must carefully consider these sensitivities to ensure effective training and optimal performance.

This work is particularly focused on low-resource languages, such as Greek, where the availability of labeled data is limited. By leveraging weakly labeled datasets, our framework aims to improve ASR performance in these languages, highlighting the potential for effective adaptation strategies in resource-constrained environments. This approach not only addresses the challenges associated with low-resource languages but also contributes to the broader goal of enhancing speech recognition systems across diverse linguistic contexts.

1.3 Outline

This thesis is structured as follows:

- In Chapter 2, we present the foundational concepts of machine learning, covering key principles and basic terminology. We then focus on deep learning architectures and models that are particularly relevant to this work.
- In Chapter 3, we explore the challenges associated with Automatic Speech Recognition and review the solutions and methodologies that have been proposed over the years.
- In Chapter 4 we provide an extensive literature overview of unsupervised domain adaptation techniques used in speech recognition. We explore the basic concepts of the most common approaches, including pseudo-labeling / semi-supervised learning, and domain adversarial training.
- In Chapter 5, we present our proposed framework, detailing the techniques employed and the methodology developed. This chapter includes an analysis of our approach, a description of the experimental setup and data utilized, an overview of our experimental results, and a discussion of the challenges encountered, particularly in combining self-supervised and semi-supervised learning methods.
- In Chapter 6 we provide our conclusions and an outlook into the future.

Chapter 2

Machine & Deep Learning

In this chapter, we'll explore the key concepts of machine learning, highlighting the fundamental principles that drive this field.

2.1 Definition

Machine learning (ML) [25, 26, 11]is a branch of artificial intelligence (AI) focused on developing and studying statistical algorithms that enable systems to learn from data, generalize to new, unseen data, and perform tasks without being explicitly programmed. These algorithms are particularly valuable in domains where manually designing algorithms is complex and challenging, such as in computer vision and speech recognition. By leveraging data-driven approaches, ML systems can automatically improve their performance and adapt to a wide range of applications.

The core objective of a machine learning model is to be able to generalize based on its experience, i.e, to be able to perform accurately on new, unseen examples and data after its training.

A common example in machine learning involves using pairs of data where x represents the input and $f(x)$ denotes the corresponding output produced by the function f . he objective is to identify a hypothesis function h that approximates f, as closely as possible, based on a finite set of these input-output pairs $(x, f(x))$. This task is challenging because the true form of the function $f$is unknown and must be inferred from the data. A good approach should be able to generalize successfully, that is it should be able to predict correctly on unseen examples and data.

2.2 Types of Learning

Machine learning approaches are divided into three main categories, depending on the type of used data and the feedback used during training: supervised learning, unsupervised learning and reinforcement learning.

Supervised Learning

In supervised learning [25, 11, 27, 26], the goal is to learn a function from pairs of input and output examples. The system is provided with labeled training data, which consists

of input-output pairs where the correct output for each input is known and provided by a teacher. The objective is to derive a general rule or function that maps inputs to their corresponding outputs. A supervised learning algorithm processes this labeled data to create an inferred function that can predict outputs for new, unseen inputs. Ideally, this learned function will accurately determine the outputs for these new instances, demonstrating the model's ability to generalize from the training data.

Unsupervised Learning
Unsupervised learning [28, 29, 11] focuses on discovering patterns or structures in data without the guidance of output labels. Also referred to as self-organization, this approach enables the modeling of probability distributions over the inputs. By analyzing the inherent structure of the data, unsupervised learning methods can uncover hidden relationships and groupings, providing insights that are not immediately apparent from labeled examples.

Reinforcement Learning
Reinforcement Learning (RL) [30, 31] is a branch of machine learning focused on how systems should make decisions to maximize cumulative rewards over time. Unlike supervised learning, RL does not require labeled input/output pairs or explicit corrections for sub-optimal actions. Instead, RL involves learning from interactions with an environment, where the system receives feedback in the form of rewards or penalties. The problem is often framed as a Markov Decision Process![32] (MDP), and many RL algorithms use dynamic programming techniques. However, RL algorithms differ from classical dynamic programming methods in that they do not rely on an exact mathematical model of the MDP. Instead, they are designed to handle situations where precise models are impractical, allowing them to learn effective strategies through trial and error.

Finally, there is a fourth approach in ML, semi-supervised learning, that lies between supervised and unsupervised learning.

Semi-supervised Learning
Semi-supervised learning [33, 34] is a machine learning approach that integrates a small amount of labeled data with a larger set of unlabeled data during training. This method merges elements of both supervised learning, which relies solely on labeled data, and unsupervised learning, which uses only unlabeled data. By combining these two types of data, semi-supervised learning can enhance model performance, leveraging the abundance of unlabeled data to complement the limited labeled data. The process of labeling data can be expensive and time-consuming, often requiring specialized knowledge or physical experiments, making it impractical to obtain large, fully labeled datasets. In contrast, unlabeled data is typically easier and cheaper to collect, making semi-supervised learning a highly valuable approach in practice.

2.3 Fundamentals

Here, we will cover fundamental concepts, ideas, and algorithms that are crucial in machine learning, including the perceptron [35] algorithm, activation functions, and feed-forward networks [11].

2.3.1 Perceptron Algorithm

The perceptron [35] is an algorithm used for supervised learning to create binary classifiers. It is a type of linear classifier that makes predictions based on a linear combination of input features and associated weights. Essentially, the perceptron is a function that maps an input $\mathbf{x}$ to an output value $f(\mathbf{x})$, which is a single binary result:

$$f(x) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Where $\mathbf{w}$ represents a vector of weights, $\mathbf{w} \cdot \mathbf{x}$ is the dot product and b is the bias. Since $f(\mathbf{x})$ returns two values, it is used in binary classification problems. The main limitation of the perceptron algorithm is that it can only generalize successfully to linearly separable problems. This restriction limits its applicability, but it serves as a foundational basis for developing more complex and advanced algorithms and models.

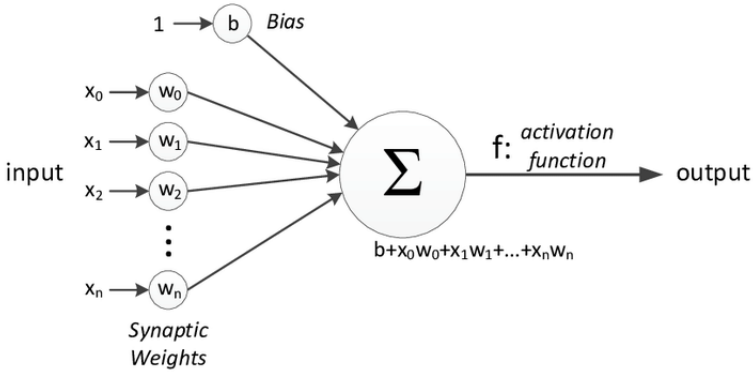


Figure 2.1. *A perceptron unit with an activation function*

2.3.2 Activation Function

To overcome the limitations of linear separability in models, it is essential to introduce non-linearities, which allow for the approximation of arbitrarily complex functions. This can be achieved through the use of activation functions [27, 11]. These functions are applied to the output of a linear unit or node and perform a non-linear transformation, enabling more sophisticated decision-making.

Given the perceptron model in equation 2.1, and an activation function g, we can define an non-linear clasiffier as:

$$f' = g(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.2}$$

Some commonly used activation functions are:

Sigmoid:
A sigmoid function [36], named for its characteristic S-shaped curve, is defined by the following formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{2.3}$$

A sigmoid function is a bounded, differentiable real function defined for all real input values, with a non-negative derivative at each point. It is generally monotonic, with a bell-shaped first derivative. The sigmoid function is constrained by horizontal asymptotes as $x \to \pm\infty$. It is convex for values less than 0 and concave for values greater than 0, which can lead to multiple optima in the sigmoid function and its affine compositions.

However, the sigmoid function has two major disadvantages. Firstly, when $\sigma(x)$ is close to 1 or 0, its gradient approaches 0, which can lead to very small weight updates and slow learning. Secondly, the output of the sigmoid function is not centered at 0. This means that positive inputs will always produce positive outputs, which can result in consistently positive or negative gradients, leading to undesirable oscillations and potentially hindered learning in the model.

Hyperbolic tangent:
Hyperbolic tangent is defined by the formula:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

From equations 2.3 and 2.4 we get that $tanh(x) = 2\sigma(2x) - 1$, thus it is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point. It's only advantage over sigmoid is that *tanh* is not zero-centered.

Rectified Linear Unit (ReLU):
ReLU [37], also know as *ramp function*, is defined as the positive part of its argument:

$$relu(x) = x^+ = max(0, x) \tag{2.5}$$

A key advantage of ReLU is its sparse activation. In a randomly initialized network, approximately 50% of the hidden units are activated, meaning they have a non-zero output. Other significant benefits of ReLU include improved gradient propagation, efficient computation, and scale invariance. However, ReLU also has potential drawbacks, such as a lack of differentiability at zero, non-zero-centered outputs, and being unbounded. A notable issue is the "dying ReLU" problem, where neurons can become inactive for almost

all inputs, resulting in no gradients flowing through them and leaving the neurons in a perpetually inactive state.

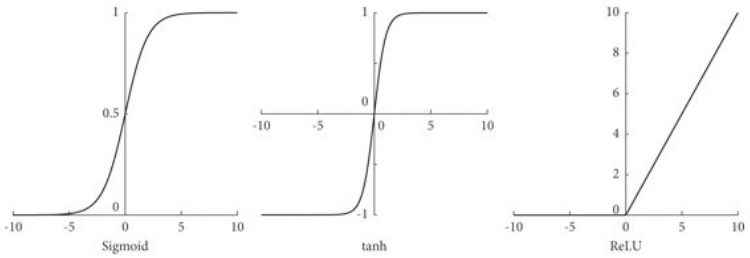


Figure 2.2. *The aforementioned activation functions*

2.3.3 Feed-Forward Networks

Multilayer perceptrons (MLPs) [38, 26], also known as feed-forward networks, are composed of several layers of perceptron units arranged in a stack. Each perceptron functions as a computational unit that processes the input from the preceding layer and applies a function to produce a scalar output. Within a single layer, units are typically not interconnected.

A feed-forward network comprises at least three layers: an input layer, one or more hidden layers, and an output layer. The input layer receives the raw data, the hidden layers process this data through successive transformations, and the output layer delivers the final result. Information flows from the input to the output through a process known as forward propagation. The model's depth is determined by the number of hidden layers, while its width is defined by the number of units within each hidden layer.

Unlike a linear perceptron, an MLP is capable of handling non-linearly separable data due to its multiple layers and non-linear activation functions. The distinction between a simple neural network and a deep neural network is illustrated in Fiq. 2.3

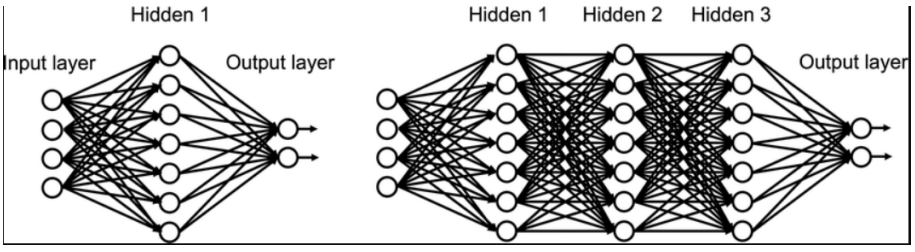


Figure 2.3. *One the left, a simple neural network. On the right a deep neural network*

2.4 Model Training

Training is a crucial process for developing a model that can perform effectively on its designated task. This process involves a loss function [25, 26], which measures how well the model is currently performing, and a training algorithm that leverages this performance metric to adjust the model's parameters for improvement.

2.4.1 Loss Function

Neural networks are trained using optimization methods that aim to find the best set of model parameters to minimize prediction error. For a model f with parameters w, the prediction error is estimated using a loss function $J(w)$. This loss function quantifies the "distance" between the model's predicted output and the desired target output.

$$L(f(x_i; w), y_i) \tag{2.6}$$

The total prediction loss over the dataset is calculated using the Eq. 2.6.

$$J(w) = \frac{1}{N} \sum_{i=0}^{N} L(f(x_i; w), y_i) \tag{2.7}$$

Eq. 2.7 is also known as objective loss, cost function of empirical risk.

There are various loss functions used in model training, each suited to different types of tasks the model is designed to solve. Below, we outline some of the most commonly used loss functions.

There are various loss functions used in model training, each suited to different types of tasks the model is designed to solve. Below, we outline some of the most commonly used loss functions.

Binary Cross Entropy Loss:
Used for binary classification tasks [39] where the model predicts a probability between 0 and 1. It measures the difference between the predicted probabilities and the actual binary labels, penalizing incorrect predictions more severely. This loss function is ideal for scenarios where the outputs represent probabilities of a binary outcome.

$$J(w) = \frac{1}{N} \sum_{i=0}^{N} y_i log(f(x_i; w)) + (1 - y_i) log(1 - f(x_i; w)) \tag{2.8}$$

Mean Squared Loss (MSE):
Commonly used for regression models that make predictions over continuous real numbers [40]. MSE measures the average of the squared differences between the predicted and actual values, penalizing larger errors more heavily.

$$J(w) = \frac{1}{N}\sum_{i=0}^{N}(y_i - f(x_i; w))^2 \tag{2.9}$$

The goal of training is to find the optimal values of w that minimize the chosen loss function.

$$w^* = \arg\min_{w} J(w) \tag{2.10}$$

This is achieved through the training algorithm, which we'll discuss below.

2.4.2 Gradient Descent

Gradient descent [41, 42, 26] is an optimization algorithm used to minimize the cost or loss function in machine learning and statistical models. It's an iterative process that helps find the optimal parameters (weights) of a model that minimize the error between predicted and actual values. The fundamental idea behind gradient descent is to adjust the parameters of the model incrementally in the direction that reduces the loss function and reaching a local minimum.

This is achieved by calculating the gradient of the loss function $\nabla J(\mathbf{w})$, which is the vector of partial derivatives of the cost function with respect to each parameter.

$$\nabla J(\mathbf{w}) = [\frac{\partial J(\mathbf{w})}{\partial w_1}, ..., \frac{\partial J(\mathbf{w})}{\partial w_n}] \tag{2.11}$$

The gradient indicates the direction and rate of the steepest increase in the cost function. By moving in the opposite direction of the gradient, we aim to find the minimum point of the cost function.

Gradient descent starts with an initial set of parameters w_0, often chosen randomly, and iteratively updates them to find the minimum. The parameters are updated using the formula:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha\nabla J(\mathbf{w}_n) \tag{2.12}$$

where α represents the learning rate.

The process described in Eq. 2.12 is repeated until convergence or for a defined number of steps Given some certain assumptions, such as J is convex and ∇J is Lipschitz, convergence to a local minimum is guaranteed. The Lipschitz continuity of the gradient ensures that the function has a controlled rate of change, which is essential for the stable convergence of gradient descent.

However, in the context of deep neural networks, the situation becomes more complex. Deep neural networks introduce significant non-linearity through their activation functions and their layered structure. This non-linearity causes the loss surface to become highly non-convex, with many local minima, saddle points, and flat regions. As a result, the smooth and predictable landscape assumed in convex optimization no longer applies.

Learning Rate: The gradient defines the direction of the weights update, while learning rate dictates the magnitude of each update step. Learning rate is a crucial hyperparameter in gradient-based optimization algorithms, since it determines the extent to which new information, modifies the existing model parameters. Essentially, it dictates the speed at which a model learns.

Choosing an appropriate learning rate is essential because it directly impacts the trade-off between convergence and the risk of overshooting the minimum. A small learning rate ensures that the model slowly progresses towards the minimum, but the convergence process may become excessively slow. On the other hand, a large learning rate can accelerate the convergence process. However it can cause the updates to overshoot the minimum.

In practice, it is common to start with a relatively large learning rate and gradually decrease it as training progresses. This approach allows the model to make quick initial progress and then fine-tune the weights as it gets closer to the minimum. Techniques such as learning rate schedules and adaptive learning rate algorithms (like AdaGrad, RMSprop, and Adam) automate this process by adjusting the learning rate dynamically based on the optimization history.

Backpropagation: Backpropagation is a fundamental algorithm used for training neural networks by minimizing the loss function. It involves a forward pass, where the input is propagated through the network to compute the output, and a backward pass, where the error is propagated back through the network to update the weights. The key idea is to compute the gradient of the loss function with respect to each weight using the chain rule of calculus.

2.5 Deep Architectures

Speech recognition is fundamentally a sequence-to-sequence problem because it involves mapping a variable-length input sequence (the spoken audio) to a variable-length output sequence (the transcribed text). The challenge lies in aligning the input speech sequence with the corresponding output text sequence, as the length and structure of the sequences can vary significantly. This alignment requires the model to understand temporal dependencies and handle varying input lengths effectively.

In this section, we'll discuss and analyze some deep neural architectures that can effectively address sequence-to-sequence problems like speech recognition, focusing on Recurrent Neural Networks (RNNs) and Transformers.

2.5.1 Recurrent Neural Networks (RNNs):

Recurrent Neural Networks (RNNs) [43] are a type of neural network designed to process sequences of data. Unlike feedforward neural networks, which assume that each input instance is independent, RNNs consider the temporal dependencies in the data by maintaining an internal state, or hidden layer h, that evolves over time. This internal state allows RNNs to incorporate information from previous time steps into their current computations.

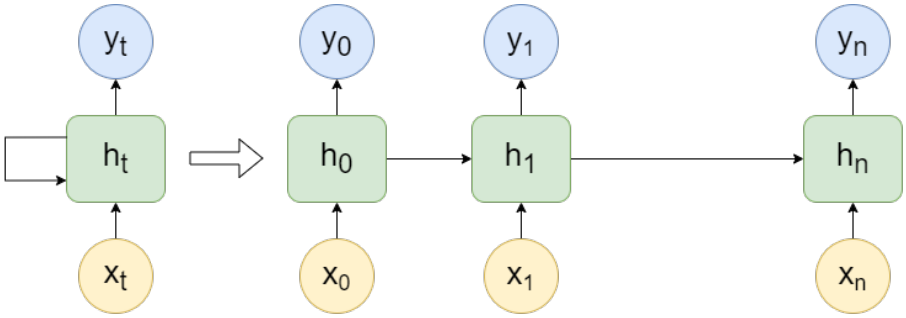


Figure 2.4. *A RNN unit, as it loops and unfolds over time.*

The core idea of an RNN is to capture the temporal dynamics of a data sequence by updating its hidden state at each time step as new data is processed. Specifically, at each time step t, an RNN takes the input x_t and combines it with the hidden state from the previous time step h_{t-1} to produce the new hidden state h_t. This process can be described mathematically as:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b) \tag{2.13}$$

where ϕ is a non-linear activation function, W_h and W_x are weight matrices, and b is a bias term. The output at each time step can be computed using:

$$y_t = \psi(W_y h_t + b_y) \tag{2.14}$$

where ψ is an non-linear activation function, W_y is a weight matrix and b_y is a bias term.

This architecture allows RNNs to learn from and make predictions based on the sequence of data, making them well-suited for tasks like speech recognition, language modeling, and time series prediction. However, RNNs can struggle with long-term dependencies due to issues such as vanishing or exploding gradients, which can make it difficult for the network to learn and remember information over extended sequences.

2.5.2 Vanishing and Exploding Gradient:

The vanishing gradient problem [44, 45] occurs when gradients, during backpropagation, become extremely small as they propagate through each time step, leading to minimal updates to the network's weights. This issue hinders the network's ability to learn long-term dependencies, as the influence of earlier time steps diminishes exponentially. Conversely, the exploding gradient problem arises when gradients become excessively large, causing the network's weights to update too aggressively. This can lead to unstable training, where the network's parameters oscillate wildly or diverge. Both problems stem from the nature of RNNs, where gradients are computed through a series of matrix multiplications, amplifying or diminishing their values. The vanishing gradient problem particularly affects RNNs with many layers or long sequences, making it difficult for them to capture long-term dependencies .

Variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address these challenges by introducing mechanisms to better capture long-range dependencies and stabilize training.

2.5.3 Long Short-term Memory (LSTM) networks:

Long Short-Term Memory (LSTM) [46] networks are a specialized type of RNN designed to address the vanishing gradient problem and enhance the learning of long-term dependencies in sequential data. Unlike traditional RNNs, LSTMs incorporate a more complex architecture that includes memory cells, input gates, output gates, and forget gates. These components work together to regulate the flow of information through the network, allowing it to maintain and access information over long sequences.

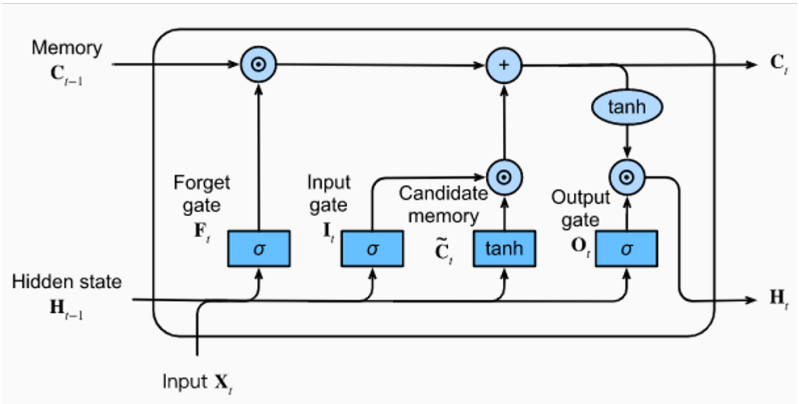


Figure 2.5. *LSTM's unit internal architecture*

The key innovation in LSTMs is the memory cell C_t, which stores information across many time steps and is updated by various gates. The forget gate F_t controls what information is discarded from the memory cell, the input gate I_t determines what new information is added, and the output gate O_t regulates what information is used to compute the output. This gating mechanism enables LSTMs to effectively manage long-term dependencies and avoid the issues associated with vanishing or exploding gradients.

The mathematical relations between these gates are described below:

$$F_t = \sigma(W_{fh}H_{t-1} + W_{fx}x_t) \tag{2.15}$$

$$I_t = \sigma(W_{ih}H_{t-1} + W_{ix}x_t) \tag{2.16}$$

$$O_t = \sigma(W_{oh}H_{t-1} + W_{ox}x_t) \tag{2.17}$$

$$\tilde{C}_t = tanh(W_{ch}H_{t-1} + W_{cx}x_t) \tag{2.18}$$

$$C_t = F_t \circ \tilde{C}_t + I_t \circ \tilde{C}_t \tag{2.19}$$

$$H_t = O_t \circ tanh(C_t) \tag{2.20}$$

2.5.4 Gated Recurrent Units (GRUs):

Gated Recurrent Units (GRUs) [47] are a simpler architecture that combines the memory cell and gating mechanisms into a single structure.

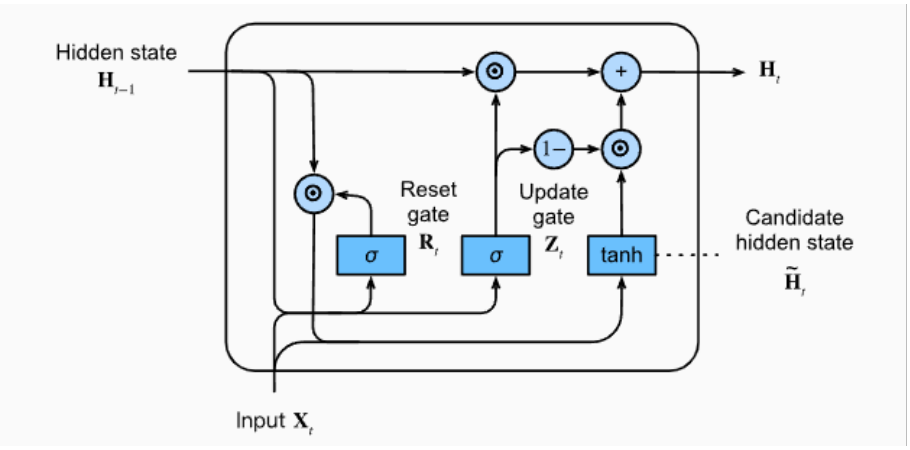


Figure 2.6. *GRU's internal architecture*

They utilize two primary gates: the update gate and the reset gate. The update gate controls how much of the past information is carried forward and how much of the new information is incorporated, while the reset gate determines how much of the past information is discarded. This design allows GRUs to maintain and adjust the flow of information more effectively across sequences, facilitating the learning of long-term dependencies without the complexity of LSTMs.

2.6 Attention and Transformers

The hidden layer of RNNs faces limitations primarily due to the challenge of maintaining long-term dependencies. RNNs try to compress the entire input sequence into a single fixed-size hidden state, limiting the model's ability to represent complex and lengthy sequences. These issue is solved using the Attention Mechanism [1].

2.6.1 Attention Mechanism

Attention mechanisms address this issue by allowing the model to directly access and focus on different parts of the input sequence. This dynamic focus enables the network to capture long-range dependencies more effectively, as it no longer relies on a single context vector to represent the entire sequence.

Each element of the input sequence has different importance, which help the attention module to focus on it. This importance is measured by their alignment with a query vector q. Given an input sequence $x_1, x_2, ..., x_N$ of length N, the attention module uses an alignment function to calculate the relevance of each x_i with the query q.

$$s_i = align(q, x_i) \tag{2.21}$$

These alignment scores s_i are then normalized using the softmax function to produce attention weights $\alpha_1, ...\alpha_N$, where $\sum_i^N \alpha_i = 1$.

$$\alpha_i = softmax(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{2.22}$$

The final representation $\hat{x}$ of the input sequence is the weighted average of each input element.

$$\hat{x} = \sum_i^N x_i \tag{2.23}$$

This final representation captures much more information about the input sequence than the hidden state h of RNNs. Below we present some popular alignment score functions.
General attention:

$$align(q, x_i) = s^T W_\alpha x_i \tag{2.24}$$

Dot-product attention:

$$align(q, x_i) = s^T x_i \tag{2.25}$$

Scaled dot-product attention:

$$align(q, x_i) = \frac{s^T x_i}{\sqrt{N}} \tag{2.26}$$

2.6.2 Transformers

Transformers [1] are a groundbreaking neural network architecture that leverages the attention mechanism to process sequences of data more efficiently than traditional RNNs. Unlike RNNs, which process inputs sequentially, Transformers allow for parallel processing by using self-attention to weigh the importance of different elements within a sequence. This self-attention mechanism enables the model to capture relationships between words regardless of their distance in the sequence, providing a more flexible way to understand context and dependencies.

Transformers are the model of choice for many machine learning tasks, from Natural Language Processing and Image segmentation to Speech Recognition. Their versatility and effectiveness have made them the foundation for the latest Large Language Models, such as OpenAI's ChatGPT and Meta's LLaMA.

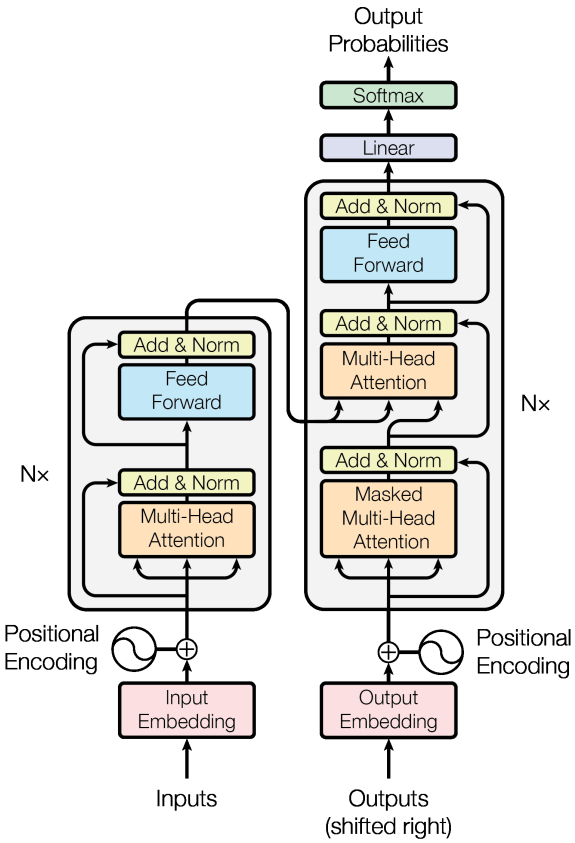


Figure 2.7. *The Transformer internal architecture. Source [1]*

Their architecture is based on the encoder-decoder model. The encoder consists of a set of encoding layers that processes the input iteratively from layer to layer, while the decoder consists of a set of decoding layers that applies the same processing to the output of the encoder. Encoder's and decoder's internal architecture is identical, but they use different parameter values since they are used for different purposes.

The encoder consists of a self-attention layer, which enables the generation of robust representations by capturing long-term dependencies in the input sequence. Following the self-attention layer, a feed-forward layer provides additional processing and transformation of these representations. The decoder mirrors this architecture but includes an additional encoder-decoder attention module. This intermediate module allows the decoder to identify and focus on specific elements of the encoded input, facilitating the generation of output sequences that are contextually relevant and accurate.

The self-attention module is essential for capturing contextual information from the input data. This is achieved via a two-step process. At the first step, the module creates three vectors based on the input, the Query (Q), Key (K) and Value (V) vectors. These vectors are created by applying specific matrices W_Q, W_K, W_V on the input sequence , whose desired values are acquired through training. At the second step, we calculate the self-attention using these three vectors, following the equation below:

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

where d_k is a scaling factor, based on the dimension of vector K.

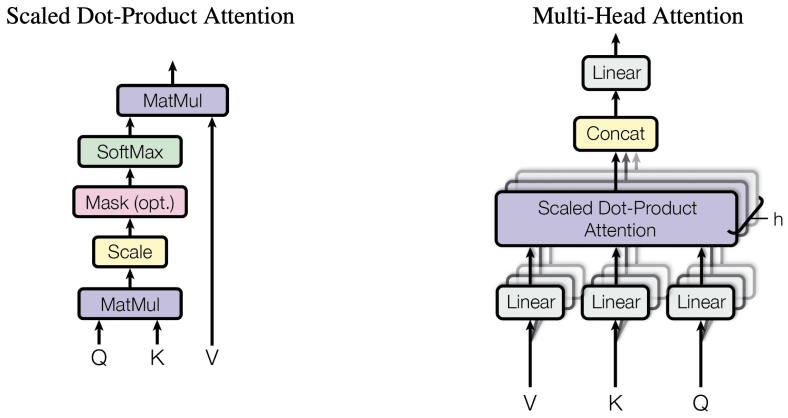


Figure 2.8. *Left: Attention Head, Right: Multi-head Attention. Source [1]*

Each set of W_Q, W_K, W_V is called an *attention-head*. To obtain more contextual information and capture different aspects of the input sequence, multiple attention heads are employed, a technique known as *multi-head attention*. This approach allows the model to focus on various parts of the input simultaneously, enhancing its ability to understand complex patterns and relationships within the data.

2.7 Generalization and overfitting

Training is considered successful when the produced model can make accurate predictions on previously unseen inputs, demonstrating its ability to generalize. Generalization is crucial for a model's effectiveness, as it indicates that the model has not only memorized the training data but has also learned underlying patterns that apply to new, unseen data. This capability is essential for the model to be useful in real-world applications where it encounters diverse and novel inputs beyond those it was explicitly trained on.

However, there are occasions when the model minimizes the loss function during training but fails to generalize to unseen data. This phenomenon is known as overfitting [48]. Overfitting occurs when the model becomes too closely aligned with the training data, effectively memorizing it rather than learning general patterns. As a result, the model performs well on the training set but poorly on new, unseen data.

Conversely, there is the issue of underfitting, where the model fails to learn adequately from the data. This typically happens when the model's capacity is too limited to capture the underlying patterns in the data. In such cases, a more complex model with additional parameters or more sophisticated architectures might be required to improve performance. Balancing between overfitting and underfitting is crucial for developing models that generalize well and perform effectively on unseen data.

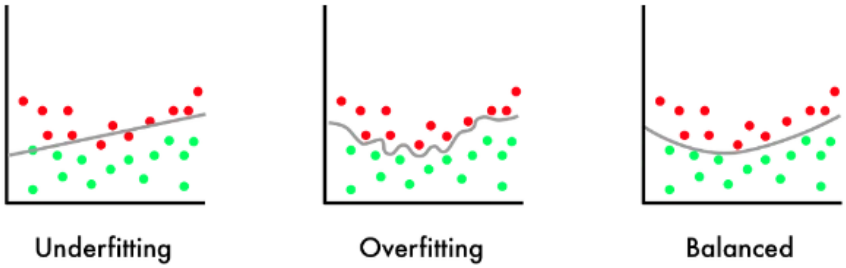


Figure 2.9. *From left to right: cases of underfitting, overfiting and successful generalization.*

2.7.1 Regularization

To combat overfitting, various regularization techniques [49] have been developed, including dropout, weight decay, and early stopping. These methods aim to balance the bias-variance trade-off by managing the model's complexity. The goal is to achieve a lower generalization error by preventing the model from fitting the training data too closely, which often results in a higher training error. One approach to this balance is by regulating the model's capacity; however, simply choosing a fixed number of parameters does not guarantee an optimal fit. Instead, considering complex function families and employing regularization techniques to control model complexity is typically more effective.

Regularization helps ensure that the model generalizes well to new data while avoiding the pitfalls of both overfitting and underfitting. Below we discuss some commonly used techniques.

Parameter norm penalties

Parameter norm penalties are used to limit the model's complexity by penalising the norm of its parameters. This is achieved by adding a regularization term $\Omega(\mathbf{w})$ to the loss function $J(\mathbf{w})$ scaled by a parameter $\lambda \in [0, 1]$. The overall loss is:

$$\hat{J}(\mathbf{w}) = \frac{1}{N}\sum_{i}^{N} L(f(x_i; \mathbf{w}), y_i) + \lambda\Omega(\mathbf{w}) \tag{2.28}$$

The two most commonly used penalties are L_1 [50] and L_2 [51] regularization.
L1 regularization: Here the $\Omega(\mathbf{w})$ is as follows

$$\Omega(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_{i}^{N} |w_i| \tag{2.29}$$

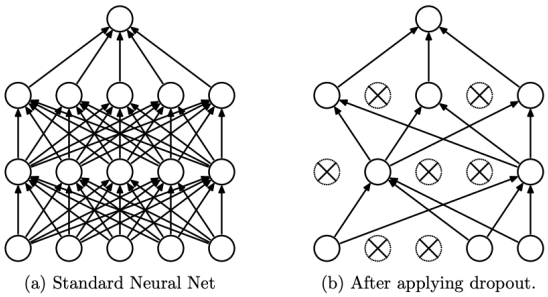
This adds the term $\lambda sign(w)$ to the update rule of eq. 2.12, which results to sparse model. This means that after training, due to the effects of this term, some weights may be close to zero, thus not contributing to the final result. Thus, L_1 acts a feature selection mechanism.
L2 regularization: Here the $\Omega(\mathbf{w})$ is as follows

$$\Omega(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2 \tag{2.30}$$

This adds the term λw to the update rule of eq. 2.12 This means that during training a term proportional to their magnitude is subtracted from all weights. This causes the weights to decay over time, so the technique is named weight decay.

Dropout:

Dropout [52] is a simple yet effective, frequently used regularization method for neural networks. During training each unit has a probability $p > 0$ to be dropped out, i.e to be ignored during forward and backward pass. This can be observed in the Fig. 2.10. At test time, all units are employed, but they are scaled by a factor $1/p$ to account for the missing activations during training. Dropout forces network to not rely on any node and prevents units from forming co-dependencies amongst each other. A usual value for p is 0.5.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.10. *The effect of dropout during training*

Dropout can be viewed as creating a set of different sub-networks during training. By randomly deactivating a subset of neurons in each training iteration, dropout effectively generates various configurations of the network. Given a model with N parameters, dropout can potentially create 2^N different sub-models, each sharing some parameters but with different neurons active or inactive. This process helps prevent overfitting by ensuring that the network does not rely too heavily on any single set of neurons, promoting a more robust and generalized learning.

Data augmentation

A simple way to reduce the risk of overfitting is to use more training data. However, when collecting additional data is not feasible, synthetic data can be generated from the existing dataset through a process known as data augmentation [53]. This technique involves creating new data points by applying various transformations or modifications to the original data. In the domain of images, common data augmentation methods include flipping, rotating, re-scaling, and cropping. For natural language data, augmentation is less common but can be achieved through techniques like synonym replacement. Additionally, adding small random noise to the data can help improve model robustness by encouraging the model to learn more general patterns rather than memorizing specific details. Data augmentation thus serves as a valuable strategy to enhance model performance and generalization without requiring extensive additional data collection.

Early stopping

Early stopping [54] is a crucial regularization technique used to prevent overfitting during the training of machine learning models. It involves monitoring the model's performance on a validation set while training. The key idea is to track a specific performance metric, such as validation loss or accuracy, and halt the training process when this metric starts to deteriorate, even if the predefined number of training epochs has not been reached.

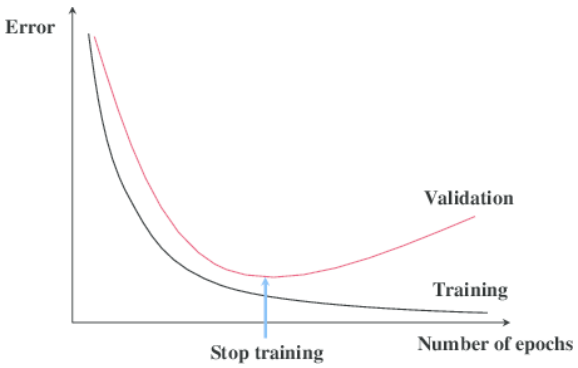


Figure 2.11. *Validation and training error during trainind. Early stopping is applied when the validation loss starts to increase*

Monitoring the model's performance is essential because, after a certain point, continued training can lead to overfitting, where the model becomes too specialized to the training data and loses its ability to generalize to unseen data. Early stopping helps mitigate this by ensuring that training is halted at an optimal point where the model performs well on both the training and validation sets, thereby maintaining its robustness and generalization capability. Additionally, early stopping helps save computational resources by avoiding unnecessary training epochs, making it a practical and efficient technique for improving model performance and efficiency.

2.8 Representation learning & Mode collapse

Representation learning [55] is a aspect of machine learning that focuses on learning useful and meaningful features or representations from raw data. The primary goal is to transform input data into a format that makes it easier for the model to perform tasks such as classification, regression, or generation. Effective representations capture the underlying structure and relationships within the data, enabling models to generalize better and improve performance on a variety of tasks.

Mode collapse is a significant challenge in representation learning when models fail to capture the full diversity of the data. It occurs when a model learns to produce or represent only a limited subset of the data's possible variations, resulting in a narrow representation of the underlying data distribution. This issue compromises the model's ability to generalize and accurately reflect the complexity of the data, as it focuses on only a few modes.

Contrastive losses [56] are a powerful tool in representation learning to mitigate mode collapse and enhance the quality of learned features. These losses encourage models to learn representations that differentiate between similar and dissimilar data points by focusing on maintaining proximity for similar pairs and distance for dissimilar pairs. By using contrastive losses during training, models are guided to develop more informative and varied representations of the input data. This approach helps prevent mode collapse by promoting a richer understanding of the data.

Chapter 3

Automatic Speech Recognition

In this chapter, we'll delve into the problem of Automatic Speech Recognition (ASR), exploring how it is formulated and the key challenges it faces.

3.1 Speech Recognition Fundamentals

The problem of Automatic Speech Recognition (ASR) is to develop a system is to convert a speech signal to text accurately and efficiently [10, 9], regardless the recording conditions, the accent and vocabulary of the speaker.

The foundation of modern speech recognition systems is illustrated in Fig. 3.1. This system operates in two distinct steps. First, it receives a speech signal $s[n]$ from the speaker and processes it using an audio processor to extract a sequence of feature vectors $X = [X_1, X_2, ..., X_T]$. In the second step, a language decoder interprets these feature vectors and decodes them into a sequence of characters or words, producing the resulting text. The feature vectors X_i are usually the Mel-Frequency Cepstral Coefficients (MFCCs) [10, 9]

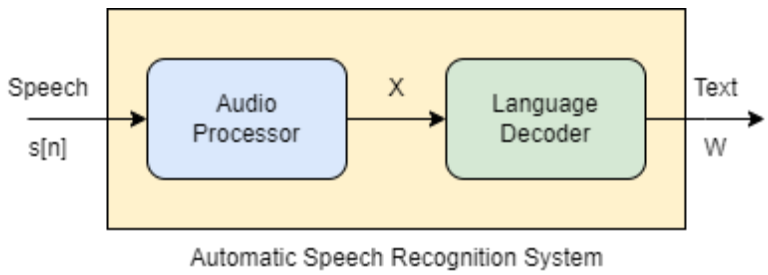


Figure 3.1. *A typical ASR system*

of the speech signal $s[n]$. MFCCS are a widely used feature representation in speech recognition that capture the power spectrum of a speech signal in a way that mimics the human ear's perception of sound.

They are extracted through several key steps. First, a pre-emphasis filter amplifies high-frequency components of the speech signal. The signal is then divided into short overlapping frames of 25-40ms, and windowed to reduce edge effects. Each frame undergoes a Fast Fourier Transform (FFT) to obtain the power spectrum. The spectrum is filtered through a Mel-filter bank, emphasizing frequencies according to the Mel scale, which mimics human hearing. The logarithm of the filter bank energies is taken, and then

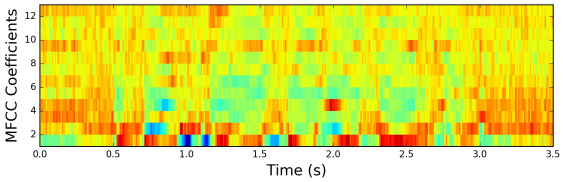


Figure 3.2. *Example of extracted MFCCs*

a Discrete Cosine Transform (DCT) is applied to produce the MFCCs. These coefficients capture essential speech characteristics for recognition.

Automatic Speech Recognition (ASR) is viewed as a sequence-to-sequence problem, as it involves mapping a sequence of processed input features to a corresponding word sequence. Below, we explore conventional methods designed to accomplish this task.

3.2 Conventional Approaches

3.2.1 Bayesian Formulation

The simplest and most common approach to the Automatic Speech Recognition (ASR) problem is to treat it as a statistical decision problem [11, 10]. In this framework, the goal is to find the most likely sequence of words given the observed acoustic signal. This is achieved by modeling the relationship between the speech signal and the corresponding text using probabilistic models.

The process involves estimating the posterior probability of a word or phoneme sequence W given the audio features X. The goal is to find the sequence $\hat{W}$ has maximizes the a posteriori probability$P(W|X)$:

$$\hat{W} = \arg\max_{W} P(W|X) \tag{3.1}$$

Using Bayes' rule, this can be formulated as:

$$\hat{W} = \arg\max_{W} \frac{P(X|W)P(W)}{P(X)} = arg\max_{W} P(X|W)P(W) \tag{3.2}$$

since $P(X)$ does not dependent on W.

The term $P(X|W)$ is generated by the acoustic model, which estimates the likelihood of the audio features X given a sequence of words W. The term $P(W)$ is provided by the language model, which estimates the probability of a word or phoneme sequence occurring in the given language. The most common approach for constructing the acoustic model is the *Hidden Markov Model*. For the language model, *N-grams* are typically used.

3.2.2 Hidden Markov Model

Hidden Markov Models (HMMs) [11] are statistical models that represent systems which transition between a series of hidden states $S = [s_1, s_2, ..., s_N]$ over time, where

each state can emit observable outputs $O = [o_1, o_2, ..., o_T]$, which in the case of speech recognition are word segments or phonemes.

Their key assumptions are based on the Markov property, which posits that the future state and the past state are independent given the current state:

$$s_{t-1} \perp\!\!\!\perp s_{t+1} | \, s_t \tag{3.3}$$

and the observation independence assumption, meaning the current output is conditionally independent of other previous observations given the current state.

$$o_t \perp\!\!\!\perp o_{t-1} | \, s_t \tag{3.4}$$

HMMs require the aforementioned sets of states S and observations O, as well as transition probabilities a_{ij} and emission probabilities bj. The transition probabilities a_{ij} are modeled as:

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i) \tag{3.5}$$

where they represent the probability of transitioning from state s_i to state s_j. The emission probabilities bj are modeled as:

$$b_j(o_t) = P(o_t | q_t = s_t) \tag{3.6}$$

where they represent the probability of observing o_t given the state s_t.

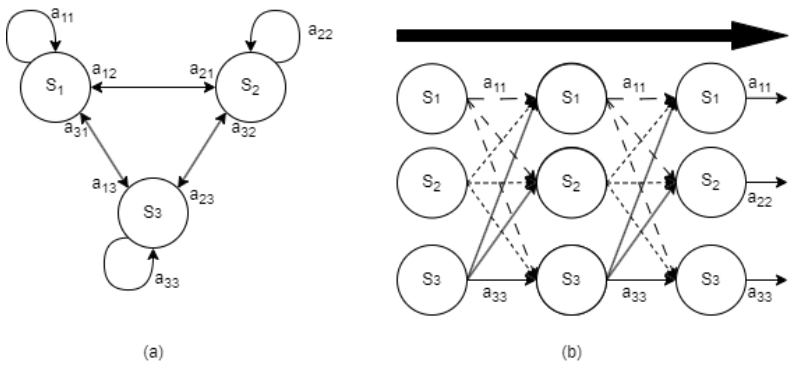


Figure 3.3. *(a) HMM with 3 states and the transition probabilities between states. (b) If a HMM is "unrolled" on time, a lattice of all possible states at each step is created*

The goal of the an HMM is to model successfully the transitions between states s_i and the output emissions o_i, given a dataset $X = [x_1, ..., x_M]$, i.e. to find the best path in the lattice shown in Fig. 3.3. This requires to calculate the probabilities $P(s_i|X)$ and $P(s_{i-1}, s_i|X)$, since they represent the posterior probability and the posterior transition probability given the data X. That can be effectively achieved using the *Baum-Welch* [57] or *Forward-Backward* [58] algorithm.

3.2.3 Forward-Backward Algorithm

First we have to define the terms $\alpha(s_i), b(s_i)$:

$$\alpha(s_i) = P(x_1, ..., x_i, s_i) \tag{3.7}$$

$$b(s_i) = P(x_{i+1}, ..., x_M | s_i) \tag{3.8}$$

The term $\alpha(s_i)$ (forward variable) represents the distribution over all observed data x and state s until the time step i. The term $b(s_i)$ (backward variable) represents the distribution over all future data until M, given the current state s_i.

The following two terms can be computed recursively using these equations:

$$\alpha(s_i) = P(x_i|s_i) \sum_{s_{i-1}} \alpha(s_{i-1}) P(s_i|s_{i-1}) \tag{3.9}$$

$$b(s_i) = \sum_{s_{i+1}} b(s_{i+1}) P(x_{i+1}|s_{i+1}) P(s_{i+1}|s_i) \tag{3.10}$$

Given these recursive equations, $P(s_i|X)$, $P(s_{i-1}, s_i|X)$ are calculated as:

$$P(s_i|X) = \frac{\alpha(s_i) b(s_i)}{P(X)} \tag{3.11}$$

$$P(s_{i-1}, s_i|X) = \frac{\alpha(s_{i-1}) P(x_i|s_i) P(s_i|s_{i-1}) b(s_i)}{P(X)} \tag{3.12}$$

where $P(X) = \sum_{s_i} \alpha(s_i) b(s_i)$.

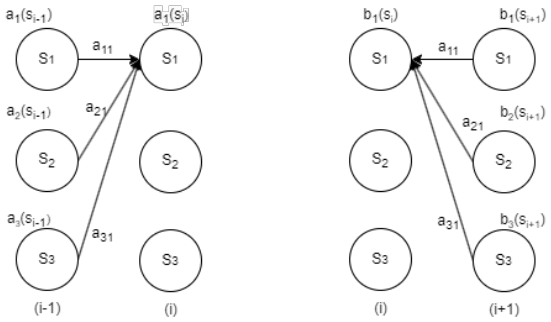


Figure 3.4. *Left: Forward pass, Right: Backward pass, used for calculating the terms $\alpha(s_i)$ and $b(s_i)$*

3.2.4 Viterbi Algorithm

The next step after creating an HMM model is to use an algorithm to determine the most likely sequence of states for a given input sequence. This can be accomplished using the *Viterbi* algorithm [59], which consists of the following four steps:

Step 1: Initialization
Initialize the probability $\delta(i)$ of each starting state s_i as:

$$\delta(i) = \pi_i b_i(o_1) \tag{3.13}$$

where π_i is the initial probability of state s_i. and the back-pointer ψ_i:

$$\psi_i = 0 \tag{3.14}$$

Step 2: Recursion
For each time step $t = 2, 3, ..., T$ and state s_i calculate

$$\delta_t(j) = \max_i(\delta_{t-1}\alpha_{ij}b_j(o_t) \tag{3.15}$$

and update back-pointer as:

$$\psi_t(j) = arg\max_i[\delta_{t-1}\alpha_{ij}] \tag{3.16}$$

Step 3: Termination
At the final time step T find the state with the highest probability:

$$P^* = max_i\delta_T(i) \tag{3.17}$$

and update back-pointer as:

$$\psi_t(j) = arg\max_i[\delta_{t-1}\alpha_{ij}] \tag{3.18}$$

Step 4: Path Backtracking
To retrieve the most probable sequence of states, trace back from the best final state using the back-pointer ψ_t :

$$s_t^* = \psi_{t+1}(s_{t+1}^*) \tag{3.19}$$

and update back-pointer as:

$$\psi_t(j) = arg\max_i[\delta_{t-1}\alpha_{ij}] \tag{3.20}$$

This algorithm computes the most likely sequence of states by exploring all possible paths in the lattice in Fig. 3.3, and keeping track of only the most probable ones at each step, discarding less likely options as it moves forward. At the end, it traces back the best sequence, ensuring the overall highest probability

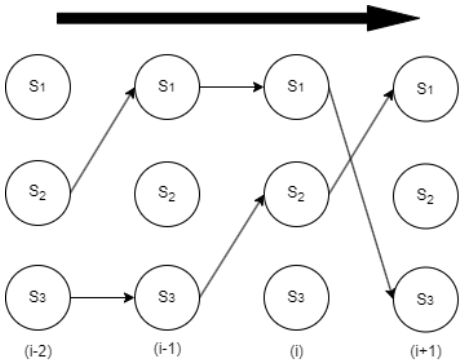


Figure 3.5. *Possible paths discovered by the Viterbi algorithm*

3.2.5 HMM-GMM Models & N-grams

HMM-GMM:

We already discussed how HMM's are trained and decoded. What remains is to define how the transition probabilities a_{ij} are computed and how the $b_j(o_t)$ are modeled. Finally we'll discuss about the most common approach of modeling a language model $P(W)$, the N-grams.

The transition probabilities a_{ij} are usually calculated from the given training data, while b_t are represented as Gaussian Mixture Models (GMM) [11, 10]:

$$b_t(o_t) = \sum_{k=1}^{K} \pi_{ik} N(o_t | m_{ik}, \Sigma_{ik}) \tag{3.21}$$

where π_{ik} represents the initial state probability.

The combination of HMM transition modeling and GMM emmision modeling produces the HMM-GMM model, which is a powerful framework particularly in speech recognition and other time-series data analysis. In the context of speech recognition, HMMs is used to model the transition from each phoneme or word to another. HMM-GMM's parameters are trained using algorithm like Expectation Maximization [REF] and Forward-Backward Algorithm.

N-grams:
The most simple approach of calculating $P(W)$, i.e. the probability of a word or phoneme W being emitted, is to calculate how frequently it appears in a 'window' of previous $N-1$ terms in our available corpora. N-grams [10] model this exact approach, where the probability of the term W is calculated given the previous $N-1$ terms:

$$P(W) = P(W_1, ..., W_M) = \prod_M P(W_n | W_{n-1}, ..., W_{n-N+1}) \tag{3.22}$$

For example, for $N = 3$, the probability of a term W_i is:

$$P(W_i | W_{i-1}, W_{i-2}) = \frac{C(W_{i-2}, W_{i-1}, W_i)}{C(W_{i-2}, W_{i-1})} \tag{3.23}$$

where $C(X)$ is the number of sequence of terms X found in the given corpora.

3.3 Modern Approaches

3.3.1 DNN-HMM Models

While GMMs are effective for modeling emission probabilities, they are limited by their assumption that emissions follow a Gaussian mixture distribution, which does not adequately capture the true nature of speech.

To overcome these limitations, the DNN-HMM (Deep Neural Network-Hidden Markov Model) [12] framework was introduced, where the DNN replaces the GMM in modeling emission probabilities for the HMM. Thanks to their complex, multi-layered architecture, DNNs excel at capturing intricate patterns in high-dimensional data, making them a much more effective alternative to GMMs for acoustic modeling.

Training this hybrid model occurs in two stages: first, the HMM is trained using traditional techniques like Maximum Likelihood Estimation (MLE) or Expectation-Maximization (EM) on pre-aligned data, so it learns the alignment between the states-phonemes and acoustic features. Second, the DNN is trained with labeled data, where the labels correspond to the HMM states. The DNN takes acoustic feature vectors as input and outputs the posterior probabilities of the HMM states. During decoding, the DNN-HMM model utilizes the Viterbi algorithm to find the most probable sequence of HMM states, similar to HMM-GMM decoding.

DNN-HMM models were a major improvement over GMM-HMMs, since DNNs can capture more complex, non-linear relationships in the data than GMMs, leading to better discrimination between different phonemes or acoustic units, and they are more robust to noise and variations in speaker accent.

However, they have been increasingly replaced by end-to-end models like RNNs and Transformers in speech recognition. These models remove the need for HMMs and directly map input features to transcriptions, simplifying the architecture and further improving accuracy.

3.3.2 Challenges of end-to-end approaches

The major challenge that end-to-end approaches faces is the uncertainty in alignment between input and output sequences. Speech and text sequences typically differ in length, and the alignment between them is often unknown. In traditional models like HMM-GMM, explicit alignment is required, whereas in end-to-end ASR, the model directly produces text from speech without relying on predefined phonetic alignments.

The input speech may vary in speaking rate, resulting in different temporal lengths for the same words or phonemes. An end-to-end model must learn these variations and adapt to generate the correct output dynamically. Additionally, during transcription, some parts of the input, such as pauses or silence, may not correspond to any output symbol. Thus, a mechanism to handle non-emitting states is essential in end-to-end models.

To address these challenges, the Connectionist Temporal Classification (CTC) [2] loss function and the Recurrent Neural Network Transducer (RNN-T) were developed [13].

3.3.3 Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification (CTC) [2] is a loss function that enables RNNs [43] and Transformers [1] to be trained for sequence transcription tasks without requiring pre-aligned input and target sequences. To address the lack of alignment, CTC introduces a special *blank* token, which represents a null output.

The blank token is used to produce many possible alignments a between the input x and the output y. For example, if the blank token is denoted as $'-'$, we have the alignments $(a, -, b, -, -, c), (-, a, -, b, c, -)$ for the sequence (a, b, c). Also, in cases where the same label appears on successive time-steps in an alignment a, the repeats are removed: therefore (a, b, b, b, c, c) and $(a, -, b, c, -)$ correspond to the same sequence, (a, b, c).

CTC loss can be applied to both RNN and Transformer architectures, as it only requires an output classification layer with units corresponding to each transcription label, along with an extra unit for the blank token. For a given input feature sequence x of length T, the output vectors y_t generated during inference are normalized using the softmax function. Each value represents the probability of emitting the label (or blank) with index k at time t.

$$P(k, t|x) = \frac{e^{y_t^k}}{\sum_{k'} e^{y_t^{k'}}} \tag{3.24}$$

where y_t^k is the k-ith element of y_t. Given $P(k, t|x)$, we can calculate the probability of each token α_t in the alignment α.

The goal is to find the alignment a which maximizes the probability $P(y|x)$. To achieve this, we must first define the probability of each alignment a:

$$P(a|x) = \prod_{t=1}^{T} P(a_t, t|x) \tag{3.25}$$

where a_t is the emitting token at step t. Given 3.25, we can define $P(y|x)$ as:

$$P(y|x) = \sum_{a \in B} P(a|x) \tag{3.26}$$

where B represents a set that contains all possible alignments between output y and input x. The intuition behind the 'integration' of all possible alignments in 3.26 is that, because we don't know where the labels within a particular transcription will occur, we sum over all the places where they could occur. Given a target sequence y^*, the network can be trained using the CTC objective loss:

$$L_{CTC} = -logP(y^*|x) \tag{3.27}$$

Given the above approach, we conclude that the CTC modeling makes the following assumptions:
Conditional Independence Between Outputs: CTC assumes that the output predictions at each time step are conditionally independent of one another, given the input. This means that the model doesn't explicitly capture dependencies between different output elements, which can be limiting for tasks that require strong contextual understanding.

Monotonic Progression: It assumes that the target sequence appears in a monotonically ordered manner within the input, meaning that the order of the output sequence is preserved in the input sequence.

These assumption lead to the following limitations:
Monotonic Alignment Requirement: CTC assumes that the target sequence has a monotonic alignment with the input, i.e, it requires $|y| \leq |x|$.
Alignment Ambiguity: CTC doesn't explicitly learn the alignment between the input and target sequences. While it marginalizes over all possible alignments, this can sometimes lead to suboptimal training or incorrect sequence predictions, especially in noisy data.

Despite these limitations, CTC remains the most commonly used objective function in speech recognition tasks due to its effectiveness and simplicity. However, RNN-T (Recurrent Neural Network Transducer) was introduced to address some of the limitations of CTC, offering more flexibility in handling variable-length sequences and improving alignment during transcription.

3.3.4 Recurrent Neural Network Transducer (RNN-T)

As we previously discussed, when the output sequence y is longer than the input sequence x, CTC struggles to effectively capture the inter-dependencies between the output elements. The Recurrent Neural Network Transducer (RNN-T) [13] addresses this limitation by extending CTC, defining a probability distribution over output sequences of varying lengths and jointly modeling both input-output and output-output relationships.

Assume an input sequence $x = [x_1, .., x_T]$ of length T, and an output sequence $y = [y_1, ..., y_U]$ of length U. As in CTC modeling, RNN-T introduces a new output token , $\varnothing$, which denotes the *null* output. This is used similarly as the blank token of CTC, as an 'nothing' output which enables the alignment between input and output. For example, the sequence $(a, b, \varnothing, \varnothing, c, \varnothing)$ is equivalent to (a, b, c). We call sequences like $(a, b, \varnothing, \varnothing, c, \varnothing)$ *alignments* $\boldsymbol{a}$, since the location of the null symbols determines an alignment between the input and output sequence. Given the input x, the RNN-T defines a conditional distribution $P(a|x)$, which is then collapsed onto the following distribution:

$$P(y|x) = \sum_{a \in B} P(a|x) \tag{3.28}$$

where B represents a set that contains all possible alignments between output y and input x.

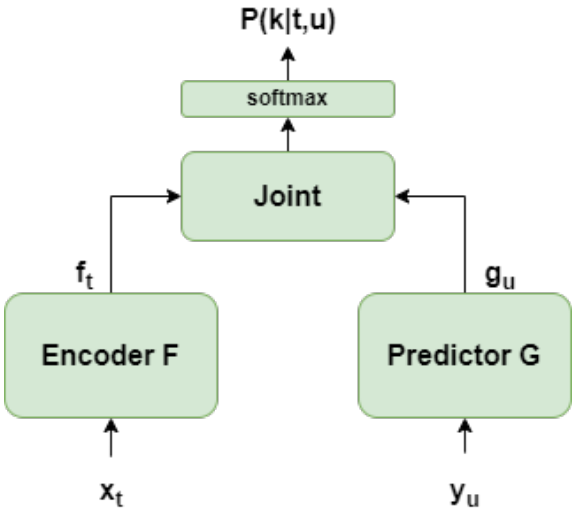


Figure 3.6. *The RNN-T architecture*

Two distinct networks are used to determine $P(a|x)$. The first network, referred to as the *transcription network* or *encoder* F, scan the input sequence x and extracts feature vectors f_i that capture the acoustic representations.

The second network, referred to as the *prediction network* or *predictor* G, scans the output sequence y and extracts linguistic features g_i. Given an acoustic vector $f_t, 0 \leq t \leq T$ and and linguistic vector $g_u, 0 \leq u \leq U$, the output distribution over all k labels (including the null output) is:

$$P(k|t,u) = \frac{h(k,t,u)}{\sum_{k'} h(k',t,u)} \tag{3.29}$$

where $h(k,t,u)$ represents the density function that is defined as:

$$h(k,t,u) = \phi(f_t^k + g_u^k) \tag{3.30}$$

where superscript k denotes the k-th element of the vectors, and ϕ is an activation function. The density function h is generated by a straightforward feed-forward network known as the *joint network*, and the final distribution is obtained by applying a softmax function to the output of the joint network. The complete architecture of the RNN-T is illustrated in Fig. 3.6.

The probability $P(k|t,u)$ is used to determine the transition probabilities in the lattice shown in Fiq. 3.7. The set of possible paths from the bottom left to the terminal node in the top right corresponds to the complete set of alignments between x and y, therefore all possible input-output alignments are assigned a probability, the sum of which is the total probability $P(y|x)$.

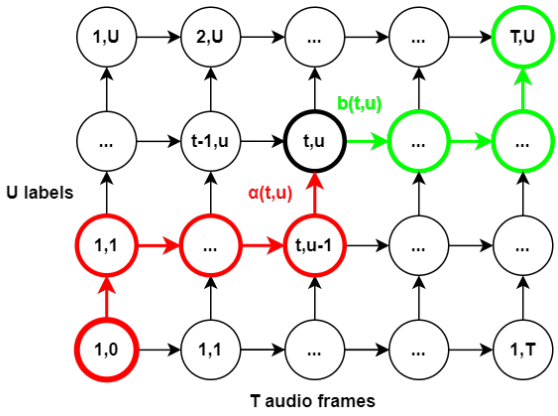


Figure 3.7. *The training lattice and the best alignment path. WIth red the forward path and forward probability, with green the backward path and the backward probability. Source [2]*

$P(y|x)$ can be efficiently calculated through a forward-backward algorithm that we'll describe below. First we define the *forward variable* $\alpha(t,u)$ and *backward variable* $b(t,u)$.

Forward variable $\alpha(t,u)$ is defined as the probability of emitting the labels $y_{[1:u]}$ while using the linguistic features $f_{[1:t]}$. It can be calculated recursively for all $0 \leq u \leq U, 0 \leq t \leq T$ as:

$$\alpha(t,u) = \alpha(t-1,u)\varnothing(t-1,u) + \alpha(t,u-1)y(t,u-1) \tag{3.31}$$

where

$$y(t,u) = P(y_{u+1}|t,u) \tag{3.32}$$

$$\varnothing(t,u) = P(\varnothing|t,u) \tag{3.33}$$

are the probabilities of emitting the label y_{u+1} or the null label $\varnothing$ at step (t,u) of the lattice. The initial condition for 3.31 is $\alpha(1,0) = 1$. The total output sequence probability is equal to the forward variable at the terminal node:

$$P(y|x) = \alpha(T,U)\varnothing(T,U) \tag{3.34}$$

Backward variable $b(t,u)$ is defined as the probability of emitting the labels $y_{[u+1:U]}$ while using the linguistic features $f_{[t+1:T]}$. Then

$$b(t,u) = b(t+1,u)\varnothing(t,u) + b(t,u+1)y(t,u) \tag{3.35}$$

with initial condition $b(T,U) = \varnothing(T,U)$.

From the definition of forward and backward variables, it follows that their product $\alpha(t,u)b(t,u)$ at any point (t,u) is equal to the probability of emitting the complete output sequence, if the label emitted at the step (t,u) is the label y_u.

Given a target output sequence y^*, the goal of training is to maximize the probability $P(y^*|x)$, i.e. to find the best possible path in the lattice of Fiq. 3.7. The loss used during training is the log-loss:

$$L_{RNNT} = -lnP(y^*|x) \tag{3.36}$$

Analysing the diffusion of probability through the output lattice shows that $Pr(y^*|x)$ is equal to the sum of $\alpha(t,u)\beta(t,u)$ over any top-left to bottom-right diagonal through the nodes, thus we can write that:

$$P(y^*|x) = \sum_{(t,u):t+u=n} \alpha(t,u)b(t,u), \forall n : 1 \leq n \leq T+U \tag{3.37}$$

This shows that the gradients produced by L_{RNNT} aim to maximize the forward-backward variables of the best path while minimizing them on all other paths. Thanks to the two sub-nets, this approach can better cover the input-output and output-output correlations, offering a more robust modeling of the speech recognition problem.

3.3.5 Wav2vec2.0

Another approach to improving speech recognition is by extracting more robust features from raw audio, which enhances the alignment quality in the previously mentioned methods. Better feature extraction allows models to capture more relevant information from the speech signal, resulting in improved performance across various transcription techniques.

Wav2Vec 2.0 [3] is a self-supervised learning framework developed for automatic speech recognition (ASR), designed to automatically discover and extract rich, high-level features from raw speech data. This automated process enhances the model's capacity to capture complex audio patterns, resulting in better training efficiency and improved performance across diverse ASR tasks. Additionally, Wav2Vec 2.0 supports effective self-supervised training on acoustic data, allowing the model to learn from vast amounts of unlabeled speech.

The core idea behind the Wav2Vec 2.0 framework is to replicate the Masked Language Modeling (MLM) [14] approach, originally introduced in Natural Language Processing (NLP), for speech data. In MLM, portions of an input sentence are 'masked,' and the model is trained to accurately predict the masked words. Similarly, Wav2Vec 2.0 masks segments of the processed input audio and trains the model to predict which pseudo-phonemes correspond to the masked parts. This enables the model to learn meaningful speech representations in a self-supervised manner.

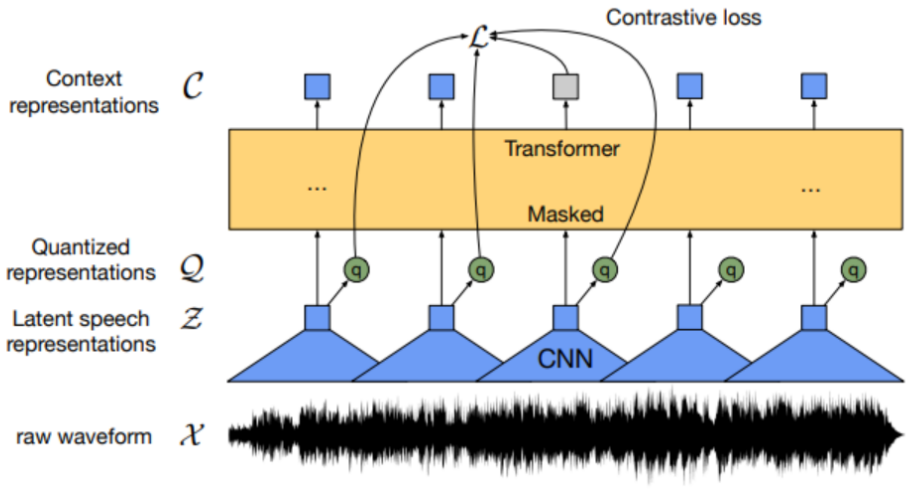


Figure 3.8. *The Wav2Vec2.0 architecture. Source [3]*

As illustrated in 3.8, the model consists of 3 components:
Feature encoder: A multi-layer convolutional [27] network which takes as input raw audio X and output speech representations $z = [z_1, ..., z_T]$ for T time-steps.

Transformer: A transformer model [1] which receives the extracted z speech representations and builds contextual representations $c = [c_1, ..., c_T]$ which capture information from the entire sequence

Quantization module: This module handles the discretization of speech representations z into a finite set of units, functioning as a form of pseudo-phonemes. This process is carried out using product quantization. The representations z_i are divided into groups, or codebooks, denoted as G, with each codebook containing V entries. For each group, an entry e_i is chosen, and these entries $e_1, ..., e_G$ are concatenated to form a resulting vector. A linear transformation is then applied to this vector, yielding the final discrete representation q. Gumbel softmax [60] is used to achieve differentiable selection of the discrete codebooks.

During pre-training, the goal is to learn latent speech representation from the raw audio in a self-supervised manner. This is achieved by solving a contastrive task L_m , which requires to identify the true discrete representation q from a group of distractors. This is augmented by a diversity loss L_d to encourage the model to use the codebook entries equally often:

$$L = L_m + \alpha L_d \tag{3.38}$$

where α is a tuned hyper-parameter.

Contrastive loss: It is applied between the context feature c_t and the discrete representations q_t. As we mentioned, the goal for the model is to identify the true quantized latent speech representation q_t from a set of candidate represantations $\hat{q}$, which includes q_t and K distractors. The loss is defined as:

$$L_m = -log \frac{exp(sim(c_t, q_t))}{\sum_{\hat{q}} exp(sim(c_t, \hat{q}))} \tag{3.39}$$

where $sim(a, b)$ represents the cosine similarity between the vectors a, b.

Diversity loss: Contrastive loss requires that every codevectors offer positive and negative examples, in order to avoid mode collapse, i.e. to learn only specific representations. L_d is designed to increase the use of every codebook representation:

$$L_d = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} p_{g,v} log p_{g,v} \tag{3.40}$$

where $p_{g,v}$ represent the discrete representation v of the the codebook g.

Wav2vec2.0's ability to learn high-level representations from raw audio makes it particularly powerful, enabling it to adapt to different accents, background noises, and varying speech patterns. Its innovative approach marks a significant advancement in the field of speech processing, bridging the gap between traditional methods and modern deep learning techniques.

Chapter 4

Unsupervised Domain Adaptation

In this chapter, we will define the problem of Unsupervised Domain Adaptation (UDA) [61, 8] and provide an overview of various adaptation approaches. In particular, we will examine semi-supervised methods, self-supervised techniques, and domain adversarial strategies.

4.1 Problem definition

The problem of UDA can be formally defined as follows:
Consider two different distributions: the source domain distribution $S(x, y)$ and the target domain distribution $T(x, y)$. Here $x \in X$ represents feature vectors that belong to a real-valued space X, while $y \in Y$ are labels from a finite set Y. The goal is to train a model that maps target domain feature vectors x_T to their corresponding labels y_T , for samples (x_T, y_T) drawn from the target distribution T.

However, during training, we only have access to labeled samples from the source distribution $S(x, y)$,and no label information for the target distribution. In other words, only the feature vectors $T(x)$ are available. The challenge is to develop a model that can effectively bridge the gap between the two domains by leveraging the labeled source data $S(x, y)$ to predict labels for the target data, despite the absence of target labels.

4.2 Teacher-student training

Teacher-student training or pseudo-labeling is the most common approach [62, 63] for UDA and one of the earliest semi-supervised learning, since it effectively transforms the unsupervised learning problem into a supervised one. In this method, a model already trained on the source domain, the teacher model g_S , generates pseudo-labels $\hat{y}_T = g_S(x_T)$ for the target domain, which are then used by a student model for supervised training. his process is usually repeated, with the student model serving as the teacher model for the next iteration, until no further improvement is observed. More recently, soft target Teacher-Student learning has been explored for ASR [64, **?**, 64], where the KL divergence between the teacher and student output label distributions is used as the loss function.

4.2.1 Utterance filtering

Since the teacher model is solely trained on the source domain, it may produce inaccurate labels, which can lead to error propagation. To address this issue, filtering techniques are commonly used to discard any utterances that could result in poor quality outputs . In [15, 16], a Confidence Estimation Module (CEM) is used to filter out unreliable ,utterances based on the confidence scores of the transcribed labels. In [15], a multi-task training objective with confidence loss is implemented to reduce the binary cross-entropy between the estimated confidence and the binary target sequence. Furthermore, in [18] the concept of Noisy Student Training (NST) is introduced, where the student model is trained on heavily augmented input data to develop more robust representations. The augmentation in this work is performed using techniques like SpecAugment [19].

4.2.2 Teacher update

Kaizen

Another approach to address error propagation involves updating teacher's parameters to improve the quality of the pseudo-labels. In Kaizen [4], the teacher is updated every Δ number of steps as the Exponential Moving Average (EMA) of the student's parameters:

$$\theta_{T,t} = (1-\alpha)\theta_{T,t} + \alpha\theta_{S,t} \tag{4.1}$$

where $\theta_{T,t}, \theta_{S,t}$ are the teacher's and student's parameters at step t, respectively, and α is a discount coefficient. The factor α controls how much the teacher's parameters will change. During their experiments, they found that large α cause the teacher model to closely resemble the student model, promoting the generation of easily predictable targets, which can lead to mode collapse. On the other hand, low α values results to a more "static" teacher model, which may lead to worse performance.

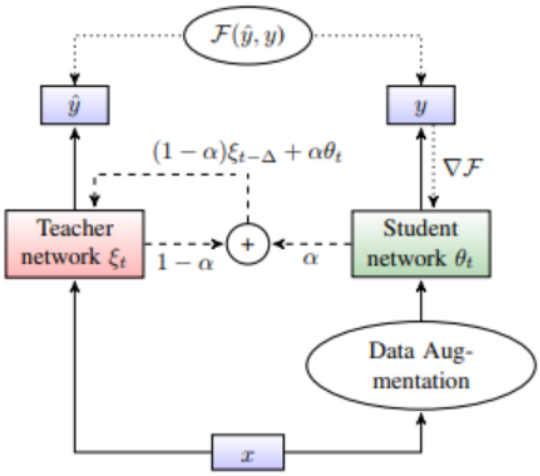


Figure 4.1. *The KAIZEN framework. Source [4]*

Meta Pseudo Labels

In Meta Pseudo Labels [5], it is observed that the student's loss on the source domain, denoted as $L_s(\theta_S)$, depends on the parameters of the teacher model θ_T, since the students model is trained of the pseudo-labels $\hat{y}_T = g(x_T; \theta_T)$. Based on this observation, it is proposed that optimizing the teacher's parameters based on the student's performance on the source domain can result in the generation of better pseudo-labels. This, in turn, improves the student's performance on both the source and target domains. The feedback loss used by the teacher model is:

$$L_s(\theta_S - \eta \nabla_{\theta_S} L_T(\theta_S; \theta_T)) \tag{4.2}$$

where L_s is the task's supervised loss applied on source domain, using the updated student model, L_T is the loss used during pseudo-labeling and η is the learning rate.
In essence, the procedure can be described as follows:

1. The teacher produces pseudo-labels $\hat{y_T}$ on target domain data x_T
2. The student is trained in a supervised manner using the pairs $(\hat{y_T}, x_T)$
3. The teacher is updated using a feedback loss from the updated student

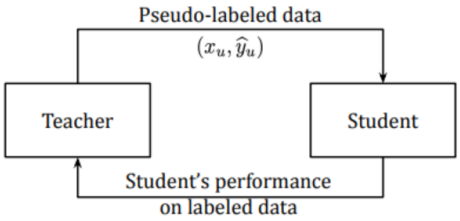


Figure 4.2. *Meta Pseudo Labels. The student is trained based on the pseudo labels generated by the teacher (top arrow). The teacher is trained based on the performance of the student on labeled data (bottom arrow). Source [5]*

This framework performs well on its own, but it has been shown that integrating auxiliary tasks, such as the teacher's supervised loss on the source domain, leads to improved performance.

4.3 Self-supervised training

Another predominant method is self-supervision. This approach was initially deployed in Natural Language Processing (NLP) tasks [6, 8] and has proven to be an effective and straigh-forward pre-training technique that facilitates domain adaptation. The core idea involves leveraging self-supervised loss during pre-training to adapt large pre-trained models to new domains. This can be accomplished either through Continual Pre-Training (CPT) [65, 66] or by constructing a multitask objective [6, 8] for the model's training.

4.3.1 Continual Pre-training

Continual Pre-Training (CPT) involves extending the training of a pre-trained language model on additional data to better suit new tasks, domains, or languages, while preserving the knowledge acquired during its initial pre-training. The core concept is to leverage the knowledge gained in the pre-training phase and apply it to specific tasks, by applying the pre-training loss on the additional data.

Continual Pre-Training (CPT) has been investigated for adapting ASR models. Robust wav2vec2 [67] examines the effectiveness of CPT for domain adaptation, highlighting the value of leveraging unlabeled in-domain data. In CASTLE [21], CPT is paired with an online pseudo-labeling strategy to adapt wav2vec2 to new domains. Cross-dataset evaluations using popular English speech corpora show that CPT helps lower error rates in the target domain. Additionally, in [68] and [69], CPT is applied for cross-lingual adaptation of wav2vec2 for Korean and Ainu, respectively.

4.3.2 Multi-task training

In multi-task training, self-supervision is employed to build a multi-task objective for the model's training.

UDALM

In UDALM [6] BERT [70], a large pre-trained NLP model, is pre-trained on target domain data using the masked language modeling (MLM) [14] task. During the final, fine-tuning stage, the model uses both a task-specific loss on the source domain data and masked language modeling loss on the target domain data. The overall loss used during fine-tuning is:

$$L = \lambda L_{TASK} + (1 - \lambda) L_{MLM} \tag{4.3}$$

where λ is a weighting factor which balances the influence of the source and target domain data. The whole process can be observed in Fiq. 4.3.

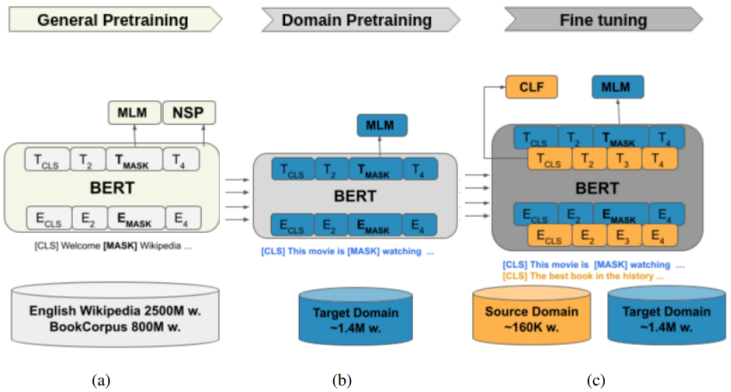


Figure 4.3. *(a) BERT pre-trained model (b) Pre-training BERT on target domain data using the MLM task. (c) Finetuning the model using the supervised task objective on source domain data, while keeping the MLM objective on target domain data. Source [6]*

During this process, the model learns the task using a task objective based on labeled source domain samples, while simultaneously adapting to the target domain data through the masked language modeling (MLM) objective. This approach leads to effective adaptation results while successfully mastering the task at hand.

Mixed Multi-Domain Self-Supervision (M2DS2)

Similar with UDALM, in [8] self-supervision is employed during the fine-tuning phase of the training to help the XLSR-53 [22] pre-trained model learn the task at hand while being adapted to the target domain. The multi-task loss used is:

$$L = L_{CTC}(x_S, y_S) + \alpha L_s(x_S) + \beta L_s(x_T) \tag{4.4}$$

where L_{CTC} is the CTC objective loss used for ASR task, L_s is the contrastive loss described in Eq. 3.39 and α, β are discount coefficients that scale the contribution of each term. The training process can be observed at Fig. 4.4.

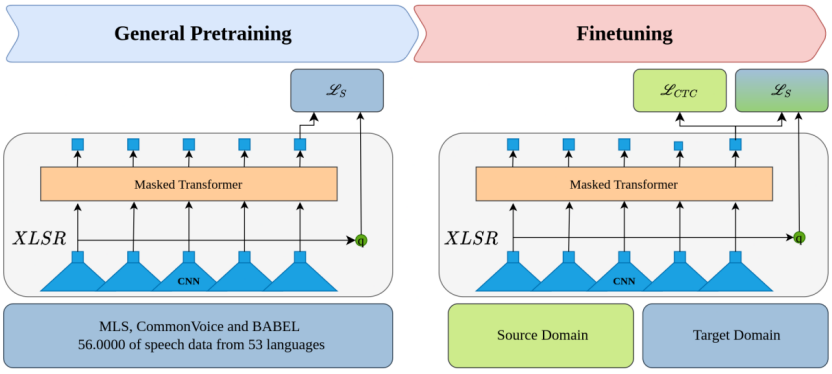


Figure 4.4. *In the left the general pre-training stage of XLSR-53 pre-trained model using the self-supervised loss L_s. In the right, the domain-adaptive finetuning stage, where both domains are used in the self-supervised task.*

Unlike [6], this approach uses data from both the source and target domains for the self-supervised task. This is crucial for preventing mode collapse, where the model relies on only a small subset of available code vectors to generate outputs. Incorporating self-supervision from both domains helps avoid this issue by encouraging the source and target code vector spaces to develop a similar structure, ensuring a more diverse and robust output.

4.4 Domain Adversarial Training

The main objective of Domain Adversarial Training (DAT) [71] is to train a model that learns deep features capable of addressing the task in the source domain, while remaining unaffected by the shift between domains. In [7] two DAT methods are proposed to address of adaptation from high-resource to low-resource languages, which share a common acoustic space.

4.4.1 Adversarial training using gradient reversal layer (GRL)

In GRL-based adversarial training, we try to learn a feature representation invariant to the domain difference, but good enough in discriminating the senone labels. The architecture used in this task involves a feature extractor, a senone classifier and a domain classifier, and it is shown at Fig. 4.5. The aim of training is to optimize the feature ex-

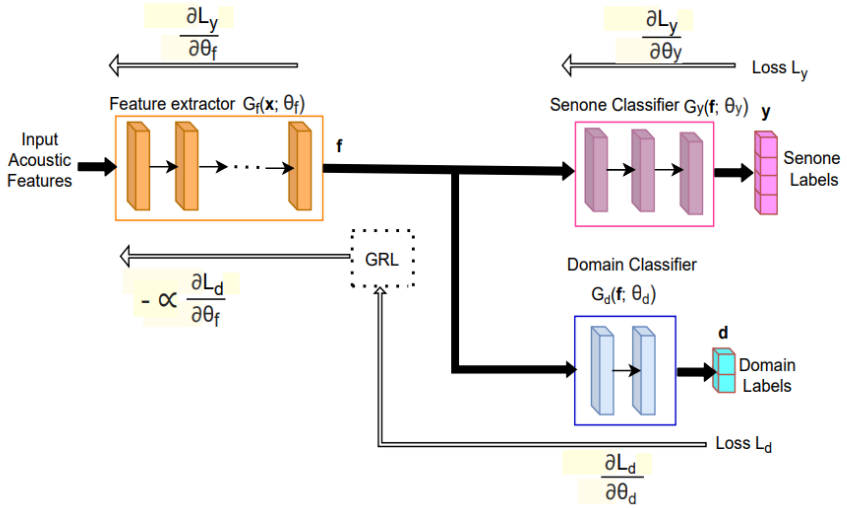


Figure 4.5. *Adversarial training using GRL. θ_f , θ_y and θ_d represent the parameters of the feature extractor, senone classifier, and domain classifier, respectively. Source [7]*

tractor such that it minimizes the senone classification loss while maximizing the domain classification loss. This encourages the network to extract features that retain senone information while being invariant to domain differences. This is achieved by using a GRL between the feature extractor and the domain classifier, which "transports" the negative of the resulting gradient of the domain classifier to the feature extractor.

4.4.2 Domain Separation Networks (DSN)

Domain separation networks (DSN) function similarly to GRL-based adversarial training. Their goal is to extract features from both domains that reduce domain variance while preserving domain-specific classification. A block diagram of the DSN architecture used in [HINDI] is shown in Fig . Private encoders E_p^t, E_p^s are used to extract domain specific features, while the shared encoder E_c is used to extract common features from both do-

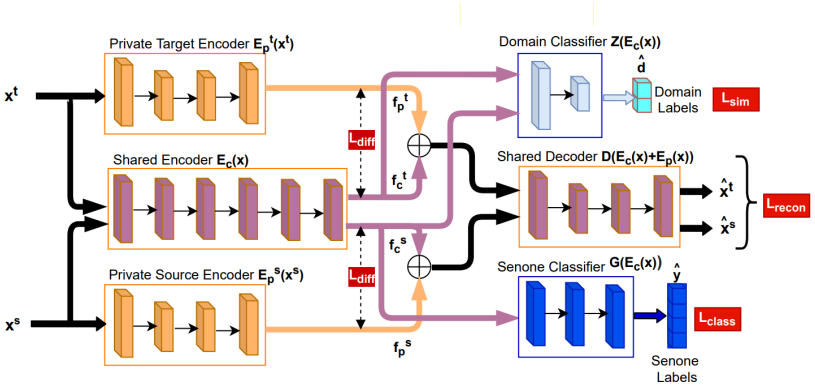


Figure 4.6. *Block diagram of DSN with the private and shared components. Source [7]*

mains. The shared decoder D is used to reconstruct the input using both private and shared features. The senone classifier G maps the shared features to the senone labels y, while the domain classifier Z is used to map the common features to their domains. The overall loss used is:

$$L = L_{class} + \beta L_{sim} + \gamma L_{diff} + \delta L_{recon} \tag{4.5}$$

where L_{class} represents the senone classification loss, L_{sim} is the similarity loss of the domain classifier Z, L_{diff} is the difference between the shared and common extracted features, L_{recon} is the input reconstruction loss, and β, γ, δ are hyper-parameters.

Chapter 5

Multi-Stage Domain Adaptation

In this chapter, we present our proposed training methodology for UDA, along with the experimental setup and the results of our experiments.

Inspired by the successful domain adaptation results in image recognition tasks using the Meta PL framework [5], we investigate whether it can be effectively applied to sequence-to-sequence tasks, such as Automatic Speech Recognition (ASR). Additionally, we introduce Multi-Stage Domain Adaptation (MSDA), a two-stage domain adaptation approach that integrates self-supervised techniques from M2DS2 [8] with semi-supervised methods from Meta PL, all within the Wav2Vec2.0 [3] framework. Our findings demonstrate that MSDA can substantially enhance model performance in the target domain, particularly for low-resource languages and weakly supervised data.

5.1 Methodology

Our approach is illustrated in Fig. 5.1. It consists of two stages of adaptation: a self-supervised stage followed by a semi-supervised stage, which we collectively refer to as Multi-Stage Domain Adaptation (MSDA).

Stage 1:
In this stage, we follow the method proposed in [15] to develop a teacher model trained on both domains in a self-supervised manner, using the following objective:

$$L_{M2DS2} = L_{CTC}(x_s, y_s) + \alpha L_s(x_s) + \beta L_s(x_s) \tag{5.1}$$

Here L_{CTC} is the CTC objective loss we discussed in 3.3.3, applied on the source domain batch (x_s, y_s), ensuring accurate alignment between the input speech and the target transcription. The terms $L_s(x_s), L_s(x_t)$ are the self-supervision losses from Eq. 3.38, applied to the source and target speech data respectively, as described in [3]. These losses encourage the model to learn robust, domain-invariant representations from both domains, successfully achieving an initial level of adaptation. The parameters α, β are discount coefficients.

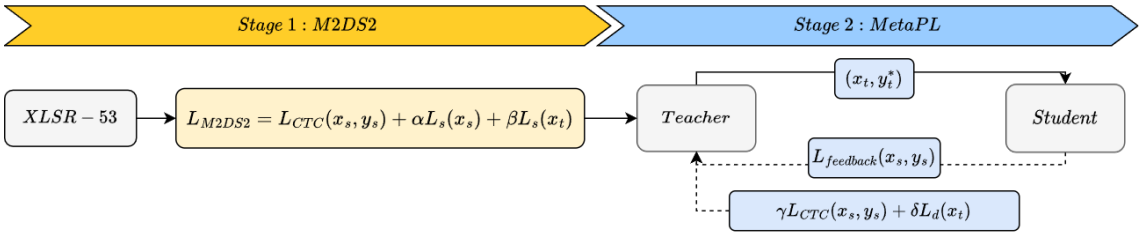


Figure 5.1. *The Multi-Stage Domain Adaptation (MSDA). At Stage 1, we use self-supervision to adapt our pre-trained model to both source (x_s, y_s) and target (x_Tt domain, using the loss objective described in Eq. 4.4. At Stage 2, we apply an enhanced version of Meta PL to further improve the overall adaptation. In this stage, the teacher provides pseudo-labelsy_t^* for student training, and itself is trained using the objective described in Eq 5.3.*

Stage 2:
In this stage we follow the framework proposed in [12], using the teacher from *Stage 1* to generate pseudo-labels y_t^* for the target domain data x_t. The student model utilizes these pseudo-labels to train on using the CTC objective:

$$L_S = L_{CTC}(x_t, y_t^*) \tag{5.2}$$

Additionally, the teacher is updated according the following objective:

$$L_T = L_{feedback}(x_s, y_s) + \gamma L_{CTC}(x_s, y_s) + \delta L_d(x_t) \tag{5.3}$$

where $L_{feedback}$ represents student's CTC loss on the source domain, which provides feedback to the teacher, guiding it to generate better pseudo-labels. L_{CTC} is the teacher's CTC loss on source domain, while L_d is the diversity loss described in [3]. The parameters γ, δ are discount coefficient.

At *Stage 1*, we apply SpecAugment [10] to both domain audio inputs to train a more robust teacher. However, in *Stage 2*, SpecAugment is used only on student's inputs. We avoid applying it to the teacher's inputs in this stage, as the added noise and augmentation lead to the generation of low-quality pseudo-labels.

We employ M2DS2 training as the first stage of adaptation, as multi-domain self-supervision not only enables effective domain adaptation but also mitigates the risk of mode collapse. This method is crucial for producing high-quality pseudo-labels, which are vital for the success of the second stage.

Furthermore, applying diversity loss L_d to the target domain during *Stage 2* is essential to prevent mode collapse in the teacher model. Simultaneously, the CTC loss L_{CTC} applied to the source domain ensures that the teacher model retains its adaptability to the source domain despite the adjustments made the feedback loss $L_{feedback}$.

5.2 Experimental Setup

5.2.1 Pre-trained model

For our base model we utilized XLSR-53 [22], a state-of-the-art pre-trained speech model developed on the Wav2Vec 2.0 [3] architecture. It is trained on a diverse corpus and extensive corpus, comprising 56,000 hours of speech data across 53 languages. This cross-lingual training enables XLSR-53 to learn cross-lingual speech representations, by creating a single quantized speech representation space which is shared across languages.

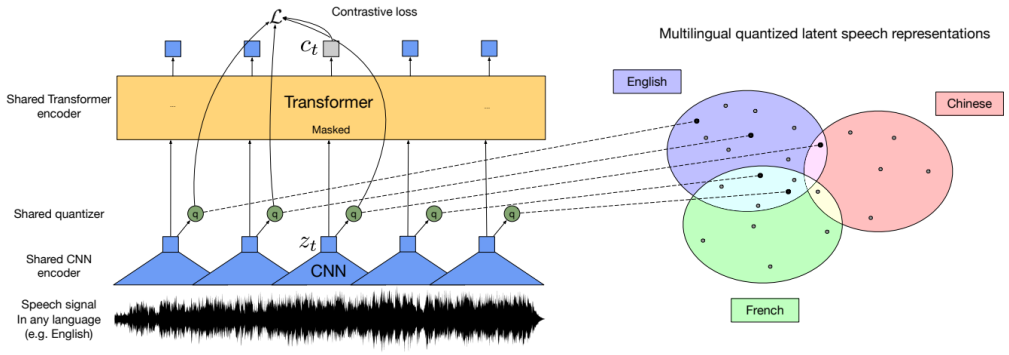


Figure 5.2. *XLSR-53: Shared quantized speech representation space created by a shared quantization module. The model created bridges between languages*

We employed the openly available version of XLSR-53[1] , which offers a pre-trained implementation ready for fine-tuning and adaptation tasks.

5.2.2 Dataset

Our dataset of choice is the Greek Podcast Corpus (GPC) [23], which consists of 3,124 hours of Greek-language podcast audio across 16 diverse categories, including *True Crime*, *News*, *Art*, and others. To enable experiments of various scales, the dataset is divided into smaller subsets: GPC-50, GPC-20, and GPC-10, which contain 50, 20, and 10 hours of audio per category, respectively. The transcriptions for these subsets were generated automatically using the WhisperX pipeline [24], resulting in weakly labeled audio.

For our experiments, we focused on the GPC-20 subset, treating specific categories as distinct domains:

1. **TrueCrime:** Podcasts that explore true crime stories and police investigations.
2. **Education**: Podcasts covering topics such as history , sociology and other educational subjects.
3. **Business:** Podcasts focused on business administration, start-ups, and investments.
4. **Comedy:** Podcasts dedicated to stand-up comedy and comedic performances.

Each domain has different content, distinct linguistic characteristics and uses specific vocabulary. For example, *True Crime* podcasts often include formal and investigative

[1] https://huggingface.co/facebook/wav2vec2-large-xlsr-53

terminology. In contrast, *Education* podcasts use more academic language, making use of vocabulary related to history, science, and humanities. *Business* podcasts' lexicon contains specialized economic terms, and entrepreneurial phrases, while Comedy podcasts, have primarily informal tone, and rely on humor and slang.

To indicate the linguistic and speech differences the selected domains, in [24] the authors used whisper-large-v2 to obtain 8%, 5%, 15%, 28% WER for *TrueCrime, Education, Business*, and *Comedy* domains respectively. Furthermore, we assessed the similarity between these domains using the ATDS [17] (Acoustic and Textual Domain Similarity) measure, which accounts for both acoustic and linguistic differences between datasets. In Table 5.1 we present the similarity scores between the selected domains. We observe that the metric skews towards 1 in all cases, which is logical since ADTS metric was developed for cross-language comparison and in our case is applied on monolingual data.

Similarity	**TrueCrime**	**Education**	**Business**	**Comedy**
TrueCrime	-	0.93937	0.91852	0.89848
Education	0.93937	-	0.96935	0.94209
Business	0.91852	0.96935	-	0.96998
Comedy	0.89848	0.94209	0.96998	-

Table 5.1. *ATDS measure between selected domains.*

5.2.3 Baselines & Experiments

We evaluate the effectiveness of Multi-Stage Domain Adaptation (MSDA) for UDA using the Word Error Rate (WER) metric. The Word Error Rate (WER) [11, 10] is a common evaluation metric used to measure the performance of speech recognition systems. It calculates the proportion of words that were incorrectly predicted by comparing the transcription output to the ground truth, accounting for substitutions, insertions, and deletions.

$$WER = \frac{S + D + I}{N} \tag{5.4}$$

where S is number of word substitutions, D is the number of word deletions, I is the number of word insertions and N is the total number of words in the ground truth transcription.

Our approach is compared with the following four baselines:

1. **Finetuned:** Supervised fine-tuning of XLSR-53 on the source domain, following the Eq. 4.4 with $\alpha = \beta = 0$.

2. **M2DS2:** Training with M2DS2 using both source and target domains, following the Eq. 4.4 with $\alpha = \beta = 0.01$.

3. **Finetuned - Meta PL:** Application of the Meta PL framework to the fine-tuned model on the target domain, following Eq. 5.3 with $\gamma = \delta = 0$, i.e., using only student's feedback.

4. **M2DS2 - Meta PL:** application of the Meta PL framework to the M2DS2-trained models, following Eq. 5.3 with $\gamma = \delta = 0$.

For MSDA training we set $\alpha = \beta = 0.01$ in Eq. 4.4 and$\gamma = \delta = 0.0001$ in Eq. 5.3. Furthermore, we experiment with different settings of γ and δ values, to examine their effects on the model's performance. Specifically, we conduct two experiments, each focusing on one of the coefficients. In each experiment, one coefficient is fixed at 0.001 while the other is varied between 1 and 0.00001.

5.3 Results

In Table 5.2 we compare MSDA with the aforementioned baselines across twelve adaptation scenarios, i.e., cross domain evaluation between the four selected categories of GPC-20. Each column represent the result of the corresponding training on the target domain, which can be identified in the second column of each row.

Domains		**Finetuned**	**Finetuned Meta PL**	**M2DS2**	**M2DS2 Meta PL**	**MSDA**
source	**target**					
TrueCrime	Education	48.8	43.47	45.32	42.35	**37.98**
	Business	62.7	57.16	58.26	56.27	**50.3**
	Comedy	61.9	57.56	68.77	57.1	**49.21**
Education	TrueCrime	37.28	**33.3**	43.23	42.07	39.89
	Business	50.93	48.25	56.01	55.17	**47.03**
	Comedy	52.54	**50.7**	71.14	58.06	56.16
Business	TrueCrime	41.07	**38.5**	52.2	43.02	41.01
	Education	42.38	**38.9**	55.32	48.27	42.4
	Comedy	51.93	53.5	66.98	64.3	**50.02**
Comedy	TrueCrime	43.32	40.01	50.27	42.67	**39.08**
	Education	42.78	40.8	46.5	39.71	**38.77**
	Business	51.41	49.17	59.52	50.12	**47.3**
Average WER		48.92	45.94	56.13	49.93	**44.93**

Table 5.2. *Performance results on GPC-20 domain pairs, using the WER metric. We observe that even when the teacher model (M2DS2) performs poorly on the target domain, the MSDA-trained model effectively reduces WER by 4% to 14%.*

First, we focus on the performance of the Finetuned-Meta PL model. We observe that employing the Meta PL adaptation method reduces the Finetuned model's initial WER by 2%-5% across all test cases and, moreover, outperforms the M2DS2 adaptation in every scenario. This demonstrates that the Meta PL framework is effectively applicable on ASR tasks and performs better than M2DS2 adaptation.

Next, we evaluate the performance of the M2DS2-Meta PL model. While the Meta PL adaptation method results in a WER reduction and improves performance compared to the initial M2DS2 model, it performs worse than the Finetuned-Meta PL model. We attribute this performance gap to potential mode collapse of the teacher model during Meta PL training.

Lastly we examine the MSDA model. We observe that the MSDA model outperforms the previous models in most cases, achieving WER reduction up to 10%-12% in some instances. The Finetuned-Meta PL model only surpasses MSDA in scenarios where M2DS2 training fails to achieve any adaptation and results in a worse WER than the Finetuned model. Even in these cases, the MSDA model manages to reduce M2DS2 model's WER by 5%-14%, indicating its ability to result to significantly more robust student even in cases where the teacher is really bad. We attribute M2DS2's inability to achieve adaptation in certain cases to the weakly supervised nature of our training dataset, as well as the minimal hyper-parameter tuning during development.

In Fig. 5.3 we examine the effects of varying γ and δ values on the model's performance in the *TrueCrime - Education* source-target domain adaptation setting. The figure illustrates how different combinations of these hyperparameters influence the model's ability to generalize from the TrueCrime domain (source) to the Education domain (target).

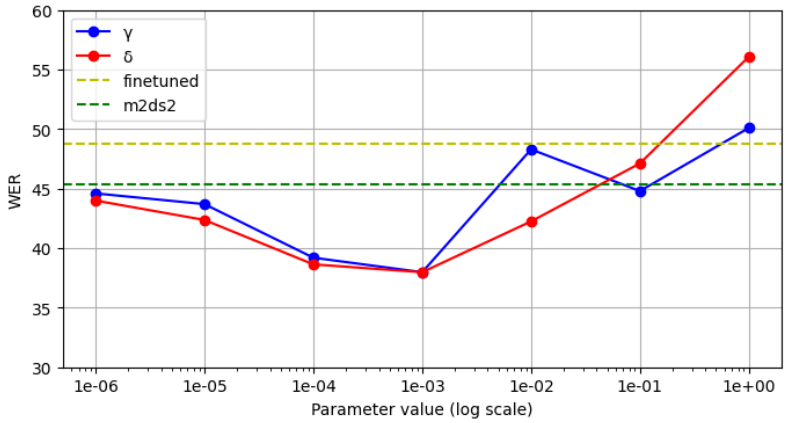


Figure 5.3. *WER comparison between different setting of γ (blue) and δ (red), finetuned (yellow) and M2DS2 (green) baselines , evaluated in the TrueCrime - Education source-target domain adaptation setting.*

We observe that for $\gamma, \delta > 10^{-3}$ MSDA fails to surpass the performance of the Finetuned and M2DS2 baselines. We hypothesize that for $\gamma > 10^{-3}$ the teacher model begins to overfit on the source domain, leading to poor pseudo-label generation. On the other hand, when $\delta > 10^{-3}$, the influence of L_d becomes substantial enough to undermine any adaptation achieved during Stage 1 . Furthermore, when $\gamma, \delta < 10^{-4}$, MSDA leads to poorer adaptation compared to the case where $\gamma, \delta \in [10^{-4}, 10^{-3}]$. We believe this is due to potential mode collapse, driven by student's feedback. In these scenarios, the impact of L_{CTC} and L_d during training are diminished, failing to preserve the adaptation achieved in Stage 1 and preventing the mitigation of mode collapse.

5.4 Earlier Approaches

During the development of the MSDA framework, we undertook several experiments to investigate different strategies for combining self-supervised objectives with the Meta PL framework. This section will detail our experimental approaches and analyze the limitations we encountered.

In these approaches we used both Finetuned and M2DS2-trained teachers, and were conducted and evaluated in the *TrueCrime - Education* source-target domain setting. SpecAugment was applied only to the student's input data to prevent the teacher model from generating poor-quality labels.

Self-supervision on both domains

In this approach, the student model is trained using the objective loss in Eq. 5.2, while the teacher model uses:

$$L_T = L_{feedback}(x_s, y_s) + L_s(x_s) + L_s(x_t) \tag{5.5}$$

where $L_{feedback}$ is the student's CTC loss on source domain, and L_s is the self-supervised objective from Eq. 3.38.

The main idea behind this approach was that the teacher's continuous adaptation to the target domain, guided by the self-supervised loss, would produce better pseudo-labels. This, in turn, would help the student model adapt more successfully.

The adaptation results are presented in Table 5.3. The *Before Meta PL* column sdisplays the teacher's Word Error Rate (WER) for both the source and target domains, while the *After Meta PL column* presents the student's WER for the same domains.

Teacher type	**Domains**		**Before Meta PL**		**After Meta PL**	
	source	**target**	**source**	**target**	**source**	**target**
Finetuned	TrueCrime	Education	37.83	48.8	43.01	97
M2DS2	TrueCrime	Education	36.9	45.32	41.6	95.83

Table 5.3. *Adaptation results using self-supervision on both domains during teacher's training*

We observe that the student model fails to adapt at all, as the WER on the target domains is close to 100. We believe this outcome is due to the self-supervised loss Ls, which likely caused the teacher to generate pseudo-labels of poor quality. These pseudo-labels, combined with the resulting feedback from the student, seem to create a vicious training cycle, preventing the student model from converging and adapting

Student self-supervision

In this approach, the student model is trained using the following multi-task objective loss:

$$L_S = L_{CTC}(x_t, y_t^*) + L_s(x_t) \tag{5.6}$$

where L_{CTC} is the CTC objective loss, y_t^* is the teacher-generated pseudo-label and L_s is the self-supervised loss from Eq. 3.38. The teacher model was trained using only the student's feedback loss $L_{feedback}$.

The idea behind this approach was that the student's simultaneous adaptation to the target domain, driven by semi-supervised and self-supervised training, would result in successful overall adaptation. The results are presented in Table 5.4.

Teacher type	**Domains**		**Before Meta PL**		**After Meta PL**	
	source	**target**	**source**	**target**	**source**	**target**
Finetuned	TrueCrime	Education	37.83	48.8	35.2	58
M2DS2	TrueCrime	Education	36.9	45.32	33.4	55.1

Table 5.4. *Adaptation results using self-supervision on target domain during student's training*

We observe that the student model struggles to adapt effectively in this scenario, performing worse in the target domain than the teacher model. We believe this outcome is due to the self-supervised loss L_s, which likely caused the student model to provide a 'misleading' feedback back to the teacher model. This, in turn, may have resulted in the generation of poor-quality pseudo-labels, preventing the semi-supervised training from achieving further adaptation

Chapter 6

Conclusions & Future Work

6.1 Conclusions

Unsupervised Domain Adaptation for Automatic Speech Recognition (ASR) is a significant challenge with important real-world implications. Even with the use of pre-trained models, this issue persists, as the wide variety of speech data continues to complicate effective adaptation to the selected target domain.

In the literature, Unsupervised Domain Adaptation approaches can be classified into three main categories: pseudo-labeling (or teacher-student training), self-supervised methods, and domain adversarial techniques. This work focuses on applying teacher-student training methods, previously used in image recognition, to Automatic Speech Recognition tasks and explores how these methods can be successfully integrated with self-supervised strategies.

In this work, we explore the application of the Meta PL framework [5] within ASR tasks. We find that Meta PL, by itself, is a straightforward and easily implementable method for UDA that achieves significant adaptation results, resulting to reduction of WER metric by 3-5% in all adaptation scenarios. Furthermore, we investigated the integration of self-supervision objectives within the Meta PL framework, aiming to develop a strategy that yields even greater adaptation results in ASR task.

Based on the experiments detailed in Section 5.4, we conclude that for the Meta PL framework to be successfully integrated with self-supervised objectives, it is essential that the student's feedback not be influenced by any additional losses. The student should only be trained on the target domain using the pseudo-labels. Furthermore, we find that the teacher's ability to produce high-quality pseudo-labels is crucial. Therefore, any additional tasks should focus on improving the quality of these pseudo-labels without causing significant changes to the teacher's current state.

Based on these observations, we propose a new adaptation method called Multi-Stage Domain Adaptation (MSDA), which successfully combines techniques from both self-supervision and semi-supervision. Our method achieves significantly better adaptation results compared to the Meta PL and M2DS2 methods, as highlighted in Table 5.2, owing to its two-stage approach. We demonstrate that an already adapted teacher, trained within the Meta PL framework while preserving its initial adaptation, results in the creation of significantly more effective student models, achieving up to a 10-12% reduction in WER

in certain scenarios.

In the cases outlined in Table 5.2 where MSDA fails to achieve any adaptation compared to the fine-tuned model, we identify the poor initial adaptation of the teacher model during the M2DS2 stage as a contributing factor. We believe this limitation arises from the weakly supervised nature of our data and minimal tuning of M2DS2's hyper-parameters, specifically the α and β parameters. However, we are optimistic that with careful hyper-parameter tuning and improved data quality, MSDA has the potential to overcome the limitations imposed by M2DS2's failure to adapt, ultimately delivering superior adaptation performance across all examined scenarios.

6.2 Future Work

Several extensions and variations could be considered for future work, which can be categorized into three main areas. The first category focuses on applying our work to other speech-related tasks. The second area of exploration involves investigating alternative domain adaptation settings . Finally, the third category involves applying our work to various fields within machine learning and deep learning.

The proposed method holds promise for application in various speech-related tasks that experience domain shifts, such as speaker and emotion recognition. These tasks are particularly intriguing because they represent classification problems, in contrast to the sequence-to-sequence nature of Automatic Speech Recognition. By assessing the effectiveness of our method in these domains, we can uncover valuable insights into its strengths and limitations.

This work tackles the challenge of Unsupervised Domain Adaptation within the specific context of single source and single target domains. However, real-world applications frequently demand the implementation of multi-target models that are trained on data spanning from multiple source domains. Consequently, we propose to investigate a multi-domain adaptation framework, where information from various source domains is utilized to effectively adapt to several target domains. This approach will enable us to comprehensively evaluate the method's performance and adaptability across a range of diverse domains, each characterized by distinct data distributions.

Furthermore, we are eager to explore the applications of our method across various research domains within machine learning, including Natural Language Processing and Image Processing. Adapting our approach to incorporate field-specific loss objectives will not only enhance its versatility but also offer significant insights into its effectiveness across different tasks. For instance, in Natural Language Processing, our method could improve performance in tasks such as sentiment analysis or text classification by leveraging contextual embeddings. In Image Processing, the application of our method could facilitate advancements in image classification and object detection, particularly when dealing with images from diverse sources that exhibit different characteristics.

Bibliography

[1] A Vaswani. *Attention is all you need*. *Advances in Neural Information Processing Systems*, 2017.

[2] AlexGraves et al. *Towards end-to-end speech recognition with recurrent neural networks*. *Proc. ICML*, σελίδες 1764–1772. PMLR, 2014.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed και Michael Auli. *wav2vec 2.0: A framework for self-supervised learning of speech representations*. *Proc. NeurIps*, τόμος 33, σελίδες 12449–12460, 2020.

[4] VimalManohar et al. *Kaizen: Continuously improving teacher using exponential moving average for semi-supervised speech recognition*. *Proc. ASRU*, σελίδες 518–525. IEEE, 2021.

[5] HieuPham et al. *Meta pseudo labels*. *Proc. CVPR*, σελίδες 11557–11568. IEEE/CVF, 2021.

[6] ConstantinosKarouzos et al. *UDALM: Unsupervised Domain Adaptation through Language Modeling*. *Proc. NAACL: HLT*, σελίδες 2579–2590, 2021.

[7] Chandran SavithriAnoop et al. *Unsupervised domain adaptation schemes for building ASR in low-resource languages*. *Proc. ASRU*, σελίδες 342–349. IEEE, 2021.

[8] GeorgiosParaskevopoulos et al. *Sample-Efficient Unsupervised Domain Adaptation of Speech Recognition Systems: A Case Study for Modern Greek*. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:286–299, 2024.

[9] D. Yu και L. Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Signals and Communication Technology. Springer London, 2016.

[10] Daniel Jurafsky και James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. Prentice Hall, 3η έκδοση, 2024. Online manuscript released August 20, 2024.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[12] Longfei Li, Yong Zhao, Dongmei Jiang, Yanning Zhang, Fengna Wang, Isabel Gonzalez, Enescu Valentin και Hichem Sahli. *Hybrid Deep Neural Network–Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition*. *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, σελίδες 312–317, 2013.

[13] Alex Graves. *Sequence Transduction with Recurrent Neural Networks*. *ArXiv*, abs/1211.3711, 2012.

[14] Julian Salazar, Davis Liang, Toan Q Nguyen και Katrin Kirchhoff. *Masked language model scoring*. *arXiv preprint arXiv:1910.14659*, 2019.

[15] DongseongHwang et al. *Large-scale asr domain adaptation using self-and semi-supervised learning*. *Proc. ICASSP*, σελίδες 6627–6631. IEEE, 2022.

[16] Dongseong Hwang, Khe Chai Sim, Zhouyuan Huo και Trevor Strohman. *Pseudo Label Is Better Than Human Label*. *Interspeech 2022*, σελίδες 1421–1425, 2022.

[17] Nay San και et al. *Predicting positive transfer for improved low-resource speech recognition using acoustic pseudo-tokens*. *Proc. of the 6th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, σελίδες 100–112. ACL, 2024.

[18] Daniel S.Park et al. *Improved noisy student training for automatic speech recognition*. *Proc. Interspeech*, σελίδες 2817–2821. ISCA, 2020.

[19] Daniel S.Park et al. *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*. *Proc. Interspeech*, σελίδες 2613–2617. ISCA, 2019.

[20] Wei Ning Hsuet al. *Robust wav2vec 2.0: Analyzing Domain Shift in Self-Supervised Pre-Training*. *Proc. Interspeech*, σελίδες 721–725. ISCA, 2021.

[21] Han.Zhu et al. *Boosting cross-domain speech recognition with self-supervision*. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:471–485, 2024.

[22] AlexisConneau et al. *Unsupervised Cross-Lingual Representation Learning for Speech Recognition*. *Proc. Interspeech*, σελίδες 2426–2430. ISCA, 2021.

[23] GeorgiosParaskevopoulos et al. *The Greek podcast corpus: Competitive speech models for low-resourced languages with weakly supervised data*. *Proc. Interspeech*, σελίδες 3969–3973. ISCA, 2024.

[24] Max Bainet al. *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*. *Proc. Interspeech*, σελίδες 4489–4493. ISCA, 2023.

[25] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[26] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Ian Goodfellow, Yoshua Bengio και Aaron Courville. *Convolutional networks*. *Deep learning*, 2016:330–372, 2016.

[28] Diederik P Kingma. *Auto-encoding variational bayes*. *arXiv preprint arXiv:1312.6114*, 2013.

[29] D. Littau και D. Boley. *Clustering Very Large Data Sets with Principal Direction Divisive Partitioning*, σελίδες 99–126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[30] Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

[31] Richard S. Sutton και Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[32] William Uther. *Markov Decision Processes*, σελίδες 642–646. Springer US, Boston, MA, 2010.

[33] Xiaojin Jerry Zhu. *Semi-supervised learning literature survey*. 2005.

[34] Olivier Chapelle, Bernhard Schölkopf και Alexander Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[35] Frank Rosenblatt. *The Perceptron: A Probabilistic Model for Information Storage and Organization (1958)*. 2021.

[36] Nikolay Kyurkchiev και Svetoslav Markov. *Sigmoid functions: some approximation and modelling aspects*. *LAP LAMBERT Academic Publishing, Saarbrucken*, 4, 2015.

[37] Yuanzhi Li και Yang Yuan. *Convergence analysis of two-layer neural networks with relu activation*. *Advances in neural information processing systems*, 30, 2017.

[38] Marius Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu και Nikos Mastorakis. *Multilayer perceptron and neural networks*. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

[39] Usha Ruby και Vamsidhar Yendapalli. *Binary cross entropy with deep learning technique for image classification*. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(10), 2020.

[40] Zhou Wang και Alan C Bovik. *Mean squared error: Love it or leave it? A new look at signal fidelity measures*. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[41] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford και Nando De Freitas. *Learning to learn by gradient descent by gradient descent*. *Advances in neural information processing systems*, 29, 2016.

[42] Shun ichi Amari. *Backpropagation and stochastic gradient descent method*. *Neurocomputing*, 5(4-5):185–196, 1993.

[43] Robin M Schmidt. *Recurrent neural networks (rnns): A gentle introduction and overview*. *arXiv preprint arXiv:1912.05911*, 2019.

[44] Alexander Rehmer και Andreas Kroll. *On the vanishing and exploding gradient problem in Gated Recurrent Units*. *IFAC-PapersOnLine*, 53(2):1243–1248, 2020.

[45] Roger Grosse. *Lecture 15: Exploding and vanishing gradients*. *University of Toronto Computer Science*, 2017.

[46] Alex Graves και Alex Graves. *Long short-term memory*. *Supervised sequence labelling with recurrent neural networks*, σελίδες 37–45, 2012.

[47] Rajib Rana. *Gated recurrent unit (GRU) for emotion classification from noisy speech*. *arXiv preprint arXiv:1612.07778*, 2016.

[48] Tom Dietterich. *Overfitting and undercomputing in machine learning*. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[49] Yingjie Tian και Yuqi Zhang. *A comprehensive survey on regularization strategies in machine learning*. *Information Fusion*, 80:146–166, 2022.

[50] Diego Vidaurre, Concha Bielza και Pedro Larranaga. *A survey of L1 regression*. *International Statistical Review*, 81(3):361–387, 2013.

[51] Andrew Y Ng. *Feature selection, L 1 vs. L 2 regularization, and rotational invariance*. *Proceedings of the twenty-first international conference on Machine learning*, σελίδα 78, 2004.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever και Ruslan Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting*. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[53] Kiran Maharana, Surajit Mondal και Bhushankumar Nemade. *A review: Data preprocessing and data augmentation techniques*. *Global Transitions Proceedings*, 3(1):91–99, 2022.

[54] Lutz Prechelt. *Early stopping-but when?* *Neural Networks: Tricks of the trade*, σελίδες 55–69. Springer, 2002.

[55] Yoshua Bengio, Aaron Courville και Pascal Vincent. *Representation learning: A review and new perspectives*. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[56] Ting Chen, Calvin Luo και Lala Li. *Intriguing properties of contrastive losses*. *Advances in Neural Information Processing Systems*, 34:11834–11845, 2021.

[57] PeterHiggins et al. *Baum-Welch Algorithm*. 2009.

[58] L.R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[59] G.D. Forney. *The viterbi algorithm*. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[60] Eric Jang, Shixiang Gu και Ben Poole. *CATEGORICAL REPARAMETERIZATION WITH GUMBEL-SOFTMAX*. *stat*, 1050:5, 2017.

[61] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong και Quoc Le. *Unsupervised data augmentation for consistency training*. *Advances in neural information processing systems*, 33:6256–6268, 2020.

[62] Henry Scudder. *Probability of error of some adaptive pattern-recognition machines*. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[63] Ellen Riloff και Janyce Wiebe. *Learning extraction patterns for subjective expressions*. *Proc. EMNLP*, σελίδες 105–112, 2003.

[64] DongseongHwang et al. *Comparison of soft and hard target rnn-t distillation for large-scale asr*. *ICASSP 2023-2023*, σελίδες 1–5. IEEE, 2023.

[65] SuchinGururangan et al. *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. *Proc. 58th Annual Meeting of ACL*, σελίδες 8342–8360, 2020.

[66] Andrea Cossu, Antonio Carta, Lucia Passaro, Vincenzo Lomonaco, Tinne Tuytelaars και Davide Bacciu. *Continual pre-training mitigates forgetting in language and vision*. *Neural Networks*, 179:106492, 2024.

[67] Zih Ching Chen, Chin Lun Fu, Chih Ying Liu, Shang Wen Daniel Li και Hung yi Lee. *Exploring efficient-tuning methods in self-supervised speech models*. *2022 IEEE spoken language technology workshop (SLT)*, σελίδες 1120–1127. IEEE, 2023.

[68] Jounghee Kim και Pilsung Kang. *K-wav2vec 2.0: Automatic speech recognition based on joint decoding of graphemes and syllables*. *arXiv preprint arXiv:2110.05172*, 2021.

[69] Karol Nowakowski, Michal Ptaszynski, Kyoko Murasaki και Jagna Nieuważny. *Adapting multilingual speech representation model for a new, underresourced language through multilingual fine-tuning and continued pretraining*. *Information Processing & Management*, 60(2):103148, 2023.

[70] Jacob Devlin Ming Wei Chang Kenton και Lee Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. *Proceedings of naacL-HLT*, τόμος 1, σελίδα 2. Minneapolis, Minnesota, 2019.

[71] YaroslavGanin et al. *Domain-adversarial training of neural networks*. *Journal of machine learning research*, 17(59):1–35, 2016.

List of Abbreviations

ASR	Automatic Speech Recognition
UDA	Unsupervised Domain Adaptation
PL	Pseudo Labels
MSDA	Multi-Stage Domain Adaptation
ML	Machine Learning
AI	Artificial Intelligence
RL	Reinforcement Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Loss
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
MFCC	Mel-Frequency Cepstral Coefficient
DCT	Discrete Cosine Transform
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
CTC	Connectionist Temporal Classification
CEM	Confidence Estimation Module
NST	Noisy Student Training
CPT	Continual Pre-Training
MLM	Masked Language Modeling
M2DS2	Mixed Multi-Domain Self-Supervision
DAT	Domain Adversarial Training
GRL	Gradient Reversal Layer
DSN	Domain Separation Network
ATDS	Acoustic and Textual Domain Similarity
WER	Word Error Rate