BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

PARAMETER EFFICIENT MODELS FOR DOMAIN SPECIFIC SPEECH AND LANGUAGE RECOGNITION

PARAMETRICKY EFEKTIVNÍ MODELY PRO AUTOMATICKÉ ROZPOZNÁVÁNÍ ŘEČI A JAZYKA

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR — **AMRUTHA PRASAD**
AUTOR PRÁCE

SUPERVISOR — **PETR MOTLÍČEK**
ŠKOLITEL

BRNO 2026

Abstract

Large pretrained acoustic models have become foundational to modern speech technologies, achieving state-of-the-art performance across a variety of tasks. However, their massive parameter counts pose significant challenges for deployment in resource-constrained and safety-critical environments. This doctoral thesis addresses these limitations by proposing and evaluating model compression and parameter-efficient methodologies for Automatic Speech Recognition (ASR) and Language Identification (LID). The research primarily focuses on the Air Traffic Control (ATC) domain—a highly demanding scenario characterized by reduced linguistic context, elevated background noise, and high speaker diversity, making it an ideal testbed for evaluating robust speech processing under real-world constraints.

To mitigate computational costs without sacrificing performance, the thesis first targets task-specific back-end discriminators built on top of the XLS-R model. For the LID task, the traditional Time-Delay Neural Network (TDNN) backend discriminator is replaced with Factorized Time-Delay Neural Networks (TDNN-F), achieving a 30–50% parameter reduction while maintaining full discriminative capacity. Furthermore, the thesis explores the use of Normalizing Flows to model complex embedding distributions. This effectively circumvents the traditional Gaussian assumptions of Probabilistic Linear Discriminant Analysis (PLDA) scoring and removes the need for length normalization. For two of the three datasets, the Flow model consistently yielded higher accuracy than the PLDA scoring method.

The thesis then investigates structural compression within the foundational models themselves. Low-rank matrix factorization via Singular Value Decomposition (SVD) is applied to the feed-forward layers of transformer architectures (XLS-R and Whisper), both independently and in conjunction with Low-Rank Adaptation (LoRA). Unlike standard parameter-efficient methods that constrain the parameter budget only during fine-tuning, this factorization reduces model size during both training and inference. The effectiveness of the proposed techniques is tested on three datasets for LID and on two datasets for ASR. For the XLS-R model, parameters are reduced by 23–50% with minimal to no performance degradation. For Whisper, a 28% parameter reduction yields a 1.8% absolute improvement in Word Error Rate (WER), while a 50% reduction introduces no degradation.

Abstrakt

Rozsáhlé předtrénované akustické modely se staly základem moderních řečových technologií a dosahují špičkového výkonu v celé řadě úloh. Jejich obrovský počet parametrů však představuje značné výzvy pro nasazení v prostředích s omezenými zdroji a kritickými pro bezpečnost. Tato disertační práce se zabývá návrhem a evaluací redukce modelů a parametricky efektivních metodologií pro automatické rozpoznávání řeči (ASR) a identifikaci jazyka (LID). Výzkum se primárně zaměřuje na oblast řízení letového provozu (ATC), t.j., vysoce náročný scénář charakterizovaný omezeným jazykovým kontextem, silným šumem pozadí a vysokou diverzitou mluvčích, což z něj činí ideální testovací prostředí pro hodnocení robustního zpracování řeči v reálných podmínkách.

Abychom zmírnili výpočetní náklady bez snížení výkonu, práce se nejprve zaměřuje na specifické diskriminátory, které jsou implementovány v modelech typu XLS-R. Pro úlohu LID je tradiční diskriminátor typu TDNN nahrazen faktorizovanou neuronovou sítí (TDNN-F), čímž se dosahuje 30–50% redukce parametrů při zachování plné diskriminační kapacity. Dále práce zkoumá použití "Normalizing Flows" metody k modelování komplexních distribucí. To efektivně obchází tradiční Gaussovské předpoklady pravděpodobnostní lineární

diskriminační analýzy (PLDA) a odstraňuje potřebu normalizace délky audio nahrávek. U dvou ze tří datových sad, "Flow" model konzistentně poskytoval vyšší přesnost než metoda PLDA.

Práce poté zkoumá strukturální kompresi samotných základních modelů. Faktorizace matic pomocí SVD dekompozice je aplikována na dopředné vrstvy neuronových sítí (v modelech XLS-R a Whisper), a to jak samostatně, tak ve spojení s adaptací LoRA. Na rozdíl od standardních metod, které omezují množství parametrů pouze během adaptace modelů, tato faktorizace zmenšuje velikost modelu jak během trénování, tak i inference. Účinnost navržených technik je testována na třech datových sadách pro detekci jazyka (LID) a dvou sadách pro rozpoznávání řeči (ASR). U modelu XLS-R je množství parametrů sníženo o 23–50 % s minimálním snížením přesnosti modelů. U Whisper modelů vede 28% redukce parametrů ke snížení WER o 1,8%, zatímco 50% redukce nemá žádný vliv na WER.

Keywords

Automatic speech recognition, Language recognition, Low-rank matrix factorization, Parameter reduction

Klíčová slova

Automatické rozpoznávání řeči, Rozpoznání jazyka, Faktorizace matic, Redukce množství parametrů

Reference

PRASAD, Amrutha. *Parameter efficient models for domain specific speech and language recognition.* Brno, 2026. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Petr Motlíček

Parameter efficient models for domain specific speech and language recognition

Declaration

I hereby declare that this PhD thesis was prepared as an original work by the author under the supervision of Dr. Petr Motlíček. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis. All the information derived from the published and unpublished work of others has been acknowledged in the text and included in the list of references. A Large Language Model (LLM) was used to refine the grammar, spelling, and overall clarity of this thesis.

.....
Amrutha Prasad
July 10, 2026

Acknowledgements

I would like to express my deepest gratitude to my supervisor, **Dr. Petr Motlíček**, for his exceptional guidance, encouragement, and unwavering support throughout the course of my Masters and Ph.D. His expertise, insightful suggestions, and patience have been invaluable in shaping both this thesis and my growth as a researcher. I am also sincerely grateful to **Prof. Jan "Honza" Černocký** for his constructive feedback, continuous support, and for creating an environment in which research and collaboration can thrive.

My heartfelt thanks go to my husband, **Dr. Srikanth Madikeri**, whose constant support, understanding, and belief in me have carried me through the most challenging stages of this journey. His help—both in my research and in managing responsibilities at home—has been fundamental to the completion of this work. I am equally and deeply grateful to my family, who encouraged me from the very beginning to pursue advanced studies, first my Master's and then my PhD, and who have always believed in my abilities.

I would also like to thank my colleagues at **Idiap Research Institute**—**Iuliia Nigmatulina**, **Juan-Pablo Gomez**, **Driss Khalil**, **Shashi Kumar**, **Pradeep Rangappa**, and **Andres Carofilis**. Our discussions, brainstorming sessions, and shared challenges have significantly enriched my research experience. Their support, expertise, and friendship have made my time at Idiap both productive and enjoyable.

My sincere thanks extend to my colleagues at **Brno University of Technology (BUT)**—**Olda Plchot**, **Lukáš Burget**, **Karel Veselý**, and **Anna Silnova**—for their help in answering my questions regarding models, experimental setups, and technical issues. I am also thankful to **Murali Karthick Baskar**, **Honza Švec**, **Martin Kocur**, and **Santosh Kesiraju** for their assistance with administrative matters, translations into Czech, and for the many helpful and supportive work-related and non-work-related conversations.

I wish to acknowledge the invaluable help of the administrative staff—**Renata Kohlová**, **Sylva Sadovská**, **Svatava Nunvářová**, **Patricie Břečková**, **Sylvie Meier**, **Laura Coppey** and others—who assisted with the numerous organizational and administrative

tasks that form the backbone of academic life. Their efficiency, kindness, and willingness to help made many difficult processes smooth and manageable.

Finally, I extend my appreciation to all friends, colleagues, and collaborators who, in one way or another, contributed to this thesis or supported me along the way. I am profoundly grateful for the community around me, without whom this work would not have been possible.

Contents

List of Figures

List of Tables

List of Acronyms

ABSR Assistance-Based Speech Recognition
AM Acoustic Model
ASR Automatic Speech Recognition
ASP Attentive Statistics Pooling
ATC Air Traffic Control
ATCO Air Traffic Controller
ATM Air Traffic Management
BPTT Backpropagation Through Time
BRNN Bidirectional RNN
CNN Convolutional Neural Network
CTC Connectionist Temporal Classification
DCT Discrete Cosine Transform
DNN-HMM Deep Neural Network- Hidden Markov Model
DTW Dynamic Time Warping
E2E end-to-end
ECAPA-TDNN Emphasized Channel Attention, Propagation and Aggregation Time-Delay Neural Network
EM Expectation-Maximization
FAIR Facebook AI Research
FC Fully Connected
FFT Fast Fourier Transform
FNN Feed-forward Neural Network
FST Finite-State Transducer
GLC Gaussian Linear Classifier
GMM Gaussian Mixture Model
GRU Gated Recurrent Unit
HAAWAII Highly Automated Air Traffic Controller Workstations with Artificial Intelligence Integration (Project)
HMM Hidden Markov Model
ICAO International Civil Aviation Organization
IHM Individual Headset Microphone
LDA Linear Discriminant Analysis
LFMMI Lattice-Free Maximum Mutual Information

LID Language Identification
LLM Large Language Model
LM Language Model
LoRA Low-rank adaptation
LSTM Long Short-Term Memory
LVCSR Large Vocabulary Continuous Speech Recognition
MALORCA MAchine Learning Of speech Recognition models for Controller Assistance (Project)
MAM Masked Acoustic Modeling
MAP Maximum A Posteriori
MFCC Mel-Frequency Cepstral Coefficient
MHA Multi-Head self-Attention
ML Maximum Likelihood
MLS Multilingual LibriSpeech
MMI Maximum Mutual Information
MLP Multilayer Perceptron
NLP Natural Language Processing
NER Named Entity Recognition
PEFT Parameter Efficient FineTuning
PLDA Probabilistic Linear Discriminant Analysis
PTQ Post-Training Quantization
QAT Quantization-Aware Training
ReLU Rectified Linear Unit
RNN Recurrent Neural Network
ResNet Residual Network
SE Squeeze-and-Excitation
SGMM Subspace Gaussian Mixture Moddel
SGD Stochastic Gradient Descent
SL Sequence Labeling
SRC Speaker Role Classification
SVD Singular Value Decomposition
SVM Support Vector Machine
TDNN Time-Delay Neural Network
TDNN-F Factorized Time-Delay Neural Network
TMA Terminal Manoeuvring Area
UBM Universal Background Model
VAD Voice Activity Detection
VHF Very High Frequency
W2V2 wav2vec 2.0
WER Word Error Rate
WFST Weighted Finite-State Transducer

XLSR Cross-Lingual Speech Representation

Chapter 1

Introduction

Automatic Speech Recognition (ASR) and Language Identification (LID) are two fundamental tasks in the field of speech processing. ASR systems aim to convert spoken language into written text, enabling machines to "understand" and respond to human speech. LID systems, on the other hand, determine the language of a spoken utterance, which is critical in multilingual environments and global communication systems. Together, these technologies support a wide range of applications, from real-time translation and virtual assistants to surveillance and transcription systems.

Over the past decade, significant advances have been made in both ASR and LID, largely driven by the introduction of deep learning methods such as Convolutional Neural Networks (CNNs) [LeCun and Bengio, 1995], Recurrent Neural Networks (RNNs) [Mikolov et al., 2010, Graves et al., 2013], and transformer-based architectures like Wav2Vec 2.0 [Baevski et al., 2020, Schneider et al., 2019] and Whisper [Radford et al., 2023]. These models, trained on increasingly large and diverse datasets, have achieved impressive results, particularly in high-resource settings such as English. They demonstrate robustness against speaker variation, background noise, and domain shifts, and have reached near-human levels of performance in some cases.

ASR and LID technologies are deployed across numerous domains. In call centers, they enable automatic transcription and routing of calls based on detected language and extracted customer intent. In criminal investigations, LID helps to monitor multilingual communications, while ASR helps to transcribe and analyze surveillance audio. In medical transcription, precise ASR output is critical to avoid clinical misinterpretations. Similarly, in legal proceedings, the accuracy of the transcription is essential due to the formal and evidentiary nature of the content. In live broadcast captioning, both latency and accuracy are vital to ensure accessibility and audience comprehension.

A particularly demanding but low-resource domain is Air Traffic Control (ATC)[1] communication where there are huge collections of audio but not many standardized annotated corpora. It is a vital component of the aviation system and is responsible for the safe, orderly, and efficient flow of air traffic in both controlled and uncontrolled airspace. One of the central tasks in ATC is the continuous verbal communication between Air Traffic Controllers (ATCOs) and pilots. This communication is typically conducted over Very High Frequency (VHF) radio channels, using the standard phraseology prescribed by the International Civil Aviation Organization (ICAO). However, in practice, these exchanges are subject to significant variability: accents, speech rates, informal deviations from standard

[1] https://en.wikipedia.org/wiki/Air_traffic_control

phraseology, background noise, and radio distortions add complexity to understanding and transcribing these interactions accurately [Karlsson, 1989].

With the growth of air traffic and the increasing demands placed on ATCOs, there is a strong drive within the aviation industry and research communities to incorporate more automation into ATC operations. Speech and language technologies, particularly ASR, offer promising tools to help in these efforts by transcribing and interpreting ATC communications in real time. Such automatic processing of voice communication can unlock a wide range of capabilities, from real-time safety monitoring and error detection to long-term analytics and workload estimation [Helmke et al., 2016, Helmke et al., 2017].

However, building reliable ASR systems for the ATC domain remains a complex task. Unlike general-purpose ASR, which is trained on diverse and typically high-quality speech data, ATC speech is noisy, specialized, and often linguistically constrained [Oualil et al., 2015, Shore et al., 2012]. Furthermore, audio collected from live ATC operations includes multiple speakers, overlapping conversations, and multilingual content, requiring a series of pre-processing steps to isolate and prepare relevant data for recognition. As a result, complete ATC speech processing systems must integrate several components— LID, Voice Activity Detection (VAD), and Speaker Role Classification (SRC)—before ASR can be applied effectively.

The significant part of the work on this thesis was situated within the context of the *Highly Automated Air Traffic Controller Workstations with Artificial Intelligence Integration (HAAWAII)*[2] project, funded by the European Commission under the Horizon 2020 program and by armasuisse[3] Science and Technology. The HAAWAII project's objective was to develop a robust and adaptable ASR-based solution tailored to operational ATC environments, focusing specifically on the London Terminal Manoeuvring Area (TMA) and the Icelandic en-route airspace. Using large-scale data sets consisting of real-world ATC communications and corresponding radar data, HAAWAII aimed to enhance Air Traffic Management (ATM) safety and reduce controller workload through intelligent automation. Among the project's proof-of-concept applications were *callsign highlighting*, enabling rapid visual identification of aircraft identifiers in transcripts, and *read-back error detection*, identifying mismatches between issued clearances and pilot responses.

In addition to these practical goals, the armasuisse project emphasized the need for efficient and scalable speech models. State-of-the-art ASR and LID systems have become increasingly complex, often comprising hundreds of millions of parameters and requiring significant computational resources. Such models are impractical for real-time deployment in ATC settings, where latency, reliability, and resource constraints are critical factors. Therefore, there is a growing need to investigate methods for *model compression and parameter reduction* that can provide comparable performance with lower computational overhead.

This thesis contributes to this line of research by exploring how parameter reduction techniques perform on two different but related tasks in the ATC domain: *language identification* (a classification problem) and *automatic speech recognition* (a sequence-to-sequence problem). These tasks represent fundamentally different problem types, detection and recognition, but both are essential in the ATC domain.

[2] https://www.haawaii.de/wp/
[3] https://www.ar.admin.ch/en

1.1 Motivation

Although the application of speech and language technologies to ATC offers significant benefits, the domain presents a range of challenges that are not encountered in typical ASR scenarios. ATC speech data are often noisy, highly domain-specific, and recorded in varying acoustic environments. In many cases, data are collected automatically from live radio transmissions, which include communication in multiple languages, especially in regions where English is not the native language. However, English remains the mandated language for international aviation communication. As a result, *LID* becomes a crucial first step in the preprocessing pipeline, ensuring that only English-language segments are selected for subsequent speech recognition and analysis.

Following LID, other preprocessing components are necessary to effectively structure the data. *VAD* is used to filter out non-speech segments, and *SRC* distinguishes between controller and pilot utterances, an important distinction for understanding dialog context, detecting errors, and structuring transcripts for downstream applications such as read-back error detection and callsign highlighting.

A further complication lies in the nature of modern speech models themselves. The pre-trained transformer models used for ASR and LID tasks are often very large, comprising hundreds of millions of parameters. These models can be expensive to run in real-time ATC environments, where computational resources are limited and response latency must be minimal. To address this issue, this thesis explores *parameter reduction techniques* aimed at compressing the size of the model while preserving performance. The effectiveness of these techniques is evaluated on two different but essential tasks in the ATC domain: a classification problem, namely LID and a sequence-to-sequence recognition problem, ASR. These tasks represent fundamentally different challenges: detection versus recognition, but both are critical for robust ATC communication processing.

By investigating parameter reduction across both tasks, this thesis contributes toward building efficient, scalable, and deployable speech systems suitable for real-world ATC applications. This aligns with the broader goals of the HAAWAII project[4] and address operational requirements in the development of next-generation ATC technologies.

1.2 Structure of the Thesis

This thesis is organized into six chapters. Following this introduction and motivation, Chapter 2 reviews the related work in ASR and LID within the ATC domain, alongside a description of the datasets used throughout this research. Chapter 3 presents baseline models and methods against which the proposed techniques are evaluated. Chapter 4 introduces the concept of orthonormal constraints applied to backend classifiers, detailing their theoretical motivation and practical implementation. Chapter 5 focuses on the parameter reduction techniques applied to large pre-trained models, demonstrating their impact on both classification (LID) and sequence-to-sequence (ASR) tasks. Finally, Chapter 6 concludes the thesis by summarizing the main findings, discussing their implications for ATC speech technology, and outlines directions for future work.

[4]The project ended in January 2023, but these goals are still relevant.

1.3 Contributions

The main contributions of this thesis are summarized as follows:

- **Application of Orthonormal Constraints in Backend Discriminators:** This work investigates the use of orthonormal constraints to improve the performance of backend discriminators, particularly in LID tasks within the ATC domain. By imposing these constraints, the classifiers demonstrate improved generalization and robustness, as presented in Chapter 4.
- **Parameter Reduction Techniques for Pre-trained Models:** This thesis investigates into and evaluates parameter reduction methods designed to compress large pre-trained speech models while preserving their classification accuracy. The effectiveness of these techniques is validated on two distinct tasks, sequence-to-sequence recognition (ASR) and classification (LID), highlighting their applicability across different model architectures and use cases. These results are detailed in Chapter 5.
- **Analysis of Normalizing Flows as Backend Classifiers:** This thesis also explores the application of normalizing flow models as a scoring method for LID. This analysis provides insights into their potential advantages and limitations compared to the models that assume the data is Gaussian distribution.

1.4 List of Publications

1.4.1 Publications used in the thesis

- **A. Prasad**, A. Carofilis, G. Vanderreydt, D. Khalil, S. Madikeri, P. Motlicek, C. Schuepbach. *"Fine-Tuning Self-Supervised Models for Language Identification Using Orthonormal Constraint,"* 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Korea, Republic of, 2024, pp. 11921-11925, doi: 10.1109/ICASSP48485.2024.10446751.
- **A. Prasad**, S. Madikeri, D. Khalil, P. Motlicek, C. Schuepbach. (2024) *"Speech and Language Recognition with Low-rank Adaptation of Pretrained Models."* Proc. Interspeech 2024, 2825-2829, doi: 10.21437/Interspeech.2024-2187
- A. Espuna, **A. Prasad**, P. Motlicek, S. Madikeri, C. Schuepbach. *"Normalizing Flows for Speaker and Language Recognition Backend."* Proc. The Speaker and Language Recognition Workshop (Odyssey 2024), 74-80, doi: 10.21437/odyssey.2024-11.

1.4.2 All Other Publications

Journals

- J. Zuluaga-Gomez, **A. Prasad**, I. Nigmatulina, P. Motlicek, M. Kleinert. *"A Virtual Simulation-Pilot Agent for Training of Air Traffic Controllers."* Aerospace, 10(5), 490. https://doi.org/10.3390/aerospace10050490
- D. Khalil, **A. Prasad**, P. Motlicek, J. Zuluaga-Gomez, I. Nigmatulina, S. Madikeri, C. Schuepbach. (2023). *"An automatic speaker clustering pipeline for the air traffic communication domain."* Aerospace, 10(10), 876.

- J. Zuluaga-Gomez, I. Nigmatulina, **A. Prasad**, P. Motlicek, D. Khalil, S. Madikeri, A. Tart, I. Szoke, V. Lenders, M. Rigault, K. Choukri. *"Lessons learned in transcribing 5000 h of air traffic control communications for robust automatic speech understanding."* Aerospace, 10(10), 898.
- J. Zuluaga-Gomez, K. Veselý, A. Blatt, P. Motlicek, D. Klakow, A. Tart, I. Szöke, **A. Prasad**, S. Sarfjoo, P. Kolčárek, M. Kocour, H. Černocký, C. Cevenini, K. Choukri, M. Rigault, F. Landis. "*Automatic call sign detection: Matching air surveillance data with air traffic spoken communications.*" In Proceedings (Vol. 59, No. 1, p. 14). MDPI. 2020.

Conferences

1. M. Bhattacharjee, I. Nigmatulina, **A. Prasad**, P. Rangappa, S. Madikeri, P. Motlicek, M. Kleinert. "*Contextual Biasing Methods for Improving Rare Word Detection in Automatic Speech Recognition.*" In 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 12652-12656). IEEE.
2. G. Vanderreydt, **A. Prasad**, D. Khalil, S. Madikeri, K. Demuynck, P. Motlicek. (2023, December). "*Parameter-efficient tuning with adaptive bottlenecks for automatic speech recognition.*" In 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) (pp. 1-7). IEEE.
3. J. Zuluaga-Gomez, S. S. Sarfjoo, **A. Prasad**, I. Nigmatulina, P. Motlicek, K. Ondrej, H. Helmke. "*Bertraffic: BERT-based joint speaker role and speaker change detection for air traffic control communications.*" In 2022 IEEE Spoken Language Technology Workshop (SLT) (pp. 633-640). IEEE.
4. J. Zuluaga-Gomez, **A. Prasad**, I. Nigmatulina, S. S. Sarfjoo, P. Motlicek, M. Kleinert, Q. Zhan. "*How does pre-trained wav2vec 2.0 perform on domain-shifted asr? an extensive benchmark on air traffic control communications.*" In 2022 IEEE Spoken Language Technology Workshop (SLT) (pp. 205-212). IEEE.
5. I. Nigmatulina, J. Zuluaga-Gomez, **A. Prasad**, S. S. Sarfjoo, P. Motlicek. (2022, May). "*A two-step approach to leverage contextual data: Speech recognition in air-traffic communications.*" In 2022 IEEE international conference on acoustics, speech and signal processing (ICASSP) (pp. 6282-6286). IEEE.
6. J. Zuluaga-Gomez, I. Nigmatulina, **A. Prasad**, P. Motlicek, K. Veselý, M. Kocour, I. Szöke. (2021) "*Contextual Semi-Supervised Learning: An Approach to Leverage Air-Surveillance and Untranscribed ATC Data in ASR Systems.*" Proc. Interspeech 2021, 3296-3300, doi: 10.21437/Interspeech.2021-1373

Air traffic conferences, seminars, workshops and project events

1. P. Motlicek, S. Kumar, D. Khalil, **A. Prasad**, Christof, Schüpbach. "*Leveraging Untranscribed Data for End-to-End Speech and Callsign Recognition in Air-Traffic Communication.*" 15th SESAR Innovation Days 2025, SIDS 2025.
2. P. Motlicek, **A. Prasad**, I. Nigmatulina, H. Helmke, O. Ohneiser, M. Kleinert. "*Automatic speech analysis framework for ATC communication in HAAWAII.*" 13th SESAR Innovation Days 2023, SIDS 2023.

3. **A. Prasad**, J. Zuluaga-Gomez, P. Motlicek, S. S. Sarfjoo, I. Nigmatulina, K. Vesely. "*Speech and Natural Language Processing Technologies for Pseudo-Pilot Simulator.*" 12th SESAR Innovation Days 2022, SIDS 2022.

4. **A. Prasad**, J. Zuluaga-Gomez, P. Motlicek, S. S. Sarfjoo, I. Nigmatulina, O. Ohneiser, H. Helmke. "*Grammar Based Speaker Role Identification for Air Traffic Control Speech Recognition*". 12th SESAR Innovation Days 2022, SIDS 2022.

5. H. Helmke, M. Kleinert, N. Ahrenhold, H. Ehr, T. Mühlhausen, O. Ohneiser, P. Motlicek, **A. Prasad**, J. Zuluaga-Gomez, L. Klamert, J. Dokic, E. Pinska Chauvin. "*Automatic Speech Recognition and Understanding for Radar Label Maintenance Support Increases Safety and Reduces Air Traffic Controllers' Workload.*" In Fifteenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2023).

6. M. Kleinert, H. Helmke, S. Shetty, O. Ohneiser, H. Ehr, **A. Prasad**, P. Motlicek, J. Harfmann. "*Automated interpretation of air traffic control communication: The journey from spoken words to a deeper understanding of the meaning.*" In 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC) (pp. 1-9). IEEE.

7. M. Kocour, V. Veselý, I. Szöke, S. Kesiraju, J. Zuluaga-Gomez, A. Blatt, **A. Prasad**, I. Nigmatulina, P. Motlíček, D. Klakow, A. Tart, H. Atassi, P. Kolčárek, H. Černocký, C. Cevenini, K. Choukri, M. Rigault, F. Landis, S. S. Sarfjoo, C. Salamin. "*Automatic processing pipeline for collecting and annotating air-traffic voice communication data.*" Engineering Proceedings, 13(1), 8.

8. H. Helmke, S. Shetty, M. Kleinert, O. Ohneiser, H. Ehr, **A. Prasad**, P. Motlicek, A. Cerna, C. Windisch. "*Measuring Speech Recognition And Understanding Performance in Air Traffic Control Domain Beyond Word Error Rates.*" Proceedings of the 11th SESAR Innovation Days, Virtual, 7-9.

9. H. Helmke, M. Kleinert, S. Shetty, O. Ohneiser, H. Ehr, H. Arilíusson, T. S. Simiganoschi, **A. Prasad**, P. Motlicek, K. Veselý, K. Ondřej, P. Smrz, J. Harfmann, C. Windisch. "*Readback error detection by automatic speech recognition to increase ATM safety.*" In Proceedings of the fourteenth USA/Europe air traffic management research and development seminar (ATM2021), virtual event (pp. 20-23).

Chapter 2

Background and Datasets

This chapter provides a background on the foundational components relevant to the thesis. It begins with an overview of ASR, detailing the evolution of features and modeling techniques, ranging from early approaches like Dynamic Time Warping (DTW) and Hidden Markov Models (HMM), to hybrid Deep Neural Network-HMM (DNN-HMM) systems, and more recent end-to-end (E2E) architectures leveraging Transformer models. Evaluation metrics and the datasets used for training and evaluation in this thesis are also discussed. The chapter then transitions to LID, outlining various embedding systems and scoring techniques, including cosine distance, Probabilistic Linear Discriminant Analysis (PLDA), and Gaussian Linear Classifier (GLC), along with the corresponding evaluation metrics and datasets employed for both training and testing. Lastly, it provides details on the pretrained models utilized in this work, specifically XLSR, XLS-R, and Whisper, highlighting their relevance and role in the experiments and analyses that follow.

2.1 Automatic Speech Recognition

ASR converts spoken language into written text, and its performance has improved considerably in the past few decades. Figure 2.1, shows a conventional architecture for ASR, which is still being used today and is referred to as a hybrid ASR. The ASR consists of the following blocks:

- *Feature extraction*: Traditionally, this involved the extraction of hand-crafted acoustic features such as Mel-Frequency Cepstral Coefficients (MFCCs) [Davis and Mermelstein, 1980a, Davis and Mermelstein, 1980b] or log-Mel filterbank energies [Hermansky, 1990], which capture the spectral characteristics of speech in a perceptually motivated way. More recently, deep learning models have been applied directly to raw audio waveforms, bypassing hand-crafted feature extraction, and allowing the model to learn optimal representations automatically.

- *Acoustic Model (AM)*: represents the relationship between a speech signal and the phonemes, or other linguistic units that make up the speech. It is trained with a dataset of speech recordings with their corresponding text (also referred to as transcripts).

- *Language Model (LM)*: represents a probability distribution on sequences of words. The LM provides context to distinguish between words and phrases that sound similar. It is trained on a large corpus of text data.

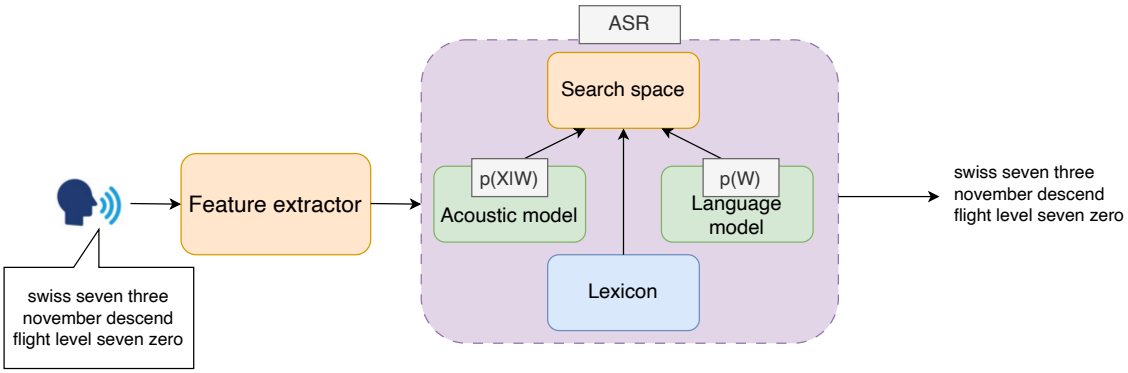

Figure 2.1: Overview of hybrid ASR architecture, including feature extractor.

- *Lexicon*: A list of words and their pronunciations that will be recognized by the ASR.
- *Decoder*: Constructs a search graph usually as a Finite-State Transducer (FST) [Mohri et al., 2002, Mohri et al., 2008, Riley et al., 2009, Povey et al., 2011] by replacing the word tokens of the LM with the corresponding phone sequences provided by the lexicon. The decoder then outputs the most probable word hypothesis by combining the acoustic model and language model scores.

The early systems were limited to isolated word recognition with small vocabularies and relied on simple techniques such as DTW [Sakoe and Chiba, 1978, Rabiner and Juang, 1993]. As research progressed, statistical methods became dominant, particularly Gaussian Mixture Models (GMMs) combined with HMMs, which were effective in modeling the variability and temporal dynamics of speech [Rabiner, 1989, Jelinek, 1997]. These GMM-HMM systems formed the backbone of ASR for many years [Hermansky et al., 2000, Young et al., 2002].

With the integration of DNNs into acoustic modeling, a significant milestone was achieved that led to substantial accuracy improvements [Hinton, Geoffrey et al., 2012, Bourlard and Morgan, 1993]. The DNN-HMM hybrid systems gradually replaced the GMM-HMM architectures on many ASR tasks. More recently, the development of E2E deep learning approaches, especially those based on transformer architectures, has further advanced the field. Models leveraging large-scale self-supervised [Baevski et al., 2020] or weakly-supervised [Radford et al., 2023] learning and implementing attention mechanisms achieve state-of-the-art performance on various speech recognition benchmarks, even in noisy and low-resource settings.

2.1.1 Acoustic Feature Representations in ASR

Acoustic feature extraction transforms raw audio waveforms into representations that capture the essential properties of speech. Commonly used features include MFCCs and filterbank energies coefficients. An overview of the comparison of these features is shown in Figure 2.2.

a. Mel-Frequency Cepstral Coefficients

MFCCs are one of the most widely used features in ASR due to their effectiveness in modeling human auditory perception [Davis and Mermelstein, 1980a]. The MFCC computation

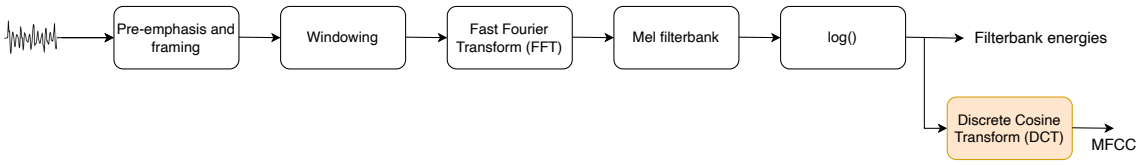

Figure 2.2: Overview of different types of features used in ASR. Applying DCT at the last step generates the MFCC features.

involves: (i) Pre-emphasis and framing of the raw waveform, (ii) Windowing, (iii) Fast Fourier Transform (FFT) to convert the signal to the frequency domain, (iv) Application of a Mel-scale filterbank, (v) Logarithmic compression of the filterbank energies [Oppenheim and Schafer, 1968, Bogert et al., 1963, Rabiner and Schafer, 1978], and (vi) Discrete Cosine Transform (DCT) converting the log-Mel spectrum into a set of cepstral coefficients.

The final MFCC features are obtained by:

$$c_n = \sum_{k=1}^{K} \log(E_k) \cdot \cos\left[n\left(k - \frac{1}{2}\right)\frac{\pi}{K}\right] \quad n = 1, 2, ..., N, \tag{2.1}$$

where E_k is the energy of the k-th Mel filter, K is the number of filters (typically $K = 23$ or $K = 40$), and N is the total number of coefficients to be extracted.

b. Filterbank Energies

E2E deep learning models avoid the DCT step in MFCCs The filterbank energy vector is defined as:

$$\mathbf{x} = [\log(E_1), \log(E_2), ..., \log(E_K)]^T. \tag{2.2}$$

These features retain more localized spectral information than MFCCs, which is beneficial for data-driven models. These features are used in our experiments with the Whisper model.

2.1.2 Evolution of Acoustic Models

ASR has undergone significant architectural transformations over the past decades, with advances in our computational capacity, data availability, and machine learning algorithms. The evolution of ASR architectures can be broadly categorized into several key architectures of acoustic models:

a. Gaussian Mixture Model-Hidden Markov Model (GMM-HMM)

The HMMs provided a probabilistic framework for modeling the temporal variability of speech. An HMM is a statistical model that assumes a sequence of hidden states $\mathbf{Q} = \{s_t\}_{t=1}^{T}$ generates the observed acoustic features $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$. The model is defined by:

$$P(\mathbf{x}, \mathbf{s}) = P(s_1) \prod_{t=2}^{T} P(s_t \mid s_{t-1}) \prod_{t=1}^{T} p(\mathbf{x}_t \mid s_t), \tag{2.3}$$

where $P(s_t \mid s_{t-1})$ are the transition probabilities and $p(\mathbf{x}_t \mid s_t)$ are the emission probabilities. In ASR, phonemes (either context-independent or context dependent) were typically

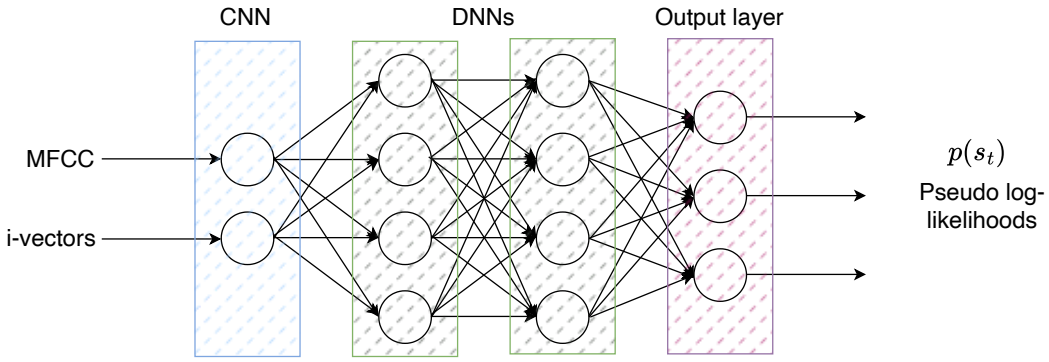

Figure 2.3: Overview of a DNN Acoustic model in ASR.

modeled as sequences of HMM states, with GMMs used to estimate the emission probabilities. The Viterbi algorithm [Viterbi, 1967, Forney, 1973] or the Forward–Backward algorithm [Baum et al., 1970] was then applied to find the most likely word sequence given the acoustic input. GMM-HMMs became the dominant paradigm for several decades, enabling Large Vocabulary Continuous Speech Recognition (LVCSR). In ASR, words are typically modeled as sequences of phonemes, and phonemes are modeled as sequences of HMM states. Modeling context-dependent phonemes (biphones, triphones, or senones) has been demonstrated to perform significantly better than context-independent (monophones) phonemes.

b. Hybrid Deep Neural Network-Hidden Markov Model (DNN-HMM)

In the 2010s, with the availability of better compute capacity and larger datasets, DNNs surpassed GMMs as acoustic models within the HMM framework [Hinton, Geoffrey et al., 2012, Povey et al., 2010, Povey et al., 2011]. The DNN takes a window of acoustic features as input and outputs a probability distribution over HMM states, or *senones* — tied triphone states clustered via decision trees [Dahl et al., 2011] — as shown in Figure 2.3.

A key challenge in training DNN is that the transcriptions should be converted to frame-level state labels. Alignment is therefore required to determine to which HMM state each frame corresponds. This is typically initialized through forced Viterbi alignment using a bootstrapped GMM-HMM system [Povey et al., 2011, Dahl et al., 2011]. These alignments provide the frame-level targets that supervise the DNN to estimate the posterior $P(s_t \mid \mathbf{x}_t)$ at each frame t [Veselỳ et al., 2013].

During decoding, the learned posteriors are converted into pseudo-log-likelihoods via Bayes' rule:

$$p(\mathbf{x}_t \mid s_t) \propto \frac{P(s_t \mid \mathbf{x}_t)}{P(s_t)}, \tag{2.4}$$

where $P(s_t)$ is the prior probability of state s_t estimated from the training alignments [Hinton, Geoffrey et al., 2012]. The HMM maintains temporal constraints through transition probabilities $P(s_t \mid s_{t-1})$. The optimal state sequence for a given word sequence $\mathbf{W}$ is then found by maximizing the joint probability [Povey et al., 2016]:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} \prod_{t=1}^{T} p(\mathbf{x}_t \mid s_t)\, P(s_t \mid s_{t-1})\, P(\mathbf{W}). \tag{2.5}$$

Training Criteria

The two most widely used criteria in hybrid DNN-HMM systems are cross-entropy (CE) training and Lattice-Free Maximum Mutual Information (LF-MMI) [Hinton, Geoffrey et al., 2012, Povey et al., 2016].

Cross-Entropy Training

Given the frame-level targets $\mathbf{s}^*$ obtained from alignment, the CE loss over all T frames is:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} \log P(s_t^* \mid \mathbf{x}_t), \tag{2.6}$$

where $P(s_t^* \mid \mathbf{x}_t)$ is the DNN's predicted posterior for the target state at frame t. Although CE training is effective and computationally efficient, it optimizes a frame-level objective that does not directly minimize WER.

Lattice-Free Maximum Mutual Information

To better align the training objective with recognition performance, sequence discriminative criteria are used [Veselỳ et al., 2013]. Lattice-Free MMI (LF-MMI) maximizes mutual information between acoustic observations and word sequence. Unlike standard MMI [Povey, 2003], LF-MMI avoids the need for pre-generated lattices by computing the denominator over all possible sequences using a phone-level n-gram language model [Stolcke, 2002, Hsu and Glass, 2008, Kneser and Ney, 1995, Jurafsky and Martin, 2008]. The LF-MMI objective is:

$$\mathcal{F}_{\text{MMI}} = \sum_{u} \log \frac{p(\mathbf{X}_u \mid \mathbf{W}_u^*)\, P(\mathbf{W}_u^*)}{\sum_{\mathbf{W}} p(\mathbf{X}_u \mid \mathbf{W})\, P(\mathbf{W})}, \tag{2.7}$$

where u indexes training utterances, $\mathbf{X}_u$ is the acoustic observation sequence for the utterance u, and $\mathbf{W}_u^*$ is the corresponding reference word sequence. $p(\mathbf{X}_u \mid \mathbf{W}_u^*)$ is the likelihood of the observations given the correct hypothesis and $P(\mathbf{W}_u^*)$ is the probability of its language model. The denominator $\sum_{\mathbf{W}} p(\mathbf{X}_u \mid \mathbf{W})\, P(\mathbf{W})$ marginalizes over all competing hypotheses, suppressing incorrect sequences. In practice, LF-MMI and CE are applied together as a multi-task objective and consistently yield significant reductions in WER over CE alone [Povey et al., 2016].

DNN architectures for ASR

The most commonly used neural network architectures for AM training in hybrid ASR are:

Time-Delay Neural Networks (TDNNs)

TDNNs [Waibel et al., 1989, Peddinti et al., 2015] are a class of feed-forward neural networks designed to model the temporal structure using time-delayed inputs, allowing them to capture contextual information in speech signals. They are widely used in acoustic modeling for ASR and will be described in more detail in Section 4.1.1.

Convolutional Neural Networks (CNNs)

CNNs are particularly well-suited for processing data with a grid-like topology, such as images (2D) or sequences like spectrograms (considering frequency and time). CNNs employ convolutional layers that apply small, learnable filters (kernels) to the input. These filters slide across the input, performing dot products, and generating feature maps. This captures local patterns (e.g., phonetic features across frequency bands). For a 2D convolution:

$$\text{Feature Map}_{ij} = \sum_{m} \sum_{n} \text{Input}_{i+m,j+n} \cdot \text{Kernel}_{mn}. \tag{2.8}$$

The pooling layers (e.g., max pooling) then downsample the feature maps, making the representation more robust to small shifts and reducing dimensionality. CNNs are used to capture local spectro-temporal patterns in the acoustic features (e.g., a few frames and a few Mel bins). They offer translation invariance and weight sharing, which makes them efficient and effective for speech representation learning.

Recurrent Neural Networks (RNNs)
RNNs are designed to process sequential data by maintaining an internal "memory" of past inputs. They are crucial for modeling temporal dependencies in speech. At each time step t, an RNN takes the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ to compute the current hidden state $\mathbf{h}_t$ and output $\mathbf{y}_t$:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{2.9}$$

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y. \tag{2.10}$$

$\mathbf{W}_{xh}$ and $\mathbf{W}_{hh}$ are the weight matrices for the input and the previous hidden state, respectively, and $\mathbf{W}_{hy}$ is the weight matrix mapping the hidden state to the output. The terms $\mathbf{b}_h$ and $\mathbf{b}_y$ represent the bias vectors for the hidden and output layers, while σ denotes an element-wise non-linear activation function, such as tanh or ReLU, which allows the model to capture non-linear dependencies in the sequential data.

Standard RNNs suffer from vanishing/exploding gradient problems, making them difficult to train for long sequences [Bengio et al., 1994, Pascanu et al., 2013]. Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Unit (GRU) [Cho et al., 2014] networks address these issues with gating mechanisms that control the flow of information into and out of memory cells, allowing them to learn long-range dependencies. RNNs (especially LSTMs/GRUs) are used to model the sequential nature of speech, capturing context over longer durations than a fixed window (as in FNNs or CNNs). Bidirectional RNNs (BRNNs) [Schuster and Paliwal, 1997] process information both in forward and backward directions, providing a richer context. Bidirectional LSTMs (BLSTMs) [Graves and Schmidhuber, 2005] further extend this by combining the gating mechanisms of LSTMs with bidirectional processing, allowing the model to exploit both acoustic context of the past and future in each frame [Graves et al., 2013]. This makes BLSTMs particularly effective for offline speech recognition, where the full utterance is available at inference time, and they have consistently demonstrated strong performance across a range of acoustic modeling tasks [Graves et al., 2013, Sak et al., 2014].

In practice, these architectures have been widely adopted within the Kaldi toolkit [Povey et al., 2011], ESPnet [Watanabe et al., 2018] and Eesen [Miao et al., 2015]. The CNN-TDNN (Convolutional Time Delay Neural Network) architecture combines convolutional layers for local feature extraction with TDNN layers that efficiently capture long-range temporal context through subsampling [Peddinti et al., 2015, Povey et al., 2018]. TDNN-LSTM models interleave TDNN layers with LSTM layers, balancing the temporal modeling strengths of both architectures while remaining computationally tractable for large-scale training [Peddinti et al., 2017]. Both architectures are trained using LF-MMI in Kaldi's chain model framework and are the standard for hybrid DNN-HMM systems.

c. E2E Systems

E2E deep learning architectures directly map acoustic features to text sequences, bypassing the alignment step in hybrid ASR. With the success of the Transformers [Vaswani et al.,

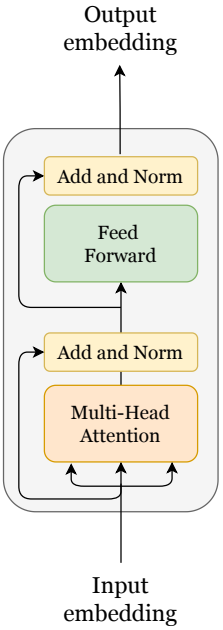

Figure 2.4: Overview of a Transformer layer adapted from [Vaswani et al., 2017a].

2017a] architecture for sequential data in combination with self-training methods for acoustic models, the adoption of E2E models became more widespread as their performance exceeded hybrid systems.

Transformers

The Transformer [Vaswani et al., 2017a] architecture shown in Figure 2.4 revolutionized sequence modeling, particularly in Natural Language Processing (NLP) and has since been successfully adapted for ASR. It shifts from recurrent or convolutional operations for sequence modeling to rely solely on attention mechanisms [Bahdanau et al., 2015], allowing parallelization and capturing long-range dependencies more effectively. The key components of a transformer layer are:

1. **Self-Attention Mechanism:** For each input element, it recomputes the representation with respect to the "context" by attending to all other elements in the sequence. Here, the input sequence $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N} \in \mathbb{R}^{N \times F}$, where N is the number of positions (e.g. time steps or tokens), and F is the feature dimension, is projected into three distinct spaces to produce *queries*, *keys*, and *values*:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \quad \mathbf{Q} \in \mathbb{R}^{N \times D} \tag{2.11}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}_K \quad \mathbf{K} \in \mathbb{R}^{N \times D} \tag{2.12}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}_V \quad \mathbf{V} \in \mathbb{R}^{N \times M}, \tag{2.13}$$

   where $\mathbf{W}_Q \in \mathbb{R}^{F \times D}$, $\mathbf{W}_K \in \mathbb{R}^{F \times D}$, and $\mathbf{W}_V \in \mathbb{R}^{F \times M}$ are learnable projection matrices. These projections serve the following roles:

- **Query (Q)**: Represents the current position (e.g., a token or time frame) that is attending to other positions.
- **Key (K)**: Encodes the content of each position in the sequence and is compared against the query to compute relevance.
- **Value (V)**: Contains the actual information to be aggregated, weighted by how well the key matches the query.

The attention mechanism uses the dot product between queries and keys to compute a similarity score that determines how much each value should contribute to the final output.

- **Scaled Dot-Product Attention:** This is the fundamental attention function. It takes Queries and Keys to compute compatibility scores, scales them, applies a softmax to get attention weights, and then multiplies these weights by the Values to get a weighted sum.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2.14}$$

  where d_k is the dimension of the keys (and queries). Scaling by $\sqrt{d_k}$ prevents the dot products from growing too large, which could push the softmax into regions with very small gradients.
- **Multi-Head Attention:** Instead of performing a single attention function, Multi-Head Attention performs h (e.g., 8) attention functions in parallel, each with independent learned $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ linear projections. The outputs from these h "heads" are then concatenated and linearly transformed to produce the final output.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W^O}, \tag{2.15}$$

  where

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V). \tag{2.16}$$

  This allows the model to jointly attend to information from different representation subspaces at different positions, enriching its capacity to capture diverse relationships.

2. **Feed-Forward Networks (FFN):** Each encoder and decoder layer contains a simple, fully connected FFN applied independently to each position. It consists of two linear transformations with a ReLU activation in between:

$$FFN(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W_1} + \mathbf{b}_1)\mathbf{W_2} + \mathbf{b}_2. \tag{2.17}$$

   where max is applied element-wise. Typically, $\mathbf{W_1}$ is an upward projection, projecting the embedding from D to κD ($\kappa = 2, 4, \ldots$) and $\mathbf{W_2}$ is a downward projection that returns the embedding to dimensions D.

3. **Layer Normalization and Residual Connections:**

- **Residual Connections:** Each sub-layer (multi-head attention, FFN) has a residual connection [He et al., 2016], where the input to the sub-layer is added to its output. This helps train very deep networks by allowing gradients to flow more easily during backpropagation.

$$\text{Output} = \text{LayerNorm}(\mathbf{x} + \text{Sublayer}(\mathbf{x})). \tag{2.18}$$

- **Layer Normalization [Ba et al., 2016]:** Applied after the residual connection, it normalizes the inputs between the features (not between batches), helping to stabilize training and accelerate convergence.

Transformer for ASR

Transformer architectures have achieved state-of-the-art performance in ASR following the success in natural language processing tasks. In [Karita et al., 2019], the authors show on 13 out of 15 benchmarks that transformers outperform LSTMs.

Beyond supervised learning, Transformer architectures have become the de facto choice for self-supervised pre-training methods trained on large amounts of unlabeled speech data. Among the most prominent approaches are: (1) the wav2vec 2.0 (W2V2) model and its variants [Baevski et al., 2020, Chung et al., 2021, Sadhu et al., 2021], which employ contrastive loss, and (2) the Masked Acoustic Modeling (MAM) variants [Liu et al., 2021, Wang et al., 2020, Hsu et al., 2021], which are based on reconstructing masked audio segments. These Transformer-based pre-trained models are subsequently fine-tuned on supervised data to improve the ASR acoustic modeling [Chen et al., 2022a, Vyas et al., 2021a, Zuluaga-Gomez et al., 2023]. Baevski et al. [Baevski et al., 2020] report a relative WER improvement of 41% and 56% on the `clean` and `other` portions of the LibriSpeech dataset, respectively, when fine-tuning the W2V2 model on only 100 hours of labeled data.

Given the widespread success of Transformer-based models, this thesis focuses on analyzing and improving various aspects of Transformer-based systems.

Variants of Transformer

The **Conformer** [Gulati et al., 2020] model is a Transformer variant specifically designed for speech recognition tasks. While the standard Transformer effectively captures global context through self-attention, it lacks the ability to model local dependencies efficiently, which are crucial in speech signals. The Conformer addresses this by integrating convolutional layers into the Transformer module. Each Conformer block consists of a combination of feed-forward modules, multi-head self-attention, and convolutional modules arranged in a sandwich structure. This design allows the model to capture both global and local contextual information effectively. As a result, Conformer significantly improves performance on a range of ASR benchmarks compared to vanilla Transformers and has become a widely adopted backbone in state-of-the-art ASR.

The **Zipformer** [Yao et al., 2023] is a recent Transformer variant introduced to address the computational inefficiencies of conventional Transformer-based ASR models, especially for streaming and real-time applications. Zipformer adopts a hierarchical attention mechanism in which the sequence length is progressively compressed and then expanded across the model's layers, effectively "zipping" the temporal resolution. This compression reduces the amount of computation required in deeper layers without sacrificing modeling capacity. Additionally, Zipformer integrates a mix of high- and low resolution attention paths, allowing it to maintain accurate recognition while significantly lowering memory usage and

inference time. These properties make Zipformer particularly suitable for deployment in latency-sensitive and resource-constrained environments.

Sequence-to-sequence Training

The two commonly used sequence training criteria used in ASR are Connectionist Temporal Classification (CTC) [Graves et al., 2006] and Lattice-Free Maximum Mutual Information (LFMMI) [Povey et al., 2016]. This section provides an overview of these criteria.

Connectionist Temporal Classification

CTC [Graves et al., 2006] is a loss function designed for sequence-to-sequence problems where the alignment between the input sequence and the output sequence is unknown.

In ASR, CTC allows an E2E training of a neural network to directly map acoustic features to a sequence of characters or phonemes without needing a pre-defined alignment between acoustic frames and labels. A special "blank" symbol is introduced to handle variable sequence lengths and repeated labels.

The neural network outputs a probability distribution over the vocabulary (plus the blank symbol) for each input frame. CTC sums the probabilities of all possible valid alignments (paths through the output sequence that collapse to the target transcription) to compute the loss. This allows the network to learn to predict the output sequence without explicit segmentation or alignment during training, simplifying the ASR pipeline.

Lattice-Free Maximum Mutual Information

LF-MMI [Povey et al., 2016] is a discriminative training objective function widely used in ASR, particularly with TDNN-HMM hybrid systems. It evolved from the Maximum Mutual Information (MMI) criterion, which aims to maximize the probability of the correct word sequence given the acoustic observations while minimizing the probability of incorrect word sequences.

- *Motivation:*
  - **Discriminative Training:** Unlike Maximum Likelihood (ML) training (e.g., Baum-Welch), which only maximizes the likelihood of the training data, discriminative training directly optimizes a measure related to classification accuracy.
  - **Avoiding Lattices:** Traditional MMI training often required generating large "confusion network" or "state-level" lattices from a decoding pass to represent alternative hypotheses. This process was computationally expensive and complex. LF-MMI eliminates the need for explicit lattice generation during training by applying forward-backward algorithm over an n-gram model that could fit in the GPU.
- *MMI Objective Function:* The MMI objective for a given utterance with acoustic observations $\mathbf{X}$ and corresponding correct word sequence $\mathbf{W}^*$ is defined as:

$$\mathcal{F}_{\text{MMI}}(\mathbf{X}, \mathbf{W}^*) = \log \frac{P(\mathbf{X}|\mathbf{W}^*)P(\mathbf{W}^*)}{\sum_W P(\mathbf{X}|\mathbf{W})P(\mathbf{W})}, \tag{2.19}$$

  where $P(\mathbf{X}|\mathbf{W})$ is the acoustic model likelihood of observations $\mathbf{X}$ given word sequence $\mathbf{W}$, and $P(\mathbf{W})$ is the language model prior probability of word sequence $\mathbf{W}$. The sum in the denominator ranges over all possible word sequences $\mathbf{W}$. Training aims to maximize this function over the entire training set.

- *Lattice-Free Implementation:* The "lattice-free" aspect comes from avoiding explicit lattice computation for the denominator term. Instead, LF-MMI typically computes the log-likelihoods for the numerator ($P(\mathbf{X}|\mathbf{W}^*)P(\mathbf{W}^*)$) and denominator ($\sum_W P(\mathbf{X}|\mathbf{W})P(\mathbf{W})$) using compact finite-state transducers (FSTs) or graphs.
  - **Numerator Graph:** Represents only the correct word sequence $\mathbf{W}^*$. It's typically a simple HMM-level graph constrained to the correct transcription.
  - **Denominator Graph:** Represents a vast set of possible word sequences, typically constructed by composing the phone HMMs with a lexicon and a pruned language model. Crucially, this graph is usually built "on-the-fly" or is an optimized static graph that does not involve explicit lattice generation per utterance.

  The Viterbi algorithm (or a scaled version of the Baum-Welsh Forward algorithm) is used on these graphs to calculate the total probability of paths. For the denominator, instead of summing over all paths explicitly, a pruned version (or a fixed-size 'denominator graph') is used to approximate the sum.

- *Optimization:* Acoustic models in LF-MMI are optimized using Natural Stochastic Gradient Descent (N-SGD) [Povey et al., 2014], allowing independent and parallel training over multiple GPUs. The "derivatives" (or state posteriors) are computed from the forward-backward probabilities on both numerator and denominator graphs.

Cross-Entropy for Encoder-Decoder
Figure 2.5 provides an overview of the encoder-decoder architecture using the transformers.

- **Encoder:** Maps an input sequence of acoustic features $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ to a sequence of continuous context-aware representations $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_T)$. A standard encoder block consists of: (i) A Multi-Head Self-Attention layer, (ii) A Position-wise Feed-Forward Network, and (iii) Each sub-layer is wrapped with a residual connection and followed by layer normalization.

- **Decoder:** Takes the encoder's output $\mathbf{Z}$ and generates an output sequence of tokens $\mathbf{Y} = (y_1, \ldots, y_{T'})$ (e.g., characters, phonemes, or subword units) one element at a time, autoregressively. A standard decoder block also has residual connections and layer normalization and contains three sub-layers:

  1. **Masked Multi-Head Self-Attention:** This attention layer is similar to the encoder's self-attention but is masked to prevent positions from attending to subsequent positions. This ensures that the predictions for position i can only depend on known output at positions less than i (autoregressive property).
  2. **Encoder-Decoder Attention:** This is a multi-head attention layer where the Queries come from the previous decoder layer, and the Keys and Values come from the output of the encoder. This allows the decoder to attend to the relevant parts of the input (acoustic) sequence when generating the output (text) sequence.
  3. **Position-wise Feed-Forward Network:** Same as in the encoder.

  The final output of the decoder stack is typically passed through a linear layer and a softmax function to produce the probability distribution over the vocabulary.

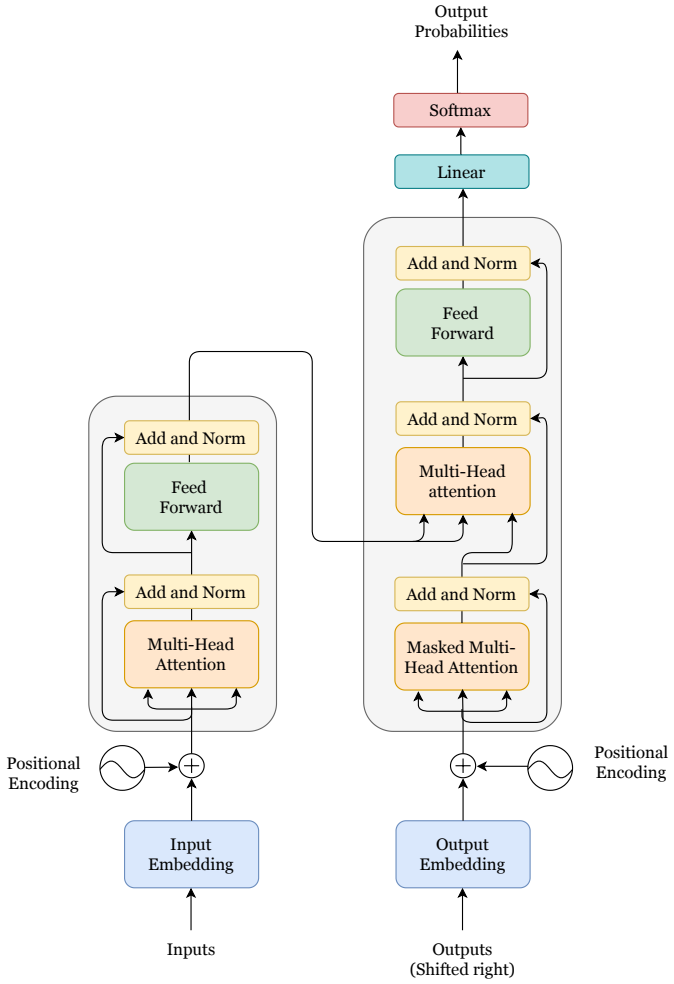

Figure 2.5: Overview of a Transformer model applied in the Encoder-Decoder architecture. The left part shows the encoder and the right part shows the decoder. The figure is adapted from [Vaswani et al., 2017a].

2.1.3 Language Models

Typical LM used in a hybrid ASR is a statistical n-gram model, which is represented by the probability distribution over a sequence of words with n being 1 (unigram), 2 (bi-gram), 3 (tri-gram), or higher. The n-gram models calculates the conditional probability of a word given n-1 previous words. In an n-gram LM, the probability $P(w_1, ..., w_N)$ of observing the sentence $w_1, ..., w_N$ is approximated as:

$$\begin{aligned} P(w_1, ..., w_N) &= \prod_{i=1}^{N} P(w_i | w_1, ..w_{i-1}) \\ &\approx \prod_{i=1}^{N} P(w_i | w_{i-(n-1)}, ..w_{i-1}), \end{aligned} \tag{2.20}$$

where n is the order of the model. Using the Markov property, in equation 2.20, the i^{th} word w_i is estimated as the probability of observing its preceding $n-1$ words. In ASR, a

typical LM trained is a tri-gram (3-gram) model. Increasing the amount of training data and order of the n-gram model to 4-gram or 5-gram typically improves system performance. Since the model size increases with its order, large LMs render decoding graphs impractical to build and use. With increase in quantity of data, the performance of the n-gram models may also decrease [Bengio et al., 2003].

The possibility of training LMs with neural networks has been investigated since many years. The first neural network based LM, also referred to as neural LMs, was proposed by Bengio [Bengio et al., 2003] in 2003. Neural LMs were intially succesfully implemented with RNNs [Goodfellow et al., 2016], with Mikolov et al. demonstrating that RNN-based LMs can significantly outperform traditional n-gram models in speech recognition [Mikolov et al., 2010].

The performance of a language model is measured in terms of perplexity. It is a measurement of how well a model predicts the next unit of text (words, tokens, etc.). Perplexity is the inverse of the probability assigned to the test set by the language model, normalized by the number of words in the test set. i.e., if a test set has $W = \{w_1, ..w_N\}$ words, then the perplexity is estimated as [Jurafsky, 2000]:

$$PP(W) = \sqrt[N]{\frac{1}{PP(w_1, w_2, ..w_N)}}. \tag{2.21}$$

Using equation 2.20 in the equation 2.21, we get

$$PP(W) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} PP(w_i|w_{i-(n-1)}, ..w_{i-1})}}. \tag{2.22}$$

2.1.4 Metric

ASR is evaluated using the WER metric. It is calculated as follows:

$$WER = \frac{S + D + I}{N}, \tag{2.23}$$

where S is the number of words that are substituted, D is the number of words that are deleted, I is the number of words that are inserted, $N = S + D + C$ is the total number of words in the reference, and C is the number of correct words. A lower WER implies better accuracy for the ASR.

2.1.5 Datasets

We use two datasets: AMI and ATC. These two datasets represent fundamentally different use-cases. AMI is a commonly used conversational English dataset, while ATC is low-resourced and domain-specific.

a. Domain Meetings - AMI

We use the AMI Meeting Corpus [Kraaij et al., 2005] for the evaluation of ASR. Only the Individual Headset Microphone (IHM) part of the training and evaluation is used. The meetings were recorded in English using three different rooms with different acoustic properties and include mostly non-native speakers. In our experiments, the training, development and test set consist of 77 hours, 9 hours and 8.7 hours respectively. A Three-fold speed perturbation is applied in the training data [Povey et al., 2011].

Table 2.1: Overview of the training and evaluation sets for ATC.

Dataset	No. of utterances (k)	Duration (hours)
Training		
Airbus	27.7	39
ATCOSIM	7.4	8
HAAWAII	39.6	43
LDC	26.9	23
MALORCA	7.7	10
NATO	2.5	10.7
SOL96 - 97	8.4	14
STARFISH	26.4	31.5
UWB	11.3	10.4
Evaluation		
NATS	0.9	1
ATCO2	0.9	1

b. Domain Air Traffic Control

Labeled ATC communication data is scarce and expensive to produce. Typically, only tens of hours of air-traffic control speech data are publicly available for ASR, Sequence Labeling (SC) or Named Entity Recognition (NER) training. The following sections provide an overview of the several datasets used for training and evaluating the ASR.

Training

The AM and LM components of the ASR are trained using the datasets described below.

- **AIRBUS** [Delpech et al., 2018]: The corpus consists of 39 hours of transcribed French-accented English ATC speech interactions between pilots and controllers.
- **ATCOSIM** [Hofbauer et al., 2008]: corpus is a publicly available dataset containing approximately 8 hours of clean ATC speech recorded in simulated real-time scenarios. It features 10 non-native English-speaking controllers of German, Swiss German, and Swiss French backgrounds, using close-talk headsets to ensure high audio quality
- **HAAWAII:** The Horizon 2020 funded HAAWAII[1] project develops a reliable, error resilient and adaptable solution to automatically transcribe voice commands from ATCO and pilots.

  The project was built on very large collections of speech data, organized with minimal expert effort, to develop a new set of speech recognition models for the complex ATM environments of the TMA and Icelandic enroute airspace. Speech and surveillance data recordings from real-life pilot-controller communications, i.e., directly from Operations rooms, were used. The data for London TMA came from London ANSP, NATS, while the data for Icelandic airspace were provided by Icelandic ANSP, Isavia.

[1] www.haawaii.de

The manually transcribed data for each of these ANSPs amounted to approximately 21 hours, totaling 40 hours used in ASR training.

- **LDC** [**Godfrey, 1994**]: is a publicly available air traffic control speech dataset, released by the Linguistic Data Consortium in 1994. It consists of approximately 70 hours of analog audio recordings from U.S. ATC environments, specifically en route communications between air traffic controllers and pilots. Applying VAD provides around 23 hours which is used for training.

- **MALORCA**[2]: The Horizon 2020 SESAR project MALORCA (MAchine Learning Of speech Recognition models for Controller Assistance) aimed to provide an efficient and reliable process for a successful deployment of Assistance Based Speech Recognition (ABSR) systems for ATC tasks, from the laboratory to the real world. To support this goal, approximately 10 hours of ATCO communication data were collected from Prague and Vienna airports.

- **NATO** [**Pigeon et al., 2007**]: corpus was developed by the NATO Speech and Language Technology group to facilitate research into multilingual and non-native speech, particularly within a military communication context. The corpus comprises approximately 10.7 hours of English speech gathered during naval communication training sessions across Germany, the Netherlands, the United Kingdom, and Canada between 2000 and 2002.

- **SOL96 and SOL97**: During three SESAR 2020 funded industrial research projects PJ.10-W2-96[3], PJ.16-W1-04[4], and PJ.05-W2-97[5] "SOL96" (the first) & "SOL97" (combination of second and third) data sets have been recorded and collected. With an ASR supported aircraft radar labels, SOL96 aims to reduce ATCOs workload. Voice utterances of ATCOs and pilots have been recorded together with ATC surveillance data for Vienna approach at the air navigation service provider site of Austrocontrol in Vienna, Austria. Approximately 4 hours of data were collected in this setting. SOL97 aims to reduce ATCO workload in an ATC tower environment, previous research and performances in [Ohneiser et al., 2021]. Approximately 10 hours of data were collected in this setting. Due to the laboratory recording environment, the SOL97 audio data is less noisy compared to SOL96. A total of 14 hours is used for training.

- **STARFISH**: was a project aiming to create and improve ASR for the airport Fraport, Germany. The collected data is ATCO speech recorded in the ATC operations room during simulations at Fraport. The data set currently contains 15 hours of transcribed and manually corrected speech.

- **UWB** [**Šmídl et al., 2019**]: corpus consists 10 hours of manually transcribed Czech-accented English speech data. The data was collected in Prague airport.

We standardized and normalized the ontologies (annotation procedure, rules, and symbols) of the databases [Zuluaga-Gomez et al., 2020, Helmke et al., 2018].

[2] http://www.malorca-project.de/wp
[3] https://cordis.europa.eu/programme/id/H2020_SESAR-IR-VLD-WAVE2-10-2019
[4] https://www.sesarju.eu/projects/cwphmi
[5] https://www.remote-tower.eu/wp/project-pj05-w2/solution-97-2

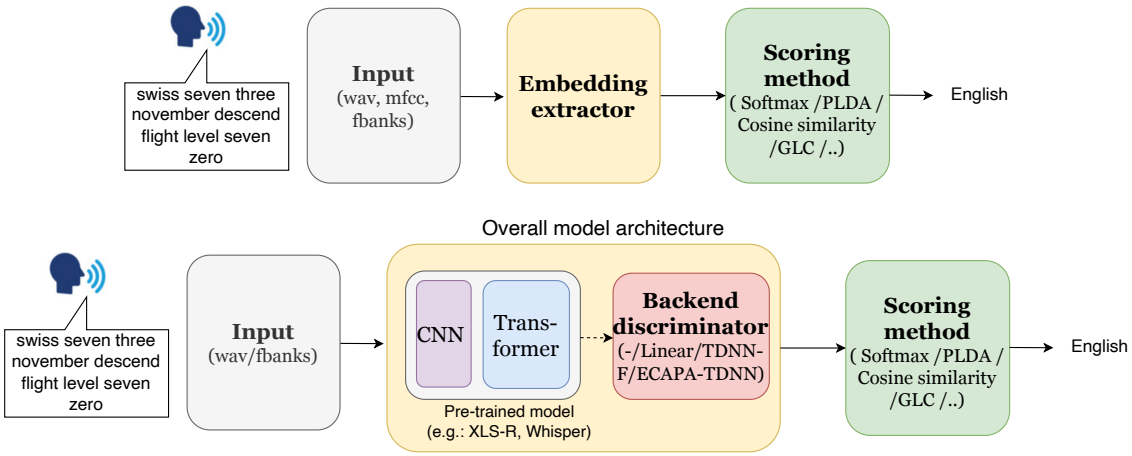

Figure 2.6: Overview of LID. Based on the type of model used in embedding extractor, the input can be raw audio or short segments or its features.

Evaluation

The ASR for the ATC domain is evaluated on the two datasets as described below.

- **NATS**: In addition to the 40 hours obtained from HAAWAII for training, we use a 1 hours set from NATS ANSP for testing.
- **ATCO2**: Clean Sky 2 Joint Undertaking (JU) and EU-H2020 funded ATCO2 project[6], developed a platform to collect, organize, pre-process and automatically annotate ATC dialogues. Approximately 5000 hours of ATC audio data were automatically collected from the Czech Republic, Switzerland, and Australia. While the majority of this data remains untranscribed, a small portion—about 4 hours—was manually transcribed. From this, a publicly available 1-hour subset is used in this work for evaluation purposes.

An overview of these sets is provided in Table 2.1.

2.2 Language Identification

LID determines the language spoken in an audio sample and is crucial in multilingual systems. A schematic diagram of the various blocks in LID is shown in Figure 2.6. Traditionally, LID systems relied on phonotactic models, using language-specific phone recognizers and modeling the sequence of phones statistically. However, modern LID systems rely on fixed-length vector representations called embeddings, extracted from variable-length speech. These embeddings should contain language discriminatory features and remain robust under varying conditions such as noise, channel variability, and speaker identity.

Prior work on LID has been significantly shaped by innovative technologies from speaker recognition tasks, such as use of Subspace GMMs (SGMM) [Motlicek et al., 2015], i-vectors [Dehak et al., 2010], online i-vectors [Dey et al., 2017] dealing with short segments, or phonetically aware d-vectors [Dey et al., 2018]. The i-vector approach represents a speech

[6] **A**u**T**omatic **CO**llection and processing of voice data from **A**ir-**T**raffic **CO**mmunications, website: https://www.atco2.org/

segment with a compact, fixed-length vector extracted from a Universal Background Model (UBM) and a total variability matrix. These vectors are then scored using cosine distance, PLDA or Support Vector Machines (SVMs). While effective, i-vectors struggle with noisy conditions and short utterances.

Following the success of neural network-based approaches for acoustic modeling, x-vectors [Snyder et al., 2019] demonstrated that i-vectors can be replaced succesfully with neural network based embeddings for modeling speaker and language identities. These are deep neural network-based embeddings learned in a supervised manner from raw acoustic features. The x-vector model consists of frame-level TDNN layers, a statistics pooling layer to aggregate temporal information, and segment-level dense layers to produce the x-vector.

Modern LID systems often combine x-vectors with scoring methods such as PLDA [Snyder et al., 2018b, Prince and Elder, 2007, Kenny, 2010] or neural classifiers [Ferrer et al., 2022]. In addition, large-scale multilingual models such as XLS-R [Babu et al., 2022] and Whisper [Radford et al., 2023] can perform LID and ASR jointly by leveraging massive amounts of unlabeled speech and cross-lingual transfer.

2.2.1 Evolution of embedding extractors

In the 2000s, the GMM-UBM approach was predominant in speaker and language recognition tasks. Reynolds et al. [Reynolds et al., 2000] introduced a speaker verification system that utilized adapted GMMs, where speaker-specific models were derived from a universal background model using maximum a posteriori (MAP) adaptation. This methodology laid the groundwork for the i-vector framework.

Dehak et al. [Dehak et al., 2010] further advanced this concept with the introduction of factor analysis for speaker verification, which decoupled speaker and channel variability using a low-dimensional i-vector representation.

Lei et al. [Lei et al., 2014] explored DNN approaches for speaker recognition, demonstrating that DNNs could effectively model complex patterns in speech data. Various architectures used are briefly described below.

a. x-Vectors [Snyder et al., 2019]

The DNN-based x-vector framework employes TDNNs to extract fixed-dimensional speaker embeddings from variable-length utterances. This progression from GMM-UBM models to DNN-based embeddings underscores the continuous evolution in embedding extraction techniques, enhancing the capabilities of speaker/language identification systems. The architecture can be divided in the following blocks:

- Frame-level TDNN layers: Capture short and medium-term context.
- Statistics pooling: Computes mean and standard deviation over frames.
- Segment-level dense layers: Extract utterance-level embeddings.
- Softmax layer: Used for speaker/language classification during training.

Given frame-level outputs $\mathbf{h}_t$, the statistics pooling layer computes:

$$\boldsymbol{\mu} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{h}_t, \quad \boldsymbol{\sigma} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\mathbf{h}_t - \boldsymbol{\mu})^2}. \tag{2.24}$$

The concatenated vector $\mathbf{s} = [\boldsymbol{\mu}, \boldsymbol{\sigma}]$ is passed through dense layers to obtain the embedding $\mathbf{z}$. The softmax layer is removed after training, and the output from a segment-level layer is used as the x-vector. These are used with cosine scoring or PLDA for language recognition.

b. ResNet

Residual Networks (ResNet) are deep convolutional architectures that use identity shortcut connections to ease training and enable very deep models. Each residual block can be expressed as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{W}_i) + \mathbf{x}, \tag{2.25}$$

where $\mathcal{F}$ is the residual function (e.g., Conv-BN-ReLU), and is the input. This structure allows the gradient to propagate more easily in deep networks.

In LID systems, ResNet architectures help in extracting hierarchical time-frequency representations. Modern adaptations like Res2Net allow multiple receptive fields within a single block, improving language-discriminative capacity.

c. ECAPA-TDNN

Emphasized Channel Attention, Propagation and Aggregation Time-Delay Neural Network (ECAPA-TDNN) [Desplanques et al., 2020] architecture improves the x-vector framework by introducing multi-scale feature extraction, channel attention, and attentive pooling. The difference blocks of the ECAPA-TDNN model are:

- *Frame-level TDNN blocks:* Capture local temporal context with dilation to expand the receptive field.
- *SE-Res2Block:* Enhances feature representation by integrating:
  - Res2Net module: Splits feature maps into multiple groups and applies convolutions in a hierarchical residual manner for multi-scale modeling.
  - Squeeze-and-Excitation (SE): Learns channel-wise dependencies to emphasize informative channels.
- *Multi-layer feature aggregation:* Concatenates outputs from multiple intermediate layers to leverage multi-level abstraction.
- *Attentive Statistics Pooling (ASP):* Computes weighted mean and standard deviation across time using learned attention weights. Given frame-level features h_t, attention weights α_t are computed as:

  $$\alpha_t = \frac{\exp(w^T \tanh(V h_t^T))}{\sum_{t=1}^{T} \exp(w^T \tanh(V h_t^T))}. \tag{2.26}$$

  Then the pooled segment embedding is:

  $$s = \sum_{t=1}^{T} \alpha_t h_t. \tag{2.27}$$

- *Segment-level fully connected layers:* Project the pooled embedding into a fixed-length vector space.

2.2.2 Scoring Method

A trained LID model can be directly detect the language by passing the input through its final softmax layer. Alternatively, an embedding at the utterance level can be extracted after the pooling layer, which can then be used with various scoring techniques. These include cosine distance, PLDA, GLC to improve the performance. In the following section, we briefly explain each of these classifiers.

a. Cosine similarity

It is one of the simplest and most widely used scoring techniques for comparing language or speaker embeddings. It measures the similarity between two vectors by calculating the cosine of the angle between them, making it effective for comparing i-vectors or x-vectors in a high-dimensional space. A neural network maps each audio sample to a fixed-length embedding that captures language-specific characteristics. By computing the cosine distance between embeddings, the system can assess the similarity between two samples—where a smaller cosine distance implies a higher likelihood that the samples originate from the same speaker or language. This makes cosine distance an effective and interpretable metric for verification and classification in speech processing applications. Given two vectors $\mathbf{a}$ and $\mathbf{b}$, the cosine similarity is defined as

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}, \tag{2.28}$$

where $\mathbf{a} \cdot \mathbf{b}$ denotes the dot product of the vectors, and $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ represent their Euclidean norms.

b. Gaussian Linear Classifier (GLC)

It is a generative classification model that assumes data from each class (e.g., each language) are drawn from a Gaussian distribution with *shared covariance matrix*. This model is closely related to Linear Discriminant Analysis (LDA) [Hastie et al., 2009] and simplified versions of PLDA [Ioffe, 2006] used in high-dimensional embedding spaces such as i-vectors or x-vectors.

In LID, each language corresponds to a class. Embeddings extracted from speech utterances are modeled by class-conditional Gaussian distributions. The classifier computes the log-likelihood of an embedding belonging to each language and assigns the class with the highest likelihood.

Mathematical Formulation: Let,

- $\mathbf{x} \in \mathbb{R}^d$ be a test embedding,
- $\boldsymbol{\mu}_c$ be the mean vector of class c,
- $\mathbf{W}$ be the shared within-class covariance matrix.

The class-conditional likelihood is modeled as:

$$p(\mathbf{x} \mid c) = \frac{1}{(2\pi)^{d/2}|\mathbf{W}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_c)^\top \mathbf{W}^{-1}(\mathbf{x} - \boldsymbol{\mu}_c)\right). \tag{2.29}$$

For classification, constants can be ignored, and the log-likelihood simplifies to:

$$\log p(\mathbf{x} \mid c) = -\frac{1}{2}\left(\mathbf{x}^\top \mathbf{W}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_c^\top \mathbf{W}^{-1}\mathbf{x} + \boldsymbol{\mu}_c^\top \mathbf{W}^{-1}\boldsymbol{\mu}_c\right). \tag{2.30}$$

The GLC operates in two main stages: training and inference. In the *training phase*, described in Algorithm 1, the within-class covariance matrix $\mathbf{W}$ is estimated from the training embeddings, and the mean vector $\boldsymbol{\mu}_c$ is computed for each language class. To enable efficient evaluation, the products $\mathbf{W}^{-1}\boldsymbol{\mu}_c$ and $\boldsymbol{\mu}_c^\top \mathbf{W}^{-1}\boldsymbol{\mu}_c$ are also precomputed. During the *inference* – to generate score, shown in Algorithm 3, a test embedding $\mathbf{x}$ is evaluated by computing the terms $\mathbf{x}^\top \mathbf{W}^{-1}\mathbf{x}$ and $\boldsymbol{\mu}_c^\top \mathbf{W}^{-1}\mathbf{x}$. These quantities are then combined to obtain a log-likelihood score for each class, enabling language identification based on the Mahalanobis distance in the transformed space.

Algorithm 1: Gaussian Linear Classifier (GLC) Training

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{d\times N}$, labels $\mathbf{y} \in \{1,\ldots,C\}^N$
Output: Within-class covariance matrix $\mathbf{W}_C$, class means $\boldsymbol{\mu}_c$, matrix $\mathbf{M}^\top\mathbf{B}$

Function `GLC_Fit(X, y):`
  $(\mathbf{W}_C, \boldsymbol{\mu}_c) \leftarrow$ `Compute_WC_Cov(X, y)`;
  $\mathbf{B}_i\mathbf{M} \leftarrow \mathbf{W}_C^{-1}\boldsymbol{\mu}_c$; `// Inverse covariance times means`
  $\mathbf{M}^\top\mathbf{B} \leftarrow \sum_c \boldsymbol{\mu}_c^\top \mathbf{W}_C^{-1}\boldsymbol{\mu}_c$; `// Quadratic class terms`
  return $(\mathbf{W}_C, \boldsymbol{\mu}_c, \mathbf{M}^\top\mathbf{B})$;

Algorithm 2: Compute Within-Class Covariance and Class Means

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{d\times N}$, labels $\mathbf{y}$
Output: Within-class covariance $\mathbf{W}_C$, class means $\boldsymbol{\mu}_c$

Initialize $\mathbf{W}_C \leftarrow \mathbf{0}$;
foreach *class* $c \in \{1,\ldots,C\}$ **do**
  $\mathbf{X}_c \leftarrow \{\mathbf{x}_i \in \mathbf{X} \mid y_i = c\}$;
  $\boldsymbol{\mu}_c \leftarrow \frac{1}{|\mathbf{X}_c|}\sum_{\mathbf{x}\in\mathbf{X}_c} \mathbf{x}$;
  $\mathbf{W}_C \leftarrow \mathbf{W}_C + \sum_{\mathbf{x}\in\mathbf{X}_c}(\mathbf{x}-\boldsymbol{\mu}_c)(\mathbf{x}-\boldsymbol{\mu}_c)^\top$;
$\mathbf{W}_C \leftarrow \frac{1}{N}\mathbf{W}_C$;
return $(\mathbf{W}_C, \boldsymbol{\mu}_c)$;

Algorithm 3: GLC Log-Likelihood Scoring

Input: Test data $\mathbf{X}_{\text{test}} \in \mathbb{R}^{d\times M}$, model parameters $\mathbf{W}_C$, $\boldsymbol{\mu}_c$, $\mathbf{M}^\top\mathbf{B}$
Output: Log-likelihood matrix $\mathbf{Y} \in \mathbb{R}^{C\times M}$

$\mathbf{B}_i\mathbf{X} \leftarrow \mathbf{W}_C^{-1}\mathbf{X}_{\text{test}}$; `// Inverse covariance applied to test data`
$\mathbf{D} \leftarrow \mathbf{M}^\top\mathbf{B} - 2\boldsymbol{\mu}_c^\top\mathbf{B}_i\mathbf{X}$; `// Class-wise Mahalanobis distances`
$\mathbf{X}^\top\mathbf{B}_i\mathbf{X} \leftarrow \sum_{\mathbf{x}} \mathbf{x}^\top\mathbf{W}_C^{-1}\mathbf{x}$; `// Sample-wise quadratic terms`
$\mathbf{Y} \leftarrow -\frac{1}{2}(\mathbf{D} + \mathbf{X}^\top\mathbf{B}_i\mathbf{X})$; `// Final log-likelihood scores`
return $\mathbf{Y}$;

Usage and Advantages In LID, GLC operates on embeddings such as i-vectors or x-vectors to model language-specific distributions with shared covariance. It is widely used due to its simplicity, interpretability, and computational efficiency, making it effective for discriminating between languages even with moderate amounts of training data.

c. Probabilistic Linear Discriminant Analysis (PLDA)

PLDA [Ioffe, 2006] is a more sophisticated scoring method that models the variability of speaker or language embeddings using a probabilistic framework. It is a generative probabilistic model designed to separate between-class and within-class variability in high-dimensional data. Originally applied to face recognition, PLDA has since become widely used in speaker and language recognition tasks. In LID, PLDA operates on fixed-dimensional embeddings such as i-vectors or x-vectors, modeling their distribution across different language classes.

The PLDA model assumes that an observed vector $\mathbf{x}$ (e.g., an i-vector) can be decomposed into a sum of a global mean $\boldsymbol{\mu}$, a latent language-dependent variable $\mathbf{z}$, and a residual noise term $\boldsymbol{\epsilon}$:

$$\mathbf{x} = \boldsymbol{\mu} + \mathbf{F}\mathbf{z} + \boldsymbol{\epsilon}, \tag{2.31}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ models between-class variability, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ models within-class variability, and $\mathbf{F}$ is a low-rank factor loading matrix.

During scoring, PLDA computes the log-likelihood ratio between the hypotheses that two embeddings come from the same class (i.e., the same language) versus different classes:

$$\text{score}(\mathbf{x}_1, \mathbf{x}_2) = \log \frac{p(\mathbf{x}_1, \mathbf{x}_2 \mid \text{same})}{p(\mathbf{x}_1 \mid \text{diff})\, p(\mathbf{x}_2 \mid \text{diff})}. \tag{2.32}$$

This scoring mechanism enables effective discrimination between languages by leveraging learned statistical differences in the embedding space.

Training Procedure Training the PLDA model involves estimating the parameters $\boldsymbol{\mu}$, $\mathbf{F}$, and $\boldsymbol{\Sigma}$ using an Expectation-Maximization (EM) algorithm applied to labeled embeddings $\{\mathbf{x}_{ij}\}$, where i indexes the language class and j indexes utterances within that class. The main computational steps are shown in Algorithm 4.

Algorithm 4: EM-based Latent Variable Estimation

Input: Embeddings $\mathbf{x}_{ij}$
Output: Optimized parameters $\boldsymbol{\mu}, \mathbf{F}, \boldsymbol{\Sigma}$

`// Step 1: Data Preparation`
$\mathbf{x}_{ij} \leftarrow \text{MeanCenter}(\mathbf{x}_{ij})$
$\mathbf{x}_{ij} \leftarrow \text{LengthNormalize}(\mathbf{x}_{ij})$
repeat
  `// Step 2: E-step`
  foreach *class* i **do**
    Compute posterior distribution of $\mathbf{z}_i$ using $(\boldsymbol{\mu}, \mathbf{F}, \boldsymbol{\Sigma})$
    Calculate posterior means $E[\mathbf{z}_i|\mathbf{x}_{ij}]$ and covariances $\text{Var}(\mathbf{z}_i|\mathbf{x}_{ij})$
  end
  `// Step 3: M-step`
  Collect sufficient statistics from E-step
  Update $\boldsymbol{\mu}, \mathbf{F}, \boldsymbol{\Sigma}$ to maximize expected complete-data log-likelihood
until *convergence or max iterations reached*

After training, the resulting PLDA model can be used for scoring embeddings to perform language verification or classification tasks.

Table 2.2: Overview of the LRE 17 language cluster and the languages

Language Cluster	Target Language
Arabic	Egyptian Arabic, Iraqi Arabic, Levantine Arabic, Maghrebi Arabic
Chinese	Mandarin, Min Nan
English	British English, General American English
Polish	Polish
Portuguese	Brazilian Portuguese
Russian	Russian
Spanish	Caribbean Spanish, European Spanish, Latin American Continental Spanish

2.2.3 Metric

The performance of our LID systems was evaluated using accuracy as the primary metric. Accuracy measures the proportion of correctly identified utterances out of the total number of utterances evaluated. It is defined as:

$$\text{Accuracy} = \frac{N_{\text{correct}}}{N_{\text{total}}}, \tag{2.33}$$

where N_{correct} is the number of utterances in which the predicted language matched the reference language (ground truth) and N_{total} is the total number of utterances evaluated. This metric provides a straightforward and interpretable measure of the performance of the LID system.

2.2.4 Datasets

Voxlingua107[7]: dataset consists of short speech segments automatically extracted and labeled from YouTube videos [Valk and Alumäe, 2021]. It consists of approximately 6,628 hours of automatically collected speech data in 107 languages, averaging 62 hours per language, and includes a development set – used as an evaluation set – of 1,609 verified utterances, which is approximately 4.5 hours. We used 2,000 utterances per language to train the different scoring methods – GLC, PLDA model – and perform an evaluation on the development set.

LRE17: The NIST Language Recognition Evaluation dataset consists of 14 languages and 3 parts: train, dev, and eval [Sadjadi et al., 2018]. The splits contain 2061 hours, 21 hours, and 236 hours of data respectively. Table 2.2 presents an overview of the languages and the various accents represented in the dataset. For evaluation with models trained on VoxLingua, the accents are grouped according to their respective language clusters, and the performance is reported per cluster. We filter out the LRE17 dev data to use only the audio files that are less than 100 seconds and use for enrollments and training the various scoring methods. On the LRE17 eval data, silence removal is applied on all audios that have length more than 100 seconds.

[7] https://cs.taltech.ee/staff/tanel.alumae/data/voxlingua107/

Table 2.3: Overview of the ATCO2-LID dataset.

	No. of utterances		Duration (minutes)	
Language	Dev	Eval	Dev	Eval
Czech	945	597	168	122
English	5358	3835	650	533
French	463	404	47	43
German	170	169	14	16

ATCO2-LID: dataset was collected as part of the ATCO2 project to develop and evaluate techniques for the classification of English/Non English, which are Czech and French and German [Szoke et al., 2021] which is publicly available[8]. This set is considered out-of-distribution data, as it belongs to a completely different domain and the amount of annotated data is limited. In addition, the speech data obtained from the air-traffic communication domain can be extremely noisy. The data annotated for LID is split between development (14.4 hours) and evaluation sets (11.6 hours). An overview of the data for each of these languages in development and evaluation is provided in Table 2.3. For each eval set, the corresponding dev data is used to train the PLDA classifier.

2.3 Pre-trained Models

Pre-trained models have revolutionized speech processing by leveraging large-scale unlabeled or weakly-labeled data to learn transferable representations for ASR and LID tasks. This section introduces the foundational models explored in this thesis, particularly the wav2vec 2.0 family and Whisper.

2.3.1 wav2vec2.0

The *wav2vec* family of models introduced by Facebook AI Research (FAIR) enabled the application of self-supervised learning techniques to raw audio waveforms. The original **wav2vec** model [Schneider et al., 2019] used a convolutional encoder to learn frame-level representations from raw waveforms, followed by context prediction objectives akin to word2vec in NLP.

Building upon this, the wav2vec 2.0 framework [Baevski et al., 2020] is a self-supervised model that learns contextualized representations directly from raw waveforms as shown in Figure 2.7. It comprises three components:

- A feature encoder made up of temporal convolutional layers that transforms raw audio $\mathbf{x}$ into latent speech representations $\mathbf{z}_t \in \mathbb{R}^d$.
- A quantization module that discretizes $\mathbf{z}_t$ into a finite set of codebook entries $\mathbf{q}_t$.
- A Transformer-based context network that produces contextual embeddings $\mathbf{c}_t$ from the masked latent features.

[8] https://www.atco2.org/data

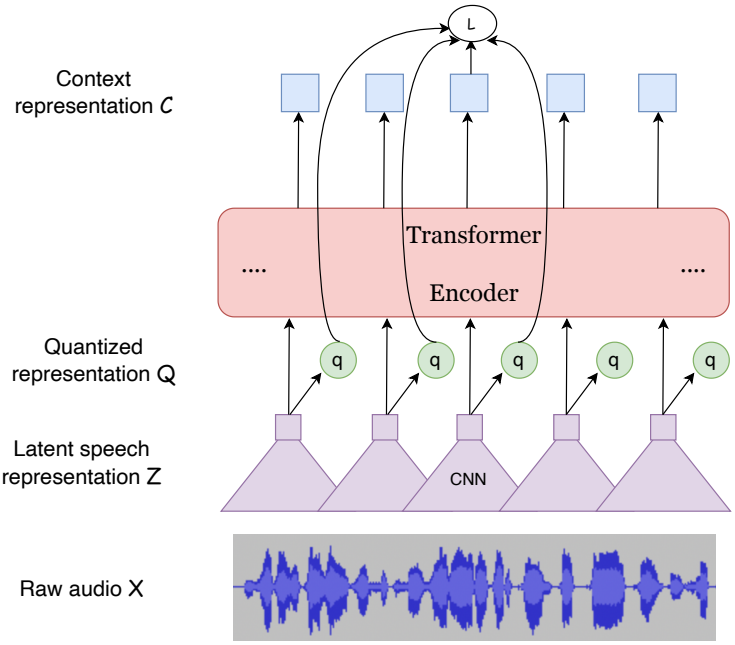

Figure 2.7: Overview of wav2vec2.0 architecture. This figure is adapted from [Baevski et al., 2020].

The model is trained by masking a portion of the latent speech features and then predicting the correct quantized target from a set of candidates using a contrastive loss:

$$\mathcal{L}_{\text{contrastive}} = -\log \frac{\exp(\text{sim}(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\mathbf{q}' \in \mathcal{Q}_t} \exp(\text{sim}(\mathbf{c}_t, \mathbf{q}')/\kappa)}, \tag{2.34}$$

where $\mathbf{c}_t$ is the contextualized embedding at masked time step t, $\mathbf{q}_t$ is the true quantized latent representation at time t, $\mathcal{Q}_t$ is the set containing $\mathbf{q}_t$ and K negative samples, $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, and κ is a temperature hyperparameter controlling the sharpness of the distribution.

To ensure all codebook entries are utilized and avoid codebook collapse, a diversity loss is added:

$$\mathcal{L}_{\text{diversity}} = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} p_{g,v} \log p_{g,v}, \tag{2.35}$$

where, G is the number of codebook groups, V is the number of entries in each group, and $p_{g,v}$ is the average probability of selecting code v from group g.

The total training objective is a weighted combination of both losses:

$$\mathcal{L} = \mathcal{L}_{\text{contrastive}} + \lambda \cdot \mathcal{L}_{\text{diversity}}, \tag{2.36}$$

where λ is a tunable hyperparameter controlling the influence of the diversity term.

XLSR [Conneau et al., 2020]: The XLSR (Cross-Lingual Speech Representation) model is a self-supervised framework designed to learn language-agnostic speech representations from raw audio. Building upon the wav2vec 2.0 architecture, XLSR employs a contrastive

learning objective over masked latent speech representations and jointly learns a quantization of these latents shared across languages. This approach enables the model to capture universal acoustic patterns, facilitating better performance in multilingual speech recognition tasks.

XLSR-53, a large-scale variant of this model, was trained on 56,000 hours of speech data spanning 53 languages. Training data was sourced from diverse multilingual datasets, including Multilingual LibriSpeech (MLS), CommonVoice, VoxPopuli, and BABEL. This extensive and diverse dataset allows XLSR-53 to learn robust and transferable speech representations, significantly improving performance in low-resource language settings and enabling a single model to perform well across multiple languages.

XLS-R [Babu et al., 2022] further scales this approach. It is trained on a large-scale multilingual corpus containing 436K hours of speech from 128 languages, comprising CommonVoice, MLS, BABEL, VoxPopuli, and other datasets. There are three variants of XLS-R, with 0.3B, 1B, and 2B parameters respectively. While maintaining the wav2vec 2.0 architecture, XLS-R increases model depth, width, and data diversity. This results in state-of-the-art performance in multilingual and zero-shot ASR benchmarks, making XLS-R one of the most powerful pre-trained speech models to date. In this thesis, we use the 0.3B version for both ASR and LID tasks.

2.3.2 Whisper

Whisper [Radford et al., 2023] is a transformer-based encoder-decoder model trained in a weakly supervised fashion using 680,000 hours of audio-text pairs scraped from the web. Unlike wav2vec 2.0 models, Whisper is trained directly on a multitask objective that includes transcription, translation, language identification, and timestamp prediction.

The model processes input audio by first computing log-mel spectrograms, which are passed to the encoder. A decoder then autoregressively generates output tokens, guided by task-specific control tokens such as `<|transcribe|>` or `<|translate|>`. This unified design allows Whisper to generalize across multiple tasks and languages within a single framework.

The model is trained using an autoregressive decoder that predicts each output token y_t given all previously generated tokens $y_{<t}$ and the input audio representation $\mathbf{x}$. The training objective is to minimize the negative log-likelihood or the cross-entropy loss between the predicted and target token sequences:

$$\mathcal{L}_{\text{Whisper}} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, \mathbf{x}), \tag{2.37}$$

where, $\mathbf{x}$ is the input log-Mel spectrogram extracted from the audio waveform, y_t is the target token at decoding time step t, $y_{<t}$ is the sequence of previous tokens (i.e., $y_1, y_2, \ldots, y_{t-1}$), and T is the total number of tokens in the output transcription or translation

Whisper excels in zero-shot and low-resource settings due to the scale and diversity of its training data. It shows robustness to noise, speaker variation, and unseen languages. Its open-domain generalization capability makes it suitable for real-world multilingual transcription, translation, and spoken language understanding. We use the 770M parameter 'medium' version of this model, which consists of 48 transformer layers and is multilingual.

Chapter 3

Baselines

This chapter establishes the baselines for both ASR and LID tasks. For ASR, we evaluate several models including a hybrid DNN-HMM system, as well as pre-trained Transformer-based models such as XLSR, XLS-R, and Whisper, with experiments conducted both with and without fine-tuning for Whisper on task-specific data. In the LID experiments, we employ embedding extractors including ECAPA-TDNN, XLSR, XLS-R, and Whisper, examining both pre-trained and fine-tuned versions of Whisper. The extracted embeddings are classified using a range of methods, specifically cosine distance, PLDA and GLC, to establish performance benchmarks across different classifier architectures.

3.1 Automatic Speech Recognition

3.1.1 AMI

For all our models in which training or fine-tuning is performed, we use 77 hours of training data described in Section 2.1.5 and apply a three-fold speed perturbation.
Kaldi s5b[1]: The AM with the best WER from the kaldi [Povey et al., 2011] recipe for AMI is trained using TDNN and LSTM architecture. 40-dimensional MFCCs and 100-dimensional i-vector is used for training, and the TDNN dimension used was 1024.

K2-Icefall: The results presented are based on the `k2/icefall` Zipformer recipe[2]. The model uses a Zipformer encoder paired with a simplified non-current decoder architecture. The decoder consists of an embedding layer, a convolutional layer with a kernel size of 2, and a linear output layer. To improve robustness, speed perturbation and MUSAN noise augmentation are applied during data preparation.

XLSR : The authors in [Vyas et al., 2021b] propose to use the LF-MMI criterion (similar to hybrid-based ASR) for the supervised adaptation of the self-supervised pre-trained models. Following the setup, our AM uses the pre-trained XLSR-53 model, followed by 3 Factorized Time-Delay Neural Network (TDNN-F) [Povey et al., 2018] layers. We use the PkWrap [Madikeri et al., 2020] toolkit for the TDNN-F layer. The AM model is trained on biphone units using the E2E-LFMMI criteria. We use Adam optimizer [Kingma and Ba, 2014] with a weight decay of 1e-2 and train for 21'000 steps. The learning rate is initially set to 3e-5 and with the learning rate scheduler tuned as follows: 10% warm-up, 40% constant learning rate and 50% decay. We use Espresso [Wang et al., 2019] toolkit with Fairseq [Ott

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/RESULTS_ihm
[2] https://github.com/k2-fsa/icefall/blob/master/egs/ami/ASR/RESULTS.md

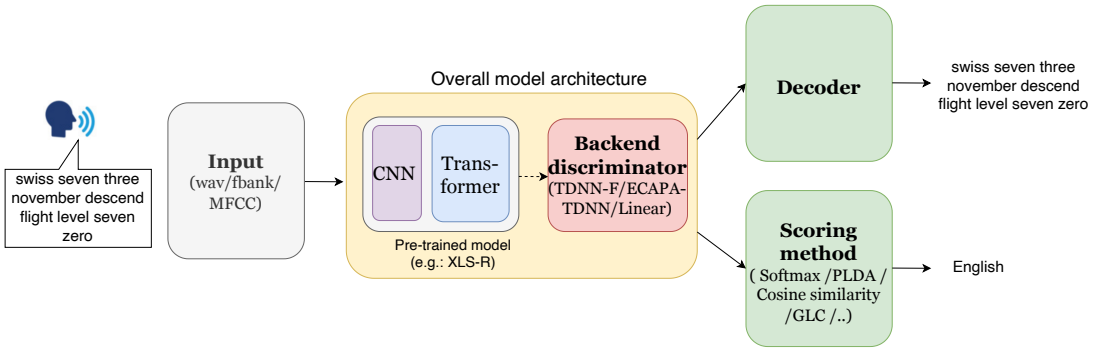

Figure 3.1: Overview of the overall E2E model architecture for ASR and LID used in this thesis.

et al., 2019] base for the calculation of the E2E-LFMMI loss that relies on PyChain [Shao et al., 2020] to implement the LFMMI loss. A 3-gram LM model is trained following the Kaldi s5b recipe for the AMI data set and the IHM microphone type. For decoding, the Kaldi Weighted FST (WFST) decoder is used with a beam width of 15.

XLS-R: The same experimental setup as described for XLSR is employed here; however, the pre-trained model used in this case is XLS-R. All training, fine-tuning, and evaluation procedures remain consistent to allow a fair comparison between the two models.

Whisper (zero-shot): The k2/Icefall framework[3] is used to decode the Whisper medium model without finetuning.

Whisper (fine-tuned): The k2/Icefall framework is used to finetune the Whisper medium. A learning rate of 1e-5 and label smoothing loss with smoothing factor 0.1 are used. The features are masked with SpecAug during training [Park et al., 2019]. AdamW [Loshchilov and Hutter, 2017] optimizer is used for all setups in this thesis. A lower-cased version of the transcripts are used for training. Finetuning is carried out for 10 epochs and the best model is chosen based on the WER on development set.

3.1.2 ATC

For all our models in which training or fine-tuning is performed, we use 190 hours of ATC training data described in Section 2.1.5 and apply a three-fold speed perturbation.
Kaldi: For the hybrid ASR based on DNN-HMMs, we use a conventional biphone model with 6 CNN [LeCun and Bengio, 1995] layers and 15 TDNN-F [Povey et al., 2018] layers. This was trained with the Kaldi [Povey et al., 2011] toolkit (that is, the nnet3 model architecture) with the LF-MMI training framework, which is considered to produce the state-of-the-art performance for hybrid ASR. A 40-dimensional MFCCs and 100-dimensional i-vector features are used. The LM is trained as a statistical 3-gram model using manual transcripts. The Kaldi WFST decoder is used for decoding.

XLSR: We use the same setup of XLSR as defined in Section 3.1.1 and train the AM for 14'000 updates. We also apply the three-fold speed perturbation. A 3-gram LM is trained using manual transcripts from 190 hours of data.

[3] https://github.com/k2-fsa/icefall

Table 3.1: ASR WER (%) on AMI dev and eval sets with various baselines defined in Section 3.1.1.

Model	# params	WER (%)	
	(M)	dev	eval
Kaldi	30	18.9	19.3
K2-Icefall	-	18.9	17.4
XLSR [Kumar et al., 2024]	320	14.6	12.6
XLS-R	320	14.3	12.4
Whisper (zero-shot)	769	23.3	22.9
Whiser (fine-tuned)	769	13.4	10.8

Table 3.2: ASR WER (%) on ATC eval sets with various baselines defined in Section 3.1.2.

Model	# params	WER (%)	
	(M)	NATS	ATCO2
Kaldi	30	6.5	29.1
K2-Icefall	66	6.5	17.6
XLSR	320	4.9	17.0
XLS-R	320	4.8	18.8
Whisper (zero-shot)	769	53.1	56.1
Whiser (fine-tuned)	769	6.9	18.6

XLS-R: The same experimental setup as described for XLSR is employed here; however, the pre-trained model used in this case is XLS-R.

Whisper (zero-shot): The k2/Icefall framework[4] is used to decode the Whisper medium model without finetuning.

Whisper (fine-tuned): The same experimental setup described in Section 3.1.1 is used with 190 hours of ATC data. We used the best model based on the WER on the NATS test set.

3.2 Language Identification

3.2.1 Embeddings models

ECAPA Model

The `SpeechBrain` [Ravanelli et al., 2021] `lang-id-voxlingua107-ecapa`[5] model is used as our baseline ECAPA model. It was trained on the **VoxLingua107** dataset, which covers 107 languages. It employs the **ECAPA-TDNN** architecture, originally developed for speaker recognition, enhanced with additional fully connected hidden layers after the

[4] https://github.com/k2-fsa/icefall

[5] https://huggingface.co/speechbrain/lang-id-voxlingua107-ecapa

Table 3.3: LID Accuracy (%) on Voxlingua107 dev, LRE17 eval and ATCO2-LID datasets evaluated on various baseline models and compared with various backend classifiers. **GLC**: Gaussian Linear Classifier, **PLDA**: Probabilistic Linear Discriminant Analysis model.

Model	**Scoring Method**	**# params (M)**	**Accuracy (%)**		
			Voxlingua dev	**LRE17**	**ATCO2**
ECAPA-TDNN	-	-	92.5	62.4	18.0
	GLC		84.7	61.6	75.2
	PLDA		94.6	42.5	81.3
XLSR	-	315	92.7	66.0	19.1
	GLC		93.1	70.8	87.6
	PLDA		95.8	75.0	87.4
XLS-R	-	315	92.7	70.5	23.3
	GLC		91.3	73.3	88.5
	PLDA		95.3	77.1	88.8
Whisper (No-FT)	-	776	92.6	86.1	56.4
Whisper	-		97.1	91.2	20.7

embedding layer. The model is trained using cross-entropy loss, enabling it to produce utterance level embeddings for downstream language identification tasks.

Due to its robust performance and open source availability in Hugging Face, this model is well suited for multilingual speech processing applications such as real-time translation, voice assistants, and other language identification tasks.

wav2vec2.0 models

For our baseline, we use the XLSR and XLS-R models described in Section 2.3.1. Both models are fine-tuned with the Voxlingua107 training set described in Section 2.2.4. The Espresso toolkit [Wang et al., 2019] which internally uses fairseq [Ott et al., 2019] is used with the loss of Cross-Entropy. The models are fine-tuned for 80'000 updates or 2 epochs.

As shown in Figure 4.1, the XLS-R model is used as an encoder and fine-tuned with LRE17 train data to extract embeddings. In addition, the PLDA [Ioffe, 2006, Prince and Elder, 2007] and GLC classifiers are trained with the dev set of each data set (depending on the evaluation condition) using the embeddings generated from the fine-tuned model.

3.2.2 Scoring Methods

Each of these baseline embedding extractors is evaluated using various scoring techniques. For the classification layer approach, the evaluation set is passed through the model, and predictions are obtained directly from its final softmax layer. For the other methods—GLC, and PLDA—embeddings are first extracted for both the development/training and evaluation sets. The development/training embeddings serve as enrollment data to compare against the evaluation utterances. In the case of GLC and PLDA, these embeddings are also used to train the model parameters.

Chapter 4

Parameter Efficient Backend for E2E Architecture

In this chapter, we present the motivation for using parameter-efficient backend layers in E2E architectures for LID. Specifically, we describe our approach that incorporates TDNN-F layers within LID models and introduce an ECAPA-TDNNF-based backend.

Fine-tuning large self-supervised, pre-trained models such as wav2vec 2.0 [Baevski et al., 2020] has been shown to significantly improve performance on downstream tasks including ASR [Conneau et al., 2020, Vyas et al., 2021c, Fan et al., 2020] and LID [Tjandra et al., 2022, Shi et al., 2023]. When adapting these models for ASR, a few feed-forward layers such as TDNN or TDNN-F are typically added prior to the final projection layer. For LID, a statistical pooling module [Snyder et al., 2018b] followed by a classification layer (e.g., softmax) is commonly used to convert frame-level outputs into utterance-level predictions.

Model architectures used for classification backends vary and include linear classifiers [Conneau et al., 2020], transformers [Vaswani et al., 2017a], TDNN, and ECAPA-TDNN [Desplanques et al., 2020]. Among these, transformer and ECAPA-TDNN architectures have demonstrated strong performance in data-rich settings. However, in low-resource scenarios—typical in language recognition evaluations [Sadjadi et al., 2018, Lee et al., 2022]—parameter-efficient discriminators are often preferred.

TDNN-F was introduced by Povey et al. [Povey et al., 2018] to reduce the parameter count of TDNN-based deep neural networks using semi-orthogonal low-rank matrix factorization. This approach achieved state-of-the-art results in Kaldi-based ASR systems. Based on this, TDNN-F layers were incorporated into wav2vec 2.0-based models, where they were placed before the final projection layer and fine-tuned using the LF-MMI criterion. This method showed improved performance for ASR [Vyas et al., 2021c]. Motivated by these findings, we explore the application of TDNN-F and ECAPA-TDNNF-style backends in LID systems to balance performance and parameter efficiency.

4.1 Automatic Speech Recognition

Speech can be viewed as a sequence-to-sequence mapping problem. Neural network architectures such as CNN and RNN were used to incorporate long temporal contexts. An important factor to be considered during the training of large neural networks is the parallelization of training so that we can use GPUs. One of the architectures effective for modeling temporal context dependencies is TDNN [Waibel et al., 1989] architecture. [Ped-

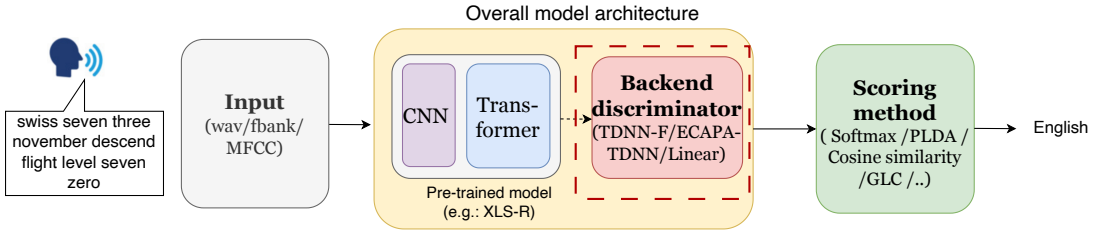

Figure 4.1: Overview of the overall architecture for language identification (LID). The embedding extractor is a pre-trained model (e.g., XLS-R) followed by backend discriminator such as TDNN-F, ECAPA-TDNN, or linear layers. At inference time, embeddings can be extracted and applied to various scoring methods such as PLDA, GLC, or cosine similarity for language prediction. The highlighted block – backend discriminator block represents the different architectures explored in this chapter.

dinti et al., 2015] showed that the TDNN architecture does not impose any relationship between the length of the input context and the number of sequential steps used during training. This allows to model training with long temporal contexts.

4.1.1 Time-Delay Neural Network

A TDNN models temporal structure by applying learned weights over a fixed-size temporal context from the preceding layer. For an output activation y_t at time t, a neuron in a TDNN layer computes its activation using inputs from specific time offsets d_i around t from the previous layer. This operation can be expressed as

$$y_t = \sigma\left(\sum_{i=0}^{C-1} \mathbf{W}_i \mathbf{x}_{t+d_i} + b\right), \tag{4.1}$$

where C is the size of the context window, d_i denote the predefined delays, $\mathbf{W}_i$ are the shared weight vectors for each delay, b is a bias term, and σ is a nonlinear activation function. As the weights $\mathbf{W}_i$ are shared across time, TDNNs learn translation-invariant feature detectors and model temporal dependencies efficiently.

In a TDNN, the lower layers operate in narrow temporal contexts, while the higher layers aggregate information over increasingly wider temporal spans, allowing the network to capture long-range dependencies. Each layer therefore uses a different temporal resolution, which increases with network depth. TDNNs are often viewed as 1D convolutional neural networks because their transforms are tied only along the time axis, and the shared parameters are updated using gradients accumulated over all time steps [Waibel et al., 1989].

At each layer, only specific time steps defined by the input context are used to compute the activations, as illustrated in Figure 4.2. For example, the input layer may consist of frames from $t-10$ to $t+10$, while the first hidden layer may use a temporal context of $[-1, 1]$, meaning that activation at time t depends on the frames $\{t-1, t, t+1\}$. As neighboring time steps have overlapping temporal contexts, many computations are redundant. To reduce this redundancy, sub-sampling is applied.

Sub-sampling introduces gaps between evaluated frames: instead of computing activations using $\{t-1, t, t+1\}$, only $\{t-1, t+1\}$ may be used. This reduces the number of time steps that must be evaluated, particularly in higher layers where the offset differences

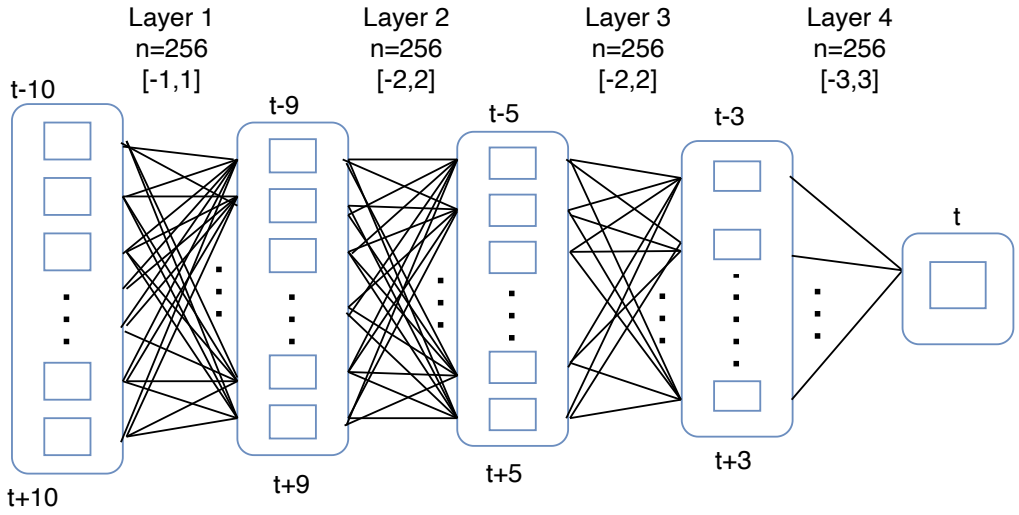

Figure 4.2: Block diagram showing the computation of output activations at each layer using temporal context in a TDNN.

are typically multiples of three. As a result, the computational load in both forward and backward passes is reduced, leading to faster training. Modern TDNN-based ASR systems typically use ReLU activation followed by batch normalization after each TDNN layer, which has been shown to improve performance.

4.1.2 Overview of TDNN-F

Singular Value Decomposition (SVD) is one of the most popular techniques that can be applied to a model to reduce its parameters. Another approach to enforce parameter reduction while training an AM is applying factorized low-rank layers [Povey et al., 2018]. In semi-orthogonal factorization, the parameter matrix $\mathbf{M}$ is factorized as a product of two matrices $\mathbf{A}$ and $\mathbf{B}$, i.e $\mathbf{M} = \mathbf{AB}$, where $\mathbf{B}$ is a semi-orthogonal matrix and $\mathbf{A}$ has a smaller "interior" (i.e. rank) than that of $\mathbf{M}$. The term "semi-orthogonal" refers to orthogonality of non-square matrices, i.e., a non square matrix $\mathbf{M}$ is semi-orthogonal if $\mathbf{M}\mathbf{M}^T = \mathbf{I}$ or $\mathbf{M}^T\mathbf{M} = \mathbf{I}$. This can be shown using SVD. Let the parameter matrix $\mathbf{M}$ have fewer rows than columns. In order for $\mathbf{M}$ to be orthogonal, it should satisfy $\mathbf{M}\mathbf{M}^T = \mathbf{I}$. Since we try to enforce a non-square matrix $\mathbf{M}$ to be orthogonal, $\mathbf{M}\mathbf{M}^T = \mathbf{P}$ is defined and the error is represented as $\mathbf{Q} = \mathbf{P} - \mathbf{I}$. In order to minimize the error $\mathbf{Q}$, a function f is defined as the sum of squared elements of $\mathbf{Q}$ which is represented as:

$$f = tr(\mathbf{Q}\mathbf{Q}^T). \tag{4.2}$$

Since $\mathbf{Q}$ is a matrix, the sum of squared elements is $tr(\mathbf{Q}\mathbf{Q}^T)$. f can be minimized by taking the derivative with respect to $\mathbf{Q}$. Since f in equation 4.2 will be a scalar, derivative of a scalar w.r.t. a matrix is not transposed w.r.t. that matrix (i.e. $\mathbf{Q}\mathbf{Q}^T$ will be considered as $\mathbf{Q}^2$). Hence

$$\frac{\partial f}{\partial \mathbf{Q}} = 2\mathbf{Q}. \tag{4.3}$$

Since $\mathbf{Q}$ is related to $\mathbf{P}$, the derivative of f w.r.t. $\mathbf{P}$ is:

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{P}} &= \frac{\partial \mathbf{Q}}{\partial \mathbf{P}} \\
&= \frac{\partial[(\mathbf{P}-\mathbf{I})(\mathbf{P}-\mathbf{I})^T]}{\partial \mathbf{P}} \\
&= \frac{\partial[\mathbf{P}\mathbf{P}^T - \mathbf{P}\mathbf{I}^T - \mathbf{I}\mathbf{P}^T + \mathbf{I}\mathbf{I}^T]}{\partial \mathbf{P}} \\
&= 2\mathbf{P} - \mathbf{I}^T - \mathbf{I} \\
&= 2(\mathbf{P}-\mathbf{I}) \\
&= 2\mathbf{Q}. \qquad (4.4)
\end{aligned}$$

Also, as we want to enforce M to be semi-orthogonal, the function f has to be minimized w.r.t. $\mathbf{M}$. The derivative of f w.r.t. $\mathbf{M}$ is the value that would be used to update the value of the parameter $\mathbf{M}$ during one iteration of SGD. A detailed derivation is as below:

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{M}} &= \frac{\partial \mathbf{Q}}{\partial \mathbf{M}} \\
&= \frac{\partial[(\mathbf{P}-\mathbf{I})(\mathbf{P}-\mathbf{I})^T]}{\partial \mathbf{M}} \\
&= \frac{\partial[\mathbf{P}\mathbf{P}^T - \mathbf{P}\mathbf{I}^T - \mathbf{I}\mathbf{P}^T + \mathbf{I}\mathbf{I}^T]}{\partial \mathbf{M}} \\
&= \frac{\partial[(\mathbf{M}\mathbf{M}^T)(\mathbf{M}\mathbf{M}^T)^T - (\mathbf{M}\mathbf{M}^T)\mathbf{I}^T - \mathbf{I}(\mathbf{M}\mathbf{M}^T)^T + \mathbf{I}\mathbf{I}^T]}{\partial \mathbf{M}} \\
&= 2\mathbf{M}(\mathbf{M}\mathbf{M}^T) + (\mathbf{M}\mathbf{M}^T)2\mathbf{M} - 2\mathbf{M}\mathbf{I}^T - 2\mathbf{M}\mathbf{I} \\
&= 4(\mathbf{M}\mathbf{M}^T - \mathbf{I})\mathbf{M} \\
&= 4\mathbf{Q}\mathbf{M}. \qquad (4.5)
\end{aligned}$$

From the above derivation we see that $\mathbf{M}$ is updated to $\mathbf{M} = \mathbf{M} - 4\nu\mathbf{Q}\mathbf{M}$, where ν is the learning rate. For quadratic convergence, the learning rate ν is chosen to be $1/8$. By substituting the value of learning rate and simplifying, we get

$$\begin{aligned}
\mathbf{M} &\leftarrow \mathbf{M} - 4(\frac{1}{8})\mathbf{Q}\mathbf{M} \\
\mathbf{M} &\leftarrow \mathbf{M} - \frac{1}{2}(\mathbf{M}\mathbf{M}^T - \mathbf{I})\mathbf{M}. \qquad (4.6)
\end{aligned}$$

If $\mathbf{M}$ is very far from being orthonormal, the above equation would not converge. Hence, in practice, Glorot (also known as Xavier) initialization [Glorot and Bengio, 2010] is used where the standard deviation of the random initial elements of $\mathbf{M}$ is the inverse of the square root of the number of columns.

In general, l_2 regularization is used (i) to control how fast the parameters of various layers change, and (ii) to reduce the scale of the parameters which helps the model to learn faster. Since the ReLU is scale invariant, l_2 regularization does not have an effect on how fast the parameters change when applied to the hidden layers. In order to have a consistent control of this, a scaling factor is added to equation 4.6. Substituting $\frac{1}{\alpha}\mathbf{M}$ and simplifying equation 4.6, a scaled version of the semi-orthogonal matrix is obtained:

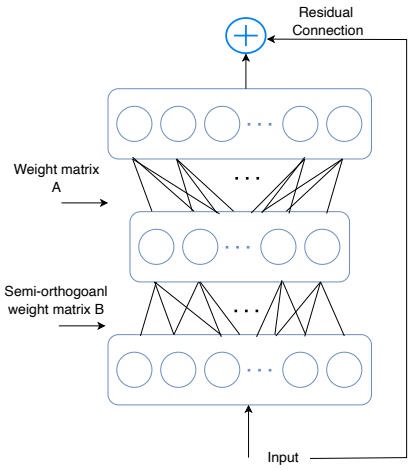

Figure 4.3: Block diagram of computation of the output activations for one layer TDNN-F.

$$\mathbf{M} \leftarrow \mathbf{M} - \frac{1}{2\alpha^2}(\mathbf{M}\mathbf{M}^T - \alpha^2\mathbf{I})\mathbf{M}. \tag{4.7}$$

The formula for α that ensures that the change in M is orthogonal to itself is given as:

$$\alpha = \sqrt{\frac{tr(\mathbf{P}\mathbf{P}^T)}{tr(\mathbf{P})}}. \tag{4.8}$$

The above method is applied to an existing topology such as TDNN and factorizes its parameter matrices into products of two smaller matrices. For example, if the TDNN model architecture has a hidden layer dimension of 625, then a typical parameter matrix would have a dimension of 625×1875. Here the number of columns (1875) corresponds to the three frame offsets of previous layers spliced together i.e., it can be viewed as a CNN with a 3×1 kernel and 625 filters. So the idea of factorization is to factorize this 625×1875 matrix $\mathbf{M}$ into two matrices as $\mathbf{M} = \mathbf{AB}$ with a smaller "interior" dimension (also called linear bottleneck dimension) d of 160 for example. This factorization will make $\mathbf{A}$ of size 625×160 and $\mathbf{B}$ of size 160×1875 with $\mathbf{B}$ constrained to be semi-orthogonal. This is shown in Figure 4.3. The number of parameters $625 \times 1875 = 1.17M$ in the TDNN model is reduced to $625 \times 160 + 160 \times 1875 = 0.4M$ in the TDNN-F model.

Thus, this technique enables training a smaller network from scratch instead of using a pre-trained network for parameter reduction. The LF-MMI training also provides a stable training procedure for semi-orthogonalized matrices.

4.1.3 Experiments

The authors in [Prasad, 2020] demonstrate the performance differences between TDNN and TDNN-F models for ASR on the LibriSpeech dataset, including the substantial reduction in parameter count achieved by replacing TDNN with TDNN-F when trained using the Kaldi toolkit. In this thesis, we present ASR performance on the AMI dataset using an XLSR-based model and compare it against TDNN and TDNN-F architectures trained with the Fairseq toolkit. The results are summarized in Table 4.1, where we also report the

Table 4.1: ASR WER (%) on AMI dev and eval sets with TDNN and TDNN-F configurations.

Model	Backend Config	# params (M)	WER (%) dev	WER (%) eval
XLS-R	TDNN	3.2	14.3	12.3
	TDNN-F	0.8	14.3	12.4

number of parameter for the TDNN and TDNN-F layers only. We observe that the number of parameters are reduced by $\approx 75\%$.

4.2 LID

We propose to apply Factorized TDNN models for LID, resulting in two classification modules: (1) TDNN-F, and (2) ECAPA-TDNN-F, where TDNN-F replaces the TDNN components in ECAPA-TDNN achieving parameter reduction up to 50%. In addition, we propose to exploit the orthonormal constraint employed in the TDNN-F architecture for the linear classifier mentioned earlier. The constraint alleviates the strong diagonal covariance assumptions in the statistics pooling layer, demonstrated with x-vector based systems [Snyder et al., 2018a].

We compare the performance of the XLS-R model [Babu et al., 2022] under low-resource and out-of-domain data conditions for LID. The model is fine-tuned with the Voxlingua107 dataset, and evaluated on Voxlingua107 dev and two unseen and out-of-distribution sets: LRE17 and air-traffic communication data[1] described in Section 2.2.4. Our experiments show that the Orthonormal linear discriminator performs the best on all test sets.

Section 4.2.1 describes the baseline approaches for fine-tuning pre-trained models for LID. In Section 4.2.2, we described the proposed approaches that apply TDNN-F and Orthonormal constraints to backend discriminators. Training, development, and test sets used in the experiments are described in Section 2.2.4. In addition to the previous scoring methods described in Section 2.2.2, we also explore Normalizing flow, which does not make the Gaussian assumption. This is described in Section 4.2.3 The results are presented in Section 4.2.4.

4.2.1 Fine-tuning XLS-R for LID

In our experiments we fine-tune the pre-trained 300M parameter version of the XLS-R [Babu et al., 2022] model[2] using the espresso [Wang et al., 2019] toolkit which internally uses fairseq [Ott et al., 2019]. As shown in Figure 4.1, the XLS-R model is employed as an encoder and fine-tuned with Voxlingua107 train set or LRE17 train data. During evaluation, as shown in Figure 4.1, the output from the backend discriminator is used for various scoring methods. For the scoring methods such as PLDA, GLC data set dependent dev set embeddings are also extracted in addition to its evaluation set. As a part of our baseline

[1] https://atco2.org/

[2] https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec/xlsr

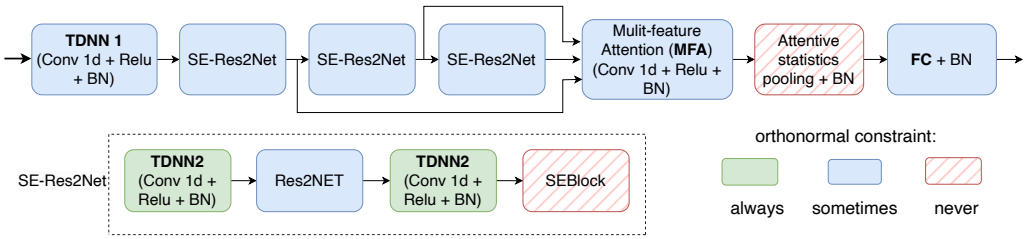

Figure 4.4: Overview of the ECAPA-TDNN model architecture. Different colours indicate the possibility of applying the orthonormal constraint in the ECAPA-TDNN-F model for each block. Green: orthonormal constraint is always applied. Blue: orthonormal constraint is applied in certain configurations. Shaded Red: the constraint is not applied.

systems, we use the following backend discriminator architectures: linear, ECAPA-TDNN, and Transformer. Each of these approaches are described below:

Linear layer: A linear layer of output dimension 512 is added prior to statistics pooling.

ECAPA-TDNN [Desplanques et al., 2020]: In language identification and speaker verification tasks, the ECAPA-TDNN model has been shown to provide state-of-the-art performance. We use the SpeechBrain implementation [Ravanelli et al., 2021] along with their default configuration.

Transformer Layer [Vaswani et al., 2017a]: a single transformer layer with the same configuration as other transformer layers in the XLS-R model is added after the XLS-R encoder.

In order to generate the audio-level embeddings, we employ statistics pooling mechanism for all the components mentioned above. The embeddings generated from the fine-tuned models is then used for the PLDA and scored using the Kaldi [Povey et al., 2011] toolkit.

4.2.2 Orthonormally constrained backend discriminators

As described above in Section 4.1.2, each TDNN-F layer reduces the number of parameters in the TDNN module by applying the orthonormal constraint (inspired by SVD). It is easy to see that the parameter reduction can be obtained by setting a low value for d. In the Kaldi implementation, the value for d is typically 160. We used the Pytorch implementation of this constraint from Pkwrap [Madikeri et al., 2020], which applies the constraint in every training iteration (in the original implementation in Kaldi, the parameters are updated with probability 0.25). The learning rate is updated accordingly.

We propose using three backend discriminators based on the orthonormal constraint:
Orthonormal Linear layer: The linear layer of an output dimension 512 is subject to orthonormal constraint. Note that there is no parameter reduction in this case. The orthonormal transform is only intended to compute diagonal covariance in the statistics pooling layer. This operation was typical in x-vector-based systems [Povey et al., 2011].
ECAPA-TDNN-F: Figure 4.4 shows the architecture of the ECAPA-TDNN model. We apply an orthonormal constraint to the Conv1d layers of the model. Since we use the implementation from Speechbrain (including hyper-parameter settings), we override the Conv1d to recast the weights from $D_1 \times D_2 \times k$ to first $D_1 \times k \times D_2$ and then to $D_1 \times kD_2$. The orthonormal constraint is then applied to this weight matrix. Here, D_1 is the number of out channels, D_2 is the number of in channels, and k is the kernel size. We experiment with the

application of the parameter reduction in different components in the model architecture. **TDNN-F**: We also experiment with the TDNN-F architecture following its use in ASR with the XLSR LF-MMI architecture [Vyas et al., 2021c]. A context size of 3 is used for all the layers, with bottleneck dimension 160 and layer dimension 1024.

4.2.3 Scoring Methods

As mentioned earlier, we use different scoring methods to detect the language from an embedding extractor. In this section, we describe how a non-linear model, specifically Normalizing Flows, can be used for scoring. Other scoring approaches, such as PLDA and GLC, are discussed in Section 2.2.2.

a. Normalizing flows

Flows are a broad class of machine learning algorithms that leverage the transformation theorem [Gut, 2009, Chapter 1.2.1] to model complex probability distributions. They achieve this by applying a series of invertible transformations to latent variables drawn from a known base distribution [Kobyzev et al., 2021, Papamakarios et al., 2021]. Although normalizing flows initially gained popularity in the context of generative modeling [Theis et al., 2016], their ability to represent explicit probability distributions also makes them theoretically applicable to tasks such as classification and regression.

The Density

According to PLDA, observations that have been generated from the same latent variable $\boldsymbol{v}$ are dependent and are considered to belong to the same class. The joint distribution of a set of dependent observations $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\theta)$ depends on the parameters θ. They are the matrix $\mathbf{A}$, the intercept $\boldsymbol{m}$, and the diagonal covariance matrix $\boldsymbol{\Psi}$. The optimal θ is found by maximizing the likelihood of the entire dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N) = \sum_{k=1}^{K} \log p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \theta), \tag{4.9}$$

where K is the total number of classes, N is the total number of observations, and $\mathcal{C}_k$ is the set of indices of observations that belong to class k. As PLDA only applies a linear transformation, Equation 4.9 can be maximized analytically if all classes have the same number of observations, or by using EM if this is not the case.

Now, instead of applying a linear transform $\boldsymbol{x} = \mathbf{A}\boldsymbol{u} + \boldsymbol{m}$, we are defining a parameterized, invertible, and non-linear transformation $\boldsymbol{x} = g(\boldsymbol{u}, \lambda)$. Transformation g involves a new set of parameters λ and has an inverse $h = g^{-1}$. At the same time, we maintain the assumptions of PLDA regarding the latent variables, i.e., $p(\boldsymbol{u}|\boldsymbol{v}) = \mathcal{N}(\boldsymbol{v}, \mathbf{I})$ and $p(\boldsymbol{v}|\boldsymbol{\Psi}) = \mathcal{N}(0, \boldsymbol{\Psi})$. Consequently, we have defined a new model with parameters λ and $\boldsymbol{\Psi}$. The remaining PLDA parameters $\mathbf{A}$ and $\boldsymbol{m}$ have disappeared. In order to train this model via maximum likelihood, we need to find how it represents the joint distribution $p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \lambda, \boldsymbol{\Psi})$ of a set of dependent observations that belong to a class k. In the following derivation, for simplicity, we use the notation $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\lambda, \boldsymbol{\Psi})$ to refer to $p(\boldsymbol{x}_i : i \in \mathcal{C}_k \mid \lambda, \boldsymbol{\Psi})$.

From PLDA, we already know which is the joint distribution $p(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n|\boldsymbol{\Psi})$ of a set of latent variables that belong to the same class [Ioffe, 2006, Equation 6]. If we now transform each variable with $\boldsymbol{x}_i = g(\boldsymbol{u}_i, \lambda)$ with inverse $\boldsymbol{u}_i = h(\boldsymbol{x}_i, \lambda)$, the transformation

theorem claims that the new joint distribution $p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n|\lambda, \boldsymbol{\Psi})$ takes the following form:

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n|\lambda, \boldsymbol{\Psi}) = p(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots \boldsymbol{u}_n|\boldsymbol{\Psi}) \prod_{i=1}^{n} |\frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i}|, \tag{4.10}$$

where $|\frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i}|$ represents the absolute value of the determinant of the Jacobian $\frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i}$. It is not difficult to prove that if we choose $\boldsymbol{x} = g(\boldsymbol{u}) = \mathbf{A}\boldsymbol{u} + \boldsymbol{m}$, we obtain the PLDA distribution. Therefore, our equations are a general case of the PLDA model. We can take the logarithm of Equation 4.10 to obtain an explicit expression for the log-likelihood:

$$\begin{aligned} \log p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\lambda, \boldsymbol{\Psi}) &= -\frac{nd}{2}\log(2\pi) \\ &+ \sum_{i=1}^{n} \log \left| \frac{\partial h(\boldsymbol{x}_i, \lambda)}{\partial \boldsymbol{x}_i} \right| - \frac{1}{2}\sum_{t=1}^{d} \log(\Psi_t + \frac{1}{n}) \\ &- \frac{1}{2}\sum_{t=1}^{d} \frac{\bar{u_t}^2}{(\Psi_t + \frac{1}{n})} - \frac{1}{2}\sum_{i=1}^{n}\sum_{t=1}^{d}(u_{it} - \bar{u}_t)^2. \end{aligned} \tag{4.11}$$

The data dimensionality is d and the scalar u_{it} is just a component t of the vector $\boldsymbol{u}_i = h(\boldsymbol{x}_i, \lambda)$. We also denote by $\bar{\boldsymbol{u}} \coloneqq \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{u}_i$ the average of all vectors $\boldsymbol{u}_i$ and $\bar{u}_t$ is the component t of vector $\bar{\boldsymbol{u}}$.

Our density assumes that the parameterized transformation $\boldsymbol{u} = h(\boldsymbol{x}, \lambda)$ generates latent variables of the same class that are normally distributed with unit variance around the center $\bar{\boldsymbol{u}}$. Moreover, the center $\bar{\boldsymbol{u}}$ is normally distributed with zero mean and covariance matrix controlled by the diagonal matrix $\boldsymbol{\Psi}$ and the number of observations within the class, just like in PLDA.

The Flow Transformation

The transformation $\boldsymbol{u} = h(\boldsymbol{x}, \lambda)$ that we use is called the coupling layer [Dinh et al., 2017]. It is a non-volume-preserving transformation that transforms the vector $\boldsymbol{x} \in \mathcal{R}^d$ into a new vector $\boldsymbol{u} \in \mathcal{R}^d$. It divides the vector $\boldsymbol{x}$ into two halves: $\boldsymbol{x}_1 \in \mathcal{R}^{\frac{d}{2}}$ and $\boldsymbol{x}_2 \in \mathcal{R}^{\frac{d}{2}}$. The first half $\boldsymbol{x}_1$ remains unmodified, forming $\boldsymbol{u}_1 \coloneqq \boldsymbol{x}_1$. The second half is transformed according to:

$$\boldsymbol{u}_2 \coloneqq (\boldsymbol{x}_2 - \boldsymbol{t}) \exp(-\boldsymbol{s}), \tag{4.12}$$

where variables $\boldsymbol{s}$ and $\boldsymbol{t}$ have been produced by a neural network function f that receives $\boldsymbol{x}_1$ as input: $\boldsymbol{s}, \boldsymbol{t} = f(\boldsymbol{x}_1, \lambda)$. The vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are concatenated to form the new vector $\boldsymbol{u}$. It is easy to check that this transformation is non-linear and invertible. The parameters λ of the coupling layer transformation are the parameters of the neural network f that has been used. The most common choice is to use a CNN [LeCun and Bengio, 1995].

The overall flow model architecture

Because the composition of invertible transformations is still an invertible transformation, we can create a powerful non-linear function by composing several coupling layers. Mathematically, we define $h(\boldsymbol{x})$ as $h(\boldsymbol{x}) = h_N \circ h_{N-1} \circ \ldots \circ h_1$, where $h_1, h_2, \ldots, h_N$ are N coupling layer transformations and $\circ$ denotes the function composition.

When composing multiple layers, it is important to vary the way in which the vector $\boldsymbol{x}$ is split at each layer. This is done to ensure that all components of the vector $\boldsymbol{x}$ are modified after the full transformation. To achieve this, we follow the simple approach of

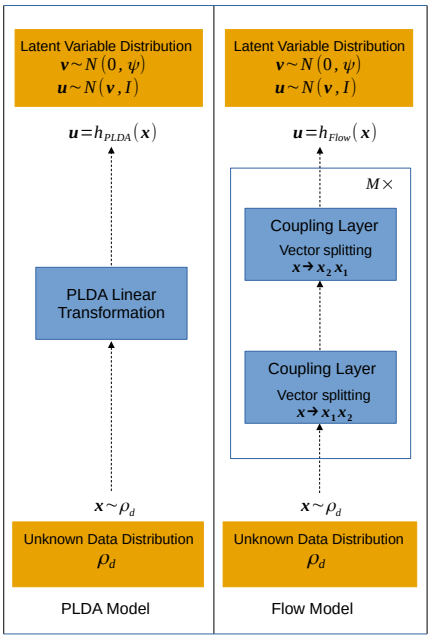

Figure 4.5: Overview of the PLDA and flow model scoring methods in SRE and LRE. PLDA receives a sample $\boldsymbol{x}$ from the unknown data distribution ρ_d, applies a linear transformation $h_{\text{PLDA}}(\boldsymbol{x})$ and assumes the resulting latent variable $\boldsymbol{u}$ belongs to a Gaussian. The flow model instead performs multiple non-linear transformations on the data sample.

just splitting in half the vector $\boldsymbol{x}$ and inverting the splitting at each layer. We are aware that there are more clever ways of achieving this, such as introducing additional invertible layers based on rotation matrices, as done in [Kingma and Dhariwal, 2018]. In this study, we decide to start with the simplest possible flow. A summary of the ideas presented is illustrated in the block diagram of Figure 4.5, which also establishes a comparison between the flow and PLDA architectures.

Finally, each coupling layer has its own CNN network with a well defined goal: it must receive one half of the vector $\boldsymbol{x} \in \mathcal{R}^d$, namely $\boldsymbol{x}_1 \in \mathcal{R}^{\frac{d}{2}}$, and it must output the scale and intercept vectors $\boldsymbol{s}$, $\boldsymbol{t}$ that are needed in Equation 4.12. Our implementation achieves this by using a CNN that starts with a linear transform from $\frac{d}{2}$ to d units. It then uses 1D convolutions with padding that preserve the dimensions of d during forward propagation. The network f returns a vector $f(\boldsymbol{x}_1) \in \mathcal{R}^d$ that is split into two to obtain $\boldsymbol{s}$ and $\boldsymbol{t}$.

Optimization

The model parameters that need to be optimized are the covariance matrix $\boldsymbol{\Psi}$ and the CNN parameters λ of each coupling layer. In this work, we take a simplified approach and first train a PLDA model via EM. Then, we take the optimal matrix $\boldsymbol{\Psi}^*$ and fix it for our flow model.

The optimization method that we use for learning λ is the stochastic gradient descent. We randomly select a batch of data and perform an update in order to maximize its likelihood. We repeat the procedure until the likelihood saturates. More specifically, we randomly select B classes $k_1, k_2, \ldots, k_B$. Our batch loss $l(\lambda, \boldsymbol{\Psi}^*)$ is the Negative Log-Likelihood

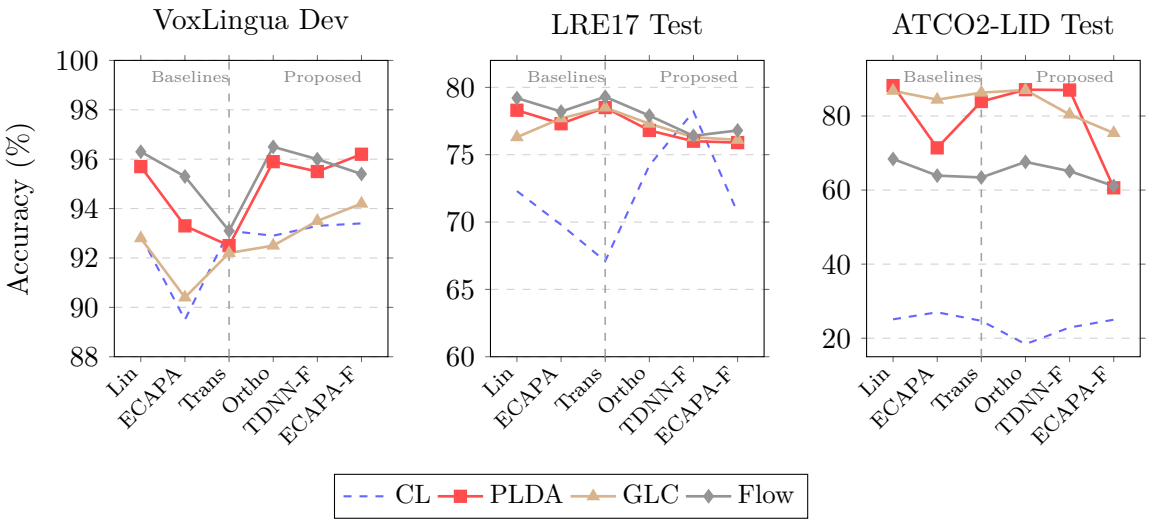

Figure 4.6: LID Accuracy (%) comparison of backend discriminator configurations across three test sets and scoring methods. **Baselines**: **Lin** = Linear layers, **ECAPA** = ECAPA-TDNN, **Trans** = Transformer layer. **Proposed**: **Ortho** = Orthonormal linear, **TDNN-F** = TDNN-F, **ECAPA-F** = ECAPA-TDNN-F. Scoring methods: **CL** = final classification layer (no backend scoring), **PLDA** = Probabilistic Linear Discriminant Analysis, **GLC** = Gaussian Linear Classifier, **Flow** = Normalizing Flows.

(NLL) of the data involved in the B classes:

$$l(\lambda, \boldsymbol{\Psi}^*) = -\frac{1}{B}\sum_{b=1}^{B} \log p(\boldsymbol{x}_i : i \in C_{k_b} | \lambda, \boldsymbol{\Psi}^*). \tag{4.13}$$

Evaluation

It is straightforward to do verification with the model. Verification consists in receiving a vector $\boldsymbol{x}^{\text{enroll}}$ and a vector $\boldsymbol{x}^{\text{test}}$ and computing the log-likelihood ratio between the target and non-target hypotheses. Following our independence assumptions, the ratio takes the following form:

$$R(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}}) := \log \left(\frac{p(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}} | \lambda, \boldsymbol{\Psi})}{p(\boldsymbol{x}^{\text{enroll}} | \lambda, \boldsymbol{\Psi}) p(\boldsymbol{x}^{\text{test}} | \lambda, \boldsymbol{\Psi})} \right). \tag{4.14}$$

At this point, we notice that when using Equation 4.10 in order to compute Equation 4.14, we keep a desirable property of PLDA: there is no need to compute the Jacobians, as they cancel each other out in the fraction. This means that our model only requires evaluating the likelihood of the Gaussian latent variables, obtaining the PLDA equation:

$$R(\boldsymbol{x}^{\text{enroll}}, \boldsymbol{x}^{\text{test}}) = \log \left(\frac{p(\boldsymbol{u}^{\text{enroll}}, \boldsymbol{u}^{\text{test}} | \boldsymbol{\Psi})}{p(\boldsymbol{u}^{\text{enroll}} | \boldsymbol{\Psi}) p(\boldsymbol{u}^{\text{test}} | \boldsymbol{\Psi})} \right), \tag{4.15}$$

where $\boldsymbol{u}^{\text{enroll}} = h(\boldsymbol{x}^{\text{enroll}}, \lambda)$ and $\boldsymbol{u}^{\text{test}} = h(\boldsymbol{x}^{\text{test}}, \lambda)$.

4.2.4 Experiments

In our experiments, we fine-tune the XLS-R model with (i) the Voxlingua107 and (ii) LRE17 training set with a learning rate of 3e-05 for 42'000 steps with a batch size of 8

Table 4.2: Results of Voxlingua dev, LRE17 and ATCO2-LID test sets evaluated with different backend discriminators and scoring methods. The number of parameters is measured only on the backend discriminator. All systems are trained on the Voxlingua107 training set, but dataset-specific PLDA models are trained for LID. **GLC**: Gaussian Linear Classifier, **PLDA**: Probabilistic Linear Discriminant Analysis model. **Flow**: Normalizing Flows. *: for the configuration of ECAPA-TDNN-F models please refer to Table 4.4. The results show that the backend discriminator with only Linear Layer or Orthonormal Linear consistently outperforms other baselines in two out of three conditions.

	Backend Discriminator	**Scoring Method**	**# params (M)**	**Accuracy (%)**		
				Voxlingua dev	**RE17 test**	**ATCO2-LID test**
Baselines	Linear layers	-	0.5	92.9	72.3	25.1
		PLDA		95.7	78.3	**88.2**
		GLC		92.8	76.3	86.8
		Flow		96.3	79.2	68.4
	ECAPA-TDNN	-	10.0	89.5	69.8	27.0
		PLDA		93.3	77.3	71.4
		GLC		90.4	77.7	84.4
		Flow		95.3	78.2	63.9
	Transformer layer	-	9.0	93.1	67.1	24.7
		PLDA		92.5	78.5	83.9
		GLC		92.2	78.5	86.3
		Flow		93.1	**79.3**	63.4
Proposed	Orthonormal linear	-	0.5	92.9	74.2	18.4
		PLDA		95.9	76.8	87.1
		GLC		92.5	77.3	87.0
		Flow		**96.5**	77.9	67.6
	TDNN-F	-	2.0	93.3	78.2	22.9
		PLDA		95.5	76.0	87.0
		GLC		93.5	76.3	80.4
		Flow		96.0	76.4	65.1
	ECAPA-TDNN-F*	-	5.0	93.4	70.7	25.0
		PLDA		96.2	75.9	60.6
		GLC		94.2	76.1	75.4
		Flow		95.4	76.8	61.1

Table 4.3: Results of LRE17, ATCO2 test sets evaluated with different backend configurations. The number of parameters is measured only on the backend discriminators. All systems are trained on the LRE17 training set, but dataset-specific PLDA models are trained for LID. * refers to the differnt configuration of ECAPA-TDNN-F models defined in Table 4.4. The results show that the backend discriminator with only Orthonormal Linear layer consistently outperforms other baselines.

	Backend Discriminator	**# params (M)**	**Accuracy (%)**	
			LRE17 test	**ATCO2-LID test**
Baselines	Linear layers	0.5	83.1	90.1
	ECAPA-TDNN	10.0	80.4	87.7
	Transformer layer	9.0	81.8	90.5
Proposed	Orthonormal linear	0.5	**83.7**	**91.0**
	TDNN-F	2.0	83.0	85.9
	ECAPA-TDNN-F*	7.0	81.7	87.4
	ECAPA-TDNN-F*	5.0	80.8	87.2
Other baselines	x-bLSTM [Padi et al., 2020]	-	68.7	-
	Whisper (medium)	-	-	56.4

along with the cross-entropy loss. The backend discriminators are trained with a gradient multiplier 20. For the model fine-tuned with Voxlingua107 train set, we evaluate it on Voxlingua107 dev, LRE17 test and ATCO2-LID tests. For the model fine-tuned with LRE17 train set, we evaluate it on LRE17 test and ATCO2-LID tests only as not all languages from Voxlingua107 dev set are present in the LRE17 languages. We report the LID Accuracy(%) in all of our experiments.

Table 4.2 and Figure 4.6 presents the results of using different backend discriminator architectures trained with Voxlingua107 training set. It is evaluated on *VoxLingua107 dev*, *LRE17* test, and *ATCO2-LID* test sets. In addition to studying various backend discriminators, we present results with different scoring methods. For PLDA, GLC and Flow models the embeddings are extracted after the backend discriminator, apply statistics pooling to convert frame level embeddings to an utterance level embedding. For each dataset, their respective development set embeddings are used to train the different scoring methods.

The results show that the backends with only linear layers (Linear Layer in baseline and Orthonormal Linear in proposed methods) discriminator consistently outperforms other baselines in 2 out 3 conditions. The Linear Layer achieves the highest performance for the low-resource ATCO2-LID dataset and competitive result for LRE17. With only 0.5 million parameters, these linear layers achieves competitive accuracy compared to the much larger baselines, such as ECAPA-TDNN (10.0M parameters) and the Transformer layer (9.0M parameters), particularly on the challenging ATCO2 dataset where it reaches 88.2% accuracy with PLDA scoring.

For the in-domain *VoxLingua107* set, we observe that the **Flow** model consistently provides higher accuracy for all backend configurations. This suggests that Normalizing Flows are highly effective when the test distribution matches the training distribution. For the *LRE17* data set, the **Flow** provides the highest performance in all cases. However, for

the *Linear layer* discriminator, Flow achieves the same accuracy (79.2%) compared to a Transformer backend. We observe the Flow model is unable to model the parameters which results in the degradation of the accuracy for this out-of-domain set

While **PLDA** and **GLC** provide better accuracy on the *ATCO2-LID* dataset, the **Flow** model exhibits a significant degradation in performance ($\approx$ 60–68% vs. $\approx$ 80–88%). This degradation can be attributed to 2 factors: (i) the difficulty of modeling noisy, low-resource speech, and (ii) the Flow model's hyperparameters, especially the number of coupling layers are not fully optimized for this specific domain.

To further validate our findings, we trained the backend discriminators on the LRE17 dataset and evaluated them on the LRE17 and ATCO2-LID test sets. The VoxLingua107 development set was excluded from this evaluation because it contains languages that are not present in the LRE17 training data. The results are presented in Table 4.3 with PLDA scoring method. The results show that systems using only linear layers (Linear layer in baseline, Orthonormal linear in proposed methods) consistently outperforms other baseline backend discriminator architectures. This could be attributed to the low-resource nature of the fine-tuning set. Moreover, the linear layers generalize better to out of domain sets. The Orthonormal linear performs better in two out of three conditions compared to the linear layer, with absolute improvement in accuracy up to 0.9% (relative improvement in error of 9%). Although ECAPA-TDNN and transformer backends provide competitive results, the number of parameters during fine-tuning increases significantly (from 8 to 10 times).

The TDNN-F model outperforms the ECAPA-TDNN and Transformer models on only the seen condition set with improvements in accuracy up to 2.6% (13.2% relative improvement in error rate), indicating its tendency to overfit. We also compared the LRE17 with the baseline x-vector system presented in [Padi et al., 2020] and observed the increase in the accuracy by 9.2% absolute. On the ATCO2 set, the accuracy was 91% compared to the Whisper model [Radford et al., 2023] which provides an accuracy of 56.4%.

Next, we compare the results of ECAPA-TDNN-F and ECAPA-TDNN in Table 4.3. The results show that the performance of ECAPA-TDNN can be preserved even after reducing the parameters by 30%, and when reduced drastically by 50% we observe a relative degradation in accuracy by 0.5% for ATCO2. Unlike TDNN-F, the ECAPA-TDNN-F maintains its performance on unseen conditions compared to ECAPA-TDNN.

Conclusion: From the overall results, it is evident that lightweight or small backend discriminators like Linear or Orthonormal linear are sufficient to obtain high performance. These compact models can match or even outperform significantly heavier architectures (e.g., those with 9.0M or 10.0M parameters), making them highly effective for practical language identification tasks. We also observe that the Flow models provides the highest performance for in-domain and out-of-domain sets compared to PLDA. However, for low-resource data, we observe huge degradation in performance for Flow models compared to PLDA or GLC.

Table 4.4 shows the effect of applying TDNN-F layers to various TDNN modules in ECAPA-TDNN model. Specifically, we study the effect of replacing with TDNN-F components. Our results indicate that the replacement is effective (i.e., reduces parameters without significantly affecting performance), as long as the final FC layer is not parameter deficient. Our results opens up new avenues of exploring the use of ECAPA-TDNN-F in tasks where ECAPA-TDNN has been shown to be effective. The severe degradation for ATCO2 when using TDNN-F for the FC component can be attributed to the size (10x smaller than LRE17) and noisy nature of the dataset.

Table 4.4: Results of LRE17 and ATCO2 test sets evaluated with different bottleneck configuration for ECAPA-TDNN-F. All systems are trained on the LRE17 train set, but dataset-specific PLDA models are trained for LID. A bottleneck dimension of 16 is used for Res2Net and 128 for the other blocks. See Figure 4.4 for the module details. The results indicate that replacing TDNN by TDNN-F reduces parameters without significantly affecting performance.

Constrained	**# params**	**Accuracy (%)**	
module	**(M)**	**LRE17 test**	**ATCO2-LID test**
-	10.0	80.4	87.7
TDNN2	9.3	80.8	87.6
+ Res2Net	9.0	80.0	87.7
+ TDNN1	7.0	81.7	87.4
+ MFA	5.0	80.8	87.2
+ FC	4.0	77.5	63.0

Chapter 5

Parameter Efficient Fine-Tuning

In the evolving field of artificial intelligence, neural networks have become the cornerstone of various state-of-the-art applications, ranging from image recognition to natural language processing. However, the increasing complexity of these models often has an impact on computational and memory requirements, making them impractical for deployment on resource-constrained devices. To address this challenge, researchers have developed a wide range of parameter reduction techniques which aim to maintain model performance while significantly decreasing the number of parameters and computational load.

Parameter reduction is a process that removes certain layers of the neural network avoiding the loss of useful information of the network required for its decision process. This process can be applied to already trained neural networks or implemented during the training. Parameter reduction methods have evolved alongside the development of neural networks, beginning with the early pruning techniques [LeCun et al., 1989a] applied to simple feedforward networks and extending to sophisticated strategies used in modern transformer architectures [Vaswani et al., 2017b]. From classical approaches such as pruning [Han et al., 2015], quantization [Jacob et al., 2018], and low-rank factorization [Denton et al., 2014] to more advanced methods like knowledge distillation [Hinton et al., 2015] and cross-layer parameter sharing in transformer models [Lan et al., 2020], these techniques have proven crucial for enhancing the efficiency, scalability, and accessibility of deep learning systems. The overall architecture of ASR and LID is illustrated in Figure 5.1. Although structurally similar to the baselines in Chapter 3, this version emphasizes the parameter reduction techniques integrated into the pre-trained model. The following section provides an overview of these various parameter reduction techniques.

5.1 Background

Reducing the number of trainable parameters in neural networks is essential to improve computational efficiency, enable deployment on resource-constrained devices, and improve generalization. This section details the evolution of parameter reduction methods from early neural networks to state-of-the-art adapter modules in transformers.

5.1.1 Early Regularization and Pruning Techniques

One of the earliest strategies for parameter reduction was network pruning, where weights deemed unnecessary (e.g., those with small magnitudes) are removed post-training. This concept was popularized by Optimal Brain Damage and Optimal Brain Surgeon, which used

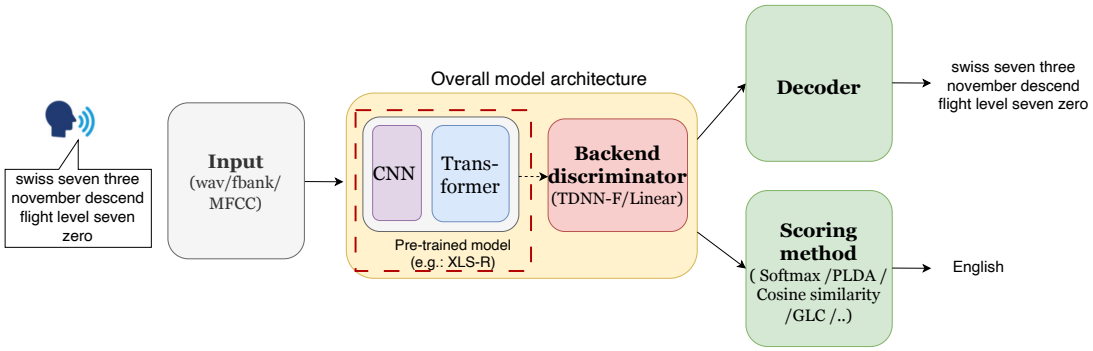

Figure 5.1: Overview of the overall model architecture for ASR and LID used in this Chapter. The aim is to reduce the parameters of the transformer part of the pre-trained model (XLS-R or Whisper) during training and inference as highlighted in the diagram.

second-order derivatives to prune weights without significantly affecting performance [LeCun et al., 1989b, Hassibi and Stork, 1993]. Later, L1 regularization was introduced to encourage sparsity by penalizing the absolute values of weights during training, effectively driving many weights to zero [Tibshirani, 1996].

There are several pruning methods:

a. Magnitude Pruning

One of the earliest and simplest approaches to parameter reduction is weight pruning based on magnitude. After training a model, weights with magnitudes below a certain threshold are removed:

$$W_{i,j}^{\text{pruned}} = \begin{cases} W_{i,j}, & \text{if } |W_{i,j}| > \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

This technique was formalized in Optimal Brain Damage (OBD) [LeCun et al., 1989b], which used the Hessian of the loss function to identify unimportant parameters:

$$\Delta E_i \approx \frac{1}{2} H_{ii} w_i^2, \tag{5.2}$$

where H_{ii} is the i-th diagonal element of the Hessian matrix and w_i is the weight.

Optimal Brain Surgeon (OBS) [Hassibi and Stork, 1993] improves this by considering full second-order information:

$$\Delta E = \frac{1}{2} \mathbf{w}^T \mathbf{H} \mathbf{w}. \tag{5.3}$$

b. L1 Regularization

L1 regularization encourages sparsity by penalizing the absolute values of weights:

$$\mathcal{L} \times \text{total} = \mathcal{L} \times \text{task} + \lambda \sum_{i,j} |W_{i,j}|. \tag{5.4}$$

This drives many weights to exactly zero, effectively reducing parameters during training [Tibshirani, 1996].

5.1.2 Quantization

Quantization is a compression technique aimed at reducing the computational complexity and memory footprint of neural networks. It achieves this by representing weights and activations with lower precision, for instance, by replacing 32-bit floating-point numbers with 8-bit integers [Jacob et al., 2018]. Mathematically, the quantization process maps a floating-point value w to a quantized integer q. A common affine mapping scheme is defined as:

$$q = \text{round}\left(\frac{w}{s} + z\right), \tag{5.5}$$

where s is a scaling factor and z is a zero-point offset (integer) [Jacob et al., 2018]. The approximate real value $\hat{w}$ can be reconstructed as $\hat{w} = s(q - z)$.Quantization strategies can be categorized into two main types:

- **Post-Training Quantization (PTQ):** This method applies quantization to a pre-trained model without further retraining. It calculates the quantization parameters (s and z) based on the distribution of weights and activations in the trained model.
- **Quantization-Aware Training (QAT):** In this approach, quantization is simulated during the training process, allowing the model to adapt to quantization noise. The gradients are typically approximated using straight-through estimation [Courbariaux et al., 2016].

a. Weight Sharing

A related compression approach is weight sharing, which clusters similar weights and stores only their cluster indices rather than individual values [Gong et al., 2014]. Given k clusters, each weight $w_{i,j}$ is approximated by a cluster centroid c:

$$w_{i,j} \approx c_{l(i,j)}, \quad l(i,j) \in 1, \ldots, k \tag{5.6}$$

where $l(i,j)$ is the index of the cluster assigned to weight $w_{i,j}$.

5.1.3 Knowledge Distillation

Knowledge distillation is a compression technique that involves training a compact „student“ model (f_S) using the supervision of a larger, pre-trained „teacher“ model (f_T) [Hinton et al., 2015]. The goal is to transfer the generalization ability of the teacher to the student.The training objective typically minimizes a combined loss function consisting of the standard task loss and a distillation loss:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{CE}(y, \hat{y}) + \lambda\mathcal{L}_{KD}(\mathbf{z}_T, \mathbf{z}_S), \tag{5.7}$$

where $\mathcal{L}_{CE}$ is the standard cross-entropy loss between ground truth labels y and student predictions $\hat{y}$, and λ is a hyperparameter balancing the two terms.The distillation loss, $\mathcal{L}_{KD}$, is often defined as the Kullback-Leibler (KL) divergence between the softened output distributions of the teacher and the student:

$$\mathcal{L}_{KD} = T^2 \cdot KL\left(\sigma\left(\frac{\mathbf{z}_T}{T}\right) \parallel \sigma\left(\frac{\mathbf{z}_S}{T}\right)\right). \tag{5.8}$$

Here, $\mathbf{z}_T$ and $\mathbf{z}_S$ represent the logits of the teacher and student respectively, σ denotes the softmax function, and T is the temperature parameter used to soften the probability distributions, revealing “dark knowledge” about inter-class relationships [Hinton et al., 2015]. The factor T^2 is applied to scale the gradients to match the magnitude of the hard-target gradients.

5.1.4 Parameter Sharing in Recurrent Networks

Recurrent Neural Networks (RNNs) inherently share parameters across time steps. For an RNN with hidden state $\mathbf{h}_t$:

$$\mathbf{h}_t = \sigma(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}), \tag{5.9}$$

where, $\mathbf{W}_h$ and $\mathbf{W}_x$ are shared across all time steps, reducing the number of unique parameters [Hochreiter and Schmidhuber, 1997].

5.1.5 Weight Tying in Language Models

Weight tying reduces the parameter count by sharing the weights between the input embedding matrix $\mathbf{E}$ and output softmax matrix $\mathbf{W}$:

$$\mathbf{W} = \mathbf{E}^T. \tag{5.10}$$

This approach improves generalization and reduces parameters [Press and Wolf, 2017, Inan et al., 2017].

5.1.6 Low-Rank Matrix Factorization

Low-rank matrix factorization reduces model size and computational complexity by decomposing large weight matrices, which are often highly redundant, into products of smaller matrices [Denton et al., 2014]. Given a weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, it can be approximated as the product of two low-rank matrices $\mathbf{U} \in \mathbb{R}^{m \times r}$ and $\mathbf{V} \in \mathbb{R}^{r \times n}$:

$$\mathbf{W} \approx \mathbf{U}\mathbf{V}, \tag{5.11}$$

where the rank r satisfies $r \ll \min(m, n)$. This decomposition reduces the number of parameters from mn to $r(m + n)$. This technique has been widely adopted to speed up Convolutional Neural Networks (CNNs) [Jaderberg et al., 2014] and compress dense layers [Denil et al., 2013]. In the context of Automatic Speech Recognition (ASR), this approach is utilized in the TDNN-F model [Povey et al., 2018], as described in Section 4.1.2. The TDNN-F applies a semi-orthogonal low-rank factorization, where Singular Value Decomposition (SVD) is combined with orthonormal constraints to improve computational efficiency without sacrificing model accuracy.

5.1.7 Adapter Modules in Transformer Models

Transformers have billions of parameters, making full fine-tuning inefficient. Adapter modules insert small bottleneck layers into each transformer block and only train these during adaptation. A typical adapter has the form [Houlsby et al., 2019]:

$$\mathbf{z} = \text{LayerNorm}(\mathbf{h}) \tag{5.12}$$

$$\tilde{\mathbf{h}} = \mathbf{h} + \mathbf{W}_{\text{up}} f(\mathbf{W}_{\text{down}} \mathbf{z}), \tag{5.13}$$

where $\mathbf{W}_{\text{down}} \in \mathbb{R}^{d \times m}$, $\mathbf{W}_{\text{up}} \in \mathbb{R}^{m \times d}$, $m \ll d$, and f is a non-linearity (usually ReLU or GELU). Only $\mathbf{W}_{\text{up}}$ and $\mathbf{W}_{\text{down}}$ are trainable. Variants like AdapterFusion [Pfeiffer et al., 2020] combine adapters from multiple tasks using attention-based gating:

$$\mathbf{h} = \sum_{i=1}^{N} \alpha_i A_i(\mathbf{h}), \tag{5.14}$$

where A_i is the adapter for task i and α_i are learned weights.Compacter [Karimi Mahabadi et al., 2021] further compresses adapters using hypercomplex multiplication and low-rank matrices, reducing memory and computation:

$$\mathbf{W} = \sum_{i=1}^{r} \mathbf{A}_i \otimes \mathbf{B}_i. \tag{5.15}$$

While standard adapters typically use a fixed bottleneck size m across all layers, in [Vanderreydt et al., 2023] we introduce ABSADAPTER an Adaptive Bottleneck Scheduler to redistribute the parameter budget dynamically. Based on the observation that lower layers (processing generic speech features) require more adaptation in domain-shifted scenarios than higher layers, this method linearly decreases the bottleneck size m as the layer depth increases. This approach allows for maintaining high performance on downstream tasks like ATC ASR while significantly reducing the total parameter count compared to fixed-size adapters.

5.2 Parameter Reduction in Transformer Model

The Transformer [Vaswani et al., 2017a] architecture provided a significant advancement in the field of natural language processing (NLP) and is the foundation for numerous state-of-the-art models now in speech processing. Each transformer layer consist of Multi-Head self-Attention (MHA) modules and FFN/MLP, each followed by layer normalization and residual connections to provide stability during training. In a commonly used configurations, both the MHA and MLP components contribute almost equally to the parameter count of models.

We consider XLS-R and Whisper medium pre-trained transformer-based models for this. The 300M params XLS-R model has 24 transformer layers of which 67% of the total parameter budget is used by the Fully Connected (FC) layers in the MLP components. Whereas the Whisper 'medium' version has 770M parameters, which consists of 48 transformer layers. The FC parameters take up 52% of the total parameter count.

We aim to reduce the parameters of the FC layers which results in fewer parameters during training and inference followed by low-rank adaptation for faster training. The choice of FC over the weights in the MHA is given as follows: the weights for query, key and value are already low-rank in a multihead configuration. Moreover, as mentioned earlier, results in [Lin et al., 2020, Hu et al., 2021] clearly demonstrate that updating only the FC layers with low-rank weight matrices provides majority of the boost in performance from adapter-based methods. Thus, we propose to study parameter pruning via low-rank matrix factorization using SVD.

5.2.1 Low-rank matrix factorization with SVD

We apply SVD [Strang, 2012] to each linear layer in the MLP component of the transformer module. Weight $\mathbf{W}$ of a linear layer is factorized as $\mathbf{U\Sigma V}$, deriving two linear layers $\mathbf{B} = \mathbf{U}$ and $\mathbf{A} = \mathbf{\Sigma V}$. Effectively, we approximate $\mathbf{W} \approx \mathbf{BA}$. In both XLS-R and Whisper models, there are two linear layers with a non-linearity inbetween. The first layer projects a 1024-dimensional embedding to 4096 dimensions, and the second one reverses this projection. Each of these linear layers is now factorized into two linear components of rank r: one $m \times r$ and another $r \times n$, where the initial weight matrix is of shape $m \times n$. For $r = 512, 256, 128,$

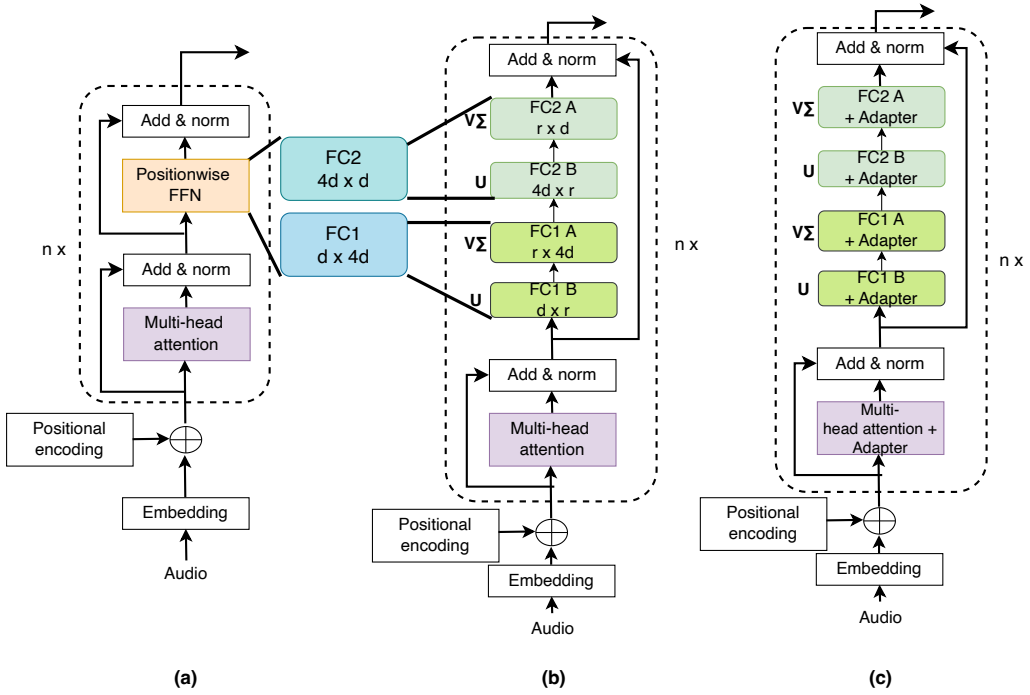

Figure 5.2: Overview of a single transformer layer for various settings: (a) overview of the standard transformer layer used during full finetuning towards downstream tasks. (b) transformer layer after applying matrix factorization. This is applied for inference only of the finetuned model and during matrix factorized finetuning. (c) transformer layer during low-rank adaptation. n is number of transformer layers which is 24 for XLS-R model and 48 for Whisper model. A rank r is the dimension used to apply SVD. In this work $r = 512, 256, 128$.

the size of the model reduces up to $\approx$ 24%, 44% and 54% respectively (the exact reduction is shared later in Section 5.4.2), thus reducing the total number of model parameters during both finetuning and inference.

SVD is applied prior to model finetuning. The directions corresponding to the lowest eigenvalues are removed. The models are then finetuned for 10 epochs for ASR and 2 epochs for LID, respectively.

In order to preserve the orthogonality of the matrices, we apply an orthonormal constraint after each update. We follow the update procedure from [Povey et al., 2018]. Unlike in AdaLoRA [Zhang et al., 2023] (Adaptive Low Rank Adapters), the constraint is applied only on $\mathbf{U}$ in the above formulation.

Model pruning with SVD has been explored in techniques such as RankDyna [Hua et al., 2023], which uses an information criterion to select the directions to prune out. One disadvantage with such methods is the additional requirement of tracking the first order and second order momentum of parameter groups. AdaLoRA uses a similar approach during parameter-efficient finetuning to impose a parameter budget on adapter layers. We consider rank pruning strategies in the aforementioned works as a part of our future investigation.

5.2.2 Low-rank Adaptation

Low-rank adaptation (LoRA) based methods are now a common approach for PEFT of large pre-trained models [Hu et al., 2021, Zhang et al., 2023, Kopiczko et al., 2023]. In LoRA, during model finetuning, instead of updating the full weight matrices during adaptation, it freezes the pre-trained weights and injects *low-rank* trainable matrices whose product approximates the change needed for each task. This is expressed as an update $\Delta\mathbf{W}$ to the original model parameters $\mathbf{W}_0$. Thus, the parameter size of $\Delta\mathbf{W}$ is significantly lesser than

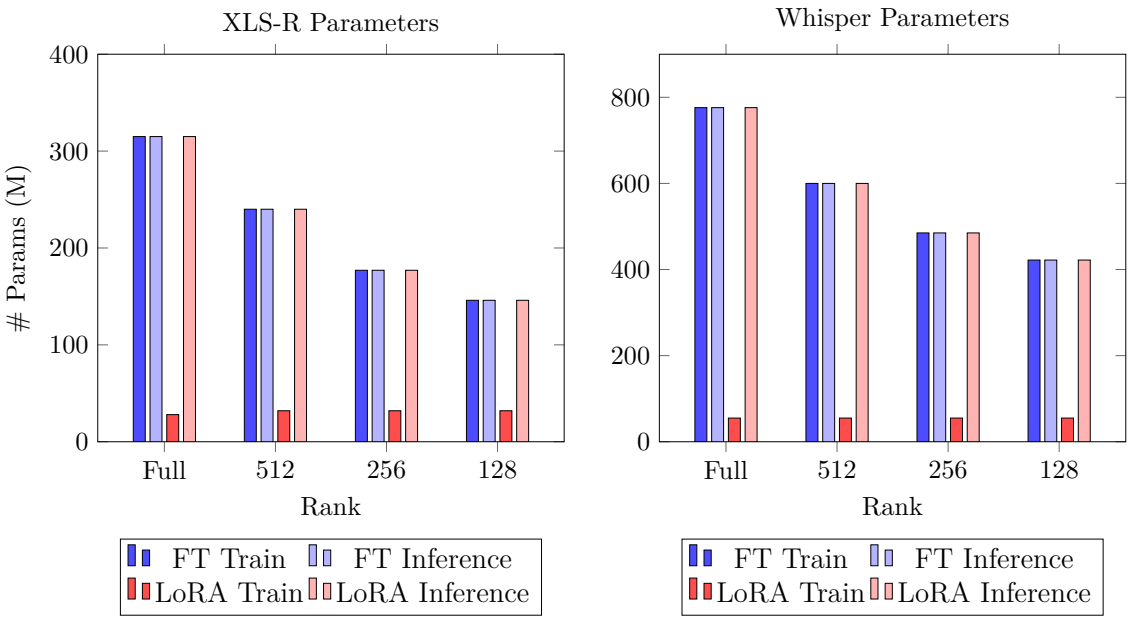

Figure 5.3: Side-by-side comparison of # training vs. inference parameters for XLS-R and Whisper. The results show that low-rank factorization reduces the model parameters during both training and inference. LoRA reduces training parameters only.

$\mathbf{W}_0$. The resource requirements for finetuning of large models are significantly reduced. For example, in experiments with GPT-3 (175B parameters), LoRA reduces the number of trainable parameters by up to $\sim$10,000$\times$ and GPU memory usage during training by $\sim$3$\times$, *with no inference latency overhead*, because at deployment time the low-rank updates can be merged into the frozen weights [Hu et al., 2021].

LoRA represents the weight update $\Delta\mathbf{W}$ for a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ as a product of two much smaller matrices $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$, where $r \ll \min(d, k)$. During training only $\mathbf{A}$ and $\mathbf{B}$ are updated, while $\mathbf{W}_0$ remains unchanged. One form of the forward pass thus becomes

$$\mathbf{h} = \mathbf{W}_0\mathbf{x} + \Delta\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{B}\mathbf{A}\mathbf{x}.$$

Empirical results show that even very small r (e.g., 1–4) are adequate for many downstream tasks without much loss in performance. Moreover, it has been observed that the learned $\Delta\mathbf{W}$ often lies in a low-dimensional subspace (low intrinsic rank), showing substantial overlap of the top singular vector directions between versions of low rank and higher rank [Hu et al., 2021].

A key difference between fine-tuning after SVD and finetuning with LoRA based methods is that the former constrains the directions of the low-rank structure implicitly by choosing the eigenvectors as initial weights. It also limits the rank options as reducing the value significantly can severely affect performance [Hua et al., 2023]. Parameter Efficient FineTuning (PEFT) with LoRA therefore can be seen as a complementary technique to SVD. Thus, after reducing the parameters with SVD, we freeze the model and update the weights with LoRA instead of full-finetuning proposed in the previous subsection.

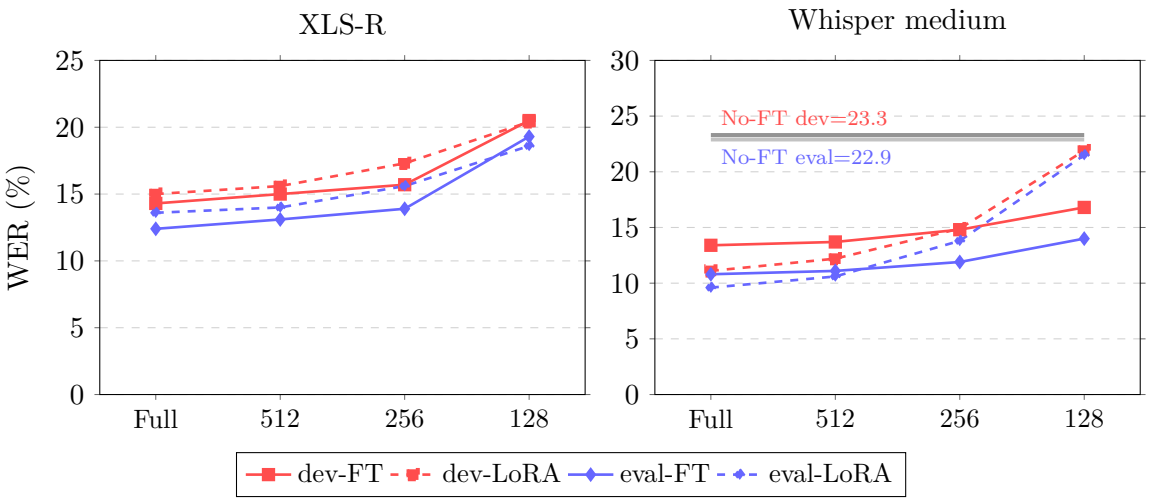

Figure 5.4: WER (%) comparison of XLS-R and Whisper medium across various ranks and configurations on AMI dev and eval test sets. **No-FT** = zero-shot evaluation (Whisper only), **FT** = full fine-tuning, **LoRA** = low-rank adaptation. Solid lines = FT, dashed lines = LoRA. Lower WER is better.

5.3 Experimental Setup

Experiments are conducted to evaluate the following: (1) full fine-tuning of models, (2) applying low-rank factorization during full fine-tuning, (3) PEFT with LoRA, and finally (4) PEFT with LoRA after low-rank factorization. Model finetuning and inference are done one a single GPU node with Nvidia RTX 3090. The next subsections provide details about model finetuning and evaluation setups. Model training after matrix factorization with SVD follow the same setup as full-finetuning.

5.3.1 ASR fine-tuning

The espresso [Wang et al., 2019] toolkit which uses fairseq [Ott et al., 2019] is used for finetuning XLS-R model. We use the E2E LF-MMI criterion [Vyas et al., 2021c] with a learning rate of 1e-5. In addition to the pre-trained parameters, three layers of TDNN-F [Povey et al., 2018] are added with a learning rate factor of 20 using the implementation from Pkwrap [Madikeri et al., 2020]. The standard AMI setup from Kaldi is modified to use only IHM recordings for training and evaluation [Povey et al., 2011]. The model is finetuned for 20000 updates with the learning rate scheduler tuned as follows: 20% warm-up, 60% constant learning rate, and 20% decay. The pre-trained model parameters are frozen for the first 500 updates.

The k2/Icefall framework[1] is used for finetuning Whisper medium. Learning rate of 1e-5 and label smoothing loss with smoothing factor 0.1 is used. The features are masked with SpecAug during training [Park et al., 2019]. AdamW [Loshchilov and Hutter, 2017] optimizer is used for all setups in this paper. A lower-cased version of the transcripts are used for training. finetuning is carried out for 10 epochs and the best model is chosen based on the WER on development set.

[1] https://github.com/k2-fsa/icefall

5.3.2 LID finetuning

The espresso [Wang et al., 2019] toolkit which uses fairseq [Ott et al., 2019] is used for finetuning XLS-R model. We use the cross-entropy criterion with a learning rate of 1e-5. In addition to the pre-trained parameters, one layer of Orthonormal Linear is added based on the observed results from Table 3.3. A learning rate factor of 20 is used from Pkwrap [Madikeri et al., 2020] implementation. Voxlingua107 data is used for training and Voxlingua107 dev, LRE17 test and ATCO2-LID test sets are used for evaluation. The model is finetuned for 40́00 updates with the learning rate scheduler tuned as follows: 10% warm-up, 40% constant learning rate, and 50% decay. The pre-trained model parameters are frozen for the first 500 updates.

Whisper model is finetuned with VoxLingua107 dataset with 33 languages subset included in the Voxlingua107 dev set to allow for rapid finetuning.

In Whisper, we used the start of the sentence token to extract only the language labels. During finetuning and evaluation, we do not constrain the languages to be one among the 33. That is, the classifier is allowed to predict any of the 99 languages. The classification is obtained by taking the language token with the maximum value over the logit vector obtained at the end of processing an audio.

5.3.3 Low rank adaptation

For all experiments we used the peft package's LoRA implementation. The parameter α, which adjusts the weight used (α/r where r is the rank of the matrix), is adjusted such that the weight is always 0.5.

5.4 Results

Rank of updates after full-finetuning

The ASR and LID systems are trained and evaluated independently. Before discussing the results on the two downstream tasks, we first analyze the rank of the updates on weight matrices obtained after regular finetuning. This is done in order to contrast the claim that task-dependent finetuning is low-rank in nature. Let $\mathbf{W}_0$ is the weight matrix of a linear layer in the pre-trained model and $\mathbf{W}_f$ is the corresponding weight matrix obtained after finetuning. According to the adapter terminology, $\Delta\mathbf{W} = \mathbf{W}_f - \mathbf{W}_0$. We then evaluate the rank of $\Delta\mathbf{W}$ for each weight matrix in each linear layer of the pre-trained model. In every case, the matrix is full-rank (the rank is 1024 in the case of FC weights in the MLP layers) suggesting that the full finetuning may be inefficient.

5.4.1 Parameter reduction

Figure 5.3 illustrates the training and inference parameter counts for the XLS-R and Whisper models under different low-rank factorization settings (Full, 512, 256, and 128), comparing standard Fine-Tuning (FT) with Low-Rank Adaptation (LoRA). For XLS-R, inference parameters decrease from approximately 315M (Full) to 240M at rank 512 ($\approx$ 23% reduction), 175M at rank 256 ($\approx$ 44% reduction), and 145M at rank 128 ($\approx$ 54% reduction). Whisper follows a similar trend, decreasing from roughly 780M (Full) to 600M at rank 512 ($\approx$ 23% reduction), 480M at rank 256 ($\approx$ 38% reduction), and 420M at rank 128 ($\approx$ 46% reduction). These results show that low-rank factorization reduces the model parameters

Table 5.1: WER (%) of AMI dev and test sets evaluated with various rank for SVD on ASR task. No FT: Zero-shot evaluation, FT: full finetuning and LoRA indicates parameter efficient finetuning with LoRA. The baseline WER (%) with Whisper is significantly better than that of the XLS-R model and applying LoRA alone provided the best ASR performance. When reducing the parameters by 23.8% using matrix factorization (rank 512), there is a degradation of 0.7% absolute in WER for XLS-R and no degradation for Whisper model.

Rank	**Config**	**XLS-R**				**Whisper medium**			
		# params (M)		**WER (%)**		**# params (M)**		**WER (%)**	
		Training	Inference	dev	eval	Training	Inference	dev	eval
Full	No FT	-	-	-	-	776	776	23.3	22.9
	FT	315	315	14.3	12.4	776	776	13.4	10.8
	LoRA	28	315	15.0	13.6	55	776	11.1	9.6
512	FT	240	240	15.0	13.1	600	600	13.7	11.1
	LoRA	32	240	15.6	14.0	55	600	12.2	10.6
256	FT	177	177	15.7	13.9	485	485	14.8	11.9
	LoRA	32	177	17.3	15.6	55	485	14.9	13.8
128	FT	146	146	20.5	19.3	422	422	16.8	14.0
	LoRA	32	146	20.4	18.6	55	422	21.9	21.5

Table 5.2: WER (%) of NATS and ATCO2 test sets evaluated with various rank for SVD on ASR task. No FT: Zero-shot evaluation, FT: full finetuning and LoRA indicates parameter efficient finetuning with LoRA. M refers to # params in millions. The baseline WER (%) with XLS-R is significantly better than that of the Whisper model on the clean NATS set. When reducing the parameters by 23.8% using matrix factorization (rank 512), there is a slight degradation in WER for the ATCO2 test set and no degradation for the NATS test set.

Rank	**Config**	**XLS-R**				**Whisper medium**			
		# params (M)		**WER (%)**		**# params (M)**		**WER (%)**	
		Training	Inference	NATS	ATCO2	Training	Inference	NATS	ATCO2
Full	No FT	-	-	-	-	776	776	53.1	56.1
	FT	315	315	4.8	17.5	776	776	6.9	18.6
	LoRA	28	315	5.3	18.5	55	776	7.5	19.2
512	FT	240	240	4.9	18.1	600	600	5.1	21.2
	LoRA	32	240	6.3	21.4	55	600	9.7	20.4
256	FT	177	177	5.2	20.0	485	485	5.6	21.8
	LoRA	32	177	5.5	19.7	55	485	11.8	22.5
128	FT	146	146	6.3	22.8	422	422	6.7	24.8
	LoRA	32	146	9.4	34.2	55	422	23.2	44.8

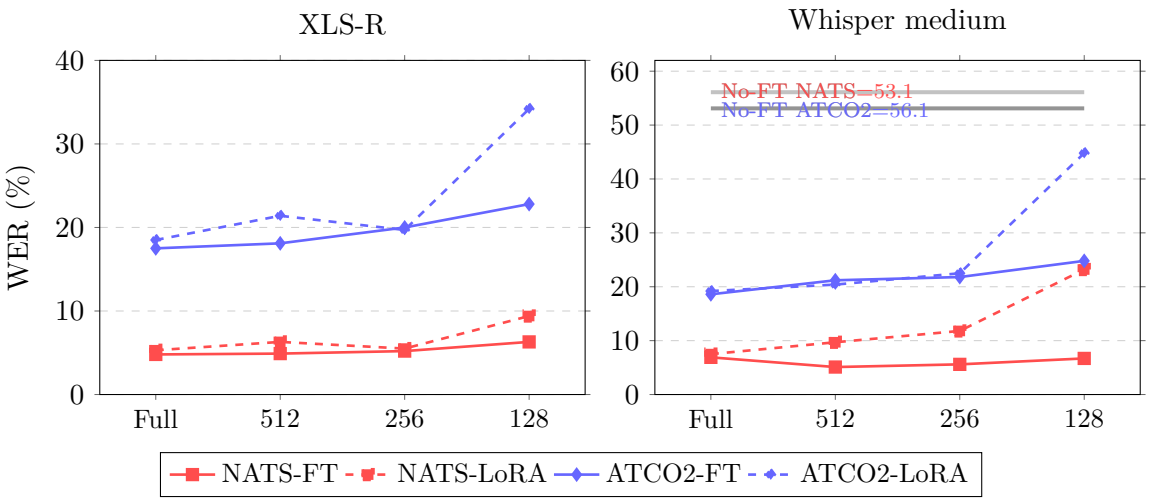

Figure 5.5: WER (%) comparison of XLS-R and Whisper medium across various ranks and configurations on NATS and ATCO2 test sets. **No-FT** = zero-shot evaluation (Whisper only), **FT** = full fine-tuning, **LoRA** = low-rank adaptation. Solid lines = FT, dashed lines = LoRA. Lower WER is better.

during both training and inference which also lowers both memory footprint and inference cost.

In contrast, LoRA drastically reduces the number of trainable parameters while leaving inference parameter counts aligned with the compressed base model at each rank. For XLS-R, LoRA requires only about 30M trainable parameters, corresponding to a $\approx 90\%$ reduction compared to full FT training at the Full configuration and still an $\approx 80\%$ reduction even at rank 128. For Whisper, LoRA uses approximately 60M trainable parameters, yielding about a $\approx 92\%$ reduction relative to full FT training at Full rank and roughly an $\approx 86\%$ reduction at rank 128.

5.4.2 ASR results

AMI: The performance of the ASR systems is presented in Table 5.1 and Figure 5.4. We report WER% for both dev and eval splits of the AMI dataset. All ASR systems are evaluated after text normalization. The baseline WER% with Whisper is significantly better than that of the XLS-R model (10.8% vs 12.4% WER). For the XLS-R system, 12.4% WER is obtained after full finetuning. Matrix factorization results in a degradation of 0.7% absolute WER with a reduction of 23.8% of the parameters. Further reduction of the rank of the matrix to 256 results in total degradation of 1.5% WER, which is not as severe as reducing to 128 dimensions with a total degradation of $\approx 7\%$ in WER. Similar trends are observed with the Whisper model. However, this could be partially attributed to the size of the model and the amount of training data used. The degradation when reducing the rank to 512 and 256 is limited (0.3% and 1% absolute).

In case of the Whisper model, applying LoRA alone provided the best ASR performance (9.6% WER), which to the best our knowledge is the best ASR performance achieved on AMI-IHM data yet. The combination of factorization and LoRA performs as well as full-finetuning even after 22% reduction in model size. In addition, we are also able to take advantage of PEFT with only 55M parameters required during training.

Table 5.3: Accuracy (%) on Voxlingua dev and LRE17 test and ATC test sets for various rank in Whisper models in LID task. †: the numbers in the parentheses indicate the percentage of reduction in the number of parameters. The performance gap is negligible when combining low-rank factorization and finetuning with LoRA.

Rank	**Config**	**# params (M) †**		**Accuracy (%)**		
		Training	Inference	Voxlingua-dev	LRE17-test	ATCO2-LID-test
Full	No FT	776	776	92.6	86.1	61.6
	FT	776		97.1	91.2	20.7
	LoRA	55 (92.9)		96.4	90.8	16.5
512	FT	600 (22.7)	600 (22.7)	95.4	84.7	28.4
	LoRA	55 (92.9)		96.0	90.5	19.0
256	FT	485 (37.5)	485 (37.5)	92.8	80.5	19.5
	LoRA	55 (92.9)		90.3	80.1	15.5
128	FT	422 (45.0)	422 (45.0)	47.6	40.6	3.1
	LoRA	55 (92.9)		93.4	76.9	10.9

When reducing the rank to 256 or 128 prior to training, a degradation in ASR performance was observed. However, the degradation with rank 256, where the model size reduces by up to 48.5%, the worst possible WER was only 11.9% with Whisper and 13.9% with XLS-R. WERs were beyond 20% when further reducing the rank from 256 to 128, suggesting the importance of information lost in the particular directions dropped.

We also observe distinct behaviors between the model architectures when applying LoRA at full rank. While LoRA significantly outperforms full finetuning for Whisper (9.6% vs 10.8% WER), it results in slightly worse performance for XLS-R (13.6% vs 12.4%). This suggests that the pre-trained Whisper model may benefit more from the regularization or parameter efficiency of LoRA on this specific dataset.

A notable divergence occurs at the lowest rank (128). Here, Whisper fails to recover performance using LoRA, degrading to 21.5% WER compared to 14.0% with full finetuning. In contrast, XLS-R with LoRA actually surpasses its full finetuning counterpart at this rank (18.6% vs 19.3%). This indicates that when model capacity is heavily constrained, the limited trainable parameters of LoRA are insufficient for Whisper, whereas they remain effective for XLS-R.

The Whisper rank 512 with LoRA achieves a WER of 10.6%, outperforming the standard Full Finetuning baseline (10.8%) while reducing inference parameters by approximately 22% (600M vs 776M). In terms of training efficiency, the gains are even more substantial; for instance, applying LoRA to XLS-R reduces the trainable parameter count from 315M to just 28M, a reduction of over 90%.

ATC: The performance of the ASR systems is presented in Table 5.2 and Figure 5.5. We report WER on both the NATS test set, which is clean and seen during training, and the ATCO2 test set, which consists of noisy data not seen during training.

The baseline WER% with XLS-R is significantly better than that of the Whisper model on the clean NATS set (4.8% vs 6.9%). When reducing the parameters by 23.8% using matrix factorization (rank 512), while there is a slight degradation in WER for the noisy

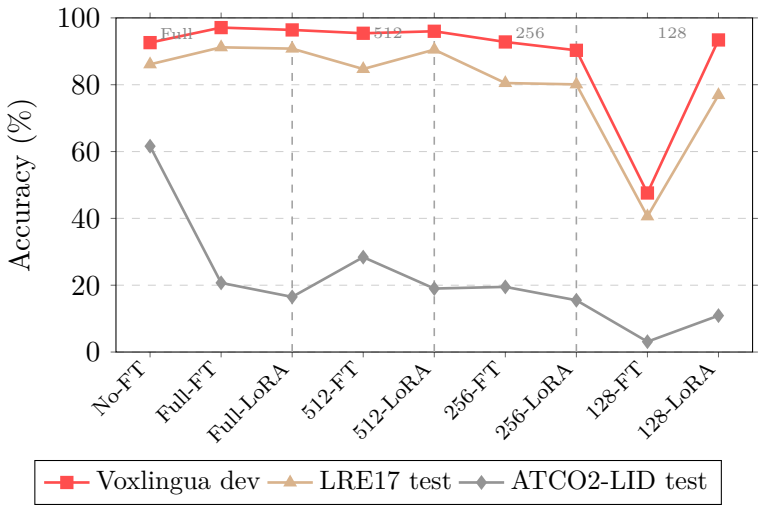

Figure 5.6: LID Accuracy (%) for Whisper models across various rank and configuration combinations. **No-FT** = no fine-tuning, **FT** = full fine-tuning, **LoRA** = low-rank adaptation. Numbers indicate SVD rank. The performance gap between FT and LoRA is negligible at higher ranks.

ATCO2 test set (18.1% vs 17.5%), we observe no degradation—and practically stable performance—for the clean NATS test set (4.9% vs 4.8%).

For the Whisper model, the zero-shot ("No FT") performance is notably poor, with WERs exceeding 50% on both datasets (53.1% NATS, 56.1% ATCO2), reconfirming the necessity of finetuning for this domain. Unlike XLS-R, applying matrix factorization to Whisper improves performance on the clean NATS dataset relative to full finetuning. The rank 512 FT configuration achieves a WER of 5.1%, which is significantly better than the Full FT baseline of 6.9%. This suggests that for Whisper, the reduction in parameters may act as a beneficial regularizer on the seen domain, preventing overfitting.

However, the robustness of these factorized models varies significantly when exposed to the noisy, out-of-domain ATCO2 data. For XLS-R, performance degrades gradually as rank decreases (from 17.5% at Full rank to 22.8% at rank 128). In contrast, Whisper with LoRA struggles severely at lower ranks. At rank 128, the Whisper LoRA model deteriorates, yielding a WER of 23.2% on NATS and 44.8% on ATCO2, compared to the much more stable Full rank LoRA performance. This indicates that while standard finetuning (FT) remains relatively robust at low ranks (6.7% WER for Whisper Rank 128 FT on NATS), the combination of low-rank factorization and LoRA is insufficient to model the complexities of the ATC data effectively.

Additional Observations There is a clear trade-off between model compression and out-of-domain generalization. While both XLS-R and Whisper maintain strong performance on the in-domain NATS set even at lower ranks (e.g., XLS-R Rank 256 FT at 5.2% vs Full FT 4.8%), the degradation is much sharper on the noisy, out-of-domain ATCO2 set (e.g., XLS-R Rank 256 FT at 20.0% vs Full FT 17.5%). This suggests that the redundant parameters pruned during factorization are likely crucial for handling noise and domain shifts.

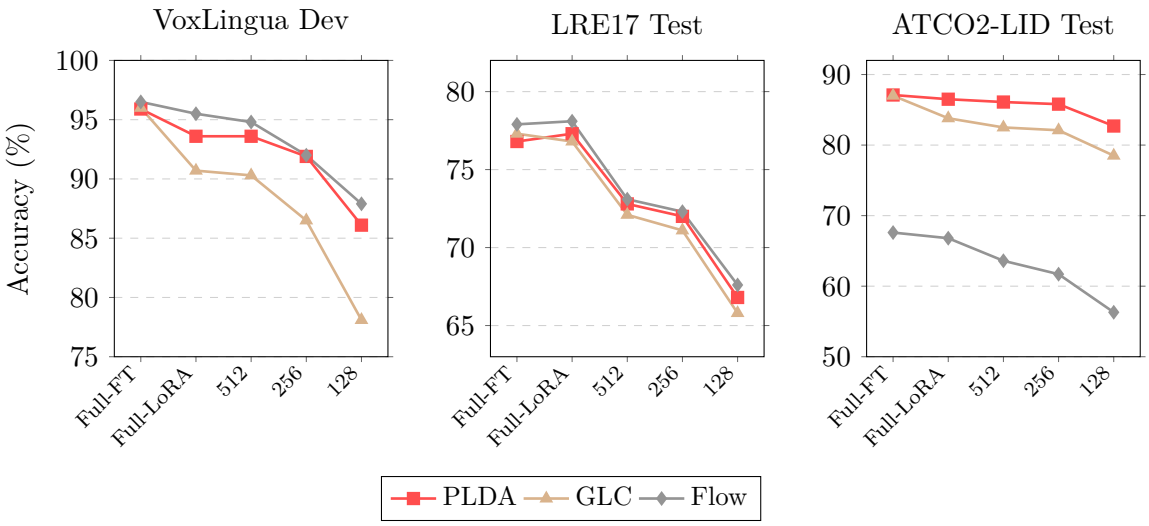

Figure 5.7: LID Accuracy (%) comparison of Full FT and LoRA (various ranks) on XLS-R across three test sets and scoring methods. **Full-FT** = full fine-tuning, **Full-LoRA** = LoRA at full inference rank, **512**, **256**, **128** = LoRA with SVD rank reduction. **PLDA** = Probabilistic Linear Discriminant Analysis, **GLC** = Gaussian Linear Classifier, **Flow** = Normalizing Flows.

The instability of LoRA at extremely low ranks (Rank 128) is more pronounced in the ATC task compared to the AMI task. For instance, on the ATCO2 test set, XLS-R LoRA degrades to 34.2% WER at Rank 128, whereas the Full Finetuning version at the same rank maintains a WER of 22.8%. This highlights that when the base model's capacity is severely restricted by SVD, the limited number of trainable parameters of LoRA is not enough to adapt the model to noisy, unseen environments.

Similarly to the AMI dataset, the Rank 512 configuration for Whisper is a favorable trade-off on the clean NATS data. Not only does it outperform the full-rank baseline (5.1% vs 6.9%) but it does so with a 22% reduction in inference parameters.

5.4.3 LID results

LID with Whisper

The accuracy (%) for the LID trained with the Whisper model with Voxlingua107 train and evaluated in Voxlingua dev, the LRE17 test, and the ATCO2-LID test sets are shown in Table 5.3 and Figure 5.6. We observe trends similar to those reported in the previous section for ASR. The performance gap is negligible when combining low-rank factorization and finetuning with LoRA. The performance drop is more noticeable on an out-of-domain test set (LRE17) when using ranks of 256 and 128 for factorization. In particular, training stability was a concern when finetuning without adapters when using a rank of 128 for factorization. In fact, for ATCO2-LID test set the effect of catastrophic forgetting is observable with the performance being worse than random in all cases.

The performance for LID system trained with XLS-R model are shown in Table 5.4 and Figure 5.7. For XLS-R models, we first generate the embeddings from the classification layer of the model. These embeddings are then used for the classifiers – PLDA, GLC and FLow. For each evaluation set, we use the corresponding development set for GLC, PLDA

Table 5.4: Accuracy (%) on Voxlingua dev, LRE17 and ATCO2-LID test sets for various ranks and scoring methods in XLS-R models on LID task. †: the numbers in the parentheses indicate the percentage of reduction in the # parameters. **PLDA**: Probabilistic Linear Discriminant Analysis model, **GLC**: Gaussian Linear Classifier, **Flow**: Normalizing Flows. The results indicate Flow model consistently outperformed other scoring methods on Voxlingua107 and LRE17 datasets and PLDA provides best performance for ATCO2-LID set.

Rank	**Config**	**# params (M) †**		**Scoring**	**Accuracy (%)**		
		Training	Inference	**Method**	Voxlingua dev	LRE17-test	ATCO2-LID test
Full	FT	315	315	PLDA	95.9	76.8	87.1
				GLC	96.0	77.3	87.0
				Flow	96.5	77.9	67.6
	LoRA	28 (91.1)	315	PLDA	93.6	77.3	86.5
				GLC	90.7	76.8	83.8
				Flow	95.5	78.1	66.8
512	LoRA	32 (89.2)	240 (23.8)	PLDA	93.6	72.8	86.1
				GLC	90.3	72.1	82.5
				Flow	94.8	73.1	63.6
256	LoRA	32 (89.2)	177 (43.8)	PLDA	91.9	72.0	85.8
				GLC	86.5	71.1	82.1
				Flow	92.0	72.3	61.7
128	LoRA	32 (89.2)	146 (53.7)	PLDA	86.1	66.8	82.7
				GLC	78.1	65.8	78.5
				Flow	87.9	67.6	56.3

and Flow model training as it provides better performance. For Voxlingua, a subset of the training data is used for these models and then evaluated on Voxlingua dev. The XLS-R baseline has an accuracy of 92.9% and 74.3% and 18.4% for Voxlingua dev, LRE17 eval and ATCO2 test sets (see Orthonormal Linear row in Table 4.2.

Each model was evaluated using on PLDA, GLC, and flow-based scoring methods. For the in-domain test set (VoxLingua dev), the system achieved a high accuracy of 92.5%, demonstrating strong generalization within the training domain. However, for out-of-domain test sets such as LRE and ATCO2, the performance decreased substantially to 74.2% and 18.4%, respectively. This suggests that for out-of-domain conditions, alternative classifiers such as PLDA, GLC, and flow models are more effective than the softmax classifier. Among all classifiers, PLDA consistently provides the best overall performance, followed by GLC and flow models. Furthermore, experiments with dimensionality reduction using SVD revealed that the SVD-512 configuration leads to a degradation in in-domain performance, whereas lower-dimensional representations (SVD-256 and SVD-128) maintain comparable accuracy without noticeable loss.

5.4.4 Effect of Orthonormal constraint

The effect of applying the orthonormal constraint after low-rank factorization is demonstrated in the results in Table 5.5. We only present the results on ASR systems for brevity. For ranks 512 and 256, the orthonormal constraint provides gains in WER% of up to 0.4% absolute. The advantage is observed only with the XLS-R system. With the Whisper models, no performance benefits are observed. When the rank is reduced to 128, a degradation

Table 5.5: Results of dev and eval sets of AMI when using orthonormal constraint during finetuning of the matrix factorized layers for XLS-R and Whisper models.

Rank	**WER (%)**			
	dev	eval	dev	eval
	XLS-R		Whisper	
512	15.6	14.6	13.7	11.1
+ Orthonormal constraint	15.0	13.1	13.7	11.2
256	16.0	14.9	14.8	11.9
+ Orthonormal constraint	15.7	13.9	14.9	12.0
128	17.3	15.7	16.8	14.0
+ Orthonormal constraint	20.5	19.3	17.0	14.3

of in WER up to 3.6% absolute is observed. Thus, for our experiments we did not employ the orthonormal constraint when using rank=128.

Chapter 6

Conclusion

6.1 Summary

This thesis investigated the development of efficient and reliable speech processing systems for the ATC domain, with a particular focus on ASR and LID. Chapters 1 and 2 introduced these two core tasks, outlining their historical development and the impact of recent self-supervised and large-scale pre-trained models such as wav2vec 2.0 and Whisper. The performance of these models for low-resource and safety-critical domains such as ATC remains challenging. In this context, the thesis emphasized robustness and efficiency for ATC systems.

Chapter 3 established baseline systems for both ASR and LID. For ASR, experiments were conducted on the AMI and ATC datasets, capturing both general and domain-specific speech characteristics. Similarly, for LID, models were trained on the VoxLingua107 dataset and evaluated on VoxLingua107, LRE17, and ATCO2-LID. The results confirmed that fine-tuned self-supervised models offer substantial gains over traditional approaches. The high parameter counts and computational demands of the models motivated the exploration of parameter-efficient architectures.

Chapter 4 focused on parameter-efficient backend discriminators for LID, motivated by the need to balance performance and model complexity in low-resource settings. Building on prior application of TDNN-F layers in Kaldi-based ASR systems, we explored their integration into E2E LID pipelines based on large pre-trained encoders. The proposed architectures incorporated TDNN-F layers and an ECAPA-TDNNF-style backend to replace more parameter-heavy classifiers. With the Orthonormal linear classifier, by adding a simple linear layer with an orthonormal constraint, we obtained improvements in accuracy in two out of three conditions over using a linear layer in the backend. We also observed that fine-tuning with a linear layer – with or without Orthonormal constraint – outperforms other common architecture choices. Using only TDNN-F instead of ECAPA-TDNN or the Transformer improved the in-domain performance by up to 3% absolute. Finally, ECAPA-TDNN-F replaces the TDNN layers in ECAPA-TDNN with TDNN-F layers, reducing the number of parameters by up to 50%. When reducing by 50% we achieved competitive results for Voxlingua107 and LRE17 datasets. However, for ATCO2-LID data, we observed degradation. Furthermore, the findings in Chapter 4 highlighted that parameter-efficient alternatives are often better suited for practical LID deployments.

In addition to exploring parameter efficient backend discriminators, Chapter 4 also explored various scoring methods. We introduced a non-linear model – Normalizing Flows. While PLDA strongly relies on length normalization to boost its performance, the flow

model can offer more robustness when the data distribution varies and it does not match the Gaussian assumption of PLDA. We present the Flow as a non-linear version of PLDA which offers the possibility to compose many invertible non-linear transformations in order to model non-Gaussian data distributions. Theoretically, the flow can be applied to the same variety of tasks as PLDA (classification, clustering, and verification) while keeping desirable properties such as Jacobian cancellation in likelihood ratios and simple Gaussian distributions for latent variables. The results show that for both an in-domain and an out-of-domain test sets, the Flow model consistently provided higher accuracy compared to PLDA and GLC, while for the out-of-domain and low-resource dataset (ATCO2-LID), GLC performed better in most of the cases. For the ATCO2-LID test set, we observed a significant performance drop for the Flow model due to the difficulty in modeling low-resource noisy speech and poor hyperparameter transfer across domains.

In Chapter 5, we studied the application of low-rank factorization with SVD on the fully connected layers of the feedforward components of the Transformer model for speech and language recognition tasks. In particular, SVD was applied to the pre-trained models to reduce parameter counts during both training and inference, addressing a limitation of popular PEFT methods such as LoRA, which primarily reduce training-time parameters. We studied the technique independently and in combination with LoRA. When reducing the rank of the fully connected layers from 1024 to 512, thereby reducing the model parameter effectively by 22.7% for the Whisper medium model and 28% for the XLS-R model, no performance degradation was observed. Further reduction of the rank to 256 introduced performance trade-offs. The combination of SVD with LoRA further reduced the number of trainable parameters during fine-tuning, demonstrating that complementary techniques can be used to balance efficiency and performance.

6.2 Future Work

For ASR, we plan to extend our work by exploring improved acoustic model architectures such as the Zipformer, and LLM-driven architectures such as SLAM-ASR (Speech-Language Aligned Modeling for ASR).

SLAM-ASR [Ma et al., 2024] leverages joint modeling of speech and text representations, typically by aligning acoustic encoders with pre-trained language models in a unified framework. In low-resource domains such as ATC, where labelled speech data is scarce but textual data is relatively easier to collect, SLAM-ASR can help bridge the gap by injecting stronger language priors into the recognition process.

Zipformer models are designed to reduce computational complexity while maintaining competitive recognition accuracy, typically using significantly fewer parameters (approximately 60M) compared to conventional Transformer-based systems. Their efficiency makes them well suited for deployment in resource-constrained or real-time environments while still providing strong modeling capacity. Another promising direction is the investigation of factorized Transducer architectures [Chen et al., 2022b, Deng et al., 2025]. In this framework, the prediction network (predictor) is decoupled from the rest of the Transducer and can be replaced or augmented by a large language model (LLM). This modular design enables the integration of strong linguistic priors learned from large-scale text corpora into the ASR system. For domains such as ATC, where annotated speech data is limited but domain-specific text may be more accessible, incorporating an LLM as the predictor can improve contextual modeling, rare phrase recognition, and overall robustness to domain shift.

Such an approach may significantly enhance performance in safety-critical and data-scarce environments.

For LID, we plan to investigate the use of large pre-trained audio models to obtain better acoustic representations. Large audio models trained in diverse multilingual corpora can capture high-level phonetic and prosodic patterns that generalize well across domains. Fine-tuning these models for LID is expected to improve robustness to noise, channel variability, and out-of-domain conditions.

Based on the observed performance drop of the Flow model in the low-resource out-of-domain ATCO2-LID dataset, our aim is to investigate and optimize hyperparameter configurations specifically for low-resource and noisy speech settings. Additionally, we plan to evaluate the impact of applying length normalization to the learned embeddings in the Flow model. This analysis will help determine whether embedding scaling or representation inconsistency contributes to the degradation observed in mismatched scenarios.

Building on the efficiency gains demonstrated by applying SVD to the fully connected layers of Transformer models, we will further investigate the training instability observed in the Whisper model when applying an SVD rank of 128. A deeper analysis of the gradient flow and loss behavior at this compression level is required to better understand the cause of the performance drop. Furthermore, we intend to explore dynamic low-rank adaptation strategies. Instead of applying a fixed rank uniformly across all layers, a more flexible approach would assign ranks dynamically based on the relative importance of each layer. This could be achieved by defining and computing a layer importance score, enabling a better trade-off between parameter efficiency and recognition performance compared to static rank selection. Ultimately, these directions aim to improve the robustness, efficiency, and adaptability of both ASR and LID systems in low-resource and domain-specific settings.

Bibliography

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

[Babu et al., 2022] Babu, A., Wang, C., Tjandra, A., Lakhotia, K., Xu, Q., Goyal, N., Singh, K., Von Platen, P., Saraf, Y., Pino, J., et al. (2022). XLS-R: Self-supervised cross-lingual speech representation learning at scale. In *Proc. of Interspeech*, pages 2278–2282.

[Baevski et al., 2020] Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA. Published as a conference paper.

[Baum et al., 1970] Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

[Bogert et al., 1963] Bogert, B. P., Healy, M. J. R., and Tukey, J. W. (1963). The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In Rosenblatt, M., editor, *Proceedings of the Symposium on Time Series Analysis*, pages 209–243. John Wiley & Sons, New York, NY.

[Bourlard and Morgan, 1993] Bourlard, H. and Morgan, N. (1993). *Connectionist Speech Recognition: A Hybrid Approach*. Springer.

[Chen et al., 2022a] Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., et al. (2022a). WavLM: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518.

[Chen et al., 2022b] Chen, X., Meng, Z., Parthasarathy, S., and Li, J. (2022b). Factorized neural transducer for efficient language model adaptation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8132–8136.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

[Chung et al., 2021] Chung, Y.-A., Zhang, Y., Han, W., Chiu, C.-C., Qin, J., Pang, R., and Wu, Y. (2021). W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 244–250. IEEE.

[Conneau et al., 2020] Conneau, A., Baevski, A., Collobert, R., Mohamed, A., and Auli, M. (2020). Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*.

[Courbariaux et al., 2016] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 4107–4115.

[Dahl et al., 2011] Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

[Davis and Mermelstein, 1980a] Davis, S. and Mermelstein, P. (1980a). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 28, pages 357–366. IEEE.

[Davis and Mermelstein, 1980b] Davis, S. B. and Mermelstein, P. (1980b). Speech recognition by machine: A review. *Proceedings of the IEEE*, 68(4):404–431.

[Dehak et al., 2010] Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2010). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

[Delpech et al., 2018] Delpech, E., Laignelet, M., Pimm, C., Raynal, C., Trzos, M., Arnold, A., and Pronto, D. (2018). A Real-life, French-accented Corpus of Air Traffic Control Communications. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

[Deng et al., 2025] Deng, K., Guo, J., Ma, Y., Moritz, N., Woodland, P. C., Kalinli, O., and Seltzer, M. (2025). Transducer-llama: Integrating llms into streamable transducer-based speech recognition. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

[Denil et al., 2013] Denil, M., Shakibi, B., Dinh, L., and de Freitas, N. (2013). Predicting parameters in deep learning. *Advances in neural information processing systems*, 26.

[Denton et al., 2014] Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*.

[Desplanques et al., 2020] Desplanques, B., Thienpondt, J., and Demuynck, K. (2020). Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification. *arXiv preprint arXiv:2005.07143*.

[Dey et al., 2018] Dey, S., Koshinaka, T., Motlicek, P., and Madikeri, S. (2018). Dnn based speaker embedding using content information for text-dependent speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5344–5348. IEEE.

[Dey et al., 2017] Dey, S., Motlicek, P., Madikeri, S., and Ferras, M. (2017). Template-matching for text-dependent speaker verification. *Speech communication*, 88:96–105.

[Dinh et al., 2017] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, Toulon.

[Fan et al., 2020] Fan, Z., Li, M., Zhou, S., and Xu, B. (2020). Exploring wav2vec 2.0 on speaker verification and language identification. *arXiv preprint arXiv:2012.06185*.

[Ferrer et al., 2022] Ferrer, L., McLaren, M., and Scheffer, N. (2022). Speaker recognition using neural networks and conventional methods. volume 30, pages 2433–2448.

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

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Godfrey, 1994] Godfrey, J. (1994). The Air Traffic Control Corpus (ATC0) - LDC94S14A.

[Gong et al., 2014] Gong, Y., Liu, L., Yang, M., and Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. In *ICLR*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Graves et al., 2006] Graves, A., Fernandez, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd International Conference on Machine Learning, ACM*, pages 369–376.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

[Graves and Schmidhuber, 2005] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5–6):602–610.

[Gulati et al., 2020] Gulati, A., Qin, K., Zhang, C., Choi, J., Cai, R., Zoph, B., Chen, Y.-H., Han, M., Wang, C.-C., Chen, J., et al. (2020). Conformer: Convolution-augmented transformer for speech recognition. *In Proc. Interspeech*, pages 5036–5040.

[Gut, 2009] Gut, A. (2009). *An Intermediate Course in Probability*. Springer Publishing Company, Incorporated.

[Han et al., 2015] Han, S., Mao, H., and Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

[Hassibi and Stork, 1993] Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Helmke et al., 2017] Helmke, H., Ohneiser, O., Buxbaum, J., and Kern, C. (2017). Increasing atm efficiency with assistant based speech recognition. In *Proc. of the 13th USA/Europe Air Traffic Management Research and Development Seminar, Seattle, USA*.

[Helmke et al., 2016] Helmke, H., Ohneiser, O., Mühlhausen, T., and Wies, M. (2016). Reducing controller workload with automatic speech recognition. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE.

[Helmke et al., 2018] Helmke, H., Slotty, M., Poiger, M., Herrer, D. F., Ohneiser, O., Vink, N., Cerna, A., Hartikainen, P., Josefsson, B., Langr, D., et al. (2018). Ontology for transcription of atc speech commands of sesar 2020 solution pj. 16-04. In *IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE.

[Hermansky, 1990] Hermansky, H. (1990). Perceptual linear predictive (plp) analysis of speech. *The Journal of the Acoustical Society of America*, 87(4):1738–1752.

[Hermansky et al., 2000] Hermansky, H., Ellis, D. P. W., and Sharma, S. (2000). Tandem connectionist feature extraction for conventional HMM systems. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 3, pages 1635–1638. IEEE.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hinton, Geoffrey et al., 2012] Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George E, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N, et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Hofbauer et al., 2008] Hofbauer, K., Petrik, S., and Hering, H. (2008). The atcosim corpus of non-prompted clean air traffic control speech. In *LREC*.

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *ICML*.

[Hsu and Glass, 2008] Hsu, B.-J. P. and Glass, J. (2008). Iterative language model estimation: Efficient data structure & algorithms. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 841–844, Brisbane, Australia.

[Hsu et al., 2021] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[Hua et al., 2023] Hua, T., Li, X., Gao, S., Hsu, Y.-C., Shen, Y., and Jin, H. (2023). Dynamic low-rank estimation for transformer-based language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9275–9287.

[Inan et al., 2017] Inan, H., Khosravi, K., and Socher, R. (2017). Tying word vectors and word classifiers: A loss framework for language modeling. In *ICLR*.

[Ioffe, 2006] Ioffe, S. (2006). Probabilistic Linear Discriminant Analysis. In *Computer Vision – ECCV 2006*, pages 531–542, Berlin, Heidelberg.

[Jacob et al., 2018] Jacob, B., Das, S. K., Sastry, S., and Mattina, M. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Jaderberg et al., 2014] Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. In *BMVC*.

[Jelinek, 1997] Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. MIT Press.

[Jurafsky, 2000] Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.

[Jurafsky and Martin, 2008] Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, Upper Saddle River, NJ, 2nd edition.

[Karimi Mahabadi et al., 2021] Karimi Mahabadi, R., Henderson, J., and Ruder, S. (2021). Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*.

[Karita et al., 2019] Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., et al. (2019). A comparative study on transformer vs rnn in speech applications. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 449–456. IEEE.

[Karlsson, 1989] Karlsson, J. (1989). Automatic speech recognition in air traffic control: A human factors perspective. *Military and Government Speech Technology*, 1:13–15.

[Kenny, 2010] Kenny, P. (2010). Bayesian speaker verification with heavy-tailed priors. In *Odyssey: The Speaker and Language Recognition Workshop*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kingma and Dhariwal, 2018] Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 181–184. IEEE.

[Kobyzev et al., 2021] Kobyzev, I., Prince, S. J., and Brubaker, M. A. (2021). Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979.

[Kopiczko et al., 2023] Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. (2023). Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

[Kraaij et al., 2005] Kraaij, W., Hain, T., Lincoln, M., and Post, W. (2005). The ami meeting corpus. In *Proc. International Conference on Methods and Techniques in Behavioral Research*.

[Kumar et al., 2024] Kumar, S., Madikeri, S., Nigmatulina, I., Villatoro-Tello, E., Motlicek, P., Pandia, K., Dubagunta, S. P., and Ganapathiraju, A. (2024). Multitask speech recognition and speaker change detection for unknown number of speakers. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12592–12596.

[Lan et al., 2020] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

[LeCun and Bengio, 1995] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[LeCun et al., 1989a] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989a). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[LeCun et al., 1989b] LeCun, Y., Denker, J. S., and Solla, S. A. (1989b). Optimal brain damage. *Advances in neural information processing systems*, 2.

[Lee et al., 2022] Lee, Y., Greenberg, C., Mason, L., and Singer, E. (2022). NIST 2022 language recognition evaluation plan.

[Lei et al., 2014] Lei, Y., Scheffer, N., Ferrer, L., and McLaren, M. (2014). A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1695–1699. IEEE.

[Lin et al., 2020] Lin, Z., Madotto, A., and Fung, P. (2020). Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*.

[Liu et al., 2021] Liu, A. T., Li, S.-W., and Lee, H.-y. (2021). Tera: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366.

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[Ma et al., 2024] Ma, Z., Yang, G., Yang, Y., Gao, Z., Wang, J., Du, Z., Yu, F., Chen, Q., Zheng, S., Zhang, S., et al. (2024). An embarrassingly simple approach for LLM with strong ASR capacity. *arXiv preprint arXiv:2402.08846*.

[Madikeri et al., 2020] Madikeri, S., Tong, S., Zuluaga-Gomez, J., Vyas, A., Motlicek, P., and Bourlard, H. (2020). Pkwrap: a pytorch package for lf-mmi training of acoustic models. *arXiv preprint arXiv:2010.03466*.

[Miao et al., 2015] Miao, Y., Gowayyed, M., and Metze, F. (2015). Eesen: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[Mohri et al., 2002] Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.

[Mohri et al., 2008] Mohri, M., Pereira, F., and Riley, M. (2008). Speech recognition with weighted finite-state transducers. *Springer Handbook of Speech Processing*, pages 559–584.

[Motlicek et al., 2015] Motlicek, P., Dey, S., Madikeri, S., and Burget, L. (2015). Employment of subspace gaussian mixture models in speaker recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4445–4449. IEEE.

[Ohneiser et al., 2021] Ohneiser, O., Sarfjoo, S., Helmke, H., Shetty, S., Motlicek, P., Kleinert, M., Ehr, H., and Murauskas, Š. (2021). Robust Command Recognition for Lithuanian Air Traffic Control Tower Utterances. In *Proc. Interspeech 2021*, pages 3291–3295.

[Oppenheim and Schafer, 1968] Oppenheim, A. V. and Schafer, R. W. (1968). Homomorphic analysis of speech. *IEEE Transactions on Audio and Electroacoustics*, 16(2):221–226.

[Ott et al., 2019] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (Demonstrations)*, pages 48–53.

[Oualil et al., 2015] Oualil, Y., Schulder, M., Helmke, H., Schmidt, A., and Klakow, D. (2015). Real-time integration of dynamic context information for improving automatic speech recognition. In *16th Annual Conference of the International Speech Communication Association*.

[Padi et al., 2020] Padi, B., Mohan, A., and Ganapathy, S. (2020). Towards relevance and sequence modeling in language recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1223–1232.

[Papamakarios et al., 2021] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64.

[Park et al., 2019] Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 1310–1318.

[Peddinti et al., 2017] Peddinti, V., Chen, G., Manohar, V., Ko, T., Povey, D., and Khudanpur, S. (2017). Low latency acoustic modeling using temporal convolution and LSTMs. *IEEE Signal Processing Letters*, 25(3):373–377.

[Peddinti et al., 2015] Peddinti, V., Povey, D., and Khudanpur, S. (2015). Time-delay deep neural networks for real-time speech recognition. *In Proc. Interspeech*, pages 3486–3490.

[Pfeiffer et al., 2020] Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. (2020). Adapterfusion: Non-destructive task composition for transfer learning. In *ACL*.

[Pigeon et al., 2007] Pigeon, S., Shen, W., Lawson, A., and Leeuwen, D. A. v. (2007). Design and characterization of the non-native military air traffic communications database (nnmatc). In *Eighth Annual Conference of the International Speech Communication Association*.

[Povey, 2003] Povey, D. (2003). *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, Cambridge, UK.

[Povey et al., 2010] Povey, D., Burget, L., Agarwal, M., Akyazi, P., Gao, F., Ghoshal, A., Glembek, O., Goel, N., Karafiát, M., Khudanpur, S., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., and Veselý, K. (2010). Subspace Gaussian mixture models for speech recognition. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4330–4333. IEEE.

[Povey et al., 2018] Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohammadi, M., and Khudanpur, S. (2018). Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Interspeech*, pages 3743–3747.

[Povey et al., 2011] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. In *IEEE workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society.

[Povey et al., 2016] Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pages 2751–2755.

[Povey et al., 2014] Povey, D., Zhang, X., and Khudanpur, S. (2014). Parallel training of DNNs with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455*.

[Prasad, 2020] Prasad, A. (2020). Automatic speech recognition engines adapted for embedded platforms. Idiap-Com Idiap-Com-01-2020, Idiap.

[Press and Wolf, 2017] Press, O. and Wolf, L. (2017). Using the output embedding to improve language models. In *EACL*.

[Prince and Elder, 2007] Prince, S. J. and Elder, J. H. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc.

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

[Rabiner and Schafer, 1978] Rabiner, L. R. and Schafer, R. W. (1978). *Digital Processing of Speech Signals*. Prentice-Hall, Englewood Cliffs, NJ.

[Radford et al., 2023] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

[Ravanelli et al., 2021] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., Mori, R. D., and Bengio, Y. (2021). SpeechBrain: A general-purpose speech toolkit. arXiv:2106.04624.

[Reynolds et al., 2000] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10:19–41.

[Riley et al., 2009] Riley, M., Allauzen, C., and Jansche, M. (2009). Openfst: An open-source, weighted finite-state transducer library and its applications to speech and language. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, pages 9–10.

[Sadhu et al., 2021] Sadhu, S., He, D., Huang, C.-W., Mallidi, S. H., Wu, M., Rastrow, A., Stolcke, A., Droppo, J., and Maas, R. (2021). Wav2vec-c: A self-supervised model for speech representation learning. *arXiv preprint arXiv:2103.08393*.

[Sadjadi et al., 2018] Sadjadi, S. O., Kheyrkhah, T., Tong, A., Greenberg, C. S., Reynolds, D. A., Singer, E., Mason, L. P., Hernandez-Cordero, J., et al. (2018). The 2017 NIST language recognition evaluation. In *Odyssey*, pages 82–89.

[Sak et al., 2014] Sak, H., Senior, A. W., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

[Schneider et al., 2019] Schneider, S., Baevski, A., Collobert, R., and Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Shao et al., 2020] Shao, Y., Wang, Y., Povey, D., and Khudanpur, S. (2020). Pychain: A fully parallelized pytorch implementation of lf-mmi for end-to-end asr. *arXiv preprint arXiv:2005.09824*.

[Shi et al., 2023] Shi, J., Berrebbi, D., Chen, W., Chung, H.-L., Hu, E.-P., Huang, W. P., Chang, X., Li, S.-W., Mohamed, A., Lee, H.-y., et al. (2023). Ml-superb: Multilingual speech universal performance benchmark. *arXiv preprint arXiv:2305.10615*.

[Shore et al., 2012] Shore, T., Faubel, F., Helmke, H., and Klakow, D. (2012). Knowledgebased word lattice rescoring in a dynamic context. In *13th Annual Conference of the International Speech Communication Association*.

[Šmídl et al., 2019] Šmídl, L., Švec, J., Tihelka, D., Matoušek, J., Romportl, J., and Ircing, P. (2019). Air traffic control communication (ATCC) speech corpora and their use for ASR and TTS development. *Language Resources and Evaluation*, 53(3):449–464.

[Snyder et al., 2018a] Snyder, D. et al. (2018a). X-vectors: Robust dnn embeddings for speaker recognition. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5329–5333. IEEE.

[Snyder et al., 2018b] Snyder, D., Garcia-Romero, D., McCree, A., Sell, G., Povey, D., and Khudanpur, S. (2018b). Spoken language recognition using x-vectors. In *Odyssey*, volume 2018, pages 105–111.

[Snyder et al., 2019] Snyder, D., Garcia-Romero, D., Sell, G., McCree, A., Povey, D., and Khudanpur, S. (2019). Speaker recognition for multi-speaker conversations using x-vectors. In *ICASSP*.

[Stolcke, 2002] Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP/INTERSPEECH)*, volume 2, pages 901–904, Denver, CO, USA.

[Strang, 2012] Strang, G. (2012). *Linear algebra and its applications*.

[Szoke et al., 2021] Szoke, I., Kesiraju, S., Novotny, O., Kocour, M., Vesely, K., Cernocky, J., et al. (2021). Detecting english speech in the air traffic control voice communication. *arXiv preprint arXiv:2104.02332*.

[Theis et al., 2016] Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations*.

[Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

[Tjandra et al., 2022] Tjandra, A., Choudhury, D. G., Zhang, F., Singh, K., Conneau, A., Baevski, A., Sela, A., Saraf, Y., and Auli, M. (2022). Improved language identification through cross-lingual self-supervised learning. In *Proc. of ICASSP*, pages 6877–6881. IEEE.

[Valk and Alumäe, 2021] Valk, J. and Alumäe, T. (2021). VoxLingua107: a dataset for spoken language recognition. In *Proc. IEEE SLT Workshop*.

[Vanderreydt et al., 2023] Vanderreydt, G., Prasad, A., Khalil, D., Madikeri, S., Demuynck, K., and Motlicek, P. (2023). Parameter-efficient tuning with adaptive bottlenecks for automatic speech recognition. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–7.

[Vaswani et al., 2017a] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. *Advances in neural information processing systems*, 30.

[Vaswani et al., 2017b] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017b). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Veselỳ et al., 2013] Veselỳ, K., Ghoshal, A., Burget, L., and Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Interspeech*, volume 2013, pages 2345–2349.

[Viterbi, 1967] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

[Vyas et al., 2021a] Vyas, A., Madikeri, S., and Bourlard, H. (2021a). Comparing CTC and LFMMI for Out-of-Domain Adaptation of wav2vec 2.0 Acoustic Model. In *Interspeech 2021*, pages 2861–2865.

[Vyas et al., 2021b] Vyas, A., Madikeri, S., and Bourlard, H. (2021b). Lattice-free mmi adaptation of self-supervised pretrained acoustic models. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6219–6223.

[Vyas et al., 2021c] Vyas, A., Madikeri, S., and Bourlard, H. (2021c). Lattice-free mmi adaptation of self-supervised pretrained acoustic models. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6219–6223. IEEE.

[Waibel et al., 1989] Waibel, A., Hanazawa, H., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

[Wang et al., 2020] Wang, W., Tang, Q., and Livescu, K. (2020). Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6889–6893. IEEE.

[Wang et al., 2019] Wang, Y., Chen, T., Xu, H., Ding, S., Lv, H., Shao, Y., Peng, N., Xie, L., Watanabe, S., and Khudanpur, S. (2019). Espresso: A fast end-to-end neural speech recognition toolkit. In *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 136–143.

[Watanabe et al., 2018] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., and Ochiai, T. (2018). ESPnet: End-to-end speech processing toolkit. In *Interspeech*, pages 2207–2211.

[Yao et al., 2023] Yao, Z., Guo, L., Yang, X., Kang, W., Kuang, F., Yang, Y., Jin, Z., Lin, L., and Povey, D. (2023). Zipformer: A faster and better encoder for automatic speech recognition. *arXiv preprint arXiv:2310.11230*.

[Young et al., 2002] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). The htk book. *Cambridge university engineering department*, 3(175):12.

[Zhang et al., 2023] Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. (2023). Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

[Zuluaga-Gomez et al., 2023] Zuluaga-Gomez, J., Prasad, A., Nigmatulina, I., Sarfjoo, S. S., Motlicek, P., Kleinert, M., Helmke, H., Ohneiser, O., and Zhan, Q. (2023). How does pre-trained wav2vec 2.0 perform on domain-shifted asr? an extensive benchmark

on air traffic control communications. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 205–212. IEEE.

[Zuluaga-Gomez et al., 2020] Zuluaga-Gomez, J., Veselỳ, K., Blatt, A., Motlicek, P., Klakow, D., Tart, A., Szöke, I., Prasad, A., Sarfjoo, S., Kolčárek, P., et al. (2020). Automatic call sign detection: Matching air surveillance data with air traffic spoken communications. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 59, pages 1–14.