AVIGNON UNIVERSITÉ

THÈSE DE DOCTORAT D'AVIGNON UNIVERSITÉ

École Doctorale n°536
Agrosciences & Sciences

Mention de doctorat :
Informatique

Laboratoire Informatique d'Avignon

Présentée par
Ryan WHETTEN

Efficient Self-Supervised Learning for Speech Processing

Soutenue publiquement le 10 juillet 2026 devant le jury composé de :

Hung-yi LEE	Professeur, National Taiwan University	Rapporteur
Shinji WATANABE	Associate Professor, Carnegie Mellon University	Rapporteur
Marcely ZANON BOITO	Ingénieure de recherche, NAVER LABS Europe	Examinatrice
Elodie GAUTHIER	Ingénieure de recherche, Orange Innovation	Examinatrice
Benjamin LECOUTEUX	Professeur, Université Grenoble Alpes, LIG	Examinateur
Yannick ESTÈVE	Professeur, Avignon Université, LIA	Directeur de thèse
Titouan PARCOLLET	Chercheur, Samsung AI, Cambridge	Co-encadrant de thèse
Marco DINARELLI	Chargé de recherche, CNRS, LIG	Co-encadrant de thèse

ACKNOWLEDGEMENTS

The decision to do a PhD was a big one, and the decision to do it in France changed my life completely. I'm extremely grateful to all those who have helped me along this journey, and I take this opportunity to thank you all.

Thank you, Shinji Watanabe and Hung-yi Lee, for taking the time to review my thesis. You are both an inspiration to me, and I enjoyed the conversations we had at conferences.

Thank you also to Marcely Zanon Boito, Elodie Gauthier, and Benjamin Lecouteux for being on my thesis committee and for providing feedback on my work.

I am grateful to my advisors who have given me this opportunity. Thank you, Yannick Estève, for always going above and beyond to help me not just navigate the PhD, but life in France. Thank you, Titouan Parcollet, for all the help in learning SpeechBrain and for the rapid, direct feedback on my work. Thank you, Marco Dinarelli, for supporting me in Grenoble and for always reminding me of the basics.

I also have to acknowledge my colleagues, Jarod Duret, Haroun Elleuch, Eva Feillet, Felix Herron, Lucas Maison, and Adel Moumen. I thoroughly enjoyed working with all of you.

I now address my family, particularly my mom and my brother. Thank you so much for your support. I appreciate all the reminders of where I come from, who I am now, and how to persevere.

Lastly, I want to give a huge thanks to my wife, Rachael. Pursuing a PhD is not easy, and it can be even harder on the people who have to support us through all the ups and downs. Thank you for being a true champion on this adventure with me.

We thank the French Agence Nationale de la Recherche (ANR) E-SSL project N°ANR-22-CE23-0013 for funding, and the Institut du Développement et des Ressources en Informatique Scientifique (dossiers AD011014732 and A0131013821 GENCI) for high-performance computing resources.

ABSTRACT

One of the primary objectives of artificial intelligence is the development of systems capable of interacting with humans via speech, our most innate form of communication. In recent years, substantial progress has been made through self-supervised learning (SSL)—a machine learning paradigm where targets for training a model are derived directly from the input itself, thereby mitigating the reliance on human-labeled data. While self-supervised speech models have achieved state-of-the-art results across diverse tasks—including automatic speech recognition, speaker verification, and speech-to-text translation—they are notoriously resource-intensive. Training these models requires vast amounts of memory, hundreds of thousands of hours of audio, and extensive time, even when training on high-performance computing clusters.

These resource requirements act as significant barriers, limiting the development and deployment of such models. Furthermore, even for those with sufficient access to compute resources, the prohibitive cost of training limits experimental freedom and the ability to explore novel architectures or training objectives. Additionally, the substantial energy consumption inherent in training large-scale SSL models raises environmental concerns.

This thesis addresses these efficiency challenges through multiple approaches, focusing on pre-training duration, inference speed, hardware constraints, data requirements, and evaluation methodologies. Specifically, we first open-source an exceptionally efficient, high-performance SSL framework for speech: BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ). Second, we propose architectural modifications that replace inefficient multi-head self-attention blocks with more streamlined alternatives while maintaining performance. Third, we investigate data selection methods that enable effective pre-training on only a fraction of the original data volume. Fourth, we propose novel, unsupervised evaluation methods that significantly reduce the need for GPU resources and labeled datasets during model assessment. Finally, we present a case study on Fongbe, a low-resource language of Benin, where our approach matches or exceeds state-of-the-art performance while requiring 600 times less data and a 1,000-fold reduction in compute cost, effectively democratizing SSL for low-resource languages.

Keywords : Self-Supervised Learning, Speech Representations, Efficiency, Deep Learning

RÉSUMÉ

L'un des objectifs fondamentaux de l'intelligence artificielle est le développement de systèmes capables d'interagir avec les humains par la parole, notre forme de communication la plus naturelle. Ces dernières années, des progrès considérables ont été réalisés grâce à l'apprentissage auto-supervisé (SSL, self-supervised learning)—un paradigme d'apprentissage automatique dans lequel les cibles d'entraînement d'un modèle sont dérivées directement des données d'entrée elles-mêmes, réduisant ainsi la dépendance aux données annotées par des humains. Bien que les modèles de parole auto-supervisés aient atteint des performances de pointe dans diverses tâches—notamment la reconnaissance automatique de la parole, la vérification du locuteur et la traduction parole-texte—ils sont notoirement gourmands en ressources. Leur entraînement requiert d'immenses quantités de mémoire, des centaines de milliers d'heures d'audio, ainsi qu'un temps considérable, même sur des infrastructures de calcul haute performance.

Ces exigences en ressources constituent des obstacles majeurs, limitant le développement et le déploiement de tels modèles. De plus, même pour ceux disposant d'un accès suffisant à ces ressources, le coût prohibitif de l'entraînement restreint la liberté d'expérimentation et la possibilité d'explorer de nouvelles architectures ou de nouveaux objectifs d'apprentissage. Par ailleurs, la consommation énergétique importante inhérente à l'entraînement des modèles SSL à grande échelle soulève des préoccupations environnementales.

Cette thèse s'attaque à ces défis d'efficacité en traitement auto-supervisé de la parole en adoptant une approche holistique du développement de modèles. Nous nous intéressons à plusieurs dimensions de l'efficacité, notamment la durée du pré-entraînement, la vitesse d'inférence, les contraintes matérielles, les besoins en données et les méthodologies d'évaluation. Plus précisément, nous mettons d'abord en open source un cadre SSL pour la parole exceptionnellement efficace et performant : BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ). Deuxièmement, nous proposons des modifications architecturales remplaçant le mécanisme inefficace de multi-head self-attention par des alternatives plus légères, tout en maintenant les performances. Troisièmement, nous étudions des méthodes de sélection de données permettant un pré-entraînement efficace avec seulement une fraction du volume initial de données. Quatrièmement, nous proposons de nouvelles méthodes d'évaluation non supervisées qui réduisent considérablement les besoins en ressources GPU et en jeux de données annotés lors de l'évaluation des modèles. Enfin, nous présentons une étude de cas sur le fongbé, une langue à faibles ressources du Bénin, dans laquelle notre approche atteint ou dépasse les performances de l'état de l'art tout

en nécessitant 600 fois moins de données et une réduction d'un facteur 1 000 du coût de calcul, démocratisant ainsi l'apprentissage auto-supervisé pour les langues à faibles ressources.

Mots-Clefs : Apprentissage auto-supervisé, Représentations de la parole, Efficacité, Apprentissage profond

Table of Contents

I Context 15

1 Introduction 17
- 1.1 Motivation . . . 19
- 1.2 Dissertation overview . . . 20

2 An Overview of Self-Supervised Learning for Speech 23
- 2.1 Representation learning . . . 24
- 2.2 Unsupervised versus self-supervised learning . . . 25
  - 2.2.1 Self-supervised learning for speech . . . 25
- 2.3 Components of self-supervised speech models . . . 26
  - 2.3.1 Acoustic feature extractor . . . 26
  - 2.3.2 Context encoder . . . 28
  - 2.3.3 Producing and predicting pseudo-targets . . . 29
- 2.4 Benchmarking and fine-tuning . . . 30
- 2.5 Recent self-supervised learning frameworks for speech . . . 31
  - 2.5.1 wav2vec 2.0 . . . 31
  - 2.5.2 HuBERT . . . 32
  - 2.5.3 WavLM . . . 34
  - 2.5.4 data2vec . . . 34
  - 2.5.5 w2v-BERT . . . 35
  - 2.5.6 BEST-RQ . . . 36
- 2.6 Inefficiencies of self-supervised speech models . . . 37
  - 2.6.1 Runtime and memory . . . 37
  - 2.6.2 Data . . . 40
  - 2.6.3 Evaluation . . . 42

II Contributions 43

3 Preliminary Study of Random-projection Quantization 45

3.1 Introduction . . . 46
3.2 Background . . . 47
3.2.1 Comparing wav2vec 2.0 and BEST-RQ . . . 47
3.2.2 Other self-supervised frameworks . . . 49
3.3 Implementation details . . . 49
3.3.1 Acoustic feature extractor and encoder . . . 50
3.3.2 Random-projection quantizer and masking . . . 50
3.4 Methods . . . 51
3.4.1 Pre-training and model settings . . . 51
3.5 Results . . . 53
3.5.1 Downstream tasks and fine-tuning . . . 54
3.5.2 Impact of the masking ratio . . . 54
3.6 Conclusion . . . 55

4 Increasing Efficiency with Attention Substitutes 57
4.1 Introduction . . . 58
4.2 Background . . . 59
4.3 Attention alternatives in BEST-RQ . . . 61
4.3.1 Fastformer . . . 61
4.3.2 HyperMixing . . . 62
4.3.3 SummaryMixing . . . 62
4.3.4 Mamba . . . 63
4.3.5 Implementation in BEST-RQ . . . 63
4.4 Methods . . . 64
4.4.1 Self-supervised pre-training . . . 64
4.4.2 Downstream evaluation . . . 65
4.5 Results . . . 66
4.5.1 Speed and memory evaluation . . . 66
4.5.2 Downstream tasks . . . 66
4.6 Conclusion . . . 69

5 Data Selection Strategies for Efficient Pre-training 71
5.1 Introduction . . . 72
5.2 Related Work . . . 73
5.2.1 Data selection for supervised speech processing . . . 73
5.2.2 Data selection for pre-training data . . . 74
5.3 Methods . . . 74
5.3.1 Data . . . 75

5.3.2 Data selection methods . . . 75
5.3.3 Training settings . . . 76
5.4 Results . . . 77
5.5 Conclusion . . . 79

6 Efficient Evaluation Strategies for Self-Supervised Speech Models 81
6.1 Introduction . . . 82
6.2 Background . . . 83
6.2.1 Architecture of BEST-RQ . . . 83
6.2.2 Pre-training loss vs downstream performance . . . 83
6.2.3 Using rank for unsupervised evaluation . . . 84
6.3 Methods . . . 85
6.3.1 Unsupervised measures for SSL model evaluation . . . 85
6.3.2 Experimental settings . . . 86
6.4 Results . . . 87
6.4.1 Preliminary experiments . . . 87
6.4.2 Automatic speech recognition . . . 88
6.4.3 Speaker verification . . . 89
6.4.4 Additional observations . . . 90
6.5 Conclusion . . . 91

7 Efficient Self-Supervised Learning for Speech in Low-Resource Settings 93
7.1 Introduction . . . 94
7.2 Related Work . . . 95
7.2.1 Training efficiency . . . 96
7.2.2 SSL speech models for low-resource languages . . . 96
7.2.3 Low-data pre-training . . . 97
7.2.4 Fongbe speech recognition and translation . . . 98
7.3 Methods . . . 98
7.3.1 Data . . . 98
7.3.2 Experiment Setting . . . 100
7.3.3 Fine tuning and evaluation . . . 101
7.4 Results . . . 101
7.5 Conclusion . . . 103

8 Conclusion 105

Appendix 111

Acronyms **113**

Personal Publications **115**

References **117**

List of Figures

2.1 Main components of self-supervised speech models for pre-training and downstream phases. Adapted from [Mohamed et al. 2022] . . . 27

2.2 Overview of reported GPU consumption for pre-training various self-supervised speech models. . . . 38

3.1 Diagrams of the core components of our BEST-RQ implementation and wav2vec 2.0 frameworks. The key differences are highlighted in red. BEST-RQ operates on mel-spectrograms, uses a static quantizer and Conformer layers. On the other hand, wav2vec 2.0 operates on raw audio, trains the quantizer, and uses Transformer layers. 48

4.1 Diagram of the BEST-RQ framework and a Conformer layer. In each Conformer layer, we replace the multi-head self-attention module with linear-complexity alternatives. . . . 64

4.2 Inference speed and peak VRAM vs. input length. Comparison between standard MHSA and linear alternatives for BEST-RQ base and large configurations. MHSA requires significantly more time and VRAM as input size increases, whereas the alternatives do not. . . . 67

5.1 Box-and-whisker plots of utterance length distributions across different dataset splits. The three plots on the left correspond to the fine-tuning split, while the remaining plots show the various pre-training subsets. Notably, the best performing models were pre-trained on subsets consisting of longer utterances (far right two plots), whose distributions differ most from the fine-tuning data. . . . 77

6.1 The effective rank of embeddings produced by the SSL model relative to the amount of audio used. The effective rank plateaus early, at less than 20 minutes of audio, and increases only marginally thereafter. . . . 87

6.2 Correlations between 4 selected measures (calculated on embeddings at 50k pre-training steps) and the future WER on the LibriSpeech *dev-clean* set (at 200k pre-training steps and 20 epochs fine-tuning). Each point represents a model with different hyper-parameters (batch size and mask percentage). Fig. 6.2a, shows that some of the models with the lowest WER (y-axis) have a relatively high pre-training loss (x-axis) resulting in a low correlation, whereas unsupervised measures have a stronger correlation with WER. 88

List of Tables

2.1 Overview of self-supervised speech models and their pre-training data. 41

3.1 Summary of downstream tasks, datasets, and probes used for model evaluation in Chapter 3. (FT) denotes the pre-trained model is fine-tuned. 53

3.2 Results on downstream tasks at 100k and 200k steps. BEST-RQ performs similarly to wav2vec but with less than half of the training time. Training time (measured in GPU hours) and number of parameters are shown under *GPU (h)* and *# Par.* respectively. 53

3.3 Results of fine-tuning on ASR with LibriSpeech *train-100*. 54

3.4 Impact of changing the masking ratio and the codebook size. The mask % concerns the start frame index chosen for the mask. Since the selected frame and the following three frames are also masked the actual masking ratio is four times as big. 55

4.1 Summary of downstream tasks, datasets, and probes. All SSL models are frozen except for the ASR Fine-tuning (FT) task. 65

4.2 Results of various alternatives to MHSA for speech SSL models on the MP3S benchmark. Base models and large models (denoted -LG) have around 94M and 315M parameters respectively. Linear complexity alternatives perform competitively with multi-head self-attention (MHSA) even surpassing depending on the task and probe. 67

4.3 Results with fine-tuning on LibriSpeech *train-100*. For the *test-clean* set, we report 95% bootstrapped confidence interval (C.I.). 68

5.1 Automatic Speech Recognition results with different pre-training subsets from the Loquacious medium and large splits. All models are fine-tuned on the small split. The *Time* and *Data* columns report the amount GPU hours and pre-training audio used. 78

6.1 Pearson Correlations between downstream performance and various unsupervised measures. Compared to pre-training loss (first row), unsupervised metrics tend to exhibit a much stronger correlation with downstream performance at either 50k or 200k steps of pre-training. Highest correlations are found in the in-domain tasks. Measures based on rank are more correlated with downstream performance than clustering measures. LS = LibriSpeech, CV = CommonVoice, VoxCel = VoxCeleb, (f.t.) = SSL Model is fine-tuned. 89
6.2 Pearson Correlations between various unsupervised measures and **future** downstream performance. The unsupervised measures are calculated from the embeddings of SSL models at 50k pre-training steps and the correlation is measured with their respective final performances at 200k pre-training steps. Correlations for in-domain ASR remains high, but not for out-of-domain tasks. LS = LibriSpeech, CV = CommonVoice, VoxCel = VoxCeleb, (f.t.) = SSL Model is fine-tuned. 90

7.1 Pre-training datasets. We create a minimal 140 hour Fongbe only pre-training dataset and combine this with audio from various sources to create larger pre-training datasets. *Eng.* and *Fre.* denote English and French. 99
7.2 Character Error Rate (CER) and Word Error Rate (WER) on Fongbe ASR with and without diacritics (lower is better). The *Data* column indicates the volume of pre-training data. Our models are pre-trained for 100k steps, except for *Fon-EFHY 400k*, which was trained for 400k steps. Parentheses indicate results achieved with augmentation during pre-training. 102
7.3 Table of BLEU scores for speech-to text-translation (higher is better). *P.T.*, *ASR*, and *ASR+NLLB* are models starting from pre-trained checkpoints, fine-tuned ASR checkpoints, and fine-tuned ASR and NLLB checkpoints, respectively. *Casc.* denotes the cascade approach. *Aug* denotes that augmentation was used in pre-training. Our models are pre-trained for 100k steps except *Fon-EFHY 400k*, which was pre-trained to 400k. 103
7.4 Pre-training efficiency comparison 103

Part I

Context

Chapter 1

Introduction

Table of Contents

1.1 Motivation 19
1.2 Dissertation overview 20

From the automatons of the Greek god Hephaestus to George Lucas's C-3PO, the idea of talking with machines has been fantasized in science fiction for many years [Silverman 2022; Lucas 1977]. Only in the most recent decade have innovations from a variety of domains begun to come together to start realizing ideas that were once thought of as fiction. These innovations involve both software and hardware improvements which, together with an increase in the amount of accessible data, have led to the development of algorithms that can process natural language.

In past decades, the approach to natural language processing has shifted from rule-based systems to data-driven machine learning methods. In contrast to traditional programming—where developers must manually define logic for every potential input—machines are now provided with vast datasets and trained to identify underlying statistical patterns. This allows models to capture the immense variability of written and spoken language more effectively than human-defined heuristics [Goodfellow et al. 2016]. In the case of transcribing speech, rather than manually programming thousands of acoustic, phonetic and linguistic rules, the model is fed audio paired with transcriptions and is trained to learn the mapping between audio and text.

While traditional machine learning often relies on relatively simple models, deep learning employs neural networks with many layers to progressively extract higher-level features from raw input. Typically, these models required massive amounts of human-annotated data to function—a bottleneck in many fields. To overcome this, self-supervised learning (SSL) has emerged as a transformative paradigm [Mohamed et al. 2022]. Unlike supervised methods that require explicit human labels (such as manual transcriptions), a self-supervised approach allows a model to learn directly from the inherent structure of the input data. By generating its own pseudo-labels from the data itself the model learns to produce high-quality, compressed representations in a phase known as pre-training.

After pre-training, self-supervised models are typically further trained using a minimal amount of data with human labels for a specific downstream task in a second phase called fine-tuning. In the speech domain, these tasks can include speech recognition, speech translation, spoken language understanding, emotion recognition, speaker verification, and many more. For this manuscript, we define a speech model to be any model that processes or produces speech for any task.

Self-supervised speech models have had a significant impact on the field of speech processing. They have led to state-of-the-art performance across a wide variety of tasks, such as speech recognition [Baevski et al. 2020; Hsu et al. 2021; Zhang et al. 2023; Chen et al. 2024], speaker identification [Li et al. 2025], and emotion recognition [Goncalves et al. 2022]. These models have also been shown to be effective for low-resource languages and domains [Zanon Boito et al. 2024; Alabi et al. 2025] and are even used to train neural audio codecs, which can then be used in speech language models [Défossez et al. 2024].

Despite the advancements that self-supervised speech models have brought to the field, there are growing concerns about their efficiency, which motivates the studies presented in this thesis.

1.1 Motivation

As the performance of speech models has improved through SSL, so too has the scale of the models, the volume of data required, and the computational hardware necessary for training. It is now common for self-supervised speech models to contain hundreds of millions of parameters, which require thousands of hours of audio and anywhere from 32 to several hundred high-end GPUs for training [Baevski et al. 2020; Hsu et al. 2021]. At the extreme, these models reach over one billion parameters and are trained on millions of hours of data [Zhang et al. 2023; Chen et al. 2024]. While these models have demonstrated remarkable potential, their development and practical application are often constrained not by the complexity of human speech, but by the amount of data and the compute budgets.

The computational inefficiency of training and evaluating self-supervised speech models limits the ability of smaller research groups, companies, and individuals to develop models. This centralization often places high-performing speech technology under the control of a handful of large corporations, while the scale of these models can also restrict deployment options such as on-device inference. Even for well-resourced entities, these inefficiencies narrow the scope of experiments that can be performed, slowing the exploration of new approaches. Accordingly, improving efficiency is not only an engineering objective, but also a scientific one, as it enables broader experimentation.

The high data requirements are similarly prohibitive. While SSL was designed to reduce dependence on labeled data, it has shifted much of that burden toward massive unlabeled corpora, with substantial storage and processing costs. Beyond these costs, the prevailing brute-force strategy of scaling training data as much as possible often leads to diminishing returns and can even degrade performance [Parcollet et al. 2024a]. Furthermore, as it is common practice to use as much data as possible, multilingual pre-training datasets remain heavily biased toward a small number of high-resource languages, leaving much of the linguistic long tail underrepresented. Consequently, these models often perform worse on low-resource languages, effectively restricting high-performing speech technologies to data-rich languages. Reducing the data requirements makes it possible to train models that perform well on low-resource languages at a lower cost.

Lowering these computational and data costs would increase accessibility, enabling broader participation in developing and deploying speech technologies while expanding the feasibility of on-device inference for privacy-preserving applications. More efficient models could also reduce dependence on centralized compute infrastructure and lower the environmental impact associated with large-scale training and inference. Ultimately, efficiency is not only a matter of reducing cost, but also of promoting accessibility, privacy, and sustainability.

1.2 Dissertation overview

In this thesis, we address the inefficiencies of self-supervised speech models through a multi-faceted approach including the SSL framework, model architecture, data requirements, and evaluation methods. We propose novel methods to improve the efficiency of each of these aspects. We characterize efficiency through GPU memory requirements (VRAM), pre-training data requirements, and the temporal costs of training, inference, and evaluation. Our goal is to make self-supervised speech models more accessible and practical for a wider range of applications and users.

This dissertation is divided into two main sections, Section I: Context and Section II: Contributions. Section I consists of Chapters 1 and 2. Section II consists of Chapters 3 through 7. This work assumes a foundational understanding of neural network architectures (multi-layer perceptrons, convolutional networks, recurrent networks, and Transformers) and optimization. Accordingly, we dedicate our discussion to modern SSL methods specifically tailored for speech.

In Chapter 2 we provide a background of representation learning, including SSL, and we review the different components of modern self-supervised speech models as well as their objectives for training, defining a taxonomy for self-supervised speech models. Using this, we then review current state-of-the-art models, classify them using the defined taxonomy and highlight the problem of efficiency.

In Chapter 3, the beginning of Section II, we open-source a highly efficient SSL framework for speech models, namely BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ). We run preliminary experiments to study the efficiency and performance on speech recognition, speaker identification, intent classification, and emotion recognition.

In Chapter 4 we adjust the architecture of BEST-RQ to address the inefficiencies associated with the quadratic complexity in multi-head self-attention. We replace multi-head self-attention with linear complexity alternatives, specifically HyperMixing, SummaryMixing, Fastformer, and Mamba.

In Chapter 5 we explore data selection methods to study what characteristics of data are most beneficial for pre-training. Data selection methods are based on a variety of criteria, such as acoustic conditions, speaker characteristics, linguistic content, and utterance length.

In Chapter 6 we address the inefficiencies of evaluation, proposing efficient unsupervised evaluation measures to gauge the quality of a model during pre-training. These measures are based on the cluster quality and the rank of the embeddings produced by self-supervised speech models.

In Chapter 7 we perform a case study of specialized pre-training for Fongbe, a low-resource language spoken in West Africa, focusing on speech recognition and translation. We explore the efficiency of pre-training and fine-tuning with limited data, comparing it to a state-of-the-art model for African languages.

In Chapter 8 we conclude by summarizing the findings throughout the dissertation and highlighting remaining challenges and defining a trajectory for future work.

Chapter 2

An Overview of Self-Supervised Learning for Speech

Table of Contents

2.1 Representation learning . . . 24
2.2 Unsupervised versus self-supervised learning . . . 25
2.2.1 Self-supervised learning for speech . . . 25
2.3 Components of self-supervised speech models . . . 26
2.3.1 Acoustic feature extractor . . . 26
2.3.2 Context encoder . . . 28
2.3.3 Producing and predicting pseudo-targets . . . 29
2.4 Benchmarking and fine-tuning . . . 30
2.5 Recent self-supervised learning frameworks for speech . . . 31
2.5.1 wav2vec 2.0 . . . 31
2.5.2 HuBERT . . . 32
2.5.3 WavLM . . . 34
2.5.4 data2vec . . . 34
2.5.5 w2v-BERT . . . 35
2.5.6 BEST-RQ . . . 36
2.6 Inefficiencies of self-supervised speech models . . . 37
2.6.1 Runtime and memory . . . 37
2.6.2 Data . . . 40
2.6.3 Evaluation . . . 42

This chapter introduces the background necessary for understanding modern self-supervised learning (SSL) methods for speech processing. We begin by defining representation learning and discussing the relationship between unsupervised learning and SSL, accompanied by a historical overview of speech representation learning.

Following this, we describe the principal architectural components commonly used in self-supervised speech models and methods for benchmarking and fine-tuning these models on downstream tasks. Together, these concepts provide the foundation necessary for understanding how SSL is applied to speech processing tasks. We then review recent SSL frameworks for speech. Lastly, we discuss the efficiency challenges of current SSL methods for speech which are the motivation of this thesis.

2.1 Representation learning

In order to understand SSL for speech processing, it is first necessary to introduce the concept of representation learning. Representation learning refers to techniques that automatically discover useful features from raw data with the goal of improving machine learning performance [Goodfellow et al. 2016]. This approach represents a paradigm shift from traditional speech recognition pipelines, which historically relied on a rigid decoupling of feature extraction and acoustic modeling.

Historically, speech processing was dominated by generative statistical models, most notably the Hidden Markov Model (HMM) paired with Gaussian Mixture Models (GMMs) [Rabiner 1989]. In these systems, representation was strictly manual; domain experts designed handcrafted features such as Mel-frequency Cepstral Coefficients (MFCCs) or log-Mel filterbank coefficients to capture the phonetic nuances of the audio signal. While effective for their time, GMM-HMM systems were difficult to scale, required complex forced-alignment to map audio to labels, and struggled to generalize across diverse acoustic environments.

With the rise of deep neural networks (DNNs), or neural networks with many layers, the landscape began to shift. This transformation initially manifested in speech as DNN-HMM hybrid architectures [Hinton et al. 2012]. Although DNNs replaced GMMs for learning more robust representations from MFCCs, the systems remained complex and still relied on the HMM framework for temporal modeling. To simplify this, the field moved toward End-to-End (E2E) architectures, such as Connectionist Temporal Classification (CTC), Attention-based Encoder-Decoders (AED), and RNN Transducers (RNN-T) [Graves et al. 2006; Chan et al. 2016; Rao et al. 2017]. These E2E models, particularly when empowered by the Transformer architecture [Zhang et al. 2020b], allowed for the direct mapping of input sequences to output text.

However, these E2E breakthroughs introduced new challenges: they are computationally expensive and require vast quantities of human-annotated data to reach high performance. Fur-

thermore, their supervised nature often leads to limited generalization when applied to unseen domains or low-resource languages. This led to the exploration of SSL methods for speech processing tasks.

2.2 Unsupervised versus self-supervised learning

Unsupervised learning refers to machine learning methods that learn from data without human-annotated labels. The goal of unsupervised learning is typically to discover hidden structure or patterns within data without explicit supervision. Classical examples include clustering algorithms such as k-means [Arthur et al. 2007] and dimensionality reduction techniques such as principal component analysis (PCA) [Jolliffe et al. 2016].

Self-supervised learning (SSL) is closely related to unsupervised learning as it also does not require human labels. In this thesis, we treat SSL as a form of unsupervised learning. However, unlike classical unsupervised methods, SSL computes supervisory labels directly from the input data itself. These automatically computed targets, often called pseudo-targets or pseudo-labels, allow models to be trained using objectives similar to those used in supervised learning.

The primary objective of SSL is not necessarily to interpret hidden structure in the data, but rather to learn representations that are useful for downstream tasks. After pre-training, these learned representations can either be directly used as features or further optimized on supervised downstream tasks. The initial SSL stage is commonly referred to as *pre-training*, while the subsequent supervised training phase is referred to as *fine-tuning*. The pre-training objective can also be referred to as the upstream task, while the supervised task can be referred to as the *downstream* task. The pre-training and downstream phases are illustrated in Figure 2.1.

2.2.1 Self-supervised learning for speech

While SSL is by no means a new concept, its application to speech has necessitated unique architectural considerations and innovations in optimizing deep neural networks due to the continuous and high-frequency nature of audio signals. Many successful approaches in speech SSL were influenced by contrastive learning, a framework where the model learns to distinguish between a true (positive) data sample and several distractor (negative) samples in a latent space [Oord et al. 2018].

One of the proposed methods was wav2vec [Schneider et al. 2019]. Unlike previous speech systems that relied on log-Mel filterbanks, wav2vec utilized a fully convolutional architecture to operate directly on raw audio waveforms. Its objective involved predicting future frames of audio by contrasting them against randomly sampled frames from the same utterance. This demonstrated that meaningful acoustic representations could be learned without any human annotation, simply by exploiting the temporal structure of speech.

Borrowing paradigms from NLP, specifically the Transformer architecture [Vaswani et al. 2017] and the Masked Language Modeling (MLM) objective introduced by BERT [Devlin et al. 2019], led to the development of wav2vec 2.0 [Baevski et al. 2020]. MLM is a SSL concept where a portion of the input data is masked, and the model learns to predict the masked portion based on the unmasked context. In wav2vec 2.0, this involves contrastive learning, in which a certain proportion of latent representations of the audio input are masked and the model is trained to identify the correct representation for the masked position from a set of distractors. wav2vec 2.0, along with other recent SSL frameworks for speech, will be described in more detail in Section 2.5.

The impact of wav2vec 2.0 was profound, achieving state-of-the-art performance on benchmarks like LibriSpeech [Panayotov et al. 2015] using only minutes of labeled data. Beyond speech recognition, these representations have proven highly transferable to other tasks such as speaker identification and emotion recognition [Mohamed et al. 2022]. Many subsequent works in speech SSL share core ideas and components introduced by wav2vec 2.0, such as the use of contrastive learning and MLM.

Despite these successes, modern SSL methods for speech are highly inefficient. These models require significant GPU memory and extensive training time, often spanning hundreds or thousands of GPU hours. Furthermore, While labels are not needed, these models often require immense amounts of unlabeled audio to converge effectively.

To understand these inefficiencies, it is essential to examine the components and training processes of self-supervised speech models in detail. We will explore these components in the next section, which will provide the necessary background to understand the efficiency challenges of current SSL frameworks for speech.

2.3 Components of self-supervised speech models

In this section, we define the main architectural components of self-supervised speech models. We divide the components into three parts: the acoustic feature extractor (AFE), the encoder, and lastly, everything else that has to do with producing and predicting targets. We consider AFE and the encoder to be the primary components of the models as they are typically the only parts of the pre-training stage needed for fine-tuning and inference. A figure highlighting these core components is shown in Figure 2.1.

2.3.1 Acoustic feature extractor

We broadly define the AFE as a function that transforms a raw acoustic signal into a latent representation suitable for downstream tasks or modeling. These extractors can be categorized into handcrafted functions, learned neural layers, or hybrid architectures that combine both paradigms.

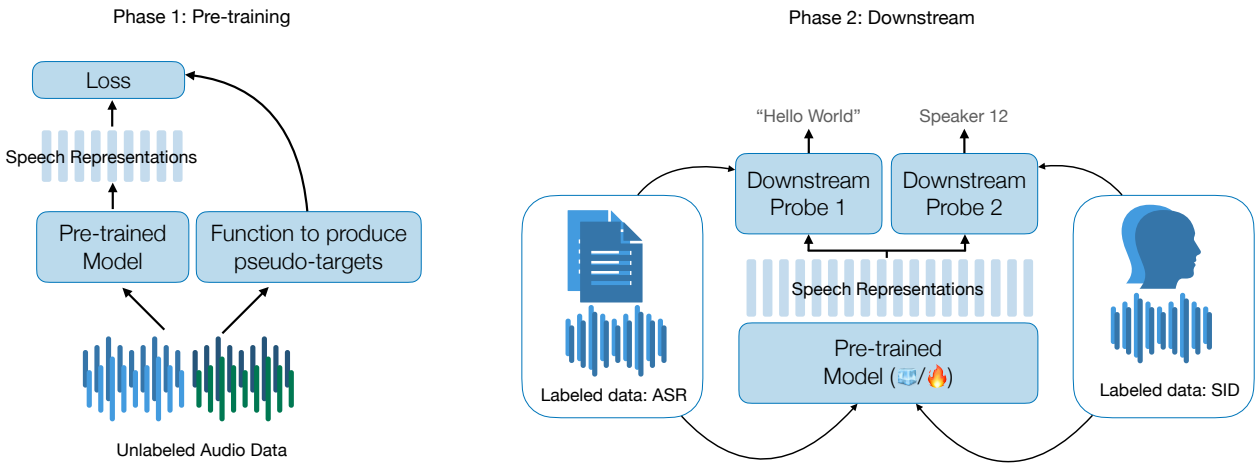


Figure 2.1: Main components of self-supervised speech models for pre-training and downstream phases. Adapted from [Mohamed et al. 2022]

The necessity of feature extraction stems from the fact that raw acoustic signals are extremely high-dimensional. Modern digital audio is typically sampled at rates between 16 and 48 kilohertz (kHz), resulting in 16,000 to 48,000 data points per second. For speech processing, where the preservation of extremely high-frequency components is less critical than in music, a sampling rate of 16 kHz is standard. Given this density, the primary goal of the AFE is to reduce the signal's complexity while simultaneously transforming it into a representation that highlights phonetic and prosodic characteristics.

Historically, handcrafted AFE dominated the field. These utilize fixed mathematical transformations, such as the Short-Time Fourier Transform (STFT), to produce Mel-frequency Cepstral Coefficients (MFCCs) [Hinton et al. 2012; Chung et al. 2018]. While these methods are computationally efficient and grounded in the physics of human hearing, they are increasingly less common as standalone extractors in end-to-end systems.

Conversely, the widespread adoption of DNNs has driven the move toward fully learned feature extractors. These are typically implemented as a multi-layer temporal convolutional neural network (CNN) [Schneider et al. 2019; Baevski et al. 2020]. In these architectures, the first layer uses a small kernel size and stride to process raw waveforms, effectively learning a bank of filters that mimic or improve upon traditional frequency-domain transforms. Many speech SSL frameworks like wav2vec 2.0 [Baevski et al. 2020], HuBERT [Hsu et al. 2021], and WavLM [Chen et al. 2022] rely on these convolutional blocks to downsample the 16 kHz signal into a more manageable latent space (typically at a frequency of around 50 Hz). While powerful and capable of capturing task-specific acoustic cues, these fully learned extractors are computationally demanding and require significant memory during the pre-training phase.

To balance between the handcrafted methods and convolutional feature extractors, hybrid ex-

tractors have emerged as a robust alternative. These systems provide log-Mel filterbank features as input to a shallow convolutional front-end such as in the w2v-BERT and BEST-RQ frameworks [Chung et al. 2021; Chiu et al. 2022]. By leveraging fixed spectral features as an initial bias, the model can bypass the computationally expensive task of learning basic frequency decompositions from scratch. This hybrid approach often results in faster training and lower GPU memory requirements, making it a common choice where efficiency is a primary concern.

2.3.2 Context encoder

The context encoder is the primary computational component of the architecture, typically comprising the vast majority of the learnable parameters in the model. It receives the latent representations from the AFE as input. In the context of speech SSL, the role of the encoder is to model the long-range dependencies of the speech signal. While early frameworks implemented this encoder using Convolutional Neural Networks (CNNs) [Schneider et al. 2019] or Recurrent Neural Networks (RNNs) [Oord et al. 2018], recent research has converged on Transformer-based architectures [Vaswani et al. 2017]. Transformers utilize self-attention mechanisms to represent the relationships between various parts of an input sequence, which is particularly effective for speech as it is inherently sequential and context-dependent. One common derivative of the Transformer designed for speech processing is the Conformer [Gulati et al. 2020], which integrates convolutional blocks to better capture both local and global dependencies.

The training of the context encoder typically relies on the concept of MLM. While the original wav2vec [Schneider et al. 2019] did not utilize this approach, it has become a cornerstone of subsequent frameworks. During training, the input from the AFE is corrupted in some fashion and the encoder must then use the remaining uncorrupted signal to reconstruct or identify the missing information in some fashion. In frameworks such as wav2vec 2.0, HuBERT, and data2vec, this corruption is achieved by replacing specific spans of the input with a learned mask embedding vector [Baevski et al. 2020; Hsu et al. 2021; Baevski et al. 2022]. Alternatively, the BEST-RQ framework simplifies this process by replacing the frames selected for masking with random noise [Chiu et al. 2022]. This objective forces the encoder to develop high-level representations that can bridge gaps in the acoustic signal. This reconstruction can be literal and regressive[1], as in data2vec [Baevski et al. 2022], or indirect and contrastive, as in wav2vec 2.0 [Baevski et al. 2020].

Despite their representational power, Transformer-based encoders introduce significant practical challenges. These architectures are notoriously computationally expensive, exhibiting quadratic complexity with respect to the sequence length due to the self-attention mechanism. Consequently, they require substantial GPU memory and long training times to reach optimal performance. Furthermore, to avoid overfitting and to learn robust features, these models require vast amounts of unlabeled data, making the pre-training phase a resource-intensive endeavor.

[1] In this context, regressive refers to a continuous-valued objective, such as Mean Squared Error.

2.3.3 Producing and predicting pseudo-targets

In order to train the AFE and encoder, a few more components are required, namely one for computing the pseudo-labels and another for calculating the loss between the predicted and pseudo-labels.

Producing useful pseudo-labels, or targets, is an active area of research with a variety of proposed methods. As mentioned previously, one common theme in many frameworks is the use of masking or MLM. The idea of MLM comes from text tasks where a portion of the text is hidden, and during pre-training the model's job is to predict the hidden portions of text. The mask can be a specific token. For example, in BERT [Devlin et al. 2019], text is segmented into tokens, usually between the character and the word level. The tokens, plus an additional few special tokens, are the dictionary of the model. The mask is one of these additional special tokens. In autoregressive models, the mask is implicit in the nature of the task, such as in the GPT series where the model's job is to predict the next token in the sequence [Radford et al. 2019]. Whether using an explicit mask token or not, the original tokens before masking become the model's pseudo-targets.

However, unlike text data, which is discrete and can be easily tokenized, speech signals are continuous and highly redundant, meaning that in the raw audio signal (and even in spectrograms), there is a high correlation from one data point to the next. As a result, solving a masked task is trivial, as the model can easily use the surrounding frames to predict the masked ones. Thus, it is common for modern frameworks for speech SSL to work with spanned masked sections [Baevski et al. 2020; Baevski et al. 2022]. While masking spans of audio makes the task less trivial, computing pseudo-targets for the masked areas is, due to the continuous nature of speech, not as obvious as it is for text (where one normally would use the token index as a target).

Targets for these masked spans of audio can be computed in various ways. The simplest way is to use the input directly as the target. This is commonly used in autoregressive frameworks [Chung et al. 2019; Chung et al. 2020]. Another method involves employing a quantizer, which is a discretization component that converts a raw audio signal into discrete tokens. One method for doing this include K-means clustering, where a K-means model is trained and the cluster IDs of a given portion of audio are used as the discretized signal [Hsu et al. 2021]. The quantizer can also involve codebooks. In this case, a codebook is a matrix that contain continuous values. A given portion of continuous input is compared with each row in the codebook using a distance metric, and the index (or indices in the case of using multiple codebooks) of the row closest to the input is used as the discrete value to represent that portion of input [Baevski et al. 2020]. Lastly, a *teacher-student* approach can be used where the *teacher* model receives the entire unmasked input and produces targets for the *student* model, which sees only a masked version. The teacher's output can be discretized, with K-means [Chen et al. 2022], or can be left as continuous if the learning objective is regressive [Baevski et al. 2022].

Besides masking spans of the input signal, another peculiarity of SSL frameworks for speech

processing compared to text is that it is not uncommon for targets to be refined throughout training. Codebooks can be updated, teachers can be refined, and clustering models can be retrained with the goal of obtaining better targets for pre-training.

In order for the acoustic feature extractor and encoder to learn from the targets, a light projection head is normally added after the encoder. The projection head is a linear layer that converts the output of the encoder into a form that can be used in conjunction with the pseudo-targets to calculate the loss value. While these additional components are typically not large in terms of the number of parameters, the process of computing the targets and loss, as well as the additional operations to refine targets (i.e. updating codebooks, refining teacher models, and re-computing clustering) can introduce additional complexity and inefficiencies. We discuss the details in more depth for specific frameworks in Section 2.5.

2.4 Benchmarking and fine-tuning

Once the self-supervised pre-training phase is complete, the resulting model must be evaluated to determine the quality and utility of its learned representations. This transition from pre-training to downstream application involves a structural change to the architecture: typically, only the AFE and encoder are used after pre-training. The components used solely for the pre-training objective—such as the projection head, quantizers, or teacher models—are discarded. A diagram illustrating the pre-training and downstream phase is shown in Figure 2.1.

In the place of these discarded components, a minimal task-specific architecture is appended to the top of the encoder. The nature of this addition and the subsequent training regime generally fall into two categories: fine-tuning and probing.

In the fine-tuning paradigm, a downstream model (such as a linear layer with a Connectionist Temporal Classification (CTC) [Graves et al. 2006] objective for speech recognition) is added to the encoder. During this phase, the weights of the AFE and encoder are not held constant; instead, they are updated alongside the downstream model using labeled data. This allows the model to shift its internal representations to better suit a specific task, such as Automatic Speech Recognition (ASR) or Intent Classification (IC).

Alternatively, the AFE and encoder can be utilized as a frozen feature extractor. In this setup, the pre-trained weights are fixed, and audio representations are extracted from the model's layers to be fed into a probe—a lightweight model trained to solve a specific task.

A common practice for probing self-supervised speech models is to use a learned weighted sum of the hidden states from all encoder layers, rather than just the final layer. This is motivated by the observation that different layers capture different acoustic and linguistic information (e.g., lower layers often capture phonetic detail while higher layers capture semantic content) [Pasad et al. 2021; Chen et al. 2022]. This frozen evaluation method is the standard for major benchmarking

protocols, such as SUPERB (Speech processing Universal PERformance Benchmark) [Yang et al. 2021] and MP3S (Multi-Probe Speech Self-Supervision Benchmark) [Zaiem et al. 2023], which both cover a wide array of tasks.

By keeping the pre-trained components frozen, researchers can isolate the quality of the pre-trained representations themselves, ensuring that the performance on downstream tasks is a direct reflection of the SSL objective rather than the capacity of the downstream model to learn the task from scratch. In Chapter 3 and 4, we will use this frozen evaluation with tasks from MP3S to evaluate our methods.

2.5 Recent self-supervised learning frameworks for speech

In this section, we now review specific SSL frameworks and neural architectures. While this is not an exhaustive list, we choose representative high-performing recent frameworks and architectures that are referenced throughout this thesis.

2.5.1 wav2vec 2.0

We have discussed briefly wav2vec 2.0 [Baevski et al. 2020] previously and how it builds upon the original wav2vec model [Schneider et al. 2019]. In this section, we go into more detail.

wav2vec 2.0 is a framework which learns contextualized speech representations with a contrastive learning objective. The AFE consists of 7 convolutional layers and the encoder consists of a series of Transformer blocks, typically either 12 or 24 in practice. The encoder learns contextualized speech representations as it captures information from the entire sequence. For discretizing the audio to compute pseudo-labels, wav2vec 2.0 employs a quantizer that consists of multiple codebooks.

The output of the AFE is discretized with product quantization [Jegou et al. 2010], where multiple codebooks are used and an entry is selected from each codebook. These entries are concatenated together, resulting in a single vector for each frame. These vectors become pseudo-targets for the contrastive learning objective. The codebooks are updated with Gumbel-Softmax [Jang et al. 2017]. Concurrently, a masked version of the acoustic features is passed to the encoder. The output of the encoder is then projected into a feature space to match the dimensions of the codebook.

For the contrastive loss, the model maximizes the similarity between the masked features and their corresponding pseudo-targets while minimizing the similarity to negative samples. This is achieved through InfoNCE [Oord et al. 2018]. However, because the codebook is updated during training, the pseudo-targets could change to become something trivial for the encoder to learn. Thus, a diversity loss is introduced to encourage the model to explore different codebook entries and prevent it from collapsing to a single solution.

Formally the contrastive loss function can be expressed with the following:

$$\mathcal{L}_c = -\log \frac{\exp(\text{sim}(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\exp(\text{sim}(\mathbf{c}_t, \mathbf{q}_t)/\kappa) + \sum_{i=1}^{K} \exp(\text{sim}(\mathbf{c}_t, \tilde{\mathbf{q}}_i)/\kappa)} \tag{2.1}$$

where $\mathbf{c}_t$ is the contextualized representation of a masked input signal at frame t (i.e. the output of the encoder at masked position t), $\mathbf{q}_t$ is the positive quantized target for frame t, and $\tilde{\mathbf{q}}_i$ are the negative quantized targets (distractors uniformly sampled from other masked frames from the same utterance). $\text{sim}(u, v)$ is the cosine similarity: $\frac{u^T v}{\|u\|\|v\|}$. κ is a temperature hyperparameter to control the sharpness of distribution.

The diversity loss encourages entries in the codebook(s) to be uniformly distributed. This is achieved by maximizing the entropy of the distribution over codebook entries for each codebook. To make this part of the loss function, we minimize the average negative entropy. Formally, the diversity loss can be expressed as:

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} (-H(\bar{p}_g)) = \frac{1}{GV} \sum_{g=1}^{G} \left(\sum_{v=1}^{V} \bar{p}_{g,v} \log(\bar{p}_{g,v}) \right) \tag{2.2}$$

where G is the number of codebooks, V is the number of entries per codebook, and $H(\bar{p}_g)$ is the entropy of the distribution over codebook entries for group g. The total loss is the simply the sum of the contrastive and diversity losses in the following way:

$$\mathcal{L}_{wav2vec\ 2.0} = \mathcal{L}_c + \alpha \mathcal{L}_d \tag{2.3}$$

where α is a hyperparameter to control the weight of the diversity loss.

By adapting MLM to the speech domain through the masking of latent feature spans, wav2vec 2.0 effectively bridges the gap between acoustic and linguistic representation. This approach of integrating MLM with a contrastive learning objective and a powerful architecture (a CNN-based AFE and a Transformer encoder) allows the model to effectively learn rich representations from raw audio data.

Despite its representative power, the architectural complexity of the AFE and encoder in wav2vec 2.0 imposes substantial computational overhead during training. Furthermore, the reliance on a multi-objective loss function (comprising both contrastive and diversity terms) increases the number of computations and can lead to training instability. This makes the framework highly sensitive to hyperparameter configurations, often requiring precise tuning to ensure convergence and avoid performance degradation [Lee et al. 2024].

2.5.2 HuBERT

Hsu et al. 2021 proposed the HuBERT framework which learns contextualized speech representations by predicting masked portions of the input with a predictive learning objective.

The main model architecture follows that of wav2vec 2.0; the AFE and encoder are the same, consisting of 7 convolutional layers followed by Transformer blocks. However, the process of computing and predicting pseudo-targets is different.

In HuBERT, pseudo-targets are initially obtained by clustering 39-dimensional MFCC features using a KMeans clustering model that has been trained offline on a random 10% portion of the dataset. Like wav2vec 2.0 the mask is applied after the feature extraction so that the encoder receives a masked input. The model is trained to predict the pseudo-targets of the masked portions of the input. After a period of training, these pseudo-targets are then refined where a new KMeans model is trained on the latent representation of a given layer of the encoder. This iterative clustering process helps to improve the quality of the pseudo-targets over time.

To calculate the loss the HuBERT framework uses a learned embedding table, where the entries represent the cluster IDs. The output of the encoder is projected to have the same dimensionality as the embedding table. Then the loss is calculated as the cross-entropy loss between the similarity of the projected vector and the embedding table, pushing output of the model and their corresponding entries in the embedding table (as defined by the clustering process) closer together while pushing other entries apart.

Let $\mathbf{c}_t$ be the output of the encoder for a masked input at a given frame t, and let $\mathbf{E}$ be the embedding table. The loss can be expressed as as the cross-entropy loss:

$$\mathcal{L}_{HuBERT} = -\sum_{t \in T} \log p(\mathbf{e}_t | \mathbf{c}_t) \tag{2.4}$$

where p is the probability of the correct entry, $\mathbf{e}_t$, given the context representation $\mathbf{c}_t$. The index of $\mathbf{E}$ that denotes $\mathbf{e}_t$ is calculated offline with KMeans. This probability is calculated using a softmax function over the cosine similarity between $\mathbf{c}_t$ and each entry in the embedding table $\mathbf{E}$:

$$p(\mathbf{e}_i | \mathbf{c}_t) = -\log \frac{\exp(\mathsf{sim}(\mathbf{c}_t, \mathbf{e}_i))}{\sum_{\mathbf{e}_j \in \mathbf{E}} \exp(\mathsf{sim}(\mathbf{c}_t, \mathbf{e}_j))} \tag{2.5}$$

where $\mathsf{sim}(u, v)$ is the cosine similarity.

Hsu et al. 2021 also perform various ablations to explore the effects of including the predictions of non-masked portions in the loss function as well as the use of an ensemble of KMeans models. However, the official implementation uses one KMeans model per iteration and only includes the predictions of masked portions in the loss function as this led to the improved performance.

By using KMeans clustering, the HuBERT framework effectively removes potential codebook instabilities present in wav2vec2.0 and obtains start-of-the-art performance. However, the iterative clustering process introduces additional computational overhead and complexity to the training process requiring one to stop training to redo the KMeans clustering. This also adds additional hyperparameters such as the number of iterations, the number of clusters, and which layer to use for clustering. Additionally, HuBERT has the same computational complexities from the wav2vec 2.0 framework, namely the use of a 7-layered convolutional AFE and Transformer blocks.

2.5.3 WavLM

Chen et al. 2022 proposed the WavLM framework which is based on HuBERT and has a predictive learning objective to learn contextualized speech representations through MLM. Similar to HuBERT and wav2vec 2.0, WavLM consists of a 7-layered convolutional AFE and an encoder built of Transformer blocks. However, in the Transformer blocks gated relative position bias (grep) [Chi et al. 2022] is incorporated.

For computing pseudo-targets, Chen et al. 2022 start by clustering latent representations from an already pre-trained HuBERT model. They report using the 6th and 9th layers of first- and second-iteration HuBERT models for the latent representations. Additionally, WavLM incorporates a noise-mixing strategy during pre-training to improve the robustness of the learned representations. To do this, they randomly select a portion of utterances to be mixed with noise. The noise is then randomly sampled from noise audio or from a secondary utterance. A mixing energy ratio is sampled and then the noise is added to the selected utterances with the selected energy ratio. They crop the noise to ensure the overlap with the original utterance is less than 50%. Thus, the encoder receives the noisy and masked speech whereas the targets are computed with the clean unmasked speech.

The loss function is the same as that in HuBERT, but, due to the added noise, the model is forced to learn not only to use the surrounding context to predict the masked targets, but also how to denoise audio in order to predict the correct pseudo-targets.

While WavLM achieves state-of-the-art performance on various benchmarks, it suffers from the computational complexities from HuBERT. Additionally, the noise-mixing strategy adds additional computational overhead during pre-training.

2.5.4 data2vec

Baevski et al. 2022 propose data2vec as a general framework for SSL for text, image, and speech tasks with a regressive learning objective. While the framework is proposed for various modalities, the proposed framework is not multi-modal in the sense that a single model does receive multi-modal input. Instead under this framework, multiple models are trained for each modality separately.

The architecture of data2vec for speech tasks is similar to the aforementioned models. The AFE consists of 7 convolutional layers and the encoder is built of Transformer blocks. However, data2vec does not use a quantizer or codebook to obtain pseudo-targets.

Instead, data2vec uses a teach-student approach where the teacher and the student share the same architecture and the Transformer blocks of the teacher are an exponential moving average of the student model. The teacher and student share the weights for the feature extractor and positional encoding for efficiency. The teacher model receives the complete input while the input

for the student in masked.

The targets are the average of the last K layers of the encoder of the teacher. As the targets are derived from the hidden layers of the teacher model, these targets contain contextual information due to the self-attention in the Transformer blocks. This is in contrast to wav2vec 2.0 and the first iteration of HuBERT in which targets do not contain global contextual information. However, due to the exponential moving average of the teacher the pseudo-targets are refined throughout training, similar to the codebook in wav2vec 2.0 and the iterative refinement in HuBERT. The student is trained to predict these continuous targets with a scaled L2 norm loss as defined below:

$$\mathcal{L}_{data2vec} = \frac{1}{\sqrt{D}} \sum_{b=1}^{B} \sum_{t=1}^{T} \|\mathbf{y}_{b,t} - \mathbf{c}_{b,t}\|^2 \tag{2.6}$$

where $\mathbf{y}_{b,t}$ are the target representations and $\mathbf{c}_{b,t}$ are the context representations at a given batch b and frame t. D is the size of the hidden dimensions, B is the batch size, and T is the number of frames. The loss is scaled by $\frac{1}{\sqrt{D}}$ to prevent the loss from becoming too large as the hidden dimension increases[2].

While data2vec achieves competitive performance with the previous mentioned frameworks, it suffers from the same computational complexities of these models due to the use of a large CNN feature extractor and Transformer encoder. Additionally, the teacher-student approach adds additional computational overhead as two models must be maintained during training.

2.5.5 w2v-BERT

Thus far, we have examined frameworks utilizing contrastive (wav2vec 2.0), predictive (HuBERT and WavLM), or regressive (data2vec) objectives. These models typically share a common architectural backbone: a 7-layer convolutional AFE followed by Transformer encoder blocks. In contrast, w2v-BERT [Chung et al. 2021] unifies contrastive and predictive learning within a single framework while introducing significant modifications to the feature extractor and encoder architectures.

Where previous models operate on raw audio via deep convolutional extractors, w2v-BERT processes 80-dimensional log-Mel filterbank coefficients. The extractor is simplified to two 2D-convolutional layers (with 2×2 strides) and a linear projection. The architecture is further distinguished by the use of Conformer blocks, which integrate a convolution module directly after the self-attention layer to better capture local and global dependencies [Gulati et al. 2020].

Additionally, w2v-BERT does not use an explicit learned mask tokens, opting instead to mask input spans by replacing them with random noise. The training architecture is designed to support a multi-objective optimization framework. The unmasked outputs from the convolutional extractor are

[2]Loss formula derived from fairseq from lines 474–496. `https://github.com/facebookresearch/fairseq/blob/main/examples/data2vec/models/data2vec_text.py`

processed by a quantization module similar to wav2vec 2.0 where the quantized representations are used as targets for the contrastive loss. These entries serve a dual purpose: they act as the targets for the contrastive loss and their respective indices serve as the pseudo-labels for the predictive loss at the end.

The contrastive loss is applied after the N-th Conformer block, utilizing the same contrastive and diversity objectives—and the same quantization-based target selection—as wav2vec 2.0. Subsequently, the remaining Conformer blocks function as a MLM encoder. The predictive loss is applied at the final layer, where a softmax layer produces a probability distribution over the codebook vocabulary. Following the general pattern of MLM, the framework optimizes the model to predict the specific discretized IDs (derived from the earlier quantization step) that correspond to the masked sections of the original input.

The final loss is the sum of the contrastive and predictive losses and can be expressed with the following:

$$\mathcal{L}_{w2v-BERT} = \mathcal{L}_{wav2vec\ 2.0} + \mathcal{L}_p \tag{2.7}$$

where $\mathcal{L}_{wav2vec\ 2.0}$ is the wav2vec 2.0 loss defined in Equation 2.3 and $\mathcal{L}_p$ is the cross-entropy loss for the masked language modeling task.

w2v-BERT enjoys the benefits of both contrastive and predictive learning objectives reaching high performance for ASR and Voice Search tasks. Furthermore, w2v-BERT contains a more simple and efficient feature extractor by starting with log-Mel filterbank coefficients instead of the raw audio waveforms.

However, the use of two separate objectives introduces additional computational overhead during training. Additionally, the use of Conformer blocks, while powerful, adds to the computational complexity of the model.

2.5.6 BEST-RQ

Chiu et al. 2022 propose BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ), a framework with a predictive learning objective. The architecture of the model BEST-RQ is similar to w2v-BERT in that the input to the model is 80-dimensional log-Mel filterbanks coefficients, the AFE is two 2D-convolutional layers, strides (2,2), and the encoder consists of Conformer blocks.

The masking strategy is also similar to w2v2-BERT in that there is no explicit mask token and the frames that are selected for masking are simply replaced with noise sample from the normal distribution with mean 0 standard deviation 0.1. After the encoder, the masked input is passed through a linear projection head to make the final prediction. The loss of the model is the cross-entropy loss between the predicted targets and the actual pseudo-targets produced by the random projection quantizer. However the way the framework creates and predicts targets is different.

For computing pseudo-targets in the BEST-RQ framework, the 80-dimensional log-Mel filterbank coefficients are stacked every 4 frames and then passed through a randomly initialized linear layer. The purpose of stacking frames is to match the time reduction of the convolutional layers layers which reduces the temporal resolution by a factor of 4 (due to having 2 layers each with a stride of 2). The randomly initialized linear layer projects the stacked frames into a lower-dimensional space to match the codebook's dimension of 16. A codebook lookup is performed with a randomly initialized codebook of size 8192 by 16. The pseudo-targets are the index of the codebook vector closest to the projected input features.

Formally, given a stacked input vector $\mathbf{x}_t$ at time t, a Xavier initialized matrix A, and codebook $E = e_1, ..., e_n$, the target label $\mathbf{y}_t$ is calculated using the following formula:

$$\mathbf{y}_t = \text{argmin}_i \, || \, \text{norm}_{l2}(e_i) - \text{norm}_{l2}(A\mathbf{x}_t)||. \tag{2.8}$$

The codebook is randomly initialized from the standard normal distribution. The codebook and the random projection are not updated during training, thus, the pseudo-targets do not change throughout training.

Despite the simplicity of the random projection quantizer, BEST-RQ achieves competitive performance with the aforementioned frameworks. Consequently, this thesis adopts the principles established by BEST-RQ as its core foundation. We detail the reasoning behind this selection and provide the first official open-source recipe for the model in Chapter 3.

2.6 Inefficiencies of self-supervised speech models

Having established the current landscape of SSL, we must now address the primary constraint of these methods: their demand for resources. As the core objective of this thesis is to improve the accessibility and utility of these models, we conclude this chapter by defining the specific axes of efficiency—runtime, data, and evaluation—that will be addressed over the remainder of the thesis.

2.6.1 Runtime and memory

When it comes to runtime efficiency, we refer to the time it takes to *run* a model with a given machine (i.e. with a fixed amount of memory). We are concerned with both training and inference runtime. For pre-training, we measure the number of GPU hours consumed. For inference time, as the runtime will change drastically according to the size of the input, we measure the runtime with various input lengths. Lastly, we also take into account amount of memory (VRAM) and the type of GPUs used for training. While not every paper reports all these details, below we highlight examples from some of the SSL frameworks mentioned previous in Section 2.5.

In the wav2vec 2.0 framework, Baevski et al. 2020 report using 64 and 128 V100 GPUs, and training in 1.6 and 2.3 - 5.2 days for their small (95M parameters) and large models (317M pa-

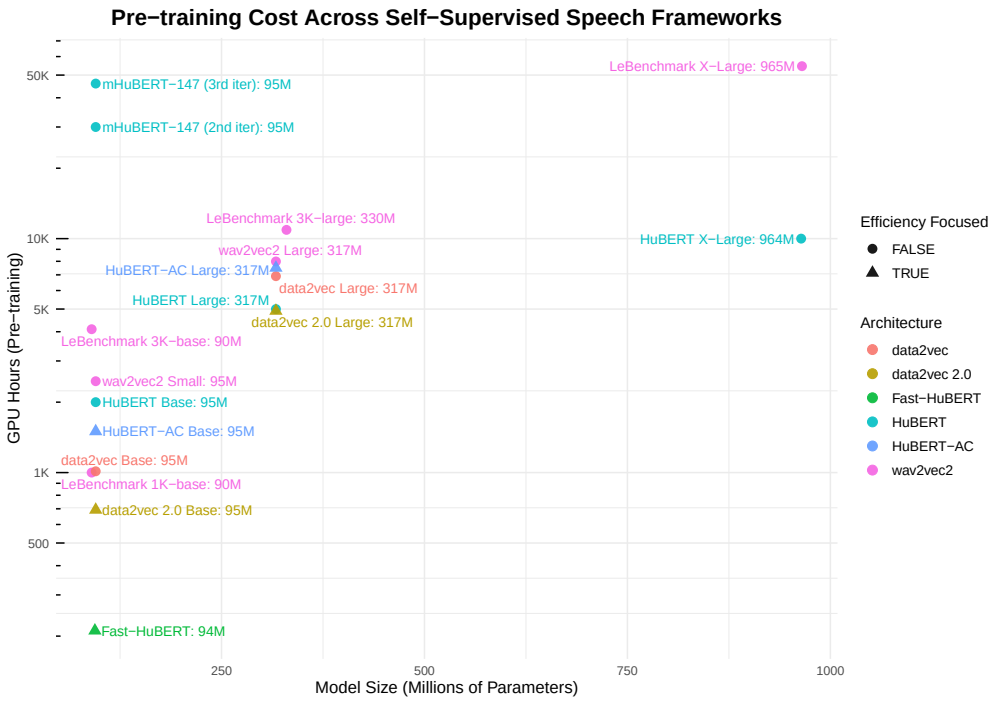


Figure 2.2: Overview of reported GPU consumption for pre-training various self-supervised speech models.

rameters) respectively. This results in a total of 2,458 and 3,532 - 7,987 V100 GPU hours. In LeBenchmark 2.0, Parcollet et al. 2024a applied the wav2vec 2.0 framework to train models for French speech processing tasks. They report pre-training consuming 1,000 to 27,904 V100 GPU hours and 54,600 A100 hours for model sizes that ranged from 90M to 965M parameters.

In HuBERT, Hsu et al. 2021 report using 32, 128, and 256 GPUs depending on model size and that 100k steps took approximately 9.5 hours. This results in a total in approximately 2,000, 5,000, and 10,000 hours to train Base (95M parameters), Large (317M parameters), and X-Large (964M parameters) models respectively. In while for the development of the model WavLM, Chen et al. 2022 do not report the total amount of time for pre-training they note using 32 V100 GPUs for base model training and 64 V100 GPUs for the large. Since their framework and architecture resembles that of HuBERT, theoretically, WavLM requires a similar amount of GPU hours in total or more due to the extra required cost for mixing audio to add noise during pre-training as described in Section 2.5.3. In a multi-lingual variant of HuBERT, mHuBERT-147, Zanon Boito et al. 2024 pre-train a 95M parameter model on 147 languages. They report training that it took approximately 20 days to train one iteration with 32 A100 80GB GPUs or 15,000 A100 GPUs hours. They report performance after the second and third iterations which would be around 30,000 to 46,000 A100 GPU hours.

Even in efforts to reduce the pre-training time we still find a substantial use of resources. In data2vec 2.0, Baevski et al. 2023 make three main changes to improve the pre-training efficiency while maintaining high performance: they do not encode masked tokens, they use a minimal convolutional decoder for the prediction of masked portions, and lastly they reuse the teacher targets for multiple masked representation of each sample. Due to these changes, compared to the original data2vec, the pre-training time was reduced from 63.3 hours to 43.3 hours and 108 to 76.7 hours for base and large models respectively. Base and large models were trained on 16 and 64 A100 GPUs respectively, resulting in a total of 693 hours and 4,909 A100 GPU hours for this optimized version of data2vec. With the same goal of reducing the amount of resources required for pre-training, Chen et al. 2023 work to improve the HuBERT framework in order to be able to train on only 8 GPUs instead of 32. They parallelize KMeans for target production to reduce memory usage per node and, for the larger Libri-Light dataset, they train the KMeans using only LibriSpeech (we will discuss these datasets in Section 2.6.2). During pre-training they use mixed precision (bfloat16) to increase training speed and employ gradient accumulation to increase the effective batch size on only 8 GPUs. Lastly, they experiment with targets from clustering the embeddings of a model already trained for ASR instead of starting with clustering MFCCs, reducing the need for multiple iterations used in the original HuBERT framework. They report using approximately 1,000 A100 GPUs per iteration. Thus, for the base models (95M parameters) it takes 1,000 to 2,000 hours depending on whether the model started from a speech recognition model or from MFCC features (which requires two iterations of training to have comparable performance). For the large models they report consuming an additional 3,000 to 7,500 GPU hours.

Other works based on improving the efficiency of the HuBERT framework include Fast-HuBERT and MelHuBERT which both propose simplifying the AFE and the loss function and they both experiment downsampling [Lin et al. 2023; Yang et al. 2023]. In these works, the AFE is simplified by starting with filterbanks and the loss is simplified to predicting the clustering ID instead of using an embedding table for the cluster IDs (refer to Section 2.5.2 for more details on the original HuBERT loss). With these changes Fast-HuBERT and MelHuBERT become almost identical to BEST-RQ with the main difference be in the use of KMeans versus a random projection for quantization (see Section 2.5.6 for more on BEST-RQ). Downsampling further improves efficiency by reducing the length of the input to the encoder. Additionally in Fast-HuBERT, Yang et al. 2023 demonstrate the benefit of starting with refined targets, achieving best performance by starting with targets clustered from a pre-trained model. Fast-HuBERT was pre-trained on 8 V100s in 1.1 days resulting in a total amount of approximately 211 GPU hours.

To summarize the above literature, pre-training a single speech model requires anywhere from 8 to a few hundred high-end GPUs (i.e. V100 or A100 with 32 to 80 GB of VRAM) and can take up to 50,000 GPU hours (see Figure 2.2). Even in efforts to reduce these requirements, pre-training can still require 8 GPUs and 1,000 or more GPU hours. It must also be noted that in practice it is

not uncommon to train multiple models in order to find the optimal set of hyperparameters and to have models of different sizes (such as base, large, and x-large). Lastly, these reported numbers do not include the cost of fine-tuning. While fine-tuning typically consumes much fewer resources than pre-training and consequently is not always reported, Hsu et al. 2021 reports fine-tuning on 8 GPUs. The computational cost involved with pre-training self-supervised speech models creates barriers for reproducibility and further exploration of self-supervised methods [Chen et al. 2023].

We focus runtime efficiency in Chapters 3, 4, and 7.

2.6.2 Data

Regarding data efficiency, we are primarily concerned with the amount of data used in pre-training. While the amount of data for fine-tuning is also important, previous studies have shown that with a strong SSL framework and vast amounts of pre-training data, one can obtain high performance with a minimal amount of labeled data (i.e. anywhere from 10 minutes to a few hundred hours of audio data) [Baevski et al. 2020; Hsu et al. 2021; Baevski et al. 2022]. Consequently, we focus on this underexplored issue of the amount of pre-training data used.

To experiment with SSL speech frameworks, it is common to employ LibriSpeech [Panayotov et al. 2015] and Libri-Light [Kahn et al. 2020] for pre-training and then fine-tune for ASR on various portions of LibriSpeech [Baevski et al. 2020; Hsu et al. 2021; Baevski et al. 2022]. LibriSpeech is a dataset for speech recognition derived from English audiobooks and contains around 1,000 hours of audio in total. Libri-Light is also derived from English audiobooks and contains 60,000 hours of audio. However, Libri-Light does not have corresponding transcriptions, making it a typical dataset for pre-training. Previous work has shown that increasing the amount of pre-training data from 1,000 hours to 60,000 or more leads to performance gains [Baevski et al. 2020; Chen et al. 2022; Baevski et al. 2022]. While these performance gains are significant, this trend has led to training models on increasingly large amounts of data. For example, large multilingual models like XEUS, w2v-BERT 2.0 v2, and Google USM were trained on 1, 4.5, and 12.5 million hours of audio respectively. As data size increases typically so does batch size, model size, and consequently, GPU hours and memory (VRAM) [Chen et al. 2025], not to mention the increased storage footprint required for multi-terabyte datasets.

Not only does adding more data usually require more resources but the naive addition of more data does not always guarantee higher performance. For example in LeBenchmark 2.0, Parcollet et al. 2024a found that models pre-trained on 3,000 and 7,000 hours of audio reach a similar performance to those trained on 14,000 hours of data for certain ASR, spoken language understanding, and speech translation tasks. Similarly, Le et al. 2026 saw no improvement in ASR and spoken language understanding tasks when increasing the data size from 14 thousand to 114 thousand hours. Furthermore, in experiments working with pre-training on a variety of scraped arabic audio, we find that actually the addition of the data actually prevented model convergence [Elleuch et al.

2026]. This brings into question how much audio is actually needed and what type of audio is *good* audio for pre-training.

There are a few works studying pre-training on extremely small amounts of data where researchers work with as low as 50 hours of pre-training data. Sriram et al. 2022 propose Wav2Vec-Aug, which introduces augmentation to the wav2vec 2.0 framework during pre-training. While performance was not state-of-the-art when pre-training with only 50 hours of LibriSpeech, the results show that when working with limited amounts of data, pre-training with augmentation outperforms pre-training without. Lodagala et al. 2023 propose data2vec-aqc which similarly adds augmentation to pre-training, yet they also add a contrastive loss similar to that of wav2vec 2.0 and a clustering step to cluster and remove less informative negative samples. They pre-train on only 360 hours of LibriSpeech and obtain good performance on ASR tasks.

Overall, aside from a few works specifically focused on data efficiency for self-supervised speech models, it is now standard to pre-train models using at minimum 1,000 hours. Furthermore, it is not uncommon to see models pre-trained on 60 thousand hours or more. See Table 2.1 for a list recent works with the amount of data used in pre-training.

Table 2.1: Overview of self-supervised speech models and their pre-training data.

Model	Parameters	Year	Pre-training Data (hours)
wav2vec 2.0 Base	95M	2020	1,000 (English)
wav2vec 2.0 Large	317M	2020	60,000 (English)
HuBERT Base	95M	2021	1,000 (English)
HuBERT Large	317M	2021	60,000 (English)
WavLM Base	95M	2022	1,000 (English)
WavLM Large	317M	2022	94,000 (English)
w2v-BERT	600M	2021	60,000 (English)
data2vec Base	95M	2022	1,000 (English)
data2vec Large	317M	2022	60,000 (English)
BEST-RQ	600M	2022	60,000 (English)
Wav2Vec-Aug	95M	2022	50 - 960 (English)
w2v-BERT 2.0 v2	600M	2023	4,500,000 (multilingual)
Whisper Large	1550M	2023	680,000 (multilingual)
data2vec-aqc	95M	2023	360 (English)
GOOGLE USM	600M and 2B	2023	12,500,000 (multilingual)
LeBenchmark	1000M	2024	14,000 (French)
mHuBERT-147	95M	2024	90,000 (multilingual)
XEUS	600M	2024	1,000,000 (multilingual)

We focus on data efficiency in Chapters 5.

2.6.3 Evaluation

As explained in Section 2.4, traditionally self-supervised speech models are evaluated by adding on a minimal neural network, or probe, and then by either fine-tuning the entire system or by using the pre-trained model as a feature extractor, meaning that the pre-trained model is frozen. Whether weights are frozen or not, most previous works report performance with a model that has been substantially pre-trained. While there is no consensus on what it means to be fully pre-trained, usually models are trained for 400 thousands to a million updates [Baevski et al. 2020; Chen et al. 2023].

During pre-training it is common to monitor a few metrics such as the pre-training loss depending on the learning objective. For example, if the learning objective involves masked prediction or a contrastive loss, one can measure the accuracy. However, these pre-training metrics do not correspond well with the final downstream performance, making it difficult to judge when a model is fully fine-tuned. Both Chen et al. 2023 and Zanon Boito et al. 2024, note that continuing pre-training even after the pre-training loss has converged increases performance to a certain point. As a result, to compare different training settings hyperparameters, architectures and frameworks, self-supervised speech models must be substantially pre-trained and then subsequently run on downstream tasks.

Alongside these initial pre-training costs, fine-tuning adds extra expenses as it requires additional training with labeled data on GPUs. Although not very many works report in detail how many GPU hours were used for fine-tuning, the authors of wav2vec 2.0, HuBERT, and WavLM provide the number of GPUs used for fine-tuning. For wav2vec 2.0, Baevski et al. 2020 mention fine-tuning on 8 to 24 GPU hours. For HuBERT, Hsu et al. 2021 report fine-tuning on 8 GPUs. For WavLM, Chen et al. 2022 note fine-tuning on 8 and 24 V100 GPUs for the base and large model respectively.

Recently, Aldeneh et al. 2025 propose using a rank based measure which shows to correlate well with downstream performance on phoneme recognition, key word spotting, and speaker identification. While promising, efficient methods for evaluating self-supervised speech model remain a relatively unexplored area. We focus on evaluation efficiency in Chapter 6.

Part II

Contributions

Chapter 3

Preliminary Study of Random-projection Quantization

Table of Contents

3.1 Introduction . . . 46
3.2 Background . . . 47
3.2.1 Comparing wav2vec 2.0 and BEST-RQ . . . 47
3.2.2 Other self-supervised frameworks . . . 49
3.3 Implementation details . . . 49
3.3.1 Acoustic feature extractor and encoder . . . 50
3.3.2 Random-projection quantizer and masking . . . 50
3.4 Methods . . . 51
3.4.1 Pre-training and model settings . . . 51
3.5 Results . . . 53
3.5.1 Downstream tasks and fine-tuning . . . 54
3.5.2 Impact of the masking ratio . . . 54
3.6 Conclusion . . . 55

Publications related to this chapter

- **ICASSPW 2024** – Open Implementation and Study of BEST-RQ for Speech Processing [Whetten et al. 2024a]

Summary

The BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) is a highly efficient frameworks proposed for self-supervised speech models. However, no open-source implementation or empirical efficiency analysis has been released. In this chapter, we describe why BEST-RQ is so efficient, introduce our BEST-RQ implementation, and run preliminary experiments comparing it to wav2vec 2.0 on four downstream tasks. We show that a random-projection quantizer can achieve downstream performance similar to wav2vec 2.0 while reducing training time by more than half.

3.1 Introduction

As discussed in Section 2.2, self-supervised learning (SSL) is a training regime in which a model extracts targets from the input data itself and is trained to predict these targets. SSL systems can thus take advantage of large amounts of unlabeled data in a pre-training stage and use a minimal amount of labeled data to achieve strong results on a wide variety of tasks. In the speech domain, SSL has led to state-of-the-art results in tasks such as Automatic Speech Recognition (ASR), Emotion Recognition (ER), Automatic Speaker Verification (ASV), and Spoken Language Understanding [Yang et al. 2021; Mohamed et al. 2022].

However, SSL pre-training is very data, memory, and computationally expensive. For example, wav2vec 2.0, our baseline for this chapter, was reported to have consumed around 2,400 V100 GPU hours base model and 7,000 for the large [Baevski et al. 2020]. On the high end for very large datasets such as *LeBenchmark*, authors have reported using 54,600 A100 GPU hours for their extra-large model, using the wav2vec 2.0 framework [Parcollet et al. 2024a].

One reason for this high cost is due to the acoustic feature extractors (AFE), which is commonly implemented as a series of convolutional neural network (CNN) layers in state-of-the-art architectures including wav2vec 2.0, HuBERT, WavLM and data2vec [Baevski et al. 2020; Hsu et al. 2021; Chen et al. 2022; Baevski et al. 2022], but studies have shown that these can be replaced with more efficient alternatives while maintaining performance [Parcollet et al. 2023]. Another reason behind this cost is the complexity of the learning objective. As described in Section 2.5 the process of generating pseudo-targets and computing the loss can involve various extra components that need to be trained.

The BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) frame-

work, reduces this cost by simplifying the AFE, the process of generating pseudo-targets, and the loss function. However, no official implementation of BEST-RQ exists from the authors, inhibiting the community from having access to a very efficient SSL training method. Furthermore, BEST-RQ performance has only been investigated on two tasks, ASR and speech translation [Chiu et al. 2022; Zhang et al. 2023]. This work aims to fill in these gaps.

In this chapter, we present our open-source implementation of the BEST-RQ framework using SpeechBrain [Ravanelli et al. 2024] and conduct a comparative analysis against the wav2vec 2.0.[1] Beyond analyzing pre-training efficiency, we evaluate the learned representations across two distinct paradigms. First, we utilize the Multi-probe Speech Self Supervision Benchmark (MP3S) [Zaiem et al. 2023] to assess performance on frozen models across four downstream tasks: ASR, ASV, Intent Classification (IC), and ER. Second, we perform a dedicated fine-tuning on the LibriSpeech *train-clean-100* split to compare how both models adapt to a standard supervised downstream task.

Results show that a random-projection quantizer can achieve similar performance on these various downstream tasks as wav2vec 2.0 with the added benefit of reducing SSL pre-training time by more than half. This open-source implementation can serve as a starting point for furthering research, facilitating exploration into various SSL architectures for speech.

3.2 Background

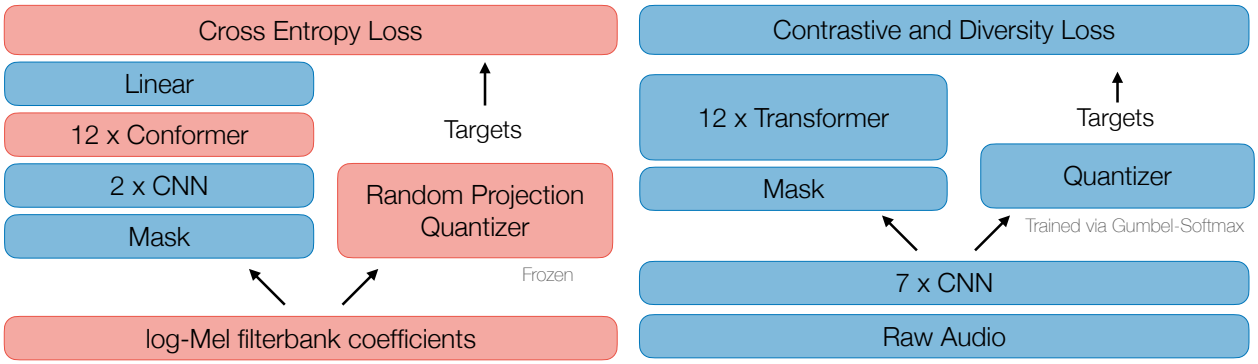
We choose wav2vec 2.0 as a representative SSL framework due to its high performance and widespread adoption. It has served as the foundation for subsequent work such as in LeBenchmark [Parcollet et al. 2024a] and Omnilingual ASR [Omnilingual ASR Team et al. 2025], and many other SSL frameworks are derivatives of wav2vec 2.0, such as data2vec [Baevski et al. 2022], HuBERT [Hsu et al. 2021] and WavLM [Chen et al. 2022] which use the same AFE and encoder as wav2vec 2.0. In this section, we give a detailed comparison of wav2vec 2.0 and BEST-RQ, highlighting the key differences. Then we give a brief comparison to other frameworks.

3.2.1 Comparing wav2vec 2.0 and BEST-RQ

BEST-RQ and wav2vec 2.0 have a few major differences, one of which is that BEST-RQ does not operate on the raw audio. Instead, BEST-RQ operates on 80-dimensional log-Mel filterbank coefficients with 2 convolutional layers for the AFE. In contrast, wav2vec 2.0 operates on the raw audio using 7 convolutional layers for the AFE.

Using a hand-crafted AFE such as log-Mel filterbank coefficients significantly reduces the amount of memory and computation needed to train the model. However, this may limit the per-

[1] Code for our BEST-RQ implementation is available at `https://github.com/speechbrain/speechbrain/tree/develop/recipes/LibriSpeech`



(a) BEST-RQ (b) wav2vec 2.0

Figure 3.1: Diagrams of the core components of our BEST-RQ implementation and wav2vec 2.0 frameworks. The key differences are highlighted in red. BEST-RQ operates on mel-spectrograms, uses a static quantizer and Conformer layers. On the other hand, wav2vec 2.0 operates on raw audio, trains the quantizer, and uses Transformer layers.

formance of the model as filterbanks are compressed views of speech based on psycho-acoustics (i.e. based on human expertise) and, consequently, could remove acoustic cues that a non-hand-crafted AFE can learn.

A second major difference is that BEST-RQ uses a randomly initialized linear layer and one codebook for quantizing and discretizing the audio. These components are both fixed throughout training. The log-Mel filterbank coefficients are projected with the linear layer to have the same dimensionality as the codebook entries. The index of the nearest codebook entry to the projection becomes the discrete pseudo-target for pre-training. The nearest codebook entry is found by taking the $\arg\min$ of the normalized distance between each entry in the codebook and the projection. We define this formally in Equation 2.8.

Afterwards, a mask is applied to portions of log-Mel filterbank coefficients and the objective of the model is to predict the correct targets for the masked sections. This is treated as a classification task using cross-entropy to calculate the loss.

Conversely, in wav2vec 2.0, there are two learned codebooks which are trained via a Gumbel-softmax. The loss function for wav2vec 2.0 is more complicated consisting of a combination of a contrastive loss (i.e. identifying the correct label among a portion of randomly selected false labels) and a diversity loss (to prevent the model from collapsing into using just one entry in the codebook). For more details on the loss function of wav2vec 2.0 see Section 2.5.1.

One last major difference is that BEST-RQ uses Conformer layers instead of Transformer layers. Note that although Conformer layers with wav2vec 2.0 have been studied [Zhang et al. 2020c], for this work we only study the wav2vec 2.0 standard architecture as our goal is not to prove one methodology is better than another, it is instead to show that our implementation works as expected

and compares favorably to a well-known alternative. See Figure 3.1 for a diagram comparing these two architectures. Chiu et al. 2022 demonstrated that competitive ASR performance can be achieved without the need to train a large AFE or codebooks. By keeping these components fixed, the model theoretically benefits from significantly reduced architectural complexity, a simpler loss function, and a more efficient backward pass. However, since the original BEST-RQ article does not contain details on the pre-training run times and resource, the exact extent of these efficiency gains remains unclear. In this work, we empirically quantify how this approach impacts pre-training speed and evaluate its performance on ASR alongside three additional downstream tasks. Finally, to facilitate further research into random-projection quantizers, we provide an open-source implementation of our framework.

3.2.2 Other self-supervised frameworks

While the focus of this work is to perform a preliminary study of our implementation of BEST-RQ with wav2vec 2.0, it is worth noting a few key differences between other self-supervised frameworks, namely HuBERT and data2vec [Hsu et al. 2021; Baevski et al. 2022]. For more details on the HuBERT and data2vec frameworks see Sections 2.5.2 and 2.5.4.

As noted in Chapter 2.5, in both HuBERT and data2vec, the AFE and encoder are the same as in wav2vec 2.0: 7 convolutional layers for the AFE and Transformer layers for the encoder. The key difference in HuBERT is that the pseudo-targets are generated through an offline K-means stage. This stage is repeated multiple times in order to refine the targets. This offline clustering adds complexity and extra hyperparameters such has which value of K to use, which layer to choose for the iterative refinement of the pseudo-targets, and how much training to perform in each iteration. The data2vec framework adds complexity to pre-training due to the need of a teacher network which is an exponentially moving average of the network being trained. Due to the similarity of the architectures and the increased complexity in HuBERT and data2vec, we select wav2vec 2.0 for comparison.

Bringing these components together, the simplification of the AFE, the process of generating targets, and the loss function make BEST-RQ theoretically more efficient than other proposed methods.

3.3 Implementation details

In this section, we describe our implementation of the BEST-RQ framework. While our approach largely follows the methodology proposed by Chiu et al. 2022, we introduced several modifications to account for the absence of low-level details in the original text and to optimize performance for our available computational resources. Most notably, we simplify the masking function to ensure that masked regions are strictly aligned with the pseudo-targets generated by the random-

projection quantizer.

3.3.1 Acoustic feature extractor and encoder

For our implementation, we use the SpeechBrain toolkit Ravanelli et al. 2024. The AFE computes 80-dimensional log-Mel filterbank coefficients using a 25 ms window and a 10 ms hop length, yielding an initial frame rate of 100 Hz. These features are then processed by two convolutional layers, each with a stride of 2. This architecture effectively downsamples the input by a factor of four, resulting in a final encoder frame rate of 25 Hz.

The encoder consists of Conformer layers. The Conformer architecture incorporates convolutional blocks into Transformer layers and has been shown to be effective in various speech processing tasks [Gulati et al. 2020].

3.3.2 Random-projection quantizer and masking

For the random-projection quantizer we follow Chiu et al. 2022, we initialize two matrices, one for the random projection and another for the codebook. Both of these are frozen, meaning they are not updated throughout training. To match the four-fold framerate reduction from the convolutional layers, the random projection is performed on stacks of every four frames of filterbank features. The codebook and the output of the output of projection are normalized with L2 norm, and the index of closest codebook entry becomes the pseudo-target. The closest index is calculate with the L2-norm distance as described in Section 2.5.6 with Equation 2.8. Following the original implementation, the codebook size is set to 8192 by 16, in other words a vocabulary size of 8192 where each entry is a vector of size 16.

We diverge from the implementation in Chiu et al. 2022 when it comes to masking. The original implementation describes a strategy of selecting a masking probability p and a mask length l. Each frame has likelihood of p of getting chosen to be masked and that frame and the subsequent frames are masked to result in a total masking span of l. In this case, masked portions of audio can overlap. Although similar strategies are used in other frameworks [Baevski et al. 2020; Hsu et al. 2021], this method of masking complicates the process of predicting pseudo-targets. Chiu et al. 2022 describe setting p to 0.01 and l to 400ms.

Hsu et al. 2021 showed that predicting only the masked sections during pre-training resulted in the best downstream performance. Following this approach, Chiu et al. 2022 limit their predictions to masked segments. However, given the masking strategy and quantization process described above, pseudo-targets may occasionally correspond to audio regions that are only partially masked. While this detail was not explicitly documented in the original paper, correspondence with the authors clarified that they only consider pseudo-targets where all four constituent frames are fully masked.

Based on these observations, we simplify the masking function to ensure that masks align perfectly with every four frames. We achieve this by sampling mask indices at the reduced 25 Hz frame rate and setting our mask length to four frames on the original log-Mel features. This ensures that each applied mask corresponds exactly to one pseudo-target from the random-projection quantizer. Alongside simplifying the masking logic, this approach is more efficient; we no longer need to calculate or track which segments are fully masked, thereby further reducing the computational overhead.

Following Chiu et al. 2022, we mask these segments using random noise sampled from a normal distribution with a mean of zero and a standard deviation of 0.1. Under this masking method, individual masks cannot overlap, though they may be adjacent, effectively creating longer spans of masked segments.

While this modification to the masking logic represents our primary departure from the BEST-RQ framework, we also adjusted several hyperparameter settings (such as batch and model size) to suit our specific experimental environment. We consider these adjustments specific to this study rather than inherent to the general framework; consequently, they can be easily tuned for other setups. Detailed descriptions of these parameters are provided in Section 3.4.1.

3.4 Methods

To validate our implementation, we pre-train models using both the wav2vec 2.0 framework and our BEST-RQ implementation. We then evaluate these models across two distinct setups. First, we use a subset of the Multi-probe Speech Self-Supervision Benchmark (MP3S) [Zaiem et al. 2023], where the pre-trained encoders remain frozen. Second, we perform full fine-tuning on the LibriSpeech *train-clean-100* subset for downstream ASR. This section details the pre-training environment, the specific model configurations, and the downstream tasks used for evaluation.

3.4.1 Pre-training and model settings

For pre-training, we use 960 hours of LibriSpeech, comprising the *train-clean-100*, *train-clean-360*, and *train-other-500* splits [Panayotov et al. 2015]. All models are trained for 42 epochs (approximately 200k steps) using eight V100 GPUs. To monitor performance during training, we save a checkpoint at epoch 21 (roughly 100k steps) and distinguish between these versions by appending the labels *100k* and *200k* to the model names.

We employ identical dynamic batching settings for both wav2vec 2.0 and BEST-RQ, with a maximum batch size of 100 seconds per GPU. Across eight GPUs, this results in a total batch size of 800 seconds (approximately 13.33 minutes). This configuration was selected to ensure compatibility with memory-constrained GPUs and to complete the experiments within a manageable timeframe of a few hundred GPU hours.

We use the wav2vec 2.0 Base architecture [Baevski et al. 2020] as our baseline, implemented via the SpeechBrain toolkit [Ravanelli et al. 2024]. To ensure a more fair comparison, we modify the standard BEST-RQ configuration to match this baseline. Specifically, we reduce the number of Conformer layers in our BEST-RQ encoder from 24 to 12. This alignment not only ensures that both models have a comparable parameter count but also allows the BEST-RQ model to fit within the memory limits of our hardware.

Our experimental environment is significantly more resource-constrained than the settings described in the original BEST-RQ literature. While Chiu et al. 2022 and Zhang et al. 2023 train with a batch size of approximately 18 hours and training on datasets ranging from 60k to 12 million hours, our setup uses a 13-minute batch size and 960 hours of data. Furthermore, we train on eight GPUs, whereas large-scale pre-training typically employs 32 to several hundred units [Baevski et al. 2020; Hsu et al. 2021].

To compensate for these differences, we introduced the following hyperparameter adjustments based on initial empirical trials:

- **Regularization:** We added a layer dropout of 0.05 to prevent overfitting on the smaller dataset; this aligns with the regularization used in the wav2vec 2.0 baseline.
- **Optimization:** The learning rate was reduced to 0.0008 which helps with convergence.
- **Masking Density:** We increased the mask ratio to approximately 60% (where 15% of frames are randomly selected as starting points for 4-frame masks). This higher masking pressure was found to be beneficial in our low-resource setting, as further discussed in Section 3.5.2.

3.4.1.1 Downstream Evaluation

To evaluate the quality of the representations learned by the BEST-RQ framework, we utilize a subset of tasks from the MP3S benchmark [Zaiem et al. 2023]. In this setup, the pre-trained SSL model remains frozen, and the input to the downstream tasks is a learned weighted sum of its hidden states. This representation is then passed to a minimal task-specific downstream model, or probe.

We select four representative tasks to cover a range of speech dimensions: content (ASR), speaker identity (ASV), semantics (IC), and paralinguistics (ER). The tasks, datasets, and specific probe architectures used in this chapter are summarized in Table 3.1. While the MP3S benchmark typically evaluates multiple probes per task, for this preliminary study we select a single representative probe for each. Our primary objective is to obtain preliminary results to verify the efficacy of our BEST-RQ implementation. Detailed descriptions of the datasets and evaluation metrics are provided in Appendix 8.

The results for these frozen-encoder tasks are summarized in Table 3.2.

Table 3.1: Summary of downstream tasks, datasets, and probes used for model evaluation in Chapter 3. (FT) denotes the pre-trained model is fine-tuned.

Task	**Dataset**	**Probe Architecture**	**Metric**
ASR	LibriSpeech-100	2-layered BiLSTM (CTC)	WER (%)
ASV	VoxCeleb1	ECAPA-TDNN	EER (%)
IC	SLURP	BiLSTM	Accuracy
ER	IEMOCAP	ECAPA-TDNN	Accuracy
ASR (FT)	LibriSpeech-100	Feed-Forward NN	WER (%)

Table 3.2: Results on downstream tasks at 100k and 200k steps. BEST-RQ performs similarly to wav2vec but with less than half of the training time. Training time (measured in GPU hours) and number of parameters are shown under *GPU (h)* and *# Par.* respectively.

Model / Task	**# Par.**	**GPU (h)**	**LibriSpeech train-100 ASR**				**VoxCeleb1**	**SLURP**	**IEMOCAP**
Metric			WER ↓				EER ↓	Acc. ↑	Acc. ↑
			Clean	Clean LM	Other	Other LM	ASV	IC	ER
W2V2 100k	90.9M	130	16.26	10.63	40.17	30.83	4.56	72.8	60.9
W2V2 200k	90.9M	262	13.89	9.45	33.55	25.49	3.83	74.5	63.0
BRQ 100k	83.0M	54	16.79	10.79	38.09	28.31	3.84	74.3	61.3
BRQ 200k	83.0M	109	15.11	9.76	34.06	24.74	3.53	74.8	63.8

3.4.1.2 Fine-tuning for ASR

Beyond frozen representations, we also evaluate the model's performance when fully updated for a specific task. We perform an end-to-end fine-tuning experiment on the LibriSpeech *train-clean-100* split, where the encoder is unfrozen and trained alongside a vanilla feed-forward decoder using a Connectionist Temporal Classification (CTC) loss [Graves et al. 2006]. These results are presented in Table 3.3.

3.5 Results

In this section, we report results for the downstream tasks and pre-training efficiency. We also discuss initial experiments regarding the impact of masking ratios and codebook sizes within our BEST-RQ implementation.

Table 3.3: Results of fine-tuning on ASR with LibriSpeech *train-100*.

Model	**Fine-Tune LibriSpeech train-100**			
	Clean	Clean LM	Other	Other LM
W2V2 100k	16.42	10.29	36.62	27.25
W2V2 200k	13.47	8.73	29.64	21.92
BRQ 100k	14.59	9.00	31.49	23.04
BRQ 200k	12.21	7.78	26.81	19.67

3.5.1 Downstream tasks and fine-tuning

The results for the downstream experiments are reported in Table 3.2. On the MP3S LibriSpeech *train-100* ASR task without a language model (LM), wav2vec 2.0 outperformed BEST-RQ. However, with the inclusion of an LM, BEST-RQ performed competitively on the *test-clean* split and slightly better on the *test-other* split. Furthermore, Table 3.3 demonstrates that BEST-RQ yields superior performance when the model is fully fine-tuned for ASR. For non-ASR tasks (ASV, IC, and ER), BEST-RQ performed marginally better than wav2vec 2.0. Overall, these results suggest that while the frameworks offer comparable performance for ASR, BEST-RQ provides a slight advantage in non-ASR tasks.

Our BEST-RQ implementation also proved to be highly efficient. Pre-training for 200k updates required approximately 109 GPU hours for BEST-RQ compared to 262 GPU hours for wav2vec 2.0; in other words, BEST-RQ trained approximately 2.4 times faster.

When examining the performance gains between 100k and 200k steps (Table 3.3), we note that wav2vec 2.0 exhibits more significant improvements in every case. For example, on the IC task, the wav2vec 2.0 200k model improved by 1.7% over its 100k version, while BEST-RQ improved by only 0.5%. We attribute this to the fact that the AFE in wav2vec 2.0 must be learned, resulting in slower initial convergence in terms of both clock time and the number of training steps.

3.5.2 Impact of the masking ratio

In this section, we report on initial experiments varying the masking ratio and the codebook vocabulary size. For these trials, all models were trained for approximately 80k steps on four 11GB RTX 2080Ti GPUs. We evaluate performance based on ASR downstream results on the LibriSpeech *dev-clean* validation set. We cross-referenced the masking ratio and codebook size variations specifically at the 1% and 10% masking levels.

As shown in Table 3.4, reducing the codebook (CB) size from the original 8192 to 1024 does not significantly or consistently alter downstream performance. At a 1% masking ratio, the smaller codebook performs slightly better, whereas at 10%, it performs slightly worse. In contrast, increas-

Table 3.4: Impact of changing the masking ratio and the codebook size. The mask % concerns the start frame index chosen for the mask. Since the selected frame and the following three frames are also masked the actual masking ratio is four times as big.

Mask %	Dev-Clean WER	CB
1%	34.08	1024
1%	36.45	8192
5%	25.24	8192
10%	21.10	1024
10%	20.68	8192
12%	20.11	8192

ing the number of masked frames has a substantial impact; as the percentage of frames selected for masking increased from 1% to 12%, the WER on the *dev-clean* split dropped from approximately 35% to just above 20%—an absolute reduction of roughly 15 percentage points. Note that this masking percentage refers specifically to the probability of sampling a starting frame index. In other words, the total amount of audio masked is four times greater, as the selected frame plus the subsequent three frames are masked.

We attribute the necessity of a higher masking ratio to our specific training conditions. Compared to the original BEST-RQ implementation—which used an estimated batch size of 18 hours and long-form audio files (30–60 seconds)—our setup utilizes the shorter segments of LibriSpeech (mostly under 20 seconds) and a smaller batch size. We believe these constraints in a low-resource setting require a higher masking density to ensure effective representation learning.

3.6 Conclusion

In this chapter, we presented an open-source implementation of a random-projection quantizer for pre-training speech models and evaluated its performance against wav2vec 2.0 across various downstream tasks. Our results demonstrate that BEST-RQ achieves comparable performance to wav2vec 2.0 while pre-training 2.4 times faster. We have released our code within the SpeechBrain toolkit [Ravanelli et al. 2024] and believe this implementation will enable researchers to further investigate these methodologies and explore diverse facets of self-supervised speech modeling.

Although this work utilized a relatively small batch size and model architecture, these preliminary experiments demonstrate the feasibility of pre-training one's own model from scratch with a minimal amount of GPU hours and memory. Moving forward, we utilize this implementation throughout the remainder of this thesis to gain further insights and improve the efficiency of self-supervised speech models.

Chapter 4

Increasing Efficiency with Attention Substitutes

Table of Contents

4.1 Introduction . . . 58
4.2 Background . . . 59
4.3 Attention alternatives in BEST-RQ . . . 61
4.3.1 Fastformer . . . 61
4.3.2 HyperMixing . . . 62
4.3.3 SummaryMixing . . . 62
4.3.4 Mamba . . . 63
4.3.5 Implementation in BEST-RQ . . . 63
4.4 Methods . . . 64
4.4.1 Self-supervised pre-training . . . 64
4.4.2 Downstream evaluation . . . 65
4.5 Results . . . 66
4.5.1 Speed and memory evaluation . . . 66
4.5.2 Downstream tasks . . . 66
4.6 Conclusion . . . 69

Publications related to this chapter

- **SLT 2024** – An Analysis of Linear Complexity Attention Substitutes With Best-RQ [Whetten et al. 2024b]
- **ICASSP 2026** – Polynomial Mixing for Efficient Self-supervised Speech Encoders [Feillet et al. 2026]

Summary

Self-Supervised Learning (SSL) frameworks have proven effective for various speech processing tasks. However, SSL is computationally and memory-intensive. This is in part due to the quadratic complexity of multi-head self-attention (MHSA). Alternatives to MHSA have been proposed and used in the speech domain but have yet to be investigated properly in an SSL setting. In this chapter, we study the effects of replacing MHSA with recent state-of-the-art alternatives that have linear complexity—namely, HyperMixing, Fastformer, SummaryMixing, and Mamba—in the BERT-based speech pre-Training with Random-projection Quantizer (BEST-RQ) framework. We evaluate these methods by examining speed, memory consumption, and performance on the Multi-Probe Speech Self-Supervision Benchmark (MP3S). Results show that these linear alternatives maintain competitive performance compared to MHSA while, on average, decreasing memory consumption by 20% to 60% and increasing speed by 7% to 65% for input sequences ranging from 20 to 80 seconds.

4.1 Introduction

In the previous chapter, we introduced our implementation of the BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) framework and conducted a preliminary set of experiments comparing it to wav2vec 2.0 to demonstrate its efficiency and performance. In this chapter, we seek to improve the efficiency of BEST-RQ by modifying the architecture.

In a study of the efficiency of self-supervised speech models from an architectural standpoint, Parcollet et al. 2023 identify three main culprits: (i) the Acoustic Feature Extractor (AFE), which transforms the raw waveform into a latent representation; (ii) the encoder, which is often a series of Transformer [Vaswani et al. 2017] or Conformer [Gulati et al. 2020] layers; and (iii) the SSL objective. For further explanation regarding the AFE and encoder, see Section 2.3.

While BEST-RQ [Chiu et al. 2022] effectively addresses efficiency in the AFE and training objective (culprits i and iii)—yielding training speeds 2.4 times faster than wav2vec 2.0 with comparable performance, as demonstrated in Chapter 3—the Conformer-based context encoder (culprit ii) remains a bottleneck. This is largely due to the quadratic complexity of MHSA. While several

linear-complexity alternatives have been proposed to mitigate this, such as HyperMixing [Mai et al. 2023b], Fastformer [Wu et al. 2021], SummaryMixing [Parcollet et al. 2024b], and Mamba [Gu et al. 2024], they have yet to be rigorously evaluated in an SSL setting for speech.

In this chapter, we address this gap by evaluating these promising MHSA alternatives within the BEST-RQ framework. Downstream experiments using the MP3S benchmark [Zaiem et al. 2023] demonstrate that our linear-time variants maintain performance parity with standard MHSA while reducing VRAM consumption by 20% to 60% and increasing inference speeds by 7% to 65% for sequences between 20 and 80 seconds. As a secondary contribution, we have open-sourced our implementation in the SpeechBrain toolkit [Ravanelli et al. 2024] to facilitate further research in efficient speech SSL[1].

4.2 Background

In this section, we briefly overview the MHSA mechanism to illustrate why its time complexity is quadratic with respect to the input length. We then describe recent efforts toward developing more efficient alternatives.

Multi-head self-attention (MHSA) was proposed by Vaswani et al. 2017 to model dependencies between different positions in a sequence. The core of MHSA is the scaled dot-product attention operation. Given an input sequence $X \in \mathbb{R}^{n \times d}$ of length n and dimension d, the model independently projects X to produce query (Q), key (K), and value (V) matrices. Scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.1}$$

where d_k represents the dimension of the queries and keys. The term QK^T computes an $n \times n$ attention matrix that represents the pairwise relevance between all tokens, providing each token with global context. This matrix multiplication results in $O(n^2)$ complexity, making it a computational bottleneck for long sequences. The softmax function converts these scores into a probability distribution, which is then used to compute a weighted sum of the values V. In the multi-head setting, this process is performed in parallel across multiple independent "heads," allowing the model to capture various relationships between tokens.

The quadratic scaling of the $n \times n$ attention matrix is particularly problematic for speech processing, where signals are significantly denser and sequences longer than those typically found in text. Despite this computational burden, the global interactions captured by MHSA are important for high performance.

Research on reducing the complexity of MHSA generally follows two overarching approaches: (i) methods aiming to approximate the pairwise token computation at a lower cost, and (ii) meth-

[1] https://github.com/whettenr/brq-att-alt-exp

ods that do not seek to mimic pairwise interaction but instead introduce global context through alternative mechanisms.

The first category includes sparse attention methods, such as BigBird [Zaheer et al. 2020] and Longformer [Beltagy et al. 2020], which employ a combination of sliding window, global, and random attention to achieve linear complexity. However, a primary limitation of sparse attention is its inability to fully model global context, as these methods primarily operate on local windows of the input. Other approaches in this category, such as Linformer [Wang et al. 2020] and the Linear Transformer [Katharopoulos et al. 2020], approximate the attention mechanism by computing low-rank approximations of the key and value matrices or by utilizing the associative property of kernel feature maps. Despite their theoretical linear complexity, these methods can remain computationally expensive in practice [Wu et al. 2021].

The second category of solutions, which do not aim to approximate the self-attention mechanism, includes HyperMixing, SummaryMixing, and Mamba [Mai et al. 2023a; Parcollet et al. 2024b; Gu et al. 2024]. HyperMixing was first introduced with the HyperMixer [Mai et al. 2023a], an extension of the MLP-Mixer [Tolstikhin et al. 2021]. A primary disadvantage of the original MLP-Mixer is its inability to handle variable-length inputs, rendering it unsuitable for text or speech tasks. HyperMixing addresses this by using hypernetworks [Ha et al. 2017] to produce token-mixing weights dynamically, thereby capturing global context for variable-length sequences. HyperMixing has demonstrated competitive performance in both natural language tasks and fully-supervised ASR via the HyperConformer architecture [Mai et al. 2023a; Mai et al. 2023b].

Alternatively, SummaryMixing avoids hypernetworks by passing the input through a parameterized function, such as a multi-layer perceptron (MLP), which is then averaged across all time steps to produce a "summary vector." To integrate global context, this summary vector is concatenated with the input at each time step and processed through a subsequent parameterized function to produce the final output. SummaryMixing has been shown to perform as well as MHSA in ASR, keyword spotting, and Spoken Language Understanding (SLU) tasks, while significantly reducing memory consumption and training time [Parcollet et al. 2024b].

Lastly, Mamba is a sequence modeling framework based on State Space Models (SSMs) [Gu et al. 2024]. SSMs can be conceptualized as a hybrid of recurrent and convolutional neural networks that scale linearly with sequence length. Mamba has shown strong performance in text and audio (within autoregressive SSL settings) as well as supervised speech tasks, ASR and speech enhancement [Zhang et al. 2025b]. HyperMixing, SummaryMixing, and Mamba all offer efficient, accessible implementations and were therefore selected as the representative linear-time alternatives for this study.

Despite the promising performance of these MHSA alternatives in supervised settings, their efficacy within an SSL framework for speech has not yet been rigorously explored. Furthermore, existing studies have primarily focused on ASR and employed models ranging from only 5M to

100M parameters [Mai et al. 2023b; Parcollet et al. 2024b; Peng et al. 2022; Zhang et al. 2025b], which contrasts sharply with the massive scale of modern SSL models. In this chapter, we address these gaps by: (i) evaluating these linear-complexity substitutes within an SSL setting; (ii) assessing performance across a diverse range of speech tasks using the MP3S benchmark; and (iii) scaling the model architecture to 300M parameters.

4.3 Attention alternatives in BEST-RQ

In this section, we describe how Fastformer, SummaryMixing, HyperMixing, and Mamba achieve linear complexity relative to multi-head self-attention (MHSA). We then detail their integration into the BEST-RQ framework. For this discussion, let the input to the attention module be $\mathbf{X} \in \mathbb{R}^{n \times d} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, representing a sequence of n time steps with d-dimensional vectors. The fundamental difference between these methods lies in how they transform $\mathbf{X}$ to incorporate global context.

4.3.1 Fastformer

Fastformer Wu et al. 2021 reduces complexity by using additive attention to summarize the query ($\mathbf{Q}$) and key ($\mathbf{K}$) matrices into individual global vectors. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$ be the standard linear transformations of $\mathbf{X}$, where $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n]$, $\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_n]$, and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$. First, $\mathbf{Q}$ is summarized into a single global query vector $\bar{\mathbf{q}} \in \mathbb{R}^d$ as follows:

$$\alpha_t = \text{softmax}\left(\frac{\mathbf{w}_q^T \mathbf{q}_t}{\sqrt{d}}\right); \quad \bar{\mathbf{q}} = \sum_{t=1}^{n} \alpha_t \mathbf{q}_t \tag{4.2}$$

where $t \in \{1, \ldots, n\}$ denotes the time step, and $\mathbf{w}_q \in \mathbb{R}^d$ is a learned vector used to calculate the attention weight α_t for each vector $\mathbf{q}_t$.

To incorporate global information, an element-wise product is performed between the global query $\bar{\mathbf{q}}$ and each key $\mathbf{k}_t$, resulting in a global context-aware key matrix $\mathbf{P}$. This matrix is then summarized into a global key vector $\bar{\mathbf{k}}$ through a similar additive attention mechanism:

$$\mathbf{p}_t = \bar{\mathbf{q}} \odot \mathbf{k}_t; \quad \beta_t = \text{softmax}\left(\frac{\mathbf{w}_k^T \mathbf{p}_t}{\sqrt{d}}\right); \quad \bar{\mathbf{k}} = \sum_{t=1}^{n} \beta_t \mathbf{p}_t \tag{4.3}$$

Finally, the global key vector $\bar{\mathbf{k}}$ is combined with the value vectors $\mathbf{v}_t$ via an element-wise product. This result is transformed by a learnable weight matrix $\mathbf{W}_o \in \mathbb{R}^{d \times d}$, and a residual connection is added from the original query $\mathbf{q}_t$ to produce the final output:

$$\mathbf{out}_t = \mathbf{W}_o(\bar{\mathbf{k}} \odot \mathbf{v}_t) + \mathbf{q}_t \tag{4.4}$$

By replacing the $n \times n$ matrix multiplication with vector summarization and element-wise products ($\odot$), Fastformer enables tokens to interact with the global sequence while maintaining linear complexity with respect to n.

4.3.2 HyperMixing

HyperMixing [Mai et al. 2023a] achieves linear complexity by replacing the pairwise dot-product of MHSA with a dynamic token-mixing mechanism. This approach is an evolution of the MLP-Mixer [Tolstikhin et al. 2021], which utilizes a Token-Mixing MLP (TM-MLP) to share information across time steps. In a standard TM-MLP, the operation is defined as:

$$\text{TM-MLP}(\mathbf{X}) = \mathbf{W}_1(\sigma(\mathbf{W}_2^T\mathbf{X}^T)) \tag{4.5}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the input. In a standard MLP-Mixer, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable parameters of a fixed size, which restricts the model to inputs of a constant length n.

To support variable-length speech signals, HyperMixing [Mai et al. 2023b] ensures that $\mathbf{W}_1$ and $\mathbf{W}_2$ are not static parameters. Instead, they are dynamically generated from the input $\mathbf{X}$ and absolute position embeddings $\mathbf{P} \in \mathbb{R}^{n\times d}$ using two hypernetworks, MLP_1 and MLP_2:

$$\mathbf{W}_k(\mathbf{X}) = \Big(\text{MLP}_k(\mathbf{x}_1 + \mathbf{p}_1) \vdots \text{MLP}_k(\mathbf{x}_n + \mathbf{p}_n)\Big) \in \mathbb{R}^{n\times d'}, \quad k \in 1, 2 \tag{4.6}$$

Here, d' represents the hidden layer size of the token mixer. The learnable parameters of the HyperMixer are contained within MLP_1 and MLP_2, rather than the mixing matrices themselves. Once these weights are generated, the operation in Equation 4.5 is applied.

This formulation results in an asymptotic complexity of $O(n \cdot d \cdot d')$, which is linear with respect to the input length n. By shifting from quadratic pairwise attention to linear weight generation, HyperMixing enables the context encoder to scale efficiently to the long sequences typical of raw speech data.

4.3.3 SummaryMixing

In SummaryMixing [Parcollet et al. 2024b], the input $\mathbf{X} \in \mathbb{R}^{n\times d} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ is processed by two functions, f and s, which are typically implemented as Multi-Layer Perceptrons (MLPs). The output of s is averaged across all time steps to produce a global summary vector $\bar{\mathbf{s}}$. To incorporate global context into the sequence, $\bar{\mathbf{s}}$ is concatenated with the output of f at each time step t, and subsequently processed by a combining function c (also an MLP). The resulting vector $\mathbf{h}_t$ serves as the final output. The SummaryMixing mechanism is formally expressed as:

$$\bar{\mathbf{s}} = \frac{1}{n}\sum_{t=1}^{n} s(\mathbf{x}_t); \quad \mathbf{h}_t = c([f(\mathbf{x}_t); \bar{\mathbf{s}}]) \tag{4.7}$$

In this method, calculating the summary vector $\bar{\mathbf{s}}$ is an averaging operation that scales linearly with the input length n, while the function c operates independently at each time step. Consequently, SummaryMixing effectively introduces global context with linear time and memory complexity.

4.3.4 Mamba

Mamba [Gu et al. 2024] reduces computational complexity to a linear scale by processing the input sequence in a recurrent-like manner. As a SSM, Mamba represents the historical information of the input $\mathbf{X}$ within a hidden state $\mathbf{h}_t \in \mathbb{R}^{d_{ssm}}$, where d_{ssm} is the state dimension. Since the model only maintains a compressed state rather than a full attention matrix, it bypasses the quadratic bottleneck of MHSA. To mitigate the information loss inherent in state compression, Mamba incorporates an input-dependent selection mechanism that allows the model to dynamically "remember" or "forget" information based on the current token.

Specifically, the discretized state transition matrices are computed as functions of the current input $\mathbf{x}_t$[2]:

$$\bar{\mathbf{A}}_t = f_{\bar{\mathbf{A}}}(\mathbf{x}_t), \quad \bar{\mathbf{B}}_t = f_{\bar{\mathbf{B}}}(\mathbf{x}_t), \quad \mathbf{C}_t = f_{\mathbf{C}}(\mathbf{x}_t). \tag{4.8}$$

The evolution of the hidden state $\mathbf{h}_t$ and the resulting output y_t at any time step t follow the recurrence:

$$\mathbf{h}_t = \bar{\mathbf{A}}_t \mathbf{h}_{t-1} + \bar{\mathbf{B}}_t \mathbf{x}_t, \quad y_t = \mathbf{C}_t \mathbf{h}_t. \tag{4.9}$$

To overcome the sequential bottleneck typically associated with recurrent architectures, Mamba utilizes a hardware-aware parallel scan algorithm [Gu et al. 2024]. This allows the model to compute the state evolution efficiently across the entire sequence, achieving a throughput significantly higher than standard Transformers. Finally, we employ a bidirectional Mamba architecture to capture full global context, as motivated by [Zhang et al. 2025b].

4.3.5 Implementation in BEST-RQ

As noted in the introduction of this chapter, the encoder employed in the BEST-RQ framework consists of a series of Conformer layers. We use the SpeechBrain [Ravanelli et al. 2024] implementation of the Conformer, which closely follows the original architecture proposed by [Gulati et al. 2020]. Each Conformer layer comprises four main components: a feed-forward module, an MHSA module, a convolutional module, and a second feed-forward module. Residual connections are integrated at each stage, with layer normalization applied after each feed-forward module. To implement our proposed modifications, we replace the MHSA module with the linear-complexity alternatives described in the previous sections. This substitution replaces the computationally expensive dot-product operations with efficient alternatives, as illustrated in Figure 4.1.

[2]While $\mathbf{A}$ is a stationary parameter, the step size Δ is derived from the current input. Given $\bar{\mathbf{A}} = \exp(\Delta_t \mathbf{A})$, the transition matrix becomes indirectly input-dependent. We simplify this notation to $f_{\bar{\mathbf{A}}}(\mathbf{x}_t)$. For further details, see [Gu et al. 2024].

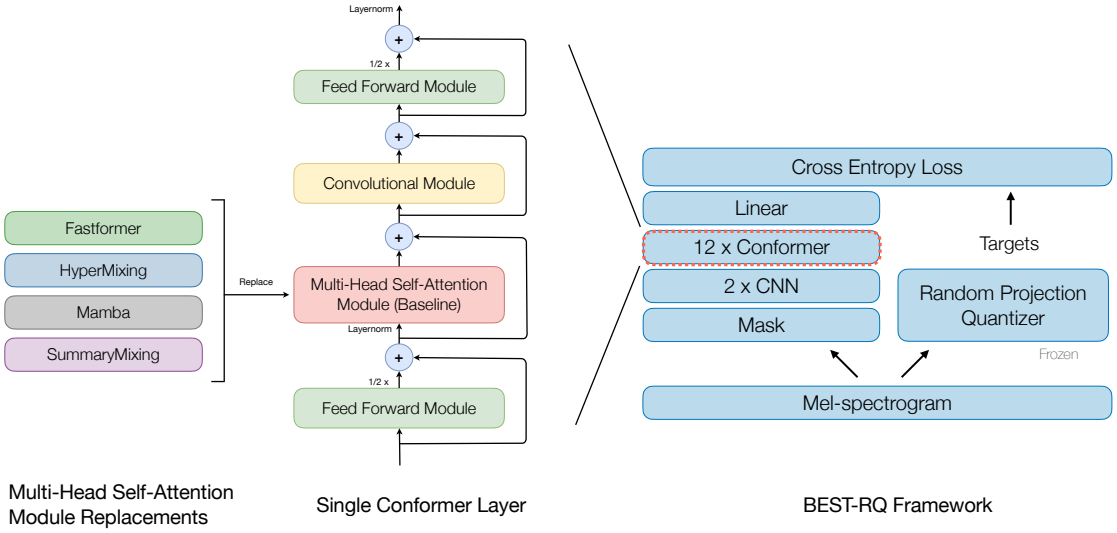


Figure 4.1: Diagram of the BEST-RQ framework and a Conformer layer. In each Conformer layer, we replace the multi-head self-attention module with linear-complexity alternatives.

4.4 Methods

In this section, we describe our SSL pre-training protocol and summarize the downstream tasks.

4.4.1 Self-supervised pre-training

We utilize the implementation of BEST-RQ introduced in Chapter 3, featuring Conformer layers [Gulati et al. 2020] that incorporate relative sinusoidal positional embeddings. For all experiments, the pre-training framework remains consistent, with the standard MHSA module being replaced by HyperMixing, Fastformer, SummaryMixing, or Mamba. The small models consist of 12 Conformer layers, while the large models contain 24 layers. For the Fastformer-LG configuration, we increased the number of heads to address convergence issues encountered at that scale.

In the case of Mamba, we employ a state dimension of $d_{ssm} = 16$, a convolution dimension of $d_{conv} = 4$, and an expansion factor of 2. To ensure a fair comparison, we adjusted the hidden dimensions of each architecture so that all models maintain a similar parameter count: approximately 95M for the small models and 315M for the large models. A detailed list of all hyperparameters is available in our open-source SpeechBrain recipes[3].

We pre-train all models for the same number of steps, 200k, using the 960 hours of training data from the LibriSpeech [Panayotov et al. 2015] dataset. Although 200k steps is fewer than the 400k or 800k steps used by some state-of-the-art models [Baevski et al. 2020; Chen et al. 2023], previous research has empirically shown that 200k steps is sufficient for comparing relative SSL

[3]Link to code provided at the end of the introduction to this chapter

model performance [Parcollet et al. 2023; Whetten et al. 2024a]. We use dynamic batching in which audio files are grouped into buckets by similar length and the number of samples per batch varies so that the total seconds of audio per batch remain approximately constant. The effective batch size is set to about 1.7 hours of audio for all experiments, following common practice [Baevski et al. 2020].

4.4.2 Downstream evaluation

To assess the quality of the learned representations, we evaluate our models on a selection of downstream tasks from the MP3S benchmark [Zaiem et al. 2023]. Following their protocol, pre-trained models remain frozen, and a weighted sum of their hidden layers serves as input to a downstream probe model.

While Chapter 3 focused on a subset of the MP3S tasks, this work expands the evaluation in two key ways to provide a more rigorous comparison of linear-complexity architectures. First, we utilize both probe architectures provided by the benchmark for each task—typically a simpler linear model and a more complex non-linear one—to ensure that our conclusions are not biased toward a specific probe capacity. Second, we introduce out-of-domain evaluation via Welsh and Basque ASR. The tasks, datasets, and probes are summarized in Table 4.1, with a comprehensive description of each provided in Appendix 8.

Table 4.1: Summary of downstream tasks, datasets, and probes. All SSL models are frozen except for the ASR Fine-tuning (FT) task.

Task	**Dataset**	**Probe 1**	**Probe 2**
ASV	VoxCeleb1	X-Vector	ECAPA-TDNN
IC	SLURP	Linear	BiLSTM
ER	IEMOCAP	Linear	ECAPA-TDNN
ASR	LibriSpeech-100	BiLSTM	ContextNet
ASR	CV (Welsh, Basque)	Linear	BiLSTM
ASR (FT)	LibriSpeech-100	Feed-Forward NN	

Similar to Chapter 3, in addition to the frozen evaluations, we perform a full fine-tuning experiment on the 100-hour LibriSpeech split. This represents a more practical deployment scenario for SSL models. In this setup, the pre-trained model weights are updated via a Connectionist Temporal Classification (CTC) loss [Graves et al. 2006], using a feed-forward network as the downstream decoder.

4.5 Results

We first evaluate the computational efficiency and memory gains of the MHSA alternatives using a controlled toy task, as shown in Figure 4.2. These findings are subsequently extended to full-scale SSL pre-training and downstream evaluation on the MP3S benchmark.

4.5.1 Speed and memory evaluation

To benchmark the speed and memory consumption of the various attention mechanisms, we utilize a synthetic task with randomly generated data. Input lengths range from 10 to 80 seconds. We set a fixed batch size of 6, which was the maximum possible to avoid Out-of-Memory (OOM) errors for the MHSA baseline at longer sequence lengths. For each input length, we perform 10 runs and record the forward pass latency and peak VRAM consumption. Only the forward pass is evaluated to reflect a real-world deployment scenario. Figure 4.2 illustrates the average computation time and peak VRAM across these runs, with 95% bootstrapped confidence intervals. All measurements were conducted on an isolated node equipped with an NVIDIA A100 80GB GPU.

At a 10-second input length, latency and VRAM requirements are relatively uniform across all models. However, significant performance gaps emerge starting at 20 seconds, where the linear alternatives consume an average of 24% less memory and run 7% faster than MHSA. At the 80-second extreme, the alternatives demonstrate an average reduction of 64% in memory usage and a 65% improvement in speed. SummaryMixing and Fastformer were the most efficient in terms of throughput, showing speed increases of 70% and 71% respectively at 80 seconds compared to MHSA. Regarding memory footprint, Mamba and Fastformer were the most efficient, reducing peak VRAM by 67% and 66% respectively at the 80-second mark.

To estimate the total pre-training resource requirements, we conduct a 5-epoch benchmark using the full 1.7-hour batch size on NVIDIA V100 32GB GPUs. These measurements are extrapolated to the full 200k-step pre-training regime to estimate total GPU hours. As shown in Table 4.2, all alternatives reduced computational overhead relative to the baseline, with the exception of the HyperMixing-LG and Mamba-LG models. This suggests that while linear complexity is theoretically superior, the overhead of weight generation in HyperMixing and the recurrent state updates in Mamba may diminish gains at certain architectural scales and batch configurations.

4.5.2 Downstream tasks

Automatic Speech Recognition on LibriSpeech For the frozen-encoder evaluation with the LSTM probe (Table 4.2), the MHSA baseline generally outperformed the alternative methods, though HyperMixing, SummaryMixing, and Mamba followed closely. For instance, on the *test-clean* split with a language model (LM), the MHSA-SM and MHSA-LG models achieved Word Er-

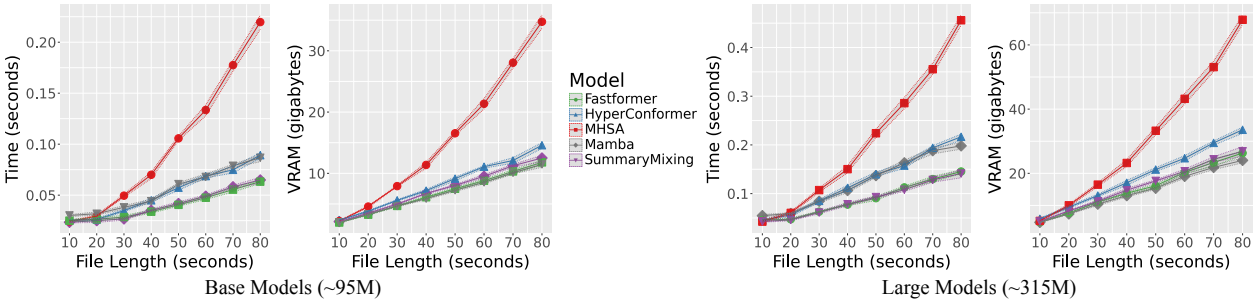


Figure 4.2: Inference speed and peak VRAM vs. input length. Comparison between standard MHSA and linear alternatives for BEST-RQ base and large configurations. MHSA requires significantly more time and VRAM as input size increases, whereas the alternatives do not.

Table 4.2: Results of various alternatives to MHSA for speech SSL models on the MP3S benchmark. Base models and large models (denoted -LG) have around 94M and 315M parameters respectively. Linear complexity alternatives perform competitively with multi-head self-attention (MHSA) even surpassing depending on the task and probe.

Model/Task	**Specs**		**LibriSpeech train-100 ASR**				**CommonVoice**		**VoxCeleb**	**SLURP**	**IEMOCAP**
Metric			WER ↓				WER ↓	WER ↓	EER ↓	Acc. ↑	Acc. ↑
1st Probe			LSTM				Lin.	Lin.	X-Vector	Lin.	Lin.
	# Params.		Clean	Clean LM	Other	Other LM	Welsh	Basque	ASV	IC	ER
MHSA	94.0M		**10.31**	**6.88**	**24.95**	**18.54**	80.04	82.56	9.36	50.3	60.1
HyperConf	94.3M		10.96	7.24	25.82	19.05	**79.65**	**81.41**	**8.54**	53.2	60.1
Fastformer	93.9M		13.21	8.59	33.11	24.24	82.03	84.62	12.27	36.2	56.4
SummaryMix	94.1M		11.92	7.76	28.25	20.80	79.97	81.99	11.13	52.1	63.0
Mamba	94.2M		10.46	6.90	26.00	18.88	80.71	83.66	11.91	**53.5**	**64.10**
MHSA-LG	313.5M		**7.10**	**4.98**	**17.63**	**12.92**	84.46	88.88	8.60	54.8	62.1
HyperConf-LG	315.4M		7.73	5.47	18.39	13.83	77.66	79.94	8.64	62.3	62.8
Fastformer-LG	315.9M		18.74	11.83	41.07	30.55	82.85	84.59	10.13	34.5	56.9
SummaryMix-LG	313.7M		7.22	5.04	17.67	13.14	**76.94**	**78.88**	**8.30**	**62.8**	65.3
Mamba-LG	314.9M		7.84	5.23	20.57	14.66	80.86	84.66	10.54	57.7	**66.15**
2nd Probe			Contextnet				LSTM	LSTM	ECAPA	LSTM+Lin.	ECAPA
	Time(h)	Mem(GB)	Clean	Clean LM	Other	Other LM	Welsh	Basque	ASV	IC	ER
MHSA	453	8.86	**10.10**	6.30	**22.84**	16.83	57.78	50.24	4.37	74.9	60.83
HyperConf	438	8.60	10.66	6.78	23.46	17.54	56.74	48.39	**3.65**	**76.9**	60.10
Fastformer	357	7.25	13.10	8.10	29.98	22.75	59.14	52.59	4.84	68.3	54.30
SummaryMix	368	7.68	11.55	7.21	25.13	18.72	58.24	50.05	4.09	76.0	**64.34**
Mamba	427	6.62	10.27	**5.96**	23.93	**16.37**	**55.22**	**47.51**	4.51	76.2	63.05
MHSA-LG	973	20.70	7.02	4.40	15.78	**11.82**	58.39	50.35	3.96	74.3	63.74
HyperConf-LG	1033	22.28	7.62	4.86	16.79	12.48	55.75	47.99	**3.45**	78.0	64.62
Fastformer-LG	780	18.76	18.41	10.97	38.64	28.92	61.18	54.49	4.26	68.6	55.44
SummaryMix-LG	805	19.49	**6.79**	**4.38**	**15.69**	11.83	58.24	48.59	4.05	**78.6**	**65.21**
Mamba-LG	1010	16.99	7.54	4.51	18.49	12.63	**55.42**	**47.59**	3.76	78.1	64.21

Table 4.3: Results with fine-tuning on LibriSpeech *train-100*. For the *test-clean* set, we report 95% bootstrapped confidence interval (C.I.).

Model	**Fine-Tune LibriSpeech train-100**			
	Clean	Clean LM (C.I.)	Other	Other LM
MHSA	**7.01**	**5.35** ($\pm$ 0.26)	**16.98**	**13.55**
HyperConf	8.22	5.77 ($\pm$ 0.28)	19.29	15.03
Fastformer	9.32	6.82 ($\pm$ 0.31)	22.75	17.95
SummaryMix	8.72	6.30 ($\pm$ 0.28)	21.78	16.97
Mamba	7.61	5.50 ($\pm$ 0.28)	19.97	15.37
MHSA-LG	**5.03**	**3.98** ($\pm$ 0.21)	**11.52**	**9.42**
HyperConf-LG	5.87	4.54 ($\pm$ 0.32)	13.13	10.78
Fastformer-LG	13.16	9.89 ($\pm$ 0.34)	31.91	26.75
SummaryMix-LG	5.28	4.20 ($\pm$ 0.25)	12.80	10.50
Mamba-LG	5.59	4.48 ($\pm$ 0.25)	15.47	12.66

ror Rates (WER) of 6.88% and 4.98%, respectively, while the Mamba counterparts reached 6.90% and 5.23%.

Interestingly, using the ContextNet probe, the Mamba-SM model outperformed MHSA-SM, reaching a WER of 5.96% compared to 6.30% on the *test-clean* split with an LM. For the large-scale models with the ContextNet probe, SummaryMixing outperformed MHSA across most settings, trailing by only 0.01% on the *test-other* split with an LM. Notably, despite extensive hyperparameter tuning, Fastformer did not scale effectively to 315M parameters; the performance of *Fastformer-LG* remains significantly behind all other architectures.

For the fine-tuning experiments on the LibriSpeech *train-100* subset (Table 4.3), we report 95% confidence intervals using 1,000 bootstraps on the *test-clean* results with an LM. Consistent with the frozen benchmark, MHSA achieved the best overall performance. However, HyperMixing, SummaryMixing, and Mamba remained competitive, with WERs ranging from only 1 to 3 points higher than the MHSA baseline.

Automatic Speech Recognition on CommonVoice The results on CommonVoice present a distinct contrast to LibriSpeech, which we hypothesize highlights a common bias in speech SSL evaluation. LibriSpeech is frequently utilized for both pre-training and downstream evaluation [Baevski et al. 2020; Hsu et al. 2021; Lodagala et al. 2023], potentially inflating performance through in-domain overlap. CommonVoice allows for a more rigorous assessment of generalization, as it is out-of-domain in terms of language (Welsh and Basque) and recording conditions; while LibriSpeech consists of clean audiobook readings, CommonVoice is a crowd-sourced dataset featuring varied environments and recording hardware.

In this out-of-domain setting, MHSA is no longer the top-performing model. With the linear probe, HyperMixing and SummaryMixing both outperformed MHSA on Welsh and Basque. With the LSTM probe, Mamba achieved the best performance, followed by HyperMixing and SummaryMixing. These results demonstrate that linear-complexity alternatives can generalize more effectively to out-of-domain and low-resource datasets than standard MHSA.

Speaker Verification For the Speaker Verification task on VoxCeleb1, results varied by probe, yet MHSA was never the optimal solution. Using the ECAPA-TDNN probe, HyperMixing achieved the best performance with an Equal Error Rate (EER) of 3.65% and 3.54% for the small and large models, respectively. With the X-Vector probe, HyperMixing performed best among the small models (8.54% EER), while SummaryMixing was superior at the large scale (8.30% EER). This confirms the ability of these alternatives to effectively capture robust speaker characteristics.

Intent Classification On the SLURP dataset, all alternatives except Fastformer outperformed MHSA across both probe types. For example, using the small models with the LSTM probe, MHSA obtained an accuracy of 74.9%. While Fastformer lagged at 68.3%, HyperMixing, SummaryMixing, and Mamba reached accuracies of 76.9%, 76.0%, and 76.2%, respectively. This indicates that these mechanisms are highly effective at learning global linguistic features such as intent.

Emotion Recognition In the Emotion Recognition task, we observed trends similar to those in CommonVoice and Speaker Verification, with MHSA not being the best performer. Mamba performed best with the linear probe, reaching accuracies of 64.10% (SM) and 66.15% (LG), compared to 60.1% and 62.1% for MHSA. For the ECAPA-TDNN probe, SummaryMixing was the top performer, achieving 64.34% and 65.21% accuracy for the small and large configurations, respectively.

4.6 Conclusion

One of the primary goals of SSL is to develop models capable of producing useful representations without the need for large-scale labeled data. However, the pre-training process is computationally expensive, making the development of resource-efficient speech models a critical area of research. A significant portion of this expense is attributed to the quadratic complexity of MHSA. While linear-complexity alternatives exist, they have primarily been evaluated on text and supervised speech tasks. In this chapter, we explored replacing MHSA with four state-of-the-art alternatives—HyperMixing, Fastformer, SummaryMixing, and Mamba—within an SSL framework for speech.

Our results demonstrate that for sequences exceeding 20 seconds, these alternatives are substantially faster and consume significantly less memory than MHSA. Empirically, HyperMixing, SummaryMixing, and Mamba proved to be effective alternatives, performing competitively with the MHSA baseline across various tasks. In contrast, Fastformer failed to scale effectively to 315M

parameters and yielded lower downstream performance. Based on our synthetic benchmarks, and considering that approximately 75% of LibriSpeech utterances are shorter than 15 seconds, we hypothesize that the efficiency gains would be even more pronounced when using datasets such as those used by [Chiu et al. 2022], where pre-training audio is cropped to between 32 and 64 seconds.

However, for shorter inputs under approximately 20 seconds, the computational time and memory consumption are relatively similar across all models. This threshold may decrease significantly if the model architecture does not utilize a two-dimensional CNN (2D-CNN) for acoustic feature extraction. The AFE plays a critical role in determining the length of the sequence that reaches the encoder; for instance, replacing the 2D-CNN with a 1D-CNN might lower the efficiency threshold to 10 seconds. Nevertheless, many common speech datasets consist primarily of short utterances, and many speech tasks do not inherently require global context exceeding 20 seconds. Furthermore, long-form audio is frequently segmented by silence for processing or streaming, which raises questions regarding the necessity of replacing MHSA in standard pipelines.

In conclusion, our findings suggest that unless researchers are working specifically with raw waveforms or long audio files, replacing MHSA with linear-complexity alternatives may not yield substantial speed or memory gains in typical settings. We believe that more significant progress toward efficient SSL for speech may be achieved through other architectural innovations, pruning or quantization techniques, and data selection.

Chapter 5

Data Selection Strategies for Efficient Pre-training

Table of Contents

5.1 Introduction . . . 72
5.2 Related Work . . . 73
5.2.1 Data selection for supervised speech processing . . . 73
5.2.2 Data selection for pre-training data . . . 74
5.3 Methods . . . 74
5.3.1 Data . . . 75
5.3.2 Data selection methods . . . 75
5.3.3 Training settings . . . 76
5.4 Results . . . 77
5.5 Conclusion . . . 79

Publications related to this chapter

- **ICASSP 2026** – A Study of Data Selection Strategies for Pre-training Self-Supervised Speech Models [Whetten et al. 2026]
- **ICASSP 2026** – Ara-Best-RQ: Multi Dialectal Arabic SSL [Elleuch et al. 2026]
- **SPECOM 2025** – In-Domain SSL Pre-training and Streaming ASR: Application to Air Traffic Control Communications [Duret et al. 2025]

Summary

Self-supervised learning (SSL) has transformed speech processing, yet its reliance on massive pre-training datasets remains a bottleneck. While robustness is often attributed to scale and diversity, the role of the data distribution is less understood. We systematically examine how curated subsets of pre-training data influence Automatic Speech Recognition (ASR) performance with the BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ). Surprisingly, optimizing for acoustic, speaker, or linguistic diversity yields no clear improvements over random sampling. Instead, we find that prioritizing the longest utterances achieves superior ASR results while using only half the original dataset, reducing pre-training time by 24% on a large corpus. These findings suggest that, for pre-training speech SSL models, data length is a more critical factor than either data diversity or overall data quantity for performance and efficiency, offering a new perspective for data collection and selection strategies in SSL speech processing.

5.1 Introduction

In the previous chapters we introduced our open-source implementation of the BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) and reported preliminary experiments. We also experimented with modifying the architecture, replacing multi-head self-attention with more efficient alternatives to improve speed and memory consumption for long audio files. Other research has similarly open-sourced models for the community [Chen et al. 2024; Peng et al. 2023] and modified architectures and pre-training frameworks to improve the efficiency of speech models [Baevski et al. 2023; Rekesh et al. 2023]. While these methods increase the accessibility and efficiency of speech models, less work has been done to decrease the data requirements for self-supervised speech models. High-performing self-supervised speech models such as Google USM [Zhang et al. 2023] and Xeus [Chen et al. 2024] are pre-trained on the order of millions of hours of speech data (corresponding to centuries of audio). Gathering, collecting, and storing these data is a significant undertaking, and the effects of changes in the composition of pre-training

data have been less explored. Recent work has found that gender diversity in the pre-training data is not a strong factor in downstream performance [Meng et al. 2022; Zanon Boito et al. 2022].

Furthermore, as noted in Section 2.6.2, increasing dataset size does not inherently guarantee performance gains. In the LeBenchmark 2.0 study, researchers demonstrated that models pre-trained on 3,000 and 7,000 hours of French audio achieved performance comparable to those trained on 14,000 hours across ASR, SLU, and speech translation tasks [Parcollet et al. 2024a]. This plateauing effect is further supported by Le et al. 2026, who observed no significant improvements when scaling data from 14,000 to 114,000 hours. Beyond diminishing returns, low-quality data can be actively detrimental; for instance, we found that incorporating large-scale scraped Arabic audio actually inhibited model convergence [Elleuch et al. 2026]. These findings challenge the paradigm that more data is always better and highlight the need to define what constitutes "high-quality" audio for effective pre-training.

To identify what constitutes "good" data for pre-training speech models, we investigate the impact of various unsupervised selection heuristics. We first explore methods that promote diversity across acoustic, speaker, and linguistic dimensions; we then examine the specific role of utterance length in model performance.

Unexpectedly, our results show that enforcing diversity in the pre-training data—whether acoustic, speaker, or linguistic—yields no significant improvement in downstream Automatic Speech Recognition (ASR) performance. In contrast, simply selecting the longest utterances for pre-training achieves superior ASR performance while requiring only half of the original dataset and reducing pre-training time by 24% on a large-scale corpus. We make the code used in the experiments public for reproducibility[1].

5.2 Related Work

In this section, we provide an overview of prior work on data selection for speech processing.

5.2.1 Data selection for supervised speech processing

While substantial work has been done on data selection for speech models, most prior research has focused on fully supervised learning, such as active learning. Active learning addresses the task of identifying the most informative data for labeling, with the goal of minimizing annotation costs. Typically, this is achieved by using uncertainty or confidence measures produced by a minimally trained ASR model. Utterances with the lowest confidence are selected for labeling and then used to train a fully supervised ASR model [Malhotra et al. 2019].

Similarly, researchers have proposed unsupervised fine-tuning data selection methods for ASR using self-supervised speech models [Gody et al. 2023]. They use a pre-trained self-supervised

[1] github.com/whettenr/sss_data_selection

model, HuBERT, and use different features such as the pre-training loss and perplexity of a minimal language model trained on the cluster targets. They found that selecting data with a high perplexity or low pre-training loss improved performance compared to selecting data with low perplexity or high pre-training loss.

However, these lines of work are focused on selecting data for fine-tuning, which is a different problem than selecting data for pre-training. In contrast, this chapter focuses on understanding what constitutes an effective pre-training dataset for self-supervised speech models, rather than on active learning to minimize annotation costs.

5.2.2 Data selection for pre-training data

For SSL-specific data selection, researchers have proposed methods that select pre-training data most similar to the target downstream domain. One such approach trains language models on quantized targets of speech from both the pre-training pool and the target domain dataset, and then computes a normalized difference in log-probability between the two models [Lu et al. 2022]. This score serves as an indicator of how similar a pre-training utterance is to the target domain. While effective for domain-matching, this line of work is distinct from ours, as we instead investigate the general properties of pre-training data that lead to strong overall performance, independent of any specific target domain.

Zanon Boito et al. 2022 pre-trained gender-specific wav2vec 2.0 models (i.e. wav2vec 2.0 models trained on either male or female speech) and compared to various levels of gender diversity. They found that gender specific models not only had lower performance over all, but did not guarantee better performance on their respective gender. Similarly, Meng et al. 2022 found that gender-balanced data is not vital for effective pre-training and that instead pre-training on data with slower speech rates lead to better overall performance. Based on these studies we do not attempt any gender-specific data selection strategies.

Most closely related to our study is the work of Berrebbi et al. 2023, who analyze the effects of varying the number of speakers and the amount of audio in pre-training datasets. Their results highlight the importance of dataset size and, to an extent, speaker diversity in SSL, though their experiments were limited to under 1,000 hours of pre-training data and require speaker labels. By contrast, our study explores methods that do not require speaker labels, consider features beyond speaker information, and scale up to 25,000 hours of audio.

5.3 Methods

In this section, we describe the dataset used, the various methods for data selection, and the training environment.

5.3.1 Data

For our experiments, we use the Loquacious dataset, which combines commercially usable English speech corpora. We select this dataset because it is both large-scale and challenging, containing 25,000 hours of diverse speech, including read, spontaneous, conversational, clean, and noisy recordings [Parcollet et al. 2025].

This diversity extends across all splits (train, development, and test), which is crucial for our study. Unlike prior work that focuses on selecting pre-training data to closely match the fine-tuning domain [Lu et al. 2022], our goal is to investigate which types of pre-training data yield the best overall performance across various speech conditions. For pre-training, we experiment with the medium split (2,500 hours) and the large split (25,000 hours). For fine-tuning for ASR, we use the small split (250 hours).

5.3.2 Data selection methods

We consider two baselines. The first, more challenging baseline, uses the entirety of the data for pre-training, while the second is a random selection of 50% of the data. We refer to these as the *all* and *random* baselines, respectively. For all of our proposed data selection methods we select 50% of the data, so that the total amount of audio used for pre-training is approximately the same as the random baseline. This allows us to directly compare the effectiveness of our selection methods against a random subset of the same size, while also assessing how close we can get to the performance achieved with the full dataset.

The next set of methods is based on diversity sampling across different feature types: acoustic, speaker, and linguistic. We target these distinct characteristics in an attempt to attribute which features contribute most to model performance. To capture basic acoustic characteristics of each utterance, we compute the first 13 Mel-Frequency Cepstral Coefficients (MFCCs) along with their first- and second-order derivatives. We then average across frames, producing a 39-dimensional vector per utterance. For speaker features, we extract 256-dimensional speaker embeddings using the WeSpeaker model [Wang et al. 2023] with pyannote [Bredin 2023], which encodes speaker-specific characteristics from a given utterance. For linguistic features, we use the SENSE model (Shared Embedding for N-lingual Speech and tExt), a high-performing, language-agnostic, open-source model that generates semantic embeddings from speech and text. This produces a 1024-dimensional vector representing the semantic content of each utterance [Mdhaffar et al. 2025].

To obtain diverse and representative samples, we apply stratified sampling guided by k-means clustering. After clustering the dataset using one of the feature representations, we sample a total of N utterances with balanced coverage across all k clusters. To avoid under-representing small clusters, we first ensure that each cluster contributes at least one data point. We then iteratively and evenly sample additional utterances from clusters until reaching the target size N.

This approach prevents larger clusters from dominating the sample and guarantees coverage of smaller ones. We set N to approximately 50% of the total utterances, adjusting slightly per method to keep the total audio duration comparable to the random baseline. For the Loquacious medium and large datasets, we use $k = 150$ and $k = 200$, respectively. These values were selected as reasonable starting points that offered adequate cluster resolution. As exhaustive tuning of k is beyond the scope of this thesis.

We further explore three methods based on utterance length. The first simply selects the longest 50% of utterances, denoted as *length*. The second combines length and speaker diversity: we use the k-means clusters on speaker embeddings from the diversity sampling method, then instead of sampling randomly within clusters, we select the longest utterances from each cluster, denoted as *speaker+len*. Due to limited resources, we only tested this feature combination, as our results showed speaker diversity sampling was slightly more effective than the *random* baseline. Lastly, to determine whether the performance gains are due to the content of the files or simply to having more context per utterance, we ran an experiment using the exact same content as the *length* method but splitting each utterance in half; this is denoted *length split in half*. For all of these experiments, the total audio duration is set to be approximately equal to the random baseline.

5.3.3 Training settings

For pre-training, we use our implementation of the BEST-RQ framework introduced in Chapter 3, which employs a random-projection quantizer to generate pseudo-targets and Conformer layers as the model backbone [Chiu et al. 2022]. However, we adjust the Conformer component by adopting rotary positional embeddings which was shown to perform comparably with Relative Position embeddings while being more efficient [Su et al. 2024]. We configure the model with 12 Conformer layers, each of which has a hidden size of 640, 8 attention heads, and a feed-forward network dimension of 2048, resulting in approximately 100M parameters in total.

For fine-tuning for speech recognition, we adopt the SpeechBrain fine-tuning recipe for BEST-RQ, which applies a feed-forward neural network with CTC loss. We set the vocabulary size to 1,024 byte-pair encoding tokens following Parcollet et al. 2025.

We pre-train all models for a total of 200,000 updates on NVIDIA A100 GPUs. To estimate total GPU time, we measure the duration required to complete 50,000 training steps on a system with 8 A100s, and then extrapolate to the full 200,000 steps. We apply dynamic batching, which groups utterances of similar lengths and adjusts batch sizes to maintain a consistent amount of audio per batch. The maximum duration per batch is capped at 800 seconds per GPU, yielding a total maximum batch size of approximately 1.77 hours.

We also employ dynamic chunking, which applies dynamic masks to both the attention mechanism [Zhang et al. 2020a] and the convolutional block [Li et al. 2023]. This limits left and right context access in a randomized manner, effectively simulating both streaming and non-streaming

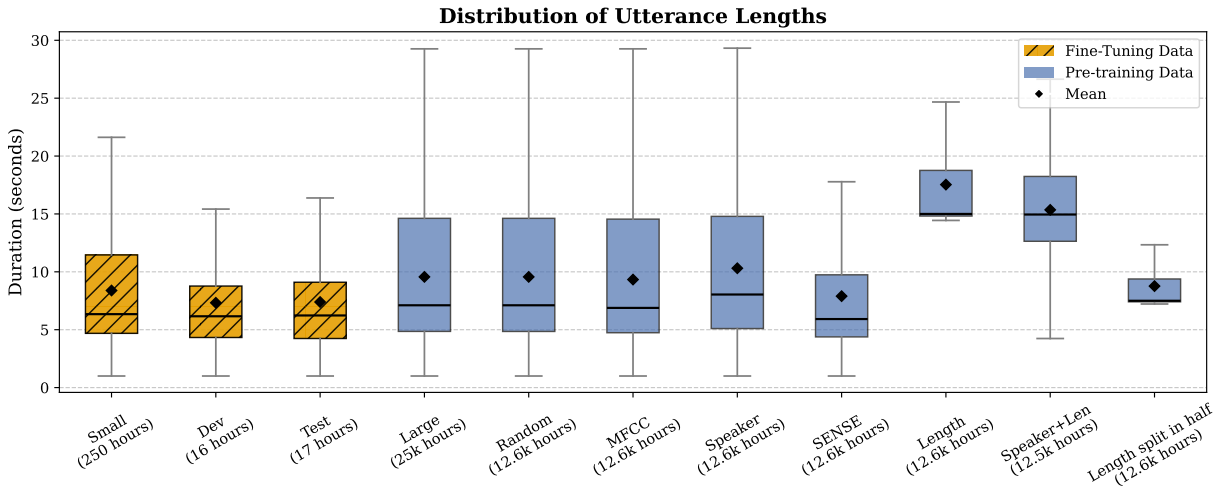


Figure 5.1: Box-and-whisker plots of utterance length distributions across different dataset splits. The three plots on the left correspond to the fine-tuning split, while the remaining plots show the various pre-training subsets. Notably, the best performing models were pre-trained on subsets consisting of longer utterances (far right two plots), whose distributions differ most from the fine-tuning data.

conditions during training. While our study does not focus on streaming, our prior work has shown that pre-training and fine-tuning with dynamic chunking improves performance for both streaming and non-streaming ASR [Duret et al. 2025]. All reported results are obtained in a non-streaming setting.

We train all models for 200,000 steps with identical hyperparameters, varying only the pre-training subsets described in Section 5.3.2. To assess statistical reliability, we perform bootstrapped paired-difference tests (1,000 samples) to calculate 95% confidence intervals; a result is deemed significant if the interval for the paired difference does not include zero.

5.4 Results

The results are presented in Table 5.1. Overall, diversity-based sampling methods did not yield significant improvements over either baseline. The only exception is the speaker-based method on the large split, which achieved a word error rate of 17.97 on the test set—significantly better than the random baseline (18.54) but not the *all* baseline (18.08).

In contrast, *length* and *speaker+len* methods consistently outperformed both baselines. On the medium split, selecting the longest utterances reduced the word error rate to 19.02 on the test set, while combining speaker diversity with utterance length further improved performance to

Table 5.1: Automatic Speech Recognition results with different pre-training subsets from the Loquacious medium and large splits. All models are fine-tuned on the small split. The *Time* and *Data* columns report the amount GPU hours and pre-training audio used.

Pre-train Split Selection method	WER *dev*	WER *test*	Time (hrs)	Data (hrs)
Medium Split				
All	18.48	19.39	195	2.50 k
Random	18.80	19.82	202	1.25 k
MFCC	19.05	19.98	202	1.24 k
Speaker	18.78	19.72	204	1.23 k
SENSE	19.23	20.39	199	1.24 k
Length	**18.08**	19.02*†	202	1.25 k
Speaker+Len	18.16	**18.97***†	205	1.24 k
Length split in half	18.47	19.46*	190	1.25 k
Large Split				
All	17.12	18.08	263	25.2 k
Random	17.53	18.54	214	12.6 k
MFCC	17.49	18.39	221	12.6 k
Speaker	17.26	17.97*	214	12.6 k
SENSE	17.67	18.42	385	12.6 k
Length	16.76	17.77*†	200	12.6 k
Speaker+Len	**16.60**	**17.42***†	201	12.5 k
Length split in half	16.95	18.01*	182	12.6 k
No pre-train [Parcollet et al. 2025]	22.30	23.80	-	-

*Significantly lower WER from *random* baseline ($p < 0.05$)
†Significantly lower WER from *all* baseline ($p < 0.05$)

18.97. On the large split, these same methods reached 17.77 and 17.42 word error rates on the test set, respectively, both significantly better than both baselines. Although the combined method performed best overall, the gains over length-only sampling were relatively small. On the other hand, the additional *length split in half* experiment led to worse results than the other length-based methods as well as the *all* baseline, yet it still outperformed the *random* baseline.

In terms of efficiency, training times for the medium split were similar across all methods. However, on the large split, length-based sampling substantially reduced pre-training time. Compared

to the *all* baseline, these methods trained about 24% faster. This improvement stems from the dynamic batching strategy: with longer utterances, batches contain fewer examples, which lowers per-step computational cost. For both medium and large splits, the *length split in half* trained the fastest, due to the benefits from dynamic batching as well as the speedup from training with smaller context lengths.

In the Loquacious dataset, prior experiments were conducted with models of similar size but using a Conformer encoder-decoder architecture trained fully supervised on the small split [Parcollet et al. 2025]. While this setup is not directly comparable, it provides a useful reference: the 100M parameter supervised model trained on 250 hours of labeled data. Notably, all of our data selection methods outperform this baseline, despite relying on a less complex architecture, demonstrating the benefits of self-supervised pre-training.

5.5 Conclusion

From our results, we find that reducing the amount of pre-training data by sampling based on acoustic, speaker, or linguistic features does not yield statistically significant improvements over a random baseline. In contrast, selecting subsets of longer utterances consistently leads to lower WER, despite these subsets being most out-of-distribution relative to the fine-tuning data. As shown in Figure 5.1, the best-performing subsets contain utterances with median lengths around 15 seconds and mean lengths above 15 seconds, whereas other subsets fall between 5 and 10 seconds.

Keeping the content the same but splitting the longest utterances led to worse results, showing that the better performance cannot be attributed solely to the content of these longer audio files but to the fact that they contain more context per utterance on average.

While further work is needed to understand the underlying cause of this effect as well as the applicability to other SSL speech models and datasets, our findings suggest that pre-training benefits from longer utterances even when fine-tuning data is dominated by shorter segments. This is potentially due to these utterances having longer, richer contexts for learning or providing more challenging learning signals. In future work, one could address these limitations and extend this work to better understand what makes a good pre-training dataset.

In summary, we systematically studied simple unsupervised data selection methods for pre-training SSL speech models. We showed that enforcing diversity across acoustic, speaker, or linguistic features does not significantly improve downstream ASR performance, whereas selecting the longest utterances achieves superior performance while using only half of the original dataset. Moreover, this approach reduces pre-training time by up to 24% on large datasets. These results highlight utterance length as a key factor in constructing efficient pre-training datasets for SSL speech models.

Chapter 6

Efficient Evaluation Strategies for Self-Supervised Speech Models

Table of Contents

6.1 Introduction . . . 82
6.2 Background . . . 83
6.2.1 Architecture of BEST-RQ . . . 83
6.2.2 Pre-training loss vs downstream performance . . . 83
6.2.3 Using rank for unsupervised evaluation . . . 84
6.3 Methods . . . 85
6.3.1 Unsupervised measures for SSL model evaluation . . . 85
6.3.2 Experimental settings . . . 86
6.4 Results . . . 87
6.4.1 Preliminary experiments . . . 87
6.4.2 Automatic speech recognition . . . 88
6.4.3 Speaker verification . . . 89
6.4.4 Additional observations . . . 90
6.5 Conclusion . . . 91

Publications related to this chapter

- **Interspeech 2025** – Towards Early Prediction of Self-Supervised Speech Model Performance [Whetten et al. 2025]

Summary

While in Self-Supervised Learning (SSL) for speech models, pre-training is the most resource-intensive stage; evaluation can also require significant resources. Furthermore, in the speech domain, current indicators of the quality of SSL models during pre-training, such as the loss, do not correlate well with downstream performance. Consequently, it is often difficult to gauge the final downstream performance in a cost-efficient manner during pre-training. In this work, we propose unsupervised, efficient methods that give insights into the pre-training quality of SSL speech models, namely measuring the cluster quality and rank of the embeddings produced by the SSL model. Results show that measures of cluster quality and rank correlate better with downstream performance than the pre-training loss, reducing the need for GPU hours and labeled data in SSL model evaluation.

6.1 Introduction

Over the course of this thesis, we have explored various strategies to enhance the training efficiency and data requirements of self-supervised speech models. In this chapter, we turn our attention to the evaluation phase. A primary challenge in evaluating self-supervised speech models is that the pre-training loss is not a reliable indicator of final performance on downstream tasks. Previous work has noted that continuing pre-training even when pre-training loss values are stagnant can improve downstream performance [Chen et al. 2023; Zanon Boito et al. 2024]. Consequently, self-supervised models are often fully pre-trained and fine-tuned on a downstream task for evaluation. However, these traditional evaluation methods of fine-tuning the model demand additional GPU hours and labeled data.

Few studies report in detail the GPU hours used for fine-tuning, but authors of wav2vec 2.0, HuBERT, and WavLM provided the number of GPUs used. For wav2vec 2.0, Baevski et al. 2020 mention fine-tuning on 8 to 24 GPU hours. For HuBERT, Hsu et al. 2021 report fine-tuning on 8 GPUs. For WavLM, Chen et al. 2022 note fine-tuning on 8 and 24 V100 GPUs for the base and large model respectively.

Previous research has explored various approaches to address these challenges. In the realm of computer vision, researchers proposed using the rank of the embeddings produced by a self-supervised vision model as an indicator of model quality, a measure they named RankMe [Garrido et al. 2023]. In the speech domain, Aldeneh et al. 2025 similarly proposed a rank measure called

RankMe-t extending RankMe to sequence tasks like speech. They apply it to the embeddings from self-supervised speech models and measure correlations with performance on phoneme recognition, keyword spotting, and speaker identification.

In this chapter, we propose unsupervised, efficient methods that provide insights into the pre-training quality of self-supervised speech models. Specifically, we focus on measuring the cluster quality and global entropic rank (GER) of the embeddings produced by these speech models. Our results demonstrate that these measures correlate better with downstream performance than the pre-training loss, thereby reducing the need for extensive GPU hours and labeled data in evaluation.

6.2 Background

In this section, we review the BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) framework and explain the calculation of the pre-training loss. Using this as a foundation, we explain theoretically why the pre-training loss is a poor indicator of downstream performance—a claim supported empirically by our results in Section 6.4.

6.2.1 Architecture of BEST-RQ

We briefly recall the main components of BEST-RQ; a full description appears in Chapter 2. The method combines a frozen random-projection quantizer that produces discrete pseudo-targets, an AFE consisting of two convolutional layers, and a Conformer-based encoder. The pre-training objective is the cross-entropy between the encoder predictions and the pseudo-targets. Because the encoder must recover masked regions from surrounding context, the masking strategy encourages the model to learn context-aware representations[1] [Chiu et al. 2022].

6.2.2 Pre-training loss vs downstream performance

In this section, we show theoretically how certain hyperparameter changes can affect the pre-training loss. Consider two self-supervised BEST-RQ models, M_a and M_b, with respective masking hyperparameters m_a and m_b, representing the percentage of audio masked during pre-training. Let $m_a < m_b$, where $m_a \approx 0$, implying that almost none of the input is masked for M_a. Assume both models use the same quantizer Q. Finally, let $\mathbf{X} \in \mathbb{R}^{n \times d} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ be an input sequence of length n composed of d-dimensional vectors.

A larger mask reduces the amount of uncorrupted input available to the encoder. Since the encoder must rely on global context to predict the targets of masked sections, model M_b—having a larger mask m_b—will exhibit greater uncertainty in its predictions. This results in higher entropy

[1] See Chapter 3 for more details on the masking strategy used in our implementation of BEST-RQ.

and, consequently, a higher loss. However, as explained below, a higher loss in this scenario does not necessarily imply inferior model quality.

Since m_a is negligible, M_a is trained to mimic Q, essentially learning a simple linear projection—a trivial task for Conformer architectures—resulting in a low loss. In this case, the model learns a mapping where $M_a(\mathbf{x}_i) \approx Q(\mathbf{x}_i)$. Conversely, because M_b has a higher masking percentage, it cannot simply replicate Q. Instead, it must leverage unmasked context to predict the quantization of missing segments, compelling the model to learn the underlying statistical properties of speech. Consequently, while M_b incurs a higher loss than M_a, it develops a more robust initialization for downstream tasks.

Therefore, while pre-training loss serves as a general guide, it is not a reliable indicator of downstream performance because hyperparameters and architectural choices directly influence the difficulty of the pre-training task. To address this limitation, recent studies have proposed using the rank of self-supervised model embeddings as an alternative evaluation measure.

6.2.3 Using rank for unsupervised evaluation

The rank of a matrix—the number of linearly independent rows or columns—serves as a proxy for its information content; a higher rank indicates lower redundancy and higher information density. Classically, the rank is computed by counting non-zero singular values. However, when processing real-valued data, many, if not all of the singular values will be non-zero, necessitating an application-dependent threshold that is difficult to tune. To address this, Roy et al. 2007 proposed the effective rank (E-Rank), a continuous, real-valued extension of rank defined as:

$$\text{E-Rank}(A) = \exp\left(-\sum_{i=1}^{\min(N,M)} p_i \log p_i\right), \tag{6.1}$$

where $A \in \mathbb{R}^{N\times M}$ and p_i represents the normalized singular values of A. This measure, referred to as RankMe in the context of SSL, has proven effective for evaluating representations in computer vision [Garrido et al. 2023].

In the speech domain, Aldeneh et al. 2025 adapted this approach to temporal data with RankMe-t. Given a dataset of D audio files, an SSL model generates a set of embedding sequences $S = \{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_n\}$. Each sequence $\mathbf{E}_i \in \mathbb{R}^{T_i\times d}$ consists of T_i vectors of dimension d, such that $\mathbf{E}_i = [\mathbf{e}_{i,1}, \mathbf{e}_{i,2}, \ldots, \mathbf{e}_{i,T_i}]$. To obtain a fixed-length representation, each sequence is summed along the time dimension:

$$\mathbf{z}_i = \sum_{t=1}^{T_i} \mathbf{e}_{i,t}. \tag{6.2}$$

These vectors are then concatenated to form the matrix $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n] \in \mathbb{R}^{D\times d}$. RankMe-$t$ is then defined as the effective rank of $\mathbf{Z}$:

$$\text{RankMe-}t(\mathbf{Z}) = \text{E-Rank}(\mathbf{Z}). \tag{6.3}$$

Although RankMe-t correlates well with downstream performance [Aldeneh et al. 2025], its correlation with pre-training loss has not been extensively reported, leaving it unclear if it serves as a superior indicator during training. Furthermore, because RankMe-t operates at the utterance level through temporal summation, we hypothesize that it loses valuable frame-level information. In this work, we expand upon this research by investigating cluster-based measures and the utility of rank-based metrics during the early stages of pre-training across two distinct tasks: Automatic Speech Recognition (ASR) and Speaker Verification (SV).

6.3 Methods

In this section, we introduce the unsupervised measures used in our experiments and then describe the experimental settings, including the models trained and the downstream tasks.

6.3.1 Unsupervised measures for SSL model evaluation

Similar to Garrido et al. 2023, we hypothesize that the dispersion (or lack thereof) of the output embeddings in the latent space correlates with the downstream performance of the model. To test this hypothesis, we employ the following clustering and rank-based measures.

For clustering measures, we train KMeans [Lloyd 1982] clustering models on the SSL embeddings from a given layer. We use MiniBatchKMeans [Sculley 2010], an efficient variant of KMeans implemented in scikit-learn [Pedregosa et al. 2011], using a KMeans++ initialization [Arthur et al. 2007]. We set the number of clusters to $K = 1024$, following recent work on self-supervised speech models [Mousavi et al. 2024].

Once a clustering model is trained, we use two metrics for evaluating the clustering quality: inertia, also known as the Within-Cluster Sum of Squares (WCSS), and the Davies-Bouldin (DB) index [Davies et al. 1979]. For both of these measures, a lower value indicates a better clustering. The WCSS measures the sum of the intra-cluster variance, defined as:

$$\text{WCSS} = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2, \tag{6.4}$$

where K is the number of clusters, C_k is the set of data points assigned to cluster k, x_i is a single data point and μ_k is the centroid of cluster k. While the WCSS only considers intra-cluster dispersion, the DB index accounts for both intra-cluster and inter-cluster distances, calculated as:

$$\frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j} \frac{s_i + s_j}{d_{ij}}, \tag{6.5}$$

where K is the number of clusters, s_i is the diameter of cluster i and d_{ij} is the distance between centroids of clusters i and j.

For rank-based evaluation, we use RankMe-t as described in Section 6.2.3 as a baseline, and propose an additional rank based measure: the Global Effective Rank (GER).

Given the series of D embedding sequences S (same as Section 6.2.3), let C represent the concatenation of all the embeddings in S shown in Equation 6.6. We propose the Global Effective Rank as the Effective rank of C. This can be expressed as follows:

$$\mathbf{C} = [\mathbf{E}_1^\top, \mathbf{E}_2^\top, \ldots, \mathbf{E}_N^\top]^\top \in \mathbb{R}^{(\sum T_i) \times d}, \tag{6.6}$$

$$\text{GER} = \text{E-Rank}(\mathbf{C}). \tag{6.7}$$

The primary distinction between these rank measures lies in the level of granularity: while RankMe-t computes the rank of utterance-level summations—thereby losing frame-specific variations through temporal averaging—the GER operates directly on the frame-level representations. GER aims to capture the true dimensionality and expressivity of the latent space more accurately.

6.3.2 Experimental settings

We conduct our experiments using the BEST-RQ framework due to its state-of-the-art performance, computational efficiency, and the open-source implementation we provide in Chapter 3. This efficiency is particularly critical given the need to train a large number of models to validate our evaluation methodology. We develop various hyperparameter configurations by varying the batch size from 15 to 120 minutes of audio and the proportion of masked audio from 40% to 80%. All models are trained for 200k steps on the 960-hour subset of LibriSpeech [Panayotov et al. 2015].

We focus our analysis on two main checkpoints: 50k steps (representing an early-stage model) and 200k steps (representing a converged model). While many self-supervised speech models are traditionally trained for longer durations [Baevski et al. 2020; Hsu et al. 2021; Chiu et al. 2022], previous research has demonstrated that 200k steps are sufficient for comparative evaluation [Parcollet et al. 2023].

For our first task, we evaluate the frozen SSL models on in-domain ASR using the ContextNet LibriSpeech-100 task from the Multi-probe Speech Self Supervision (MP3S) benchmark Zaiem et al. 2023. In this setup, a minimal ContextNet [Han et al. 2020] probe is trained on the *train-clean-100* subset of LibriSpeech. Performance is measured by the Word Error Rate (WER) on the *dev-clean* set using greedy decoding without a language model. Since fine-tuning is common in practice, we also fine-tune the SSL models on in-domain ASR data using the SpeechBrain fine-tuning recipe. Evaluation follows the same setup as the frozen model, measuring WER on the *dev-clean* set with greedy decoding and no language model. For our third task, we evaluate the frozen SSL models on out-of-domain ASR using the LSTM CommonVoice 11.0 Basque ASR

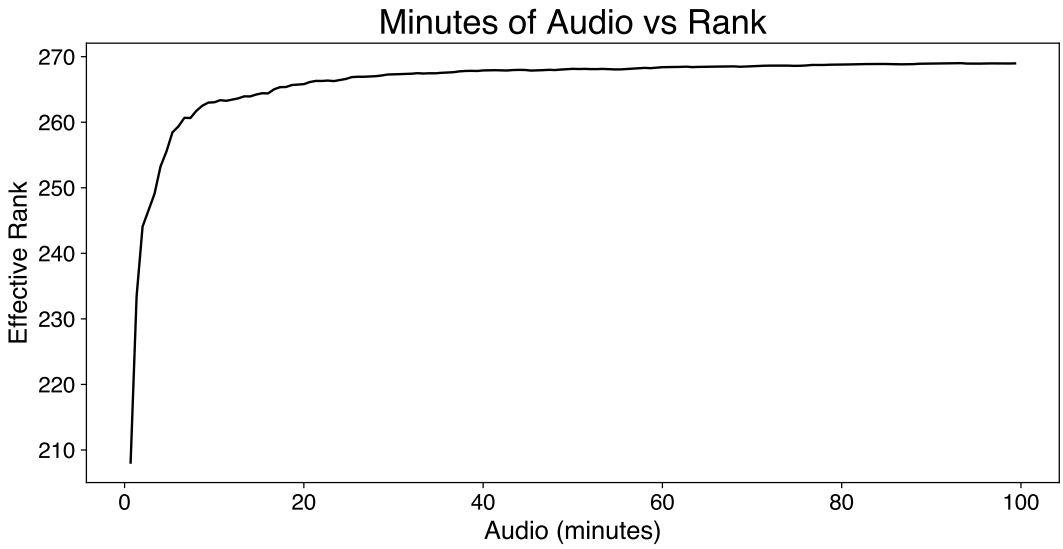


Figure 6.1: The effective rank of embeddings produced by the SSL model relative to the amount of audio used. The effective rank plateaus early, at less than 20 minutes of audio, and increases only marginally thereafter.

task from the MP3S. For this task, a 2-layer BiLSTM probe is trained on the provided train splits. Evaluation is done using the WER on the provided development set.

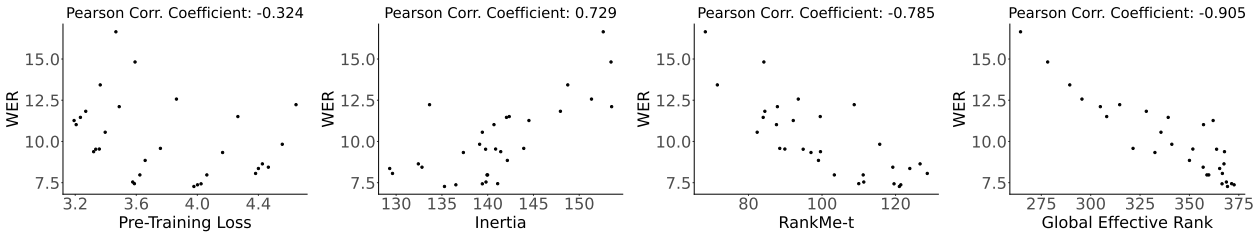
Lastly, to assess performance on a non-ASR task, we follow the MP3S benchmark to evaluate the frozen SSL models on SV using VoxCeleb1 [Nagrani et al. 2017]. In this task, an ECAPA-TDNN [Desplanques et al. 2020] probe is trained and models are evaluated using Equal Error Rate (EER) on the development set.

6.4 Results

We first report on preliminary experiments that informed our choices on the amount of audio to use for the unsupervised evaluation methods and the number of epochs for the downstream task. We then present results for each task.

6.4.1 Preliminary experiments

For obtaining the output embeddings, we randomly sample approximately 1 hour of audio from each dataset. This choice is based on preliminary experiments where we observed that the global effective rank plateaus quickly, as shown in Figure 6.1, and increases minimally when using more than one hour of audio. Computing all embeddings and metrics for a single layer of one model takes about 20 minutes on CPUs, with the clustering accounting for the majority of the computation time.



(a) WER vs Pre-Train Loss (b) WER vs Inertia (c) WER vs RankMe-t (d) WER vs GER

Figure 6.2: Correlations between 4 selected measures (calculated on embeddings at 50k pre-training steps) and the future WER on the LibriSpeech *dev-clean* set (at 200k pre-training steps and 20 epochs fine-tuning). Each point represents a model with different hyper-parameters (batch size and mask percentage). Fig. 6.2a, shows that some of the models with the lowest WER (y-axis) have a relatively high pre-training loss (x-axis) resulting in a low correlation, whereas unsupervised measures have a stronger correlation with WER.

This is demonstrated in Figure 6.1.

Another preliminary finding is the strong correlation between the Word Error Rate (WER) early in pre-training and the final WER when utilizing in-domain data. For instance, the Pearson correlations between the WER of minimally trained models (50k pre-training steps and 5 downstream epochs) and their fully trained counterparts (200k pre-training steps and 20 downstream epochs) were 0.951 and 0.967 for the frozen and fine-tuned settings, respectively. This high degree of correlation suggests that early-stage WER results are a reliable predictor of a model's final quality. Consequently, for in-domain ASR, it is not necessary to conduct full pre-training or exhaustive fine-tuning to accurately assess the comparative strengths of different model configurations. To optimize computational resources, we therefore limit the downstream training to 5 epochs for the 50k-step checkpoints, as these early results provide a clear indication of the performance trends expected at convergence.

6.4.2 Automatic speech recognition

For in-domain ASR experiments, results were similar whether the SSL models were fine-tuned or frozen; both cluster and rank methods correlate with the WER better than the pre-training loss baseline (see first four columns of Table 6.1). Some of the highest correlations came from the GER ranging from -0.63 to -0.96, meaning the higher the rank the better the performance (lower WER). In contrast, the pre-training loss did not correlate strongly with WER, with correlations ranging from -0.170 to -0.303. Graphs comparing a select few of these measures with the WER are shown in Fig. 6.2.

Similarly, we find high correlations between clustering and rank measures at 50k steps of pre-

training and final WER after 200k steps of pre-training and 20 epochs of fine-tuning. This suggests that, without any fine-tuning, we can gauge the final performance early on in pre-training for in-domain ASR. For example, the correlations between the GER and final WER were around -0.9, yet only around -0.3 for the pre-training loss (first two columns of Table 6.2).

When it comes to out-of-domain (CommonVoice) ASR experiments, correlations between unsupervised measures and WER remain high at 50k steps, with the highest being -0.885 for the GER. However, the correlations at 200k are not strong, nor are the correlations between unsupervised measures at 50k and the final WER at 200k (Table 6.2). Examining why this might be the case, we observe that at 200k all the models perform similarly, with an absolute difference in WER of only 2.8% absolute difference between the worst and best performance, suggesting a potential saturation effect where further increases in embedding rank or improvements in cluster separation capture structural variance that is no longer task-relevant.

Table 6.1: Pearson Correlations between downstream performance and various unsupervised measures. Compared to pre-training loss (first row), unsupervised metrics tend to exhibit a much stronger correlation with downstream performance at either 50k or 200k steps of pre-training. Highest correlations are found in the in-domain tasks. Measures based on rank are more correlated with downstream performance than clustering measures. LS = LibriSpeech, CV = CommonVoice, VoxCel = VoxCeleb, (f.t.) = SSL Model is fine-tuned.

Measure	**LS 50k**	**LS 200k**	**LS 50k** (f.t.)	**LS 200k** (f.t.)	**CV 50k**	**CV 200k**	**VoxCel 50k**	**VoxCel 200k**
Pre-train Loss	-0.183	-0.303	-0.170	-0.332	0.107	0.143	0.084	-0.598
Inertia (WCSS)	0.686	0.622	0.710	0.633	0.412	-0.556	-0.000	0.367
Davies-Bouldin	-0.903	-0.494	-0.929	-0.486	-0.854	-0.414	0.155	0.356
RankMe-t	-0.752	-0.735	-0.734	-0.756	-0.387	0.110	0.376	0.567
Global Rank (GER)	-0.931	-0.630	-0.962	-0.634	-0.885	-0.003	0.163	0.584
In-Domain	Yes	Yes	Yes	Yes	No	No	No	No
Epochs on DS task	5	20	5	20	5	20	20	20

6.4.3 Speaker verification

Regarding SV, we observe correlations between unsupervised measures and the error rate after 20 epochs of fine-tuning. While these correlations are not as strong as those observed for ASR, the unsupervised measures still outperform the pre-training loss as predictive indicators. Notably, for the 200k-step checkpoints, there is a strong negative correlation ($r = -0.598$) between the loss and downstream performance (see Table 6.1, last column). This indicates that a higher pre-training loss corresponds to a lower error rate, suggesting that the loss objective is inversely predictive of SV performance, which is not useful. Similar to the CommonVoice results, we find

Table 6.2: Pearson Correlations between various unsupervised measures and **future** downstream performance. The unsupervised measures are calculated from the embeddings of SSL models at 50k pre-training steps and the correlation is measured with their respective final performances at 200k pre-training steps. Correlations for in-domain ASR remains high, but not for out-of-domain tasks. LS = LibriSpeech, CV = CommonVoice, VoxCel = VoxCeleb, (f.t.) = SSL Model is fine-tuned.

Measure	**LS**	**LS (f.t.)**	**CV**	**VoxCel**
Pre-train Loss	-0.296	-0.324	0.140	-0.604
Inertia (WCSS)	0.692	0.729	-0.378	-0.027
Davies-Bouldin	-0.840	-0.837	0.095	0.056
RankMe-t	-0.771	-0.785	0.201	0.200
Global Rank (GER)	-0.903	-0.905	0.159	0.318

that the majority of models performed within a narrow margin—90% of the models exhibited error rates within a 1% range—which likely contributed to the lower overall correlations.

6.4.4 Additional observations

Previous research has identified the middle layers of SSL models as the most critical for ASR, as they contain the highest density of phonetic and word-level information [Pasad et al. 2021]. However, our results align with the findings of Aldeneh et al. 2025, where the layers most useful for the downstream task are not necessarily those that correlate most strongly with downstream performance. For ASR, we found that the measures computed on the embeddings extracted from the first and last layers correlated the most with downstream performance, indicating that the way the model processes incoming acoustic features (or reconstructs them in the case of the last layers) is an important indicator of downstream ASR performance. For SV, we found that the middle layers tended to have stronger correlations than beginning or final layers. We report the correlations from measures calculated on the outputs of layer 12 for ASR and layer 8 for SV in both Table 6.1 and 6.2.

Another significant finding is that the correlations between inertia and downstream performance are consistently positive, except in the out-of-domain ASR case at 200k steps where performance converged. This positive correlation confirms that lower inertia—indicating better cluster compactness—is associated with superior performance. Conversely, the correlations with the DB index are negative for ASR, implying that a worse clustering score (by DB standards) aligns with better performance. We believe this occurs because the DB index penalizes clusters that are in close proximity to their nearest neighbors. However, closely packed clusters do not necessarily indicate poor representation; close neighboring clusters suggest efficient use of the latent space.

6.5 Conclusion

In this chapter, we have demonstrated why the pre-training loss serves as a poor indicator of representation quality for self-supervised speech models. We have proposed the use of clustering and rank-based measures as efficient, unsupervised alternatives for evaluating self-supervised speech models, showing that these measures are better indicators of downstream performance than the pre-training loss.

To illustrate the potential computational savings, consider the training of the 30 models discussed in this chapter to 400k steps. Under our experimental setup, this would require approximately 16,800 GPU hours. If, instead, all models were pre-trained to only 50k steps and the rank measures were used to select the top five candidates for continued training to 400k steps, the total requirement would drop to approximately 6,000 hours. This approach would save approximately 10,800 GPU hours while still identifying the highest-performing architectures.

Future research may involve further refining these unsupervised metrics and developing more robust early-stage indicators to evaluate self-supervised speech models across a broader range of architectures and languages.

Chapter 7

Efficient Self-Supervised Learning for Speech in Low-Resource Settings

Table of Contents

7.1 Introduction . . . 94
7.2 Related Work . . . 95
7.2.1 Training efficiency . . . 96
7.2.2 SSL speech models for low-resource languages . . . 96
7.2.3 Low-data pre-training . . . 97
7.2.4 Fongbe speech recognition and translation . . . 98
7.3 Methods . . . 98
7.3.1 Data . . . 98
7.3.2 Experiment Setting . . . 100
7.3.3 Fine tuning and evaluation . . . 101
7.4 Results . . . 101
7.5 Conclusion . . . 103

Publications related to this chapter

- **Under Review** – Single-GPU Pre-training: Low-Resource Self-Supervised Learning for Speech
- **ICASSP 2026** – Ara-Best-RQ: Multi Dialectal Arabic SSL [Elleuch et al. 2026]
- **SPECOM 2025** – In-Domain SSL Pre-training and Streaming ASR: Application to Air Traffic Control Communications [Duret et al. 2025]

Summary

Pre-training self-supervised speech models is traditionally compute-intensive, often requiring thousands of hours of data and massive GPU clusters. In this chapter, we demonstrate that effective pre-training is achievable in low-resource settings with significantly fewer resources. Using Fongbe, a language of Benin, as a case study, we pre-train models with the BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ) framework with only 140 to 700 hours of unlabeled data. We evaluate our approach on Automatic Speech Recognition (ASR) and Speech Translation (ST) tasks using the Fongbe-to-French Speech Translation Corpus 2. Compared to AfriHuBERT—a state-of-the-art model for African languages—our results demonstrate that single-GPU pre-training achieves comparable or superior performance despite using 100 to 600 times less data and an estimated 320- to 1000-fold reduction in computational cost. These findings suggest a significantly lower barrier to entry for self-supervised speech models in low-resource settings.

7.1 Introduction

Throughout this thesis, we have examined efficiency from various perspectives, including open-sourcing the most efficient self-supervised learning (SSL) framework for speech models in Chapter 3, improving model efficiency in Chapter 4, exploring strategies for efficient pre-training data selection in Chapter 5, and developing efficient evaluation methods in Chapter 6. While these efforts have yielded significant insights, much work remains to make SSL more accessible and efficient, particularly for low-resource languages.

Although the methodologies proposed in this thesis are theoretically language-agnostic, the majority of our experiments—and indeed much of the existing literature [Baevski et al. 2020; Baevski et al. 2022; Hsu et al. 2021]—rely on clean English audiobook data for pre-training. In research, this approach is adopted to maintain standardized benchmarks for fair comparison; consequently, English audiobook datasets have become the default due to their accessibility. Outside of controlled experimental settings, however, real-world data can differ drastically from the clean

recordings of professional speakers.

In the context of low-resource languages, a common practice is to utilize models trained on massive datasets, either using them out-of-the-box [Chaparala et al. 2024] or fine-tuning them to improve performance [Kponou et al. 2024]. While effective, these methodologies offer less control over—and insight into—the underlying training data and force researchers to rely on pre-defined model architectures and sizes. Furthermore, many publicly available models remain underperforming for low-resource languages, often exhibiting significant biases toward high-resource languages [Chaparala et al. 2024].

Given these drawbacks, pre-training from scratch may seem like the ideal path to achieve full control over data, architecture, and model scale. However, data collection, labeling, and training remain prohibitively expensive and time-consuming, typically requiring vast datasets and high-performance computing clusters. While studies have explored training in very low-resource settings—such as on a single GPU [Liu et al. 2021; Lin et al. 2023; Lugo et al. 2024]—these efforts have primarily focused on clean English data.

In this chapter, we present the first application of BEST-RQ for pre-training in a noisy, low-resource setting regarding both data and compute, focusing on Fongbe, a language spoken in the Republic of Benin. Our pre-training corpus consists of 140 hours of radio broadcasts and Automatic Speech Recognition (ASR) data in Fongbe. We investigate the impact of data augmentation and the inclusion of an additional 560 hours of multilingual speech data during pre-training, ensuring the dataset remains balanced across languages to mitigate bias. Models are evaluated on ASR and Speech Translation (ST) tasks, using AfriHuBERT [Alabi et al. 2025], the current state-of-the-art for Fongbe, as our baseline [Kponou et al. 2025].

Our results demonstrate that competitive pre-training is achievable with a fraction of traditional data and compute requirements. Despite using 100–600$\times$ less pre-training data, our models achieve comparable or superior performance to AfriHuBERT across both ASR and ST tasks. Crucially, our approach is highly efficient: while AfriHuBERT (derived from mHuBERT-147) consumed approximately 30,000 A100 GPU hours, all of our experiments were conducted using only 132 to 528 hours on a single RTX 3090 GPU. This represents a 320–1,000$\times$ reduction in computational cost, highlighting the feasibility of low-cost pre-training for under-resourced languages. We release our code and models to support reproducibility and accessibility within the community[1].

7.2 Related Work

In this chapter, we bridge three distinct research directions aimed at improving Self-Supervised Learning (SSL) for speech. First, we leverage advancements in **training efficiency** to reduce the heavy computational requirements of pre-training. Second, we follow **multilingual low-resource**

[1] https://github.com/whettenr/fongbe-brq

language SSL trends to train and release models that are high-performing for low-resource languages (note that these models, however, are still often trained on large amounts of speech data from high-resource languages). Lastly, we draw inspiration from **low-data pre-training**, which focuses on learning robust representations from a limited amount of audio data. In this section, we review these domains and provide an overview of Fongbe speech recognition and translation efforts.

7.2.1 Training efficiency

To improve SSL training efficiency, researchers have focused on three primary areas: architecture, objectives, and engineering. We summarize these efforts here (see Chapter 1 for a more complete overview).

First, architectural modifications have involved replacing the 7-layer convolutional feature extractor of wav2vec 2.0 and HuBERT [Baevski et al. 2020; Hsu et al. 2021] with log-Mel filterbanks [Yang et al. 2023; Chiu et al. 2022]. Starting with log-Mel filterbanks is computationally lighter and does not require any learnable parameters, reducing the parameter count and, consequently, memory requirements. Second, efficiency is gained by simplifying the objective; for example, replacing the more complex contrastive and diversity losses of wav2vec 2.0 with a simple cross-entropy loss, as seen in Fast-HuBERT and BEST-RQ [Yang et al. 2023; Chiu et al. 2022]. Lastly, engineering optimizations have further reduced redundancy through the offline computation of features [Yang et al. 2023], the reuse of targets [Baevski et al. 2023], or by leveraging high-quality targets derived from existing pre-trained or fine-tuned models [Chen et al. 2022; Yang et al. 2023; Chen et al. 2023].

Despite these efficiency improvements, many of these models are still trained on clusters of at least 8 GPUs, and these methods are mostly tested on large, English-only audiobook datasets such as LibriSpeech [Panayotov et al. 2015] and Libri-light [Kahn et al. 2020]. See Sections 2.6.1 and 2.6.2 for more details on typical GPU and data requirements for self-supervised speech models. We take advantage of these insights by using the BEST-RQ framework and applying it in a low-resource scenario, both in terms of data (140–700 hours) and compute (maintaining model and batch sizes small enough to train on a single 24-GB GPU).

7.2.2 SSL speech models for low-resource languages

As SSL can take advantage of unlabeled data, developing models for languages that have a limited amount of labeled data is an active area of research. Recently, mHuBERT-147 was released as a minimal (95M parameters) model pre-trained on over 90k hours, covering 147 languages [Zanon Boito et al. 2024]. This model was further specialized for African languages by continuing pre-training on an additional 10k hours across 1,226 African languages, resulting in increased perfor-

mance on African language downstream tasks [Alabi et al. 2025]. Similarly, SSA-HuBERT was pre-trained on 60k hours of audio covering 21 sub-Saharan African languages [Caubrière et al. 2024]. We previously released Ara-BEST-RQ, a set of models trained on 5,600 and 13,700 hours of audio, including various Arabic dialects from North Africa [Elleuch et al. 2026].

By specializing, these models have pushed the boundaries of performance on low-resource African languages using relatively small model architectures. However, they still required large amounts of data and GPU resources to train. For example, mHuBERT-147 required 40–60 days on 32 $\times$ A100 GPUs [Zanon Boito et al. 2024]. While the total pre-training time was not reported for SSA-HuBERT, it was trained on 4 $\times$ A100 GPUs for 775k steps [Caubrière et al. 2024]. In this work, we follow these trends by pre-training a specialized minimal model, but we push this approach to the extreme by pre-training on only one to five languages.

7.2.3 Low-data pre-training

Despite the fact that SSL models can take advantage of unlabeled data, such data can still be expensive and difficult to obtain due to the costs of collection and privacy or copyright restrictions. Consequently, it remains critical to determine the extent to which robust representations can be learned under such extreme data constraints.

In Wav2Vec-Aug [Sriram et al. 2022], researchers experimented with wav2vec 2.0 [Baevski et al. 2020] in a low-data pre-training setting. By using augmentation, replacing Transformer layers with Conformer layers, substituting a portion of the convolutional layers in the feature encoder with lightweight and dynamic convolutions, and adding multilayer perceptrons before the loss, they obtained strong ASR results on LibriSpeech [Panayotov et al. 2015]. These results were achieved using only fractions of the 1,000-hour dataset for pre-training and fine-tuning. However, pre-training remained relatively expensive, requiring 48 V100 GPUs and approximately 3,840 total GPU hours for a single model (in contrast, our work utilizes between 132 and 528 RTX 3090 GPU hours).

Similarly, in data2vec-aqc [Lodagala et al. 2023], augmentation is added to data2vec [Baevski et al. 2022]. The authors also include a wav2vec 2.0-style quantizer module and a clustering module to help reduce the effect of *non-informative* negative samples in the contrastive loss task. This led to competitive results when pre-training on the 360-hour split of the LibriSpeech dataset for fewer than 90k updates, whereas original wav2vec 2.0 and data2vec models are typically trained for 400k updates. While the total training time was not reported, training was conducted on 4 A100 GPUs.

While these methods are data-efficient, the addition of extra modules increases complexity and can decrease training efficiency. Additionally, evaluations have primarily been conducted on relatively clean English ASR tasks, leaving open the question of the utility of low-data pre-training with spontaneous, noisy data.

While prior work has addressed these dimensions in isolation, there remains a lack of research

at the intersection of data efficiency, computational accessibility, and low-resource language modeling. In this chapter, we bridge this gap by evaluating the performance attainable under the extremely limited and noisy conditions—such as spontaneous speech and varied recording environments—characteristic of the Fongbe language.

7.2.4 Fongbe speech recognition and translation

Fongbe (or Fon) is a Niger-Congo tonal language spoken primarily in Benin, as well as in Nigeria and Togo. The first published ASR systems for the language were trained on a minimal dataset (10 hours) and obtained word error rates (WER) above 40% [LAleye et al. 2016; Dossou et al. 2021] when accounting for diacritics. For speech-to-text translation, researchers released a Fongbe-to-French dataset [Kponou et al. 2024] and achieved a BLEU score of 8.96 using a Transformer-based end-to-end model.

In the current state-of-the-art, researchers leverage pre-training by starting from Transformer-based models such as AfriHuBERT and NLLB. By fine-tuning these models, they have achieved a WER of 21.98 for ASR and a BLEU score of 26.32 on the Fongbe to French Speech Translation Corpus 2 [Kponou et al. 2025]. In other work focusing on Fongbe-to-French translation, Kponou et al. 2026 experimented with hybrid feature fusion strategies to combine acoustic features from two models, AfriHuBERT and XLS-R [Babu et al. 2022], reaching a BLEU score of 27.78 with their AfriHuBERT-NLLB model. To ensure the experiments in this chapter are as comparable as possible to the current state of the art, we use the same fine-tuning dataset and scripts provided by the authors [Kponou et al. 2025], modifying only the pre-trained model and the data.

7.3 Methods

In this section, we describe the data used, experimental settings, and evaluation metrics for pre-training and fine-tuning. For our experiments, we use the implementation of BEST-RQ from Chapter 3 within the SpeechBrain library [Ravanelli et al. 2024].

7.3.1 Data

The first dataset we curated is a Fongbe-only dataset consisting of radio and ASR recordings. We obtained approximately 110 hours of radio data by scraping broadcasts in the local language from Capp FM, a radio station based in the Republic of Benin. Each broadcast has an average duration of 45 minutes and contains music, jingles, code-switching, and varied speaker acoustic conditions due to the mixture of studio broadcasters and guests calling in via telephone. We split all radio recordings into 20-second intervals for training without any voice activity detection (VAD). This was done to evaluate performance when pre-training on raw, noisy data. The ASR recordings

come from two sources: 26 hours from the ASR training set provided by Kponou et al. 2025 and 4 hours from the training sets in LAleye et al. 2016, which were made accessible on HuggingFace[2]. This results in a total of 140 hours of Fongbe data. Models trained on this data are denoted as *Fon*.

Additionally, we incorporate audio from other languages. We enforced two selection criteria: (1) permissive licensing for accessibility and reproducibility, and (2) maintaining a balanced data distribution per language to prevent model bias.

Our first multilingual dataset was created by randomly sampling 140 hours each from the English and French LibriSpeech corpora Panayotov et al. 2015; Pratap et al. 2020 and combining these with our 140 hours of Fongbe for a total of 420 hours. We denote models pre-trained on this dataset as *Fon-EF*. Our second multilingual dataset focuses on West African tonal languages. We created this dataset by combining the Hausa and Yoruba training splits from VoxLingua107 [Valk et al. 2021], CommonVoice 23.0 [Ardila et al. 2020], and BibleTTS [Meyer et al. 2022]. As there was more Hausa data (180 hours) than Yoruba (120 hours), we removed 20 hours of the Hausa BibleTTS to arrive at 160 hours of Hausa and 120 hours of Yoruba. BibleTTS was selected for reduction because it contains the least diversity, consisting of only single-speaker speech. We combined this with 140 hours of Fongbe for a comparable 420-hour dataset denoted as *Fon-HY*. By creating these two datasets of similar size—one based on high-resource languages and one based on African tonal languages more closely related to Fongbe—we aim to study the impact of different types of multilingual data on model performance.

Lastly, we combined all aforementioned data to create a dataset consisting of Fongbe, English, French, Hausa, and Yoruba, denoted as *Fon-EFHY*. These datasets are summarized in Table 7.1.

Table 7.1: Pre-training datasets. We create a minimal 140 hour Fongbe only pre-training dataset and combine this with audio from various sources to create larger pre-training datasets. *Eng.* and *Fre.* denote English and French.

Dataset	Languages	Sources (Licenses)	Total Hours
Fon	Fongbe	Capp FM radio (this chapter) + ASR corpora [Kponou et al. 2025; LAleye et al. 2016]	140
Fon–EF	Fongbe, Eng., Fre.	Fon + Librispeech [Panayotov et al. 2015; Pratap et al. 2020] (CC BY 4.0)	420
Fon–HY	Fongbe, Hausa, Yoruba	Fon + VoxLingua107 [Valk et al. 2021] (CC BY 4.0) + CommonVoice 23.0 [Ardila et al. 2020] (CC0-1.0) + BibleTTS [Meyer et al. 2022] (CC BY-SA)	420
Fon-EFHY	Fongbe, Eng., Fre., Hausa, Yoruba	Combination of all above sources	700

[2]https://huggingface.co/datasets/beethogedeon/fongbe-speech

7.3.2 Experiment Setting

We maintain the same architecture for all models trained in this chapter. The architecture is based on the open-source BEST-RQ implementation in Chapter 3 within SpeechBrain, using the following configurations. The two convolutional layers are configured with 128 and 32 output channels, kernel sizes of 5, and strides of 2. The Conformer dimensions are set to 640 with 8 attention heads, and the feed-forward network dimension is set to 2048. Similar to Chapter 5, we use Rotary Positional Embedding (RoPE), which is more efficient and has proven to be high-performing compared to traditional Relative Positional Embeddings [Zhang et al. 2025a]. The codebook is configured with 8,192 entries of 16 dimensions, as described in the original implementation [Chiu et al. 2022]. This results in a model size of approximately 100 million parameters. To prevent overfitting on our minimal datasets, we set the Transformer dropout and encoder layer drop to 0.1 for all models, except for the model trained on 700 hours of data, for which we reduce the encoder layer drop to 0.05. This strategy for preventing overfitting on smaller pre-training datasets is consistent with prior work [Baevski et al. 2020; Chen et al. 2022].

For pre-training, we set the maximum learning rate to 0.0008 with 25,000 warmup steps. We employ dynamic batching, where audio data is separated into buckets of similar lengths and the batch size varies based on the total audio duration in the batch. We set the maximum number of seconds per GPU to 300. Gradient accumulation is set to 16 to achieve an effective batch size of approximately 1.3 hours. Furthermore, we employ dynamic chunking, which applies dynamic masks to both the attention mechanism [Zhang et al. 2020a] and the convolutional block [Li et al. 2023]. This randomly limits left and right context access, simulating both streaming and non-streaming conditions during training. While this study does not focus on streaming, previous research has shown that pre-training and fine-tuning with dynamic chunking improves performance for both streaming and non-streaming ASR [Duret et al. 2025]. All reported results are obtained in a non-streaming setting. Due to resource constraints, we train most models for 100k updates, with the exception of the *Fon-EFHY* models, which are trained for 400k updates.

In previous work, data augmentation was shown to be beneficial when working with minimal amounts of data [Sriram et al. 2022]. We experiment with augmentation during pre-training using the following methods: speed perturbation (randomly speeding up or slowing down audio by 5% or keeping the original speed), frequency augmentation (dropping a random number of frequency bands to zero), and chunk augmentation (dropping a random number of temporal chunks). Each model requires approximately 5.5 days (132 hours) to pre-train for 100k updates on a single RTX 3090 GPU (24 GB VRAM).

7.3.3 Fine tuning and evaluation

We evaluate all models using the same downstream architecture, data, and hyperparameters for the ASR and ST tasks from the Fongbe-to-French Speech Translation Corpus 2 (FFSTC 2), following the work of Kponou et al. 2025.

For ASR, the output of the self-supervised speech models is passed into a minimal four-layer feed-forward neural network with Batch Normalization [Ioffe et al. 2015], LeakyReLU activation [Maas et al. 2013], and a dropout [Srivastava et al. 2014] of 0.15 between each layer. Each layer contains 1024 neurons. The entire model is fine-tuned using CTC loss [Graves et al. 2006]. We evaluate performance using the character error rate (CER) and the word error rate (WER). Following the FFSTC 2 evaluation protocol, we report results both with and without diacritics.

For Speech Translation (ST), we evaluate both cascade and end-to-end systems, similar to Kponou et al. 2025. In the cascade system, Fongbe audio is transcribed by our ASR models and then translated into French text using the fine-tuned NLLB model from Kponou et al. 2025. For our end-to-end system, we experiment with three different initializations: (1) starting from the pre-trained SSL checkpoints, (2) starting from the fine-tuned ASR checkpoints, and (3) starting from the fine-tuned ASR checkpoints while also initializing the NLLB model from the fine-tuned checkpoint used in the cascade system (rather than the original NLLB checkpoint). All parameters are unfrozen during training except within the NLLB model, where fine-tuning is restricted to the encoder attention and layer normalization weights. This restriction reduces the number of trainable parameters to approximately 200 million. We evaluate translation quality using the BLEU score [Papineni et al. 2002].

Following Kponou et al. 2025, both ASR and ST models are fine-tuned for 50 epochs.

7.4 Results

When pre-training on only 140 hours of Fongbe with no augmentation, we obtain a WER of 24.28 and 18.68 on the test set with and without diacritics, respectively. These results are initially worse than AfriHuBERT, which achieves a WER of 21.98 and 17.02. However, adding augmentation during pre-training improves performance from 24.28 to 21.91 (and from 18.68 to 16.95 without diacritics), reaching comparable performance to AfriHuBERT while using significantly less data.

Increasing the amount of pre-training data from 140 to 420 hours yields two main insights. First, models trained on 420 hours achieve comparable or superior performance to AfriHuBERT without augmentation. Secondly, models trained on 420 hours do not benefit as significantly from data augmentation. For example, while the *Fon* model improved by 2.37 absolute WER on the test set with diacritics, the *Fon-EF* and *Fon-HY* models improved by only 0.01 and 0.59, respectively. The ASR results are summarized in Table 7.2.

For speech translation, we select all models pre-trained without augmentation, in addition to the

Table 7.2: Character Error Rate (CER) and Word Error Rate (WER) on Fongbe ASR with and without diacritics (lower is better). The *Data* column indicates the volume of pre-training data. Our models are pre-trained for 100k steps, except for *Fon-EFHY 400k*, which was trained for 400k steps. Parentheses indicate results achieved with augmentation during pre-training.

Pre-train Data	Data	Dev CER (w/Aug)	Test CER (w/Aug)	Dev WER (w/Aug)	Test WER (w/Aug)
with diacritics					
Fon	140h	8.00 (7.24)	11.10 (10.11)	19.61 (17.73)	24.28 (21.91)
Fon-EF	420h	6.98 (6.97)	10.02 (9.92)	17.49 (17.27)	21.83 (21.82)
Fon-HY	420h	7.15 (7.18)	10.12 (9.92)	17.54 (17.46)	22.07 (21.48)
Fon-EFHY	700h	7.10 (7.03)	10.02 (10.02)	17.57 (17.11)	22.06 (21.61)
Fon-EFHY 400k	700h	**6.90** (**6.62**)	**9.72** (**9.59**)	**16.74** (**16.01**)	**20.88** (**20.66**)
AfriHuBERT [Kponou et al. 2025]	100kh	-	-	-	21.98 (-)
without diacritics					
Fon	140h	6.15 (5.61)	9.13 (8.46)	14.23 (12.69)	18.68 (16.95)
Fon-EF	420h	5.36 (5.37)	8.10 (8.25)	12.28 (12.39)	16.41 (16.65)
Fon-HY	420h	5.48 (5.43)	8.29 (8.33)	12.80 (12.45)	16.75 (16.59)
Fon-EFHY	700h	5.29 (5.36)	8.01 (8.15)	12.19 (12.36)	16.21 (16.62)
Fon-EFHY 400k	700h	**5.21** (**5.24**)	**7.84** (**7.93**)	**11.61** (**11.66**)	**15.57** (**15.79**)
AfriHuBERT [Kponou et al. 2025]	100kh	-	-	-	17.02 (-)

Fon model pre-trained with augmentation, as the latter was the only configuration where augmentation significantly impacted performance. For the cascade system, adding augmentation during the pre-training of the *Fon* model improved performance from a BLEU score of 37.67 to 39.47, making it competitive with AfriHuBERT, which obtained a BLEU score of 39.60. Furthermore, when increasing the amount of pre-training data to 420 and 700 hours, we achieve performance superior to AfriHuBERT. These results are summarized in the last column of Table 7.3.

In our end-to-end systems, models fine-tuned directly from pre-training checkpoints (P.T.) perform slightly worse than the AfriHuBERT baseline, with the exception of the *Fon-EFHY* model pre-trained for 400k steps, which obtained a BLEU score of 26.82 compared to AfriHuBERT's 26.32 (see first column of Table 7.3). Starting from our fine-tuned ASR checkpoints leads to improvements across all models; in this configuration, every model surpasses AfriHuBERT except for the *Fon* model pre-trained without augmentation (see second column of Table 7.3). Lastly, when fine-tuning from both our ASR checkpoints and the fine-tuned NLLB model used in the cascade systems (*ASR+NLLB*), we observe further gains. In this setting, the lowest-performing model reaches a BLEU score of 29.11, while the highest-performing model achieves 31.82 (shown in the third column of Table 7.3).

Across all evaluated tasks, we observe no definitive trend favoring the addition of linguistically related tonal African languages (*Fon-HY*) over high-resource audiobook data (*Fon-EF*). While

Table 7.3: Table of BLEU scores for speech-to text-translation (higher is better). *P.T.*, *ASR*, and *ASR+NLLB* are models starting from pre-trained checkpoints, fine-tuned ASR checkpoints, and fine-tuned ASR and NLLB checkpoints, respectively. *Casc.* denotes the cascade approach. *Aug* denotes that augmentation was used in pre-training. Our models are pre-trained for 100k steps except *Fon-EFHY 400k*, which was pre-trained to 400k.

Starting point	P.T.	ASR	ASR+NLLB	Casc.
Pre-train Data				
Fon	22.39	25.14	29.11	37.67
Fon (Aug)	24.77	26.48	31.27	39.47
Fon-EF	25.69	26.99	31.33	40.07
Fon-HY	25.18	26.54	30.52	40.66
Fon-EFHY	25.94	26.42	29.76	39.61
Fon-EFHY 400k	**26.82**	**27.22**	**31.82**	**41.39**
AfriHuBERT [Kponou et al. 2025]	26.32	-	-	39.60

Hausa and Yoruba are more closely related to Fongbe than English and French, the *Fon-EF* model performed marginally better in most ASR and initialization settings. This suggests that for the BEST-RQ framework at this scale, linguistic proximity may be secondary to other data characteristics. Specifically, the *Fon-EF* dataset (mean duration 12.69s, median 13.97s) consists of longer audio segments than the *Fon-HY* dataset (mean 10.25s, median 8.71s). Given our findings in Chapter 5 regarding the positive impact of longer utterance lengths on representation learning, the performance of *Fon-EF* may be attributed to the high audio quality, phonetic and speaker diversity, and superior segment durations inherent in the LibriSpeech corpora.

7.5 Conclusion

Table 7.4: Pre-training efficiency comparison

Method	**Data (hrs)**	**Compute (hrs)**	**VRAM (GB)**
AfriHuBERT	∼90k	∼30k*	80
Ours	140–700	132–528†	24
% Reduct.	**100–600**×	**320–1000**×‡	**3.3**×

*Estimated A100 hours; †RTX 3090 hours; ‡Cost reduction in USD.

We perform a simple GPU cost estimate[3] using the following parameters. Our model takes approximately 5.5 days to train for 100k steps on a single RTX 3090 GPU. Assuming a rental cost of $0.20 per hour, a single model costs under $30 to train for 100k steps and $120 for 400k steps. While the exact pre-training time was not reported for AfriHuBERT, the model is initialized from mHuBERT-147 [Zanon Boito et al. 2024]. One iteration of mHuBERT-147 pre-training was reported to take 20 days on 32 $\times$ A100 GPUs (80 GB VRAM). Assuming two iterations—though up to three were reported in the original work—and an estimated price of $1.15 per A100 per hour, the total cost exceeds $35,000. This is approximately 1,100 times more expensive than our 100k-step models, which already achieve comparable performance. We consider these estimates conservative, as they do not account for the additional cost of the continual pre-training required to develop AfriHuBERT, nor the significant difference in storage costs between 90,000 and 700 hours of audio.

The methods presented in this chapter make pre-training speech models from scratch accessible to smaller companies and research groups with limited data or computational budgets. Furthermore, the ability to pre-train from scratch at a reasonable cost allows for greater control and transparency regarding the data used. Training on minimal amounts of data also facilitates the creation of more language-balanced datasets, whereas most massive-scale speech datasets remain heavily biased toward a few high-resource languages. By demonstrating that performance parity with state-of-the-art models is achievable with a fraction of the traditional resources, this chapter provides a viable roadmap for the inclusion of under-resourced languages in the self-supervised learning paradigm.

[3] Price estimates based on vast.ai at the time of writing.

Chapter 8

Conclusion

Speech technology has made massive strides over the past decade. It is not uncommon to see users interacting directly with smartphones, TV remotes, watches, and other AI agents, powered by speech models at the front end. To achieve high performance with a minimal amount of labeled data, these models are often trained within a self-supervised learning (SSL) framework. However, SSL is notoriously resource-intensive, requiring immense datasets, significant GPU hours, and extensive time to train a single model.

This thesis addresses the efficiency of self-supervised speech models from a broad perspective, exploring various angles to optimize performance. In Chapter 3, we open-source a highly efficient SSL framework for speech: BERT-based Speech pre-Training with Random-projection Quantizer (BEST-RQ). In Chapter 4, we investigate architectural modifications by replacing multi-head self-attention with various efficient alternatives. Chapters 5 and 6 focus on data and evaluation efficiency, respectively. Finally, Chapter 7 presents a case study on pre-training for Fongbe; we demonstrate that a state-of-the-art model can be developed for a low-resource language without the massive data and compute requirements typical of high-resource-centric models. This concluding chapter summarizes the work presented throughout the thesis, highlighting key contributions, findings, and avenues for future exploration.

Open-source Implementation of the BERT-based Speech pre-Training with Random-projection Quantizer

In Chapter 3, we describe our open-source implementation of BEST-RQ and present preliminary experiments comparing it to wav2vec 2.0. We find that our implementation maintains comparable performance to wav2vec 2.0 across multiple downstream tasks while pre-training 2.4 times faster. Providing an open-source implementation is vital for efficient knowledge dissemination. We believe that an accessible, efficient SSL framework lowers the barrier to entry, allowing others to learn from, experiment with, and ultimately improve upon this work.

While this is a crucial step, further progress is needed on the open-science front to make these models truly accessible. Beyond simply posting recipes and models on Hugging Face, establishing robust methods for interoperability between open-source environments is essential. For example, the ability to smoothly exchange models between SpeechBrain [Ravanelli et al. 2024], ESPnet [Watanabe et al. 2018], NeMo [Harper et al. n.d.], and fairseq2 [Balioglu et al. 2023], combined with a unified framework for benchmark evaluation, would significantly enhance collaboration between research groups and ensure consistency in reported results.

Increasing Efficiency with Attention Substitutes

In Chapter 4, we propose replacing multi-head self-attention (MHSA) within the BEST-RQ framework with alternatives that offer linear complexity—specifically HyperMixing, Fastformer, SummaryMixing, and Mamba. Our results demonstrate that these alternative methods, with the excep-

tion of Fastformer, maintain competitive performance relative to MHSA, even surpassing it on out-of-domain[1] downstream tasks. Furthermore, for input sequences ranging from 20 to 80 seconds, these linear alternatives decreased VRAM consumption by 20% to 60% and increased processing speed by 7% to 65%.

Regarding future work in this direction, we believe a granular analysis of various components within SSL frameworks—including our BEST-RQ implementation—to identify potential bottlenecks (e.g., data loading, pseudo-target generation, or the encoder) would be fruitful. Furthermore, the development of innovative architectures and frameworks remains a vital research frontier. For instance, recent advancements involve combining efficient methodologies, as seen in NEST [Huang et al. 2025], which employs the FastConformer [Rekesh et al. 2023] architecture within a BEST-RQ style framework, or Jamba [Lenz et al. 2025], which interleaves Transformer and Mamba layers. Finally, much like the trajectory of large language models where specialized kernels are developed to optimize Transformer performance, we anticipate that hardware-aware algorithms and algorithm-centric hardware developments will continue to play a pivotal role [Dao et al. 2022].

Data Selection Strategies for Efficient Pre-training

In Chapter 5, we explore various pre-training data selection methods. We found that selecting the longest audio files was more critical than curating a diverse or balanced pre-training dataset based on audio, linguistic, or speaker conditions. We hypothesize that when working at a large scale—such as our experiments with the 2,500-hour and 25,000-hour splits of the Loquacious dataset—large-scale datasets naturally contain substantial inherent diversity. Consequently, the speech model benefits more from having increased context per utterance. By selecting longer audio, we obtained superior results on ASR tasks using only 50% of the original dataset, with the additional benefit of accelerated training, as longer audio segments allow for more efficient batching.

As unlabeled data becomes increasingly prevalent, we believe that developing robust methods for curating high-quality pre-training datasets will be essential. Future research could involve validating these findings on other datasets across various languages. Furthermore, future work could explore a shift from diversity-based selection methods toward criterion-based or uncertainty-based selection methods.

Efficient Evaluation Strategies for Self-Supervised Speech Models

In Chapter 6, we propose using rank and clustering measures as proxies for evaluating the quality of self-supervised speech models during training. We find that our rank measure, the Global Effective Rank, and clustering measures—such as Within-Cluster Sum of Squares (WCSS)

[1] By out-of-domain, we refer to downstream tasks involving data distributions that differ from the pre-training data.

and the Davies-Bouldin (DB) index—correlate with downstream performance significantly better than the traditional pre-training loss metric.

We believe that developing efficient evaluation methods that do not require labeled data is an important research direction for self-supervised speech models. While our initial findings are promising, further research is required to validate these measures across diverse architectures and tasks. For example, rank-based measures may be less effective when comparing across different architectures, as the model's dimensionality inherently influences its rank capacity. Similarly, while clustering measures show great potential, more exploration is needed to fully understand their relationship to downstream performance and their utility as monitoring tools during training. Additionally, investigating alternative evaluation methods, such as LLM-based evaluation [Maekaku et al. 2025], could provide further insights into the representation quality of self-supervised speech models.

Efficient Self-Supervised Learning for Speech in Low-Resource Settings

In Chapter 7, we explore the application of BEST-RQ under severe resource constraints, using Fongbe, a language of Benin, as a case study. We restricted training to a single RTX 3090 GPU with 24 GB of VRAM and utilized only 140 to 700 hours for pre-training, intentionally keeping our model Fongbe-centric. Despite these constraints and the use of noisy radio data, we obtained state-of-the-art results for both automatic speech recognition and speech-to-text translation. These results demonstrate that SSL can be leveraged not only to exploit unlabeled data but also to develop high-quality, non-high-resource-language centric speech models for a low cost.

While this work involved collaboration with the Fongbe community, we recognize the importance of maintaining ongoing dialogue with the communities that stand to benefit from these developments to ensure their needs and wishes are respected. We find this direction particularly exciting, as it not only facilitates the efficient creation of models for specific communities but also empowers those communities to develop their own models at a lower cost. Ultimately, this technology can play a dual role: breaking down communication barriers for thousands of low-resource languages and supporting the preservation of languages facing extinction.

Closing Remarks

Future directions should not be limited solely to those mentioned above. We recognize a growing global effort to develop increasingly powerful Speech and Audio Large Language Models (SALLMs). The research presented in this thesis is highly complementary to the development of Speech and Audio Large Language Models, as many of these architectures rely on a foundation of pre-trained speech and language models. Consequently, as these models grow in complexity, the need for even greater compute and data efficiency becomes paramount. Furthermore, advance-

ments in quantization, streaming, and edge computing will be essential to enable the efficient, near real-time execution of these models on-device.

Speech technology allows us to communicate with machines more naturally than ever before. From personal assistants to industrial applications, the potential use cases are vast. However, despite rapid advancements, the immense amount of data and compute required to train and deploy these models creates a barrier, limiting who can participate in this research. As the integration of speech, vision, and text models continues to accelerate, we see no sign that the demand for AI will decrease. Therefore, we believe that efficiency will remain a pertinent and vital area of exploration for the foreseeable future.

We hope that the methodologies proposed here, alongside our experimental results and open-source contributions, will empower others to develop their own models and serve as a catalyst for the next generation of efficient speech technology.

APPENDIX

A. Description of tasks in the Multi-Probe Speech Self-Supervision benchmark

This appendix provides a comprehensive overview of the downstream tasks, datasets, and probe architectures used in this work, following the Multi-Probe Speech Self-Supervision (MP3S) benchmark [Zaiem et al. 2023]. A subset of these tasks is used for evaluation in Chapter 3, while the full set described below is used for evaluation in Chapter 4.

LibriSpeech-100 Automatic Speech Recognition (ASR)

ASR is the task of transcribing spoken utterances into text.

- **Datasets:** For this task we use the LibriSpeech [Panayotov et al. 2015] *train-clean-100* for training, *dev-clean* for validation, and *test-clean* and *test-other* for final testing. In our experiments this is considered in-domain, as models are pre-trained on LibriSpeech.
- **Probes:** For LibriSpeech, we employ the 2-layered BiLSTM and the ContextNet [Han et al. 2020] probes. CTC loss [Graves et al. 2006] is used for training ASR probes.
- **Metrics:** Performance is measured by the Word Error Rate (WER). We report WER both with and without the official 4-gram ARPA language model using beam search and shallow fusion.

CommonVoice Automatic Speech Recognition (ASR)

This ASR task evaluates the model's ability to generalize to out-of-domain data, as the pre-training is done on LibriSpeech, which consists of English speech, while the evaluation is performed on CommonVoice datasets in Welsh and Basque.

- **Datasets:** For this task, the MP3S benchmark utilizes the Welsh and Basque datasets from CommonVoice 11.0 [Ardila et al. 2020] to evaluate performance in an out-of-domain ASR setting.
- **Probes:** For this task the MP3S benchmark proposes Linear and LSTM probes. Similar to LibriSpeech, CTC loss [Graves et al. 2006] is used for training ASR probes.
- **Metrics:** Performance is measured by the Word Error Rate (WER) with no language model.

Automatic Speaker Verification (ASV)

ASV is a binary classification task where, given two utterances, the model must determine if the speakers are the same.

- **Dataset:** We use VoxCeleb1 [Nagrani et al. 2017], which consists of utterances from over 1,000 celebrities gathered from YouTube.
- **Probes:** We evaluate two architectures: the X-Vector probe [Snyder et al. 2018], which uses a Time Delay Neural Network (TDNN) with statistical pooling, and the ECAPA-TDNN [Desplanques et al. 2020], which enhances the TDNN with temporal and channel self-attention.
- **Metric:** Performance is measured using the Equal Error Rate (EER).

Intent Classification (IC)

IC is a classification task aimed at predicting the main purpose or scenario (e.g., *email*, *calendar*) of an utterance.

- **Dataset:** We utilize the SLURP dataset [Bastianelli et al. 2020], a challenging collection of 72k audio files (58 hours) covering 18 intents.
- **Probes:** We employ a Linear probe (average-pooling followed by a linear classifier) and an LSTM probe (a 2-layered BiLSTM followed by statistical pooling and a linear classification layer).
- **Metric:** Performance is measured by Accuracy.

Emotion Recognition (ER)

ER is the task of predicting the emotional state of a speaker (neutral, happy, sad, or angry).

- **Dataset:** We leverage the IEMOCAP dataset [Busso et al. 2008], which contains approximately 12 hours of data from 10 actors performing scripted and improvised scenes.
- **Probes:** Consistent with the other tasks, we use a Linear probe and the ECAPA-TDNN probe.
- **Metric:** Performance is measured via Accuracy using 10-fold cross-validation.

ACRONYMS

AFE	Acoustic Feature Extractor
ASV	Automatic Speaker Verification
ASR	Automatic Speech Recognition
BEST-RQ	BERT-based Speech pre-Training with Random-projection Quantizer
Bi-LSTM	Bidirectional Long Short-Term Memory
BLEU	Bilingual Evaluation Understudy
CER	Character Error Rate
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
E2E	End-to-End
EER	Equal Error Rate
EMA	Exponential Moving Average
ER	Emotion Recognition
GB	Gigabyte
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
HPC	High-Performance Computing
IC	Intent Classification
LSTM	Long Short-Term Memory
M	Million
MHSA	Multi-Head Self-Attention
NN	Neural Network
SSL	Self-Supervised Learning
SSM	State Space Model
STFT	Short-Time Fourier Transform
VRAM	Video Random Access Memory
WER	Word Error Rate

PERSONAL PUBLICATIONS

[Duret et al. 2025] Jarod Duret, Salima Mdhaffar, Gaëlle Laperrière, et al. “In-Domain SSL Pre-training and Streaming ASR: Application to Air Traffic Control Communications”. In: *International Conference on Speech and Computer*. Springer. 2025, pp. 3–12 (cit. on pp. 72, 77, 94, 100).

[Elleuch et al. 2026] Haroun Elleuch, Ryan Whetten, Salima Mdhaffar, Yannick Estève, and Fethi Bougares. “Ara-Best-RQ: Multi Dialectal Arabic SSL”. In: *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2026 (cit. on pp. 40, 72, 73, 94, 97).

[Feillet et al. 2026] Eva Feillet, Ryan Whetten, David Picard, and Alexandre Allauzen. “Polynomial Mixing for Efficient Self-supervised Speech Encoders”. In: *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2026 (cit. on p. 58).

[Whetten et al. 2025] Ryan Whetten, Lucas Maison, Titouan Parcollet, Marco Dinarelli, and Yannick Estève. “Towards Early Prediction of Self-Supervised Speech Model Performance”. In: *Interspeech 2025*. 2025, pp. 1228–1232. DOI: 10.21437/Interspeech.2025-34 (cit. on p. 82).

[Whetten et al. 2024] Ryan Whetten, Titouan Parcollet, Marco Dinarelli, and Yannick Estève. “Open Implementation and Study of Best-RQ for Speech Processing”. In: *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. 2024, pp. 460–464. DOI: 10.1109/ICASSPW62465.2024.10626364 (cit. on pp. 46, 65).

[Whetten et al. 2026] Ryan Whetten, Titouan Parcollet, Marco Dinarelli, and Yannick Estève. “A Study of Data Selection Strategies for Pre-training Self-Supervised Speech Models”. In: *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2026 (cit. on p. 72).

[Whetten et al. 2024] Ryan Whetten, Titouan Parcollet, Adel Moumen, Marco Dinarelli, and Yannick Estève. “An Analysis of Linear Complexity Attention Substitutes With Best-RQ”. In: *2024 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2024, pp. 169–176 (cit. on p. 58).

REFERENCES

[Alabi et al. 2025] Jesujoba O. Alabi, Xuechen Liu, Dietrich Klakow, and Junichi Yamagishi. “Afri-HuBERT: A self-supervised speech representation model for African languages”. In: *Interspeech 2025*. 2025, pp. 4023–4027. DOI: 10.21437/Interspeech.2025-1437 (cit. on pp. 18, 95, 97).

[Aldeneh et al. 2025] Zakaria Aldeneh, Vimal Thilak, Takuya Higuchi, Barry-John Theobald, and Tatiana Likhomanenko. “Towards automatic assessment of self-supervised speech models using rank”. In: *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2025, pp. 1–5 (cit. on pp. 42, 82, 84, 85, 90).

[Ardila et al. 2020] Rosana Ardila, Megan Branson, Kelly Davis, et al. “Common Voice: A Massively-Multilingual Speech Corpus”. English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Ed. by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, et al. Marseille, France: European Language Resources Association, May 2020, pp. 4218–4222. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.520 (cit. on pp. 99, 111).

[Arthur et al. 2007] David Arthur and Sergei Vassilvitskii. “k-means++: the advantages of careful seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245 (cit. on pp. 25, 85).

[Babu et al. 2022] Arun Babu, Changhan Wang, Andros Tjandra, et al. “XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale”. In: *Interspeech 2022*. 2022, pp. 2278–2282. DOI: 10.21437/Interspeech.2022-143 (cit. on p. 98).

[Baevski et al. 2023] Alexei Baevski, Arun Babu, Wei-Ning Hsu, and Michael Auli. “Efficient self-supervised learning with contextualized target representations for vision, speech and language”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1416–1429 (cit. on pp. 39, 72, 96).

[Baevski et al. 2022] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. “Data2vec: A general framework for self-supervised learning in speech, vision and

language". In: *International conference on machine learning*. PMLR. 2022, pp. 1298–1312 (cit. on pp. 28, 29, 34, 40, 46, 47, 49, 94, 97).

[Baevski et al. 2020] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 12449–12460. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf (cit. on pp. 18, 19, 26–29, 31, 37, 40, 42, 46, 50, 52, 64, 65, 68, 82, 86, 94, 96, 97, 100).

[Balioglu et al. 2023] Can Balioglu, Alexander Erben, Martin Gleize, Artyom Kozhevnikov, Ilia Kulikov, and Julien Yao. *fairseq2*. 2023. URL: http://github.com/facebookresearch/fairseq2 (cit. on p. 106).

[Bastianelli et al. 2020] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. "SLURP: A Spoken Language Understanding Resource Package". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 7252–7262. DOI: 10.18653/v1/2020.emnlp-main.588. URL: https://aclanthology.org/2020.emnlp-main.588 (cit. on p. 112).

[Beltagy et al. 2020] Iz Beltagy, Matthew E. Peters, and Arman Cohan. "Longformer: The Long-Document Transformer". In: *arXiv:2004.05150* (2020) (cit. on p. 60).

[Berrebbi et al. 2023] Dan Berrebbi, Ronan Collobert, Navdeep Jaitly, and Tatiana Likhomanenko. "More speaking or more speakers?" In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on p. 74).

[Bredin 2023] Hervé Bredin. "pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe". In: *Proc. INTERSPEECH 2023*. 2023, pp. 1983–1987. DOI: 10.21437/Interspeech.2023-105 (cit. on p. 75).

[Busso et al. 2008] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, et al. "IEMOCAP: Interactive emotional dyadic motion capture database". In: *Language resources and evaluation* 42 (2008), pp. 335–359 (cit. on p. 112).

[Caubrière et al. 2024] Antoine Caubrière and Elodie Gauthier. "Africa-Centric Self-Supervised Pretraining for Multilingual Speech Representation in a Sub-Saharan Context". In: *5th Workshop*

on African Natural Language Processing. 2024. URL: https://openreview.net/forum?id=zLOhcft2E7 (cit. on p. 97).

[Chan et al. 2016] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2016, pp. 4960–4964 (cit. on p. 24).

[Chaparala et al. 2024] Kaavya Chaparala, Guido Zarrella, Bruce Torres Fischer, Larry Kimura, and Oiwi Parker Jones. "Mai Ho'om\= auna i ka'Ai: Language Models Improve Automatic Speech Recognition in Hawaiian". In: *arXiv preprint arXiv:2404.03073* (2024) (cit. on p. 95).

[Chen et al. 2022] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, et al. "Wavlm: Large-scale self-supervised pre-training for full stack speech processing". In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (2022), pp. 1505–1518 (cit. on pp. 27, 29, 30, 34, 38, 40, 42, 46, 47, 82, 96, 100).

[Chen et al. 2023] William Chen, Xuankai Chang, Yifan Peng, Zhaoheng Ni, Soumi Maiti, and Shinji Watanabe. "Reducing Barriers to Self-Supervised Learning: HuBERT Pre-training with Academic Compute". In: *Interspeech 2023*. 2023, pp. 4404–4408. DOI: 10.21437/Interspeech.2023-1176 (cit. on pp. 39, 40, 42, 64, 82, 96).

[Chen et al. 2025] William Chen, Jinchuan Tian, Yifan Peng, Brian Yan, Chao-Han Huck Yang, and Shinji Watanabe. "OWLS: Scaling Laws for Multilingual Speech Recognition and Translation Models". In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. Vol. 267. Proceedings of Machine Learning Research. PMLR, 2025, pp. 9121–9145. URL: https://proceedings.mlr.press/v267/chen25bj.html (cit. on p. 40).

[Chen et al. 2024] William Chen, Wangyou Zhang, Yifan Peng, et al. "Towards Robust Speech Representation Learning for Thousands of Languages". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 10205–10224. DOI: 10.18653/v1/2024.emnlp-main.570. URL: https://aclanthology.org/2024.emnlp-main.570/ (cit. on pp. 18, 19, 72).

[Chi et al. 2022] Zewen Chi, Shaohan Huang, Li Dong, et al. "XLM-E: Cross-lingual Language Model Pre-training via ELECTRA". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav

Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 6170–6182. DOI: 10.18653/v1/2022.acl-long.427. URL: https://aclanthology.org/2022.acl-long.427/ (cit. on p. 34).

[Chiu et al. 2022] Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu. "Self-supervised learning with random-projection quantizer for speech recognition". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 3915–3924 (cit. on pp. 28, 36, 47, 49–52, 58, 70, 76, 83, 86, 96, 100).

[Chung et al. 2018] Yu-An Chung and James Glass. "Speech2Vec: A Sequence-to-Sequence Framework for Learning Word Embeddings from Speech". In: *Interspeech 2018*. 2018, pp. 811–815. DOI: 10.21437/Interspeech.2018-2341 (cit. on p. 27).

[Chung et al. 2020] Yu-An Chung and James Glass. "Generative pre-training for speech with autoregressive predictive coding". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 3497–3501 (cit. on p. 29).

[Chung et al. 2019] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. "An Unsupervised Autoregressive Model for Speech Representation Learning". In: *Interspeech 2019*. 2019, pp. 146–150. DOI: 10.21437/Interspeech.2019-1473 (cit. on p. 29).

[Chung et al. 2021] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. "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)*. IEEE. 2021, pp. 244–250 (cit. on pp. 28, 35).

[Dao et al. 2022] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. "Flashattention: Fast and memory-efficient exact attention with io-awareness". In: *Advances in neural information processing systems* 35 (2022), pp. 16344–16359 (cit. on p. 107).

[Davies et al. 1979] David L. Davies and Donald W. Bouldin. "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2 (1979), pp. 224–227. DOI: 10.1109/TPAMI.1979.4766909 (cit. on p. 85).

[Défossez et al. 2024] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. *Moshi: a speech-text foundation model for real-time dialogue*. Tech. rep. 2024. arXiv: 2410.00037 [eess.AS]. URL: https://arxiv.org/abs/2410.00037 (cit. on p. 18).

[Desplanques et al. 2020] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck. “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification”. In: *Interspeech 2020*. 2020, pp. 3830–3834. DOI: 10.21437/Interspeech.2020-2650 (cit. on pp. 87, 112).

[Devlin et al. 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423/ (cit. on pp. 26, 29).

[Dossou et al. 2021] Bonaventure F. P. Dossou and Chris Chinenye Emezue. “OkwuGbé: End-to-End Speech Recognition for Fon and Igbo”. In: *Proceedings of the Fifth Workshop on Widening Natural Language Processing*. Ed. by Erika Varis, Ryan Georgi, Alicia Tsai, et al. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1–4. URL: https://aclanthology.org/2021.winlp-1.1/ (cit. on p. 98).

[Garrido et al. 2023] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. “RankMe: Assessing the Downstream Performance of Pretrained Self-Supervised Representations by Their Rank”. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 10929–10974. URL: https://proceedings.mlr.press/v202/garrido23a.html (cit. on pp. 82, 84, 85).

[Gody et al. 2023] Reem Gody and David Harwath. “Unsupervised fine-tuning data selection for asr using self-supervised speech models”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on p. 73).

[Goncalves et al. 2022] Lucas Goncalves and Carlos Busso. “Improving Speech Emotion Recognition Using Self-Supervised Learning with Domain-Specific Audiovisual Tasks”. In: *Interspeech 2022*. 2022, pp. 1168–1172. DOI: 10.21437/Interspeech.2022-11012 (cit. on p. 18).

[Goodfellow et al. 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 18, 24).

[Graves et al. 2006] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, pp. 369–376 (cit. on pp. 24, 30, 53, 65, 101, 111).

[Gu et al. 2024] Albert Gu and Tri Dao. “Mamba: Linear-time sequence modeling with selective state spaces”. In: *First conference on language modeling*. 2024 (cit. on pp. 59, 60, 63).

[Gulati et al. 2020] Anmol Gulati, James Qin, Chung-Cheng Chiu, et al. “Conformer: Convolution-augmented Transformer for Speech Recognition”. In: *Interspeech 2020*. 2020, pp. 5036–5040. DOI: 10.21437/Interspeech.2020-3015 (cit. on pp. 28, 35, 50, 58, 63, 64).

[Ha et al. 2017] David Ha, Andrew M. Dai, and Quoc V. Le. “HyperNetworks”. In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rkpACe1lx (cit. on p. 60).

[Han et al. 2020] Wei Han, Zhengdong Zhang, Yu Zhang, et al. “ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context”. In: *Proc. Interspeech 2020*. 2020, pp. 3610–3614. DOI: 10.21437/Interspeech.2020-2059 (cit. on pp. 86, 111).

[Harper et al. n.d.] Eric Harper, Somshubra Majumdar, Oleksii Kuchaiev, et al. *NeMo: a toolkit for Conversational AI and Large Language Models*. URL: https://github.com/NVIDIA/NeMo (cit. on p. 106).

[Hinton et al. 2012] Geoffrey Hinton, Li Deng, Dong Yu, et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal processing magazine* 29.6 (2012), pp. 82–97 (cit. on pp. 24, 27).

[Hsu et al. 2021] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. “Hubert: Self-supervised speech representation learning by masked prediction of hidden units”. In: *IEEE/ACM transactions on audio, speech, and language processing* 29 (2021), pp. 3451–3460 (cit. on pp. 18, 19, 27–29, 32, 33, 38, 40, 42, 46, 47, 49, 50, 52, 68, 82, 86, 94, 96).

[Huang et al. 2025] He Huang, Taejin Park, Kunal Dhawan, et al. “NEST: Self-supervised fast conformer as all-purpose seasoning to speech processing tasks”. In: *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2025, pp. 1–5 (cit. on p. 107).

[Ioffe et al. 2015] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456 (cit. on p. 101).

[Jang et al. 2017] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical Reparametrization with Gumble-Softmax”. In: *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net. 2017 (cit. on p. 31).

[Jegou et al. 2010] Herve Jegou, Matthijs Douze, and Cordelia Schmid. “Product quantization for nearest neighbor search”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.1 (2010), pp. 117–128 (cit. on p. 31).

[Jolliffe et al. 2016] Ian T. Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016), p. 20150202. ISSN: 1364-503X. DOI: 10.1098/rsta.2015.0202. eprint: https://royalsocietypublishing.org/rsta/article-pdf/doi/10.1098/rsta.2015.0202/1381479/rsta.2015.0202.pdf. URL: https://doi.org/10.1098/rsta.2015.0202 (cit. on p. 25).

[Kahn et al. 2020] Jacob Kahn, Morgane Riviere, Weiyi Zheng, et al. “Libri-Light: A Benchmark for ASR with Limited or No Supervision”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7669–7673 (cit. on pp. 40, 96).

[Katharopoulos et al. 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. “Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 5156–5165. URL: https://proceedings.mlr.press/v119/katharopoulos20a.html (cit. on p. 60).

[Kponou et al. 2024] D. Fortuné Kponou, Fréjus A. A. Laleye, and Eugène Cokou Ezin. “FFSTC: Fongbe to French Speech Translation Corpus”. In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Ed. by Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue. Torino, Italia: ELRA and ICCL, May 2024, pp. 7270–7276. URL: https://aclanthology.org/2024.lrec-main.638/ (cit. on pp. 95, 98).

[Kponou et al. 2025] D. Fortuné Kponou, Salima Mdhaffar, Fréjus A. A. Laleye, Eugène C. Ezin, and Yannick Estève. “Extending the Fongbe to French Speech Translation Corpus: resources, models

and benchmark". In: *Interspeech 2025*. 2025, pp. 4533–4537. DOI: 10.21437/Interspeech.2025-1801 (cit. on pp. 95, 98, 99, 101–103).

[Kponou et al. 2026] Fortuné Kponou, Frejus Laleye, Eugène Ezin, and Yannick Estève. "Improving End-to-End Speech Translation for the Low Resource Language Fongbe to French". In: *ACM Transactions on Asian and Low-Resource Language Information Processing* (2026) (cit. on p. 98).

[LAleye et al. 2016] Fréjus AA LAleye, Laurent Besacier, Eugène C Ezin, and Cina Motamed. "First automatic fongbe continuous speech recognition system: Development of acoustic models and language models". In: *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2016, pp. 477–482 (cit. on pp. 98, 99).

[Le et al. 2026] Phuong-Hang Le, Valentin Pelloin, Arnault Chatelain, et al. "Pantagruel: Unified Self-Supervised Encoders for French Text and Speech". In: *arXiv preprint arXiv:2601.05911* (2026) (cit. on pp. 40, 73).

[Lee et al. 2024] Mun-Hak Lee, Jae-Hong Lee, DoHee Kim, Ye-Eun Ko, and Joon-Hyuk Chang. "Balanced-Wav2Vec: Enhancing Stability and Robustness of Representation Learning Through Sample Reweighting Techniques". In: *Interspeech 2024*. 2024, pp. 5058–5062. DOI: 10.21437/Interspeech.2024-1875 (cit. on p. 32).

[Lenz et al. 2025] Barak Lenz, Opher Lieber, Alan Arazi, et al. "Jamba: Hybrid transformer-mamba language models". In: *The thirteenth international conference on learning representations*. 2025 (cit. on p. 107).

[Li et al. 2023] Xilai Li, Goeric Huybrechts, Srikanth Ronanki, Jeff Farris, and Sravan Bodapati. "Dynamic chunk convolution for unified streaming and non-streaming conformer asr". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on pp. 76, 100).

[Li et al. 2025] Ze Li, Ming Cheng, and Ming Li. "Enhancing Speaker Verification with w2v-BERT 2.0 and Knowledge Distillation guided Structured Pruning". In: *arXiv preprint arXiv:2510.04213* (2025) (cit. on p. 18).

[Lin et al. 2023] Tzu-Quan Lin, Hung-yi Lee, and Hao Tang. "Melhubert: A simplified hubert on mel spectrograms". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2023, pp. 1–8 (cit. on pp. 39, 95).

[Liu et al. 2021] Andy T Liu, Shang-Wen Li, and Hung-yi Lee. “Tera: Self-supervised learning of transformer encoder representation for speech”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 2351–2366 (cit. on p. 95).

[Lloyd 1982] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 85).

[Lodagala et al. 2023] Vasista Sai Lodagala, Sreyan Ghosh, and Srinivasan Umesh. “data2vec-aqc: Search for the right teaching assistant in the teacher-student training setup”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on pp. 41, 68, 97).

[Lu et al. 2022] Zhiyun Lu, Yongqiang Wang, Yu Zhang, Wei Han, Zhehuai Chen, and Parisa Haghani. “Unsupervised Data Selection via Discrete Speech Representation for ASR”. In: *Interspeech 2022*. 2022, pp. 3393–3397. DOI: 10.21437/Interspeech.2022-399 (cit. on pp. 74, 75).

[Lucas 1977] George Lucas. *Star Wars: Episode IV - A New Hope*. Motion Picture. United States, 1977 (cit. on p. 18).

[Lugo et al. 2024] Luis Lugo and Valentin Vielzeuf. “Towards efficient self-supervised representation learning in speech processing”. In: *Findings of the Association for Computational Linguistics: EACL 2024*. Ed. by Yvette Graham and Matthew Purver. St. Julian’s, Malta: Association for Computational Linguistics, Mar. 2024, pp. 340–346. DOI: 10.18653/v1/2024.findings-eacl.23. URL: https://aclanthology.org/2024.findings-eacl.23/ (cit. on p. 95).

[Maas et al. 2013] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. Atlanta, GA. 2013, p. 3 (cit. on p. 101).

[Maekaku et al. 2025] Takashi Maekaku, Keita Goto, Jinchuan Tian, Yusuke Shinohara, and Shinji Watanabe. “Evaluating Self-Supervised Speech Models Via Text-Based LLMs”. In: *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2025, pp. 1–8. DOI: 10.1109/ASRU65441.2025.11434740 (cit. on p. 108).

[Mai et al. 2023] Florian Mai, Arnaud Pannatier, Fabio Fehr, Haolin Chen, Francois Marelli, François Fleuret, and James Henderson. “HyperMixer: An MLP-based Low Cost Alternative to Transformers”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2023, pp. 15632–15654 (cit. on pp. 60, 62).

[Mai et al. 2023] Florian Mai, Juan Zuluaga-Gomez, Titouan Parcollet, and Petr Motlicek. “HyperConformer: Multi-head HyperMixer for Efficient Speech Recognition”. In: *Proc. INTERSPEECH 2023*. 2023, pp. 2213–2217. DOI: 10.21437/Interspeech.2023-1611 (cit. on pp. 59–62).

[Malhotra et al. 2019] Karan Malhotra, Shubham Bansal, and Sriram Ganapathy. “Active Learning Methods for Low Resource End-to-End Speech Recognition”. In: *Interspeech 2019*. 2019, pp. 2215–2219. DOI: 10.21437/Interspeech.2019-2316 (cit. on p. 73).

[Mdhaffar et al. 2025] Salima Mdhaffar, Haroun Elleuch, Chaimae Chellaf, Ha Nguyen, and Yannick Estève. “SENSE models: an open source solution for multilingual and multimodal semantic-based tasks”. In: *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2025, pp. 1–8. DOI: 10.1109/ASRU65441.2025.11433845 (cit. on p. 75).

[Meng et al. 2022] Yen Meng, Yi-Hui Chou, Andy T Liu, and Hung-yi Lee. “Don’t speak too fast: The impact of data bias on self-supervised speech models”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 3258–3262 (cit. on pp. 73, 74).

[Meyer et al. 2022] Josh Meyer, David Adelani, Edresson Casanova, et al. “BibleTTS: a large, high-fidelity, multilingual, and uniquely African speech corpus”. In: *Interspeech 2022*. 2022, pp. 2383–2387. DOI: 10.21437/Interspeech.2022-10850 (cit. on p. 99).

[Mohamed et al. 2022] Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, et al. “Self-supervised speech representation learning: A review”. In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (2022), pp. 1179–1210 (cit. on pp. 18, 26, 27, 46).

[Mousavi et al. 2024] Pooneh Mousavi, Jarod Duret, Salah Zaiem, Luca Della Libera, Artem Ploujnikov, Cem Subakan, and Mirco Ravanelli. “How Should We Extract Discrete Audio Tokens from Self-Supervised Models?” In: *Interspeech 2024*. 2024, pp. 2554–2558. DOI: 10.21437/Interspeech.2024-2135 (cit. on p. 85).

[Nagrani et al. 2017] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. “VoxCeleb: A Large-Scale Speaker Identification Dataset”. In: *Interspeech 2017*. 2017, pp. 2616–2620. DOI: 10.21437/Interspeech.2017-950 (cit. on pp. 87, 112).

[Omnilingual ASR Team et al. 2025] Omnilingual ASR Team, Gil Keren, Artyom Kozhevnikov, et al. *Omnilingual ASR: Open-Source Multilingual Speech Recognition for 1600+ Languages*. 2025. URL: https://ai.meta.com/research/publications/omnilingual-asr-open-source-multilingual-speech-recognition-for-1600-languages/ (cit. on p. 47).

[Oord et al. 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018) (cit. on pp. 25, 28, 31).

[Panayotov et al. 2015] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. “Librispeech: an asr corpus based on public domain audio books”. In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5206–5210 (cit. on pp. 26, 40, 51, 64, 86, 96, 97, 99, 111).

[Papineni et al. 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. “Bleu: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318 (cit. on p. 101).

[Parcollet et al. 2024] Titouan Parcollet, Ha Nguyen, Solène Evain, et al. “Lebenchmark 2.0: A standardized, replicable and enhanced framework for self-supervised representations of french speech”. In: *Computer Speech & Language* 86 (2024), p. 101622 (cit. on pp. 19, 38, 40, 46, 47, 73).

[Parcollet et al. 2025] Titouan Parcollet, Yuan Tseng, Shucong Zhang, and Rogier C. van Dalen. “Loquacious Set: 25,000 Hours of Transcribed and Diverse English Speech Recognition Data for Research and Commercial Use”. In: *Interspeech 2025*. 2025, pp. 4053–4057. DOI: 10.21437/Interspeech.2025-720 (cit. on pp. 75, 76, 78, 79).

[Parcollet et al. 2024] Titouan Parcollet, Rogier van Dalen, Shucong Zhang, and Sourav Bhattacharya. “SummaryMixing: A Linear-Complexity Alternative to Self-Attention for Speech Recognition and Understanding”. In: *Interspeech 2024*. 2024, pp. 3460–3464. DOI: 10.21437/Interspeech.2024-40 (cit. on pp. 59–62).

[Parcollet et al. 2023] Titouan Parcollet, Shucong Zhang, Rogier van Dalen, Alberto Gil CP Ramos, and Sourav Bhattacharya. “On the (In) Efficiency of Acoustic Feature Extractors for Self-Supervised Speech Representation Learning”. In: *Interspeech 2023*. 2023 (cit. on pp. 46, 58, 65, 86).

[Pasad et al. 2021] Ankita Pasad, Ju-Chieh Chou, and Karen Livescu. “Layer-wise analysis of a self-supervised speech representation model”. In: *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2021, pp. 914–921 (cit. on pp. 30, 90).

[Pedregosa et al. 2011] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 85).

[Peng et al. 2022] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. “Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 17627–17643 (cit. on p. 61).

[Peng et al. 2023] Yifan Peng, Jinchuan Tian, Brian Yan, et al. “Reproducing whisper-style training using an open-source toolkit and publicly available data”. In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2023, pp. 1–8 (cit. on p. 72).

[Pratap et al. 2020] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. “MLS: A Large-Scale Multilingual Dataset for Speech Research”. In: *Interspeech 2020*. 2020, pp. 2757–2761. DOI: 10.21437/Interspeech.2020-2826 (cit. on p. 99).

[Rabiner 1989] Lawrence R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286. DOI: 10.1109/5.18626 (cit. on p. 24).

[Radford et al. 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9 (cit. on p. 29).

[Rao et al. 2017] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. “Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer”. In: *2017 IEEE automatic speech recognition and understanding workshop (ASRU)*. IEEE. 2017, pp. 193–199 (cit. on p. 24).

[Ravanelli et al. 2024] Mirco Ravanelli, Titouan Parcollet, Adel Moumen, et al. “Open-source conversational ai with speechbrain 1.0”. In: *Journal of Machine Learning Research* 25.333 (2024), pp. 1–11 (cit. on pp. 47, 50, 52, 55, 59, 63, 98, 106).

[Rekesh et al. 2023] Dima Rekesh, Nithin Rao Koluguri, Samuel Kriman, et al. “Fast conformer with linearly scalable attention for efficient speech recognition”. In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2023, pp. 1–8 (cit. on pp. 72, 107).

[Roy et al. 2007] Olivier Roy and Martin Vetterli. “The effective rank: A measure of effective dimensionality”. In: *2007 15th European Signal Processing Conference*. 2007, pp. 606–610 (cit. on p. 84).

[Schneider et al. 2019] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. "wav2vec: Unsupervised Pre-Training for Speech Recognition". In: *Interspeech 2019*. 2019, pp. 3465–3469. DOI: 10.21437/Interspeech.2019-1873 (cit. on pp. 25, 27, 28, 31).

[Sculley 2010] D. Sculley. "Web-scale k-means clustering". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178. ISBN: 9781605587998. DOI: 10.1145/1772690.1772862. URL: https://doi.org/10.1145/1772690.1772862 (cit. on p. 85).

[Silverman 2022] Kenneth Silverman. "The Androids of Hephaestus: Between Human and Machine in the Iliad". In: *Selected Proceedings of the Classics Graduate Student Symposia at the University of Florida*. Vol. 1. 2022 (cit. on p. 18).

[Snyder et al. 2018] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur. "X-vectors: Robust dnn embeddings for speaker recognition". In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2018, pp. 5329–5333 (cit. on p. 112).

[Sriram et al. 2022] Anuroop Sriram, Michael Auli, and Alexei Baevski. "Wav2Vec-Aug: Improved self-supervised training with limited data". In: *Interspeech 2022*. 2022, 4950–4954. DOI: {10.21437/Interspeech.2022-667} (cit. on pp. 41, 97, 100).

[Srivastava et al. 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958 (cit. on p. 101).

[Su et al. 2024] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. "Roformer: Enhanced transformer with rotary position embedding". In: *Neurocomputing* 568 (2024), p. 127063 (cit. on p. 76).

[Tolstikhin et al. 2021] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272 (cit. on pp. 60, 62).

[Valk et al. 2021] Jörgen Valk and Tanel Alumäe. "VoxLingua107: a dataset for spoken language recognition". In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2021, pp. 652–658 (cit. on p. 99).

[Vaswani et al. 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 26, 28, 58, 59).

[Wang et al. 2023] Hongji Wang, Chengdong Liang, Shuai Wang, Zhengyang Chen, Binbin Zhang, Xu Xiang, Yanlei Deng, and Yanmin Qian. "Wespeaker: A research and production oriented speaker embedding learning toolkit". In: *ICASSP 2023, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5 (cit. on p. 75).

[Wang et al. 2020] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. "Linformer: Self-attention with linear complexity". In: *arXiv preprint arXiv:2006.04768* (2020) (cit. on p. 60).

[Watanabe et al. 2018] Shinji Watanabe, Takaaki Hori, Shigeki Karita, et al. "ESPnet: End-to-End Speech Processing Toolkit". In: *Interspeech 2018*. 2018, pp. 2207–2211. DOI: 10.21437/Interspeech.2018-1456 (cit. on p. 106).

[Wu et al. 2021] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. "Fastformer: Additive attention can be all you need". In: *arXiv preprint arXiv:2108.09084* (2021) (cit. on pp. 59–61).

[Yang et al. 2023] Guanrou Yang, Ziyang Ma, Zhisheng Zheng, Yakun Song, Zhikang Niu, and Xie Chen. "Fast-HuBERT: an efficient training framework for self-supervised speech representation learning". In: *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2023, pp. 1–7 (cit. on pp. 39, 96).

[Yang et al. 2021] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, et al. "SUPERB: Speech Processing Universal PERformance Benchmark". In: *Interspeech 2021*. 2021, pp. 1194–1198. DOI: 10.21437/Interspeech.2021-1775 (cit. on pp. 31, 46).

[Zaheer et al. 2020] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, et al. "Big bird: Transformers for longer sequences". In: *Advances in neural information processing systems* 33 (2020), pp. 17283–17297 (cit. on p. 60).

[Zaiem et al. 2023] Salah Zaiem, Youcef Kemiche, Titouan Parcollet, Slim Essid, and Mirco Ravanelli. "Speech Self-Supervised Representation Benchmarking: Are We Doing it Right?" In: *Interspeech 2023*. 2023, pp. 2873–2877. DOI: 10.21437/Interspeech.2023-1087 (cit. on pp. 31, 47, 51, 52, 59, 65, 86, 111).

[Zanon Boito et al. 2022] Marcely Zanon Boito, Laurent Besacier, Natalia Tomashenko, and Yannick Estève. “A Study of Gender Impact in Self-supervised Models for Speech-to-Text Systems”. In: *Interspeech 2022*. 2022, pp. 1278–1282. DOI: 10.21437/Interspeech.2022-353 (cit. on pp. 73, 74).

[Zanon Boito et al. 2024] Marcely Zanon Boito, Vivek Iyer, Nikolaos Lagos, Laurent Besacier, and Ioan Calapodescu. “mHuBERT-147: A Compact Multilingual HuBERT Model”. In: *Interspeech 2024*. 2024, pp. 3939–3943. DOI: 10.21437/Interspeech.2024-938 (cit. on pp. 18, 38, 42, 82, 96, 97, 104).

[Zhang et al. 2020] Binbin Zhang, Di Wu, Zhuoyuan Yao, et al. “Unified streaming and non-streaming two-pass end-to-end model for speech recognition”. In: *arXiv preprint arXiv:2012.05481* (2020) (cit. on pp. 76, 100).

[Zhang et al. 2020] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. “Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7829–7833 (cit. on p. 24).

[Zhang et al. 2025] Shucong Zhang, Titouan Parcollet, Rogier Van Dalen, and Sourav Bhattacharya. “Benchmarking Rotary Position Embeddings for Automatic Speech Recognition”. In: *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2025, pp. 1–7. DOI: 10.1109/ASRU65441.2025.11434788 (cit. on p. 100).

[Zhang et al. 2025] Xiangyu Zhang, Qiquan Zhang, Hexin Liu, et al. “Mamba in speech: Towards an alternative to self-attention”. In: *IEEE Transactions on Audio, Speech and Language Processing* (2025) (cit. on pp. 60, 61, 63).

[Zhang et al. 2023] Yu Zhang, Wei Han, James Qin, et al. “Google usm: Scaling automatic speech recognition beyond 100 languages”. In: *arXiv preprint arXiv:2303.01037* (2023) (cit. on pp. 18, 19, 47, 52, 72).

[Zhang et al. 2020] Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. “Pushing the limits of semi-supervised learning for automatic speech recognition”. In: *arXiv preprint arXiv:2010.10504* (2020) (cit. on p. 48).