Thèse n° 9766

Efficient Transformer-Based Speech Recognition

Présentée le 13 octobre 2022

Faculté des sciences et techniques de l’ingénieur
Laboratoire de l’IDIAP
Programme doctoral en génie électrique

pour l’obtention du grade de Docteur ès Sciences

par

Apoorv VYAS

Acceptée sur proposition du jury

Prof. A. M. Alahi, président du jury
Prof. H. Bourlard, Prof. F. Fleuret, directeurs de thèse
Prof. F. Metze, rapporteur
Prof. R. Schlüter, rapporteur
Prof. M. Jaggi, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

Acknowledgements

I would like to begin by expressing my gratitude to a number of people whose support has made this thesis possible and the past four years enjoyable.

Firstly, I would like to thank my supervisors Prof. Hervé Bourlard and Prof. François Fleuret for their support, insightful discussions, and guidance on how to conduct sustainable research. They motivated me to explore research ideas and to be persistent in the face of failure. It is due to this academic freedom and encouragement that I now feel confident to pursue research on a diverse set of topics in machine learning. I want to extend my gratitude to the secretariat and the system administration group for providing a professional workspace and excellent technical resources to ease the burden of research.

I thank the Swiss National Science Foundation (SNSF) for funding my work through the grant 200021-175589 for the SHISSM project. I sincerely thank my thesis committee members Prof. Florian Metze, Dr. Ralf Schlüter, Prof. Martin Jaggi, and Prof. Alexander Alahi for taking the time to read my thesis and providing their insightful comments and questions.

Moving to new country with different culture and language can be as scary as it is exciting. I consider myself fortunate to have made wonderful friends with whom I shared countless coffees, chai, board games, hikes, sports, movies, dinners, etc. I am incredibly grateful to everyone at Idiap for making my PhD a memorable experience. To Enno, Verena, Julian, François, Sibo, Banri, Bastian, Sophie, Tom, Vedrana, Pranay, DJ, Angel, Suraj, Teja, Suhan, Despoina, Angelos, Julian, Parvaneh, Weipeng, Ina, and many more whose names are not here but have enriched my stay at Idiap.

I would also like to thank my whole family, especially my parents for their support, blessings, patience, and sacrifice which allowed me to pursue my interests all my life. Finally, and most importantly, I thank my wife, Sargam, for taking on this journey and sharing the highs and lows of PhD with me. She agreed with a smile as I made the most selfish request of pursuing PhD and asking her to give up her job and move to a new country. I could always count on her as I toiled through the PhD or the job interviews. This thesis would certainly not be possible without her constant support over the last four years.

September 16, 2022 Apoorv

Abstract

Training deep neural network based Automatic Speech Recognition (ASR) models often requires thousands of hours of transcribed data, limiting their use to only a few languages. Moreover, current state-of-the-art acoustic models are based on the Transformer architecture that scales quadratically with sequence lengths, hindering its use for long sequences. This thesis aims to reduce (a) the data and (b) the compute requirements for developing state-of-the-art ASR systems with only a few hundred hours of transcribed data or less.

The first part of this thesis focuses on reducing the amount of transcribed data required to train these models. We propose an approach that uses dropout for uncertainty-aware semi-supervised learning. We show that our approach generates better hypotheses for training with unlabelled data. We then investigate the out-of-domain and cross-lingual generalization for two popular self-supervised pre-training approaches: Masked Acoustic Modeling and wav2vec 2.0. We conclude that both pre-training approaches generalize to unseen domains and significantly outperform the models trained only with supervised data.

In the second part, we focus on reducing the computational requirements for the Transformer model, (a) by devising efficient forms of attention computation and (b) by reducing the input context length for attention computation. We first present 'linear' attention that uses a kernelized formulation for attention to express an autoregressive transformer as a recurrent neural network and reduce the computational complexity from quadratic to linear in sequence length. We then present 'clustered' attention which approximates self-attention by clustering input sequence and using centroids for computation. We show that the clustered attention outperforms the vanilla attention for a given computational budget.

For ASR, we find that linear attention results in word error rate degradation, and clustering introduces overheads when working with shorter sequences. To address this limitation, we develop a method that stochastically downsamples input using mean-pooling for efficient wav2vec 2.0 training. This enables using the same model at different compression factors during inference. We conclude that stochastic compression for wav2vec 2.0 pre-training enables building compute-efficient ASR models for languages with limited transcribed data.

Keywords: speech recognition, semi-supervised learning, self-supervision, efficient attention, wav2vec 2.0, transformers, end-to-end

Résumé

L'entraînement de modèles de reconnaissance vocale basés sur des réseaux de neurones profonds nécessite souvent des milliers d'heures de données retranscrites, ce qui limite leur utilisation à quelques langues seulement. De plus, les modèles acoustiques de l'état de l'art actuel sont basés sur l'architecture *Transformer* qui croît de manière quadratique avec la longueur des séquences, limitant leur utilisation pour de longues séquences. Cette thèse vise à réduire les besoins en (a) données et (b) calculs pour développer des systèmes de reconnaissance vocale de pointe avec seulement quelques centaines d'heures ou moins de données retranscrites.

La première partie de cette thèse se concentre sur la réduction de la quantité de données retranscrites nécessaires à l'entraînement de ces modèles. Nous proposons une approche qui utilise le *dropout* pour un apprentissage semi-supervisé qui tient compte de l'incertitude du modèle. Nous montrons que notre approche génère de meilleures hypothèses pour l'entraînement sur des données non étiquetées. Nous étudions ensuite la généralisation extra-domaine et interlinguistique de deux approches populaires de pré-entraînement auto-supervisé : *Masked Acoustic Modeling* et *wav2vec 2.0*. Nous concluons que ces deux approches généralisent à des domaines inconnus et surpassent significativement les modèles entraînés uniquement sur des données supervisées.

Dans la seconde partie, nous nous intéressons à la réduction des besoins en calcul du modèle *Transformer*, à travers (a) la conception de formes efficaces de calcul de l'attention et (b) la réduction de la longueur du contexte d'entrée pour le calcul de l'attention. Nous présentons d'abord la *linear attention* qui utilise une formulation à noyau de l'attention afin d'exprimer un *transformer* auto-régressif sous la forme d'un réseau de neurones récurrent, permettant de réduire la complexité algorithmique de quadratique à linéaire par rapport à la longueur de séquence. Nous présentons ensuite la *clustered attention* qui approxime l'auto-attention en regroupant les séquences d'entrée et en utilisant les centroïdes pour le calcul. Nous montrons que la *clustered attention* surpasse l'attention classique pour un budget de calcul donné.

Pour la reconnaissance de la parole, nous constatons que la *linear attention* entraîne une dégradation du taux d'erreur de mots, et que la *clustered attention* introduit un surcoût computationnel sur les séquences plus courtes. Pour remédier à cette limitation, nous développons une méthode qui sous-échantillonne les entrées aléatoirement par moyennage pour entraîner efficacement *wav2vec 2.0*. Cela permet d'utiliser le même modèle avec différents facteurs de compression pendant l'inférence. Nous concluons que la compression stochastique pour le pré-entraînement de *wav2vec 2.0* permet d'implémenter des modèles de reconnaissance vocale efficaces en termes de calcul pour des langues dont les données retranscrites sont limitées.

Mots-clés : reconnaissance vocale, apprentissage semi-supervisé, auto-supervision, attention efficiente, wav2vec 2.0, transformers, end-to-end

Contents

Acknowledgements **i**

Abstract (English/Français) **iii**

1 Introduction **3**
- 1.1 Motivations 3
- 1.2 Summary of contributions 5
- 1.3 Thesis outline 6

2 Background on Automatic Speech Recognition **9**
- 2.1 Key Components in ASR 9
- 2.2 Hidden Markov Models 10
  - 2.2.1 Discrete Markov Chain 10
  - 2.2.2 Hidden Markov Model 11
- 2.3 HMM based ASR System 12
- 2.4 Sequence Discriminative Training 13
  - 2.4.1 Maximum Mutual Information 14
  - 2.4.2 Lattice-Free Maximum Mutual Information 15
  - 2.4.3 Connectionist Temporal Classification 15
- 2.5 Deep Neural Networks 17
  - 2.5.1 Convolutional Neural Networks 17
  - 2.5.2 Recurrent Neural Networks 18
  - 2.5.3 Transformers 18
- 2.6 ASR Evaluation Metric 22
- 2.7 Databases 22
  - 2.7.1 Fisher Corpus 23
  - 2.7.2 Librispeech Corpus 23
  - 2.7.3 Switchboard Corpus 23
  - 2.7.4 Babel Corpus 24
  - 2.7.5 Wall Street Journal Corpus 24

3 Uncertainty Aware Semi-supervised Learning with Dropout **27**
- 3.1 Introduction 27
- 3.2 ASR Uncertainty estimation with Dropout 28

3.2.1 Word Error Rate Estimation 30
3.2.2 Error Localization Using Word-Level Confidence 31
3.3 Error localization and WER Estimation Experiments 31
3.4 Error localization and WER Estimation Results 32
3.4.1 Word Error Rate Estimation 33
3.4.2 Error Localization Using Word Confidences 34
3.5 Uncertainty Aware Semi-supervised Learning 35
3.6 Related Work Discussion 39
3.7 Semi-supervised Learning Experiments 40
3.8 Semi-supervised Learning Results 41
3.8.1 Effect of Dropout Sample Numbers from Acoustic Model 41
3.8.2 Effect of Number of Dropout Samples from Language Model 42
3.8.3 Complete Comparison 43
3.9 Conclusions 44

4 Robust Self-Supervised Pre-training 47
4.1 Introduction 47
4.1.1 Masked Acoustic Modeling 48
4.1.2 wav2vec 2.0 50
4.2 Acoustic Model Adaptation 52
4.2.1 Model Training 52
4.2.2 Decoding 53
4.3 Experimental Setup 53
4.3.1 Datasets 53
4.3.2 Baselines 54
4.4 Results 54
4.4.1 Librispeech (100h) 54
4.4.2 Switchboard (300h) 56
4.4.3 Babel: Swahili (38h) and Tagalog (84h) 57
4.5 Conclusions 58

5 Efficient Attention Models 61
5.1 Introduction 61
5.2 Related Work 62
5.2.1 Attention Improvements Before Transformers 62
5.2.2 Non-asymptotic Improvements 63
5.2.3 Improvements in Asymptotic Complexity 63
5.3 Transformers 65
5.4 Linear Transformers 65
5.4.1 Linearized Attention 66
5.4.2 Causal Masking 68
5.4.3 Transformers are RNNs 70
5.5 Clustered Transformers 70

5.5.1 Clustered Attention 71
5.5.2 Improving clustered attention 73
5.6 Automatic Speech Recognition Experiments 74
5.6.1 Evaluation on Wall Street Journal (WSJ) 75
5.6.2 Evaluation on Switchboard 77
5.7 Experiments on Other Modalities 78
5.7.1 Synthetic Tasks 79
5.7.2 Image Generation 80
5.7.3 Pre-Trained Model Approximation 83
5.8 Conclusions 86

6 Towards Data and Compute Efficient ASR 87
6.1 Introduction 87
6.2 Related Works 88
6.3 Stochastic wav2vec 2.0 90
6.3.1 Basic Notations 90
6.3.2 Query and Key-Value Mean Pooled Attention 90
6.3.3 Stochastic Compression 91
6.4 Experimental Setup 91
6.4.1 Models 91
6.4.2 Pre-training 91
6.4.3 Fine-tuning 92
6.4.4 Evaluation 92
6.5 Results 93
6.5.1 How much query and key-value pooling? 93
6.5.2 On-demand compute reduction inference 94
6.5.3 Test Set Evaluation 95
6.6 Conclusions 97

7 Conclusions and Future Work 99
7.1 Conclusions 99
7.2 Future Work 100

A Appendix for Chapter 4 103
A.1 Pre-trained Models as Feature Extractors 103
A.1.1 Librispeech (100h) 103
A.1.2 Switchboard (300h) 103
A.1.3 Babel: Swahili (38h) and Tagalog (84h) 105

B Appendix for Chapter 5 107
B.1 Linear Transformers 107
B.1.1 Forward and Backward Pass 107

B.1.2 Gradient Derivation 107
B.2 Clustered Transformers 109
B.2.1 Improved clustered attention 109
B.2.2 Quality of the approximation 111
B.3 Experiments 113
B.3.1 Synthetic Tasks 113
B.3.2 Autoregressive Image Generation: Training Evolution 115
B.3.3 Autoregressive Image Generation: Throughput Discussion 115
B.3.4 Autoregressive Image Generation: Qualitative Results 117
B.3.5 Automatic Speech Recognition 122
B.3.6 RoBERTa Approximation 124

Bibliography **127**

Curriculum Vitae **145**

List of Tables

2.1 Statistics of the LibriSpeech database. 24

2.2 Statistics of the WSJ dataset. 25

3.1 Hyperparameter values tuned on the *dev* set of the Switchboard dataset. 32

3.2 Results on estimating WER on the Switchboard *test* set using dropout uncertainty. *Dr* refers to dropout based estimation, *Nb* refers to N-best list based estimation, and *Gt* refers to the Ground truth WER. While the N-best list reasonably estimated the WER over all the sentences (last column), it significantly overestimates the WER on shorter sentence and underestimates on longer sentences. In contrast, dropout-based WER estimation consistently performs well over all sentence lengths. 33

3.3 Error localization performance on the *dev* and *test* parts of the Switchboard dataset. We use the IoU metric as it penalizes both false errors and missed detections. Dropout achieves an IoU of ~ 0.6 indicating that a significant number of errors are correctly identified. 35

3.4 We compare the averaged WER(%) and SER (%) for the combined lattice and regular decoding lattice on the Fisher-English *dev* set. We note that the dropout-based lattice combination achieves much lower sentence error rate (SER). This reduces the effect of incorrect hypotheses when the model is confident and improves the performance of semi-supervised training. 42

3.5 Comparison between combined lattice and regular decoding lattice in WER(%) obtained on the *dev* and *test* sets of the Fisher-English dataset. The 50h supervised system is used as baseline to calculate WRR. 43

4.1 Comparing the number of parameters for different choices of acoustic models. 53

4.2 Comparing the performance of MAM pre-trained model fine-tuning with E2E-LFMMI against cross entropy based hybrid model. All models are fine-tuned on the 100 hour subset of the Librispeech dataset. We report the results on the clean and other portions of the test set. Fine-tuning the pre-trained model with LFMMI (E2E-LFMMI) significantly outperforms the cross-entropy based hybrid model training. Note that the base (MAM-B) model in (Liu et al., 2021) are pre-trained with fMLLR features. For a fair comparison, we also report the result for adaptation on MAM-S models pre-trained with FBANK features. For MAM-S, the pre-trained models are used as feature extractors to compare against the results reported for hybrid model. TDNNF-L and TDNNF-S refer to the 12 and 7 layered TDNNF architectures. liGRU refers to light Gated Recurrent Unit. E2E-LFMMI refers to flat-start training with LFMMI and hybrid refers to cross-entropy based HMM-DNN models. 55

4.3 Comparison of word error rates (WER) (in %) on the clean and other parts of the Librispeech test set with and without 4-gram language model rescoring. Fine-tuning the pre-trained models significantly outperforms the baseline trained using only supervised data. MAM-S, MAM-B refers to small, and base transformer models used for pre-training. TDNNF-S and TDNNF-L refer to the 7 and 12 layered TDNNF architectures used for fine-tuning. e2e-lfmmi refers to flat-start training with LFMMI and ctc refers to Connectionist Temporal Classification criterion. 56

4.4 Comparison of word error rates (WER) (in %) on eval2000 test set portion of the Switchboard dataset. The 3-gram language model is based on Switchboard, whereas the 4-gram employs Switchboard+Fisher training set transcripts. Fine-tuning either of the pre-trained W2V2 or MAM model outperforms the baseline. Fine-tuning W2V2 model with E2E-LFMMI or CTC significantly outperforms MAM model. 57

4.5 Comparison of word error rates (WER) (in %) on dev10h set for the Swahili and Tagalog languages of the Babel dataset. Fine-tuning the pre-trained wav2vec 2.0 BASE model significantly outperforms the monolingual and MAM baselines. Note that while we use SRILM, XLSR-10 model uses KenLM for decoding and does not use speed or volume perturbation. 58

5.1 Performance comparison in automatic speech recognition on the WSJ dataset. The results are given in the form of phoneme error rate (PER) and training time per epoch. Our models outperform the LSTM and Reformer while being faster to train and evaluate. Details of the experiment can be found in Section 5.6.1. . 75

5.2 We report the validation phone error rate (PER) on the WSJ dataset (Section 5.6.1). We train with one model and evaluate with another to assess the approximation abilities of different models. Underline denotes training and testing with the same model. Improved cluster (rows 7-8) approximates the softmax and the shared-softmax significantly better than all the other fast attention methods. . 77

5.3 Performance comparison in automatic speech recognition on the eval2000 test set portion of the Switchboard dataset. The results are given in the form of word error rate (wer) (WER) and training time per epoch. We use a 4-gram language model that is trained on the Switchboard+Fisher training set transcripts. improved clustered attention model converges 2× faster than the vanilla softmax attention while being only 0.5% worse in WER. We additionally report results for the 12 layered TDNNF model from Section 4.4.2. More details of the experiment can be found in Section 5.6.2. 78

5.4 Comparison of autoregressive image generation of MNIST images. Our linear transformers achieve almost the same bits/dim as the full softmax attention but more than 300 times higher throughput in image generation. Full details in Section 5.7.2. 81

5.5 We train autoregressive transformers for 1 week on a single GPU to generate CIFAR-10 images. Our linear transformer completes 3 times more epochs than softmax, which results in better perplexity. Our model generates images 4,000× faster than the baselines. The full details of the experiment are in Section 5.7.2. 83

5.6 We measure the word error rate and inference times for wav2vec 2.0 model approximation. We report the WER on the clean and other portion of the Librispeech test sets. We use a greedy decoder without any language model to measure the WER and the inference times. We additionally report the WER obtained using 4-gram LM (in parenthesis). We find that *improved clustered* attention with only 100 clusters can approximate the base model with minimal loss in performance. The slower inference time is due to the short sequence lengths. In contrast to the FBank features extracted at 100Hz, the wav2vec 2.0 convolutional front end downsamples the input to the transformer encoder to 50Hz. 84

5.7 We report the performance of model approximation on GLUE and SQuAD benchmarks involving natural language processing tasks such as textual entailment (RTE) and question answering (SQuAD). More details about the tasks can be found in (Wang et al., 2019a; Rajpurkar et al., 2018). Following common practice, we report accuracy for all tasks except STS-B and SQuAD, where we report Pearson correlation and F1-score respectively. For all metrics higher is better. . 85

6.1 Model hyper-parameters and pre-training times in hours for base (B) and large (L) models. S_f refers to possible squeeze factors, S_q and S_k refer to the possible query and key-value pooling factors. E and D refer to the model dimension and the number of layers (depth) in transformer respectively. We estimate the pre-training times (PT) for 400K steps for models trained using 960 hour of Librispeech data on 8 NVIDIA V100 GPUs. 92

6.2 Comparing pre-training time for different choices of query and key-pooling factors for St-SEW-B models trained for 100K steps. 93

6.3 Inference times and WER result for base models on *other* and *clean* portions of the Librispeech test sets. We fine-tune the wav2vec 2.0 base model pre-trained on 960 hours of Librispeech data on the 10 hour transcribed subset of the Librispeech dataset. We report the WER obtained using greedy decoding as well as 4-gram LM (in parenthesis). 96
6.4 Inference times and WER result for large models on *other* and *clean* portions of the Librispeech test sets. We fine-tune the wav2vec 2.0 base model pre-trained on 960 hours of Librispeech data on the 10 hour transcribed subset of the Librispeech dataset. We report the WER obtained using greedy decoding as well as 4-gram LM (in parenthesis). 97

A.1 Comparing the performance of MAM pre-trained model when used as a feature extractor to fine-tuning the pre-trained model weights. Note that in both settings, we obtain better WER compared to the baseline model trained only on the supervised data. Fine-tuning the pre-trained model weights in significant improvements for the case of noisy portion of the test set. MAM-B refers to the base transformer model used for pre-training. TDNNF-L and TDNNF-S refer to the 12 and 7 layered TDNNF architectures, respectively. E2E-LFMMI refers to flat-start training with LFMMI. 104
A.2 Comparison of word error rates (WER) (in %) on eval2000 test set for the 300 hours Switchboard task. The 3-gram language model is based on Switchboard, whereas the 4-gram employs Switchboard+Fisher training set transcripts. Using a pre-trained model as a feature extractor or fine-tuning outperforms the baseline. Fine-tuning achieves the best WER on both callhome and switchboard part of eval set. 104
A.3 Comparison of word error rates (WER) (in %) on dev10h set for the Swahili and Tagalog languages of the Babel dataset. Using the pre-trained model performs worse than the baseline trained only with supervised data. Fine-tuning the pre-trained MAM model recovers the performance and significantly outperforms the monolingual and other baselines. 105

B.1 Comparison of autoregressive image generation throughput of MNIST and CIFAR-10 images. The experiment can be found in Section 5.7.2. For stateful-softmax we save the keys and values and reuse them for predicting the next element. A detailed description of this extra baseline can be found in Appendix B.3.3. 116
B.2 Comparison of the time required to generate a single image with autoregressive transformers on MNIST and CIFAR-10. We run all methods with a batch size of 1 both on CPU and GPU and report the total time in seconds. For all numbers in the table, lower is better. 117

Acronyms

AM	Acoustic Model.
ASR	Automatic speech recognition.
CNN	Convolutional Neural Network.
CTC	Connectionist Temporal Classification.
DAG	Directed Acyclic Graph.
DNN	deep neural networks.
E2E	end-to-end.
E2E-LFMMI	flat-start LFMMI.
GRU	Gated recurrent units.
HMM	Hidden Markov Models.
LFMMI	Lattice-Free Maximum Mutual Information.
LM	Language Model.
LSTM	Long short-term memory.
LVCSR	large vocabulary continuous speech recognition.
MAM	masked acoustic model.
MMI	Maximum Mutual Information.
RNN	Recurrent Neural Networks.
RNN	Recurrent Neural Network.
SEW	Squeeze and Efficient Wav2vec.
TDNN	time-delay neural network.
TDNNF	factorized time-delay neural network.
W2V2	wav2vec 2.0.
WER	word error rate.
WRR	WER Recovery Rate.
WSJ	Wall Street Journal.

1 Introduction

Speech is the most natural form of communication that humans use for effective interactions. Automatic speech recognition (ASR) is defined as the task of converting a spoken speech signal into text using a computing device such as a computer, phone, or even a watch.

The availability of large amounts of transcribed speech data, the advancements in *deep learning* (LeCun et al., 2015; Schmidhuber, 2015) coupled with the tremendous increase in computational resources have significantly improved large vocabulary continuous speech recognition (LVCSR). The immense advancements in modern ASR systems have been accompanied by commercialization in the form of smart assistants such as Siri, Alexa, and Google Assistant. Speech-to-Text as a service has also become crucial to support devices and applications such as smart home devices, video platforms like YouTube and Netflix, automatic subtitle generation, etc., that billions of users use daily. From the recent trends (Fortune Business Insights, 2019; Asavari and Chhabra, 2019), the number of applications based on ASR will only continue to grow over the next coming years.

While the ASR systems reach human performance levels on a handful of languages that have transcribed data in abundance, the performance on the majority of other languages is relatively poor. Moreover, the state-of-the-art ASR systems often are compute-intensive and require specialized hardware such as Graphical Processing Units (GPUs) (Nickolls et al., 2008) for both training and inference, thereby making them infeasible on devices such as watches. Thus, for speech recognition technology to become widely accessible, it is vitally important to develop techniques that require fewer data as well as computing resources.

1.1 Motivations

Much of the performance improvements in the modern-day state-of-the-art ASR systems can be attributed to these two main factors: (i) massive increase in the transcribed data and (ii) improvements in the network architecture.

Massive Datasets: Depending on the application, the problem of ASR could be simplified based on controlling factors such as isolated words, single speaker, vocabulary size, and environmental conditions. In this thesis, we consider the most difficult setting of speaker-independent LVCSR systems. The difficulty in building such a system arises from the large variations in speech due to accents, gender, speaker speed, and environmental conditions.

Building an ASR system that is robust to such factors often involves using transcribed datasets of the order of tens of thousands of hours (Xiong et al., 2017; Zhang et al., 2021; Hussein et al., 2022). Moreover, acquiring transcribed data is both a time-consuming and expensive process. Currently, only a few out of ~ 7000 languages support datasets of this magnitude.

While obtaining a large amount of transcribed data is costly and time-consuming, untranscribed audios are often available in abundance. Semi-supervised ASR (Zavaliagkos et al., 1998; Wessel and Ney, 2005; Veselý et al., 2013) is a technique that exploits untranscribed data by first building a seed model using a small amount of transcribed data and then using the model to decode untranscribed data using a beam search decoder. However, using only the best path for supervision can make the model reinforce its errors. In Chapter 3, we first introduce a method that uses dropout to quantify the uncertainty in the output of an ASR model. We then extend this for an uncertainty-aware semi-supervised learning method.

Semi-supervised learning requires first building a seed model using the transcribed data. Moreover, it assumes that the untranscribed data belongs to the same language as transcribed data. In contrast, recently proposed self-supervised learning methods such as the masked acoustic model (MAM) (Liu et al., 2021) and wav2vec 2.0 (W2V2) (Baevski et al., 2020) first pre-train the model using a large amount of untranscribed data. The pre-trained model is then fine-tuned using the available transcribed data. Both MAM and W2V2 models are pre-trained and fine-tuned on the Librispeech dataset (Panayotov et al., 2015), which contains *read* English speech. In Chapter 4, we investigate the efficacy of these models under strong domain mismatch conditions and cross-lingual settings, such as when the untranscribed and transcribed data come from two different languages. We also examine the sensitivity of self-supervised pre-trained models to the commonly used Connectionist Temporal Classification (CTC) (Graves et al., 2006a) and Lattice-Free Maximum Mutual Information (LFMMI) (Povey et al., 2016; Hadian et al., 2018) training criteria in ASR.

Network Architecture Improvements: The current state-of-the-art ASR models use the Transformer architecture for acoustic modeling (Baevski et al., 2020; Synnaeve et al., 2019; Gulati et al., 2020). The key component of the Transformer architecture (Vaswani et al., 2017) is the *attention* function, which maps an input sequence with N tokens to another sequence with the same number of tokens. Like convolution, the i-th output token is a weighted combination of input activations. However, attention models full input context using weights that are determined dynamically based on the input activations. Attention explicitly models contributions from each time step which helps with gradient vanishing/explosion issues encountered in other full-context models such as Recurrent Neural Networks (RNN).

Currently, transformers demonstrate impressive performance on a variety of tasks involving natural language (Devlin et al., 2019), images (Parmar et al., 2019), and audios (Baevski et al., 2020). While the transformer-based ASR systems (Sperber et al., 2018a; Gulati et al., 2020; Synnaeve et al., 2019; Dong et al., 2018) achieve state-of-the-art performance, they have a computation complexity that is quadratic with respect to the sequence length. This, coupled with the long sequences encountered in speech makes them prohibitively expensive for long sequences.

In Chapter 5, we present two efficient transformer variants with linear complexity of attention computation with respect to the sequence length. In Chapter 6, we further present a practically efficient transformer for self-supervised pre-training to achieve both data and compute efficient ASR models.

1.2 Summary of contributions

This thesis aims to reduce the data, and the compute requirements for developing state-of-the-art ASR systems for challenging datasets with only a few hundred hours of data or less (see datasets described in Section 2.7). Below, we summarize the main contributions of this thesis.

- **Uncertainty aware semi-supervised learning with Dropout**
  We propose a novel framework that uses “Dropout” at the test time to sample from the posterior predictive distribution of word sequences. We show that we can use the sampled hypotheses to capture the model uncertainty. We then systematically exploit this uncertainty to accurately localize the errors made by the ASR system (Vyas et al., 2019). In addition, we combine the hypotheses to produce unbiased supervision lattices for uncertainty-aware semi-supervised training (Tong et al., 2019). Our results on the Fisher English (Cieri et al., 2004) dataset show that semi-supervised training using supervision lattice generated with the proposed dropout sampling results in significant improvements over regular semi-supervised LFMMI training (Manohar et al., 2018).

- **Analyzing the impact of domain mismatch for self-supervised learning**
  We investigate the performance of the recently proposed self-supervised learning techniques MAM and W2V2 when the pre-trained models are fine-tuned on the out-of-domain datasets and under language mismatch conditions (Vyas et al., 2021b,a). Towards this objective, we pre-train models on 960h of Librispeech (Panayotov et al., 2015) dataset and fine-tune it on three different datasets, including out-of-domain (Switchboard) and cross-lingual (Babel) scenarios. Our findings indicate that while W2V2 pre-training outperforms MAM, both methods significantly improve performance compared to models trained from scratch only using the supervised data. We further show that the W2V2 pre-training helps mitigate the overfitting issues with CTC training to reduce its performance gap with flat-start LFMMI for ASR with limited training data.

- **Computationally efficient Transformer models**
  Transformers have been proven a successful model for a variety of tasks in sequence modeling. However, computing the attention matrix, which is their key component, has quadratic complexity with respect to the sequence length, thus making them prohibitively expensive for large sequences. To overcome this limitation, we propose alternatives to vanilla self-attention that scale linearly with sequence length. The first variant called "Linear attention" (Katharopoulos et al., 2020), expresses self-attention as a linear dot-product of kernel feature maps and makes use of the associativity property of matrix products to reduce the computational complexity. We show that this formulation permits an iterative implementation that dramatically accelerates autoregressive transformers and reveals their relationship to recurrent neural networks.

  The next variant "Clustered attention" (Vyas et al., 2020), approximates vanilla self-attention by clustering queries into groups and computes attention only for the centroids. To showcase their effectiveness, we examine the proposed techniques on tasks involving speech recognition, image generation, and natural language understanding.

- **Investigating compute efficient transformers for low resource ASR**
  In the context of LVCSR, we present a novel method for W2V2 pre-training to achieve both data and compute efficient ASR models. The proposed method uses stochastic input context compression to reduce inference time and memory requirements dramatically. Stochastic compression enables the training of a single model that provides a smooth trade-off between WER and inference latency. Our results for models pre-trained on 960h Librispeech dataset and fine-tuned on 10h of transcribed data show that using the same stochastic model; we get a smooth trade-off between word error rate (WER) and inference time with only marginal WER degradation compared to the W2V2 model trained for a specific setting. We further show that we can fine-tune the same stochastically pre-trained model to a specific configuration to recover the WER difference resulting in significant computational savings on pre- training models from scratch.

1.3 Thesis outline

In Figure 1.1, we present a schematic overview of this thesis. Below, we present the main organization of this thesis, briefly describing the main goal of each of the chapters.

Chapter 2 presents background on ASR. We discuss the key components of the ASR pipeline with a focus on deep neural networks (DNN) based acoustic models trained using CTC and LFMMI criteria. We also present an overview of the Transformers (Vaswani et al., 2017) model architecture. We briefly discuss the basics of self-attention and its computational complexity. We end this chapter with a discussion on the datasets and metrics that we consider for evaluating different approaches.

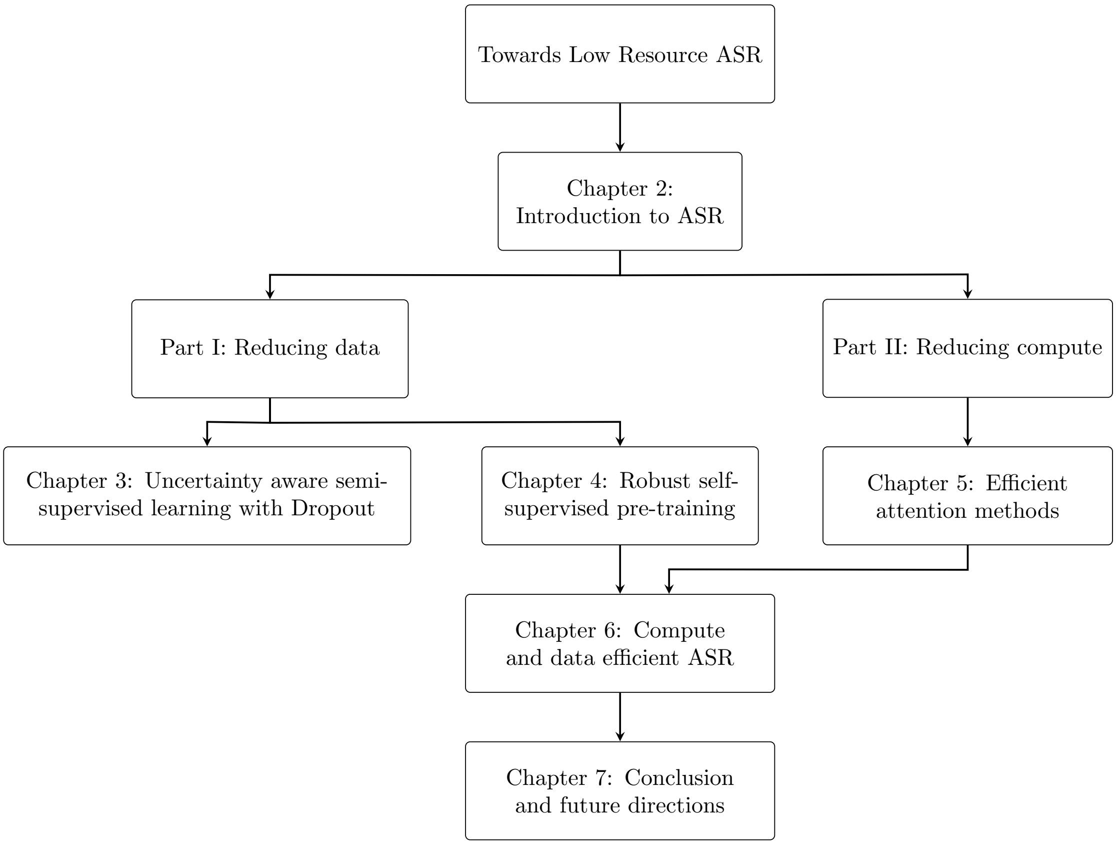

Towards Low Resource ASR
Chapter 2: Introduction to ASR
Part I: Reducing data
Part II: Reducing compute
Chapter 3: Uncertainty aware semi-supervised learning with Dropout
Chapter 4: Robust self-supervised pre-training
Chapter 5: Efficient attention methods
Chapter 6: Compute and data efficient ASR
Chapter 7: Conclusion and future directions

Figure 1.1: Schematic overview of the thesis

In Chapter 3, we propose a novel approach for semi-supervised learning with dropout. We begin with a discussion of our approach that uses dropout at test time to quantify the uncertainty in the output of the ASR system. We then discuss how we exploit the dropout uncertainty for semi-supervised learning for ASR.

Recently, self-supervised pre-training approaches to exploit a large amount of unlabelled data have received much attention. In Chapter 4, we first present two popular pre-training approaches that are based on the Transformer architecture, namely, MAM (Liu et al., 2021) and W2V2 (Baevski et al., 2020). We then investigate the domain generalization of these methods for the ASR task, i.e., when the pre-trained models are fine-tuned on out-of-domain and cross-lingual supervised datasets.

Chapter 5 focuses on tackling the computational and memory requirements of the self-attention mechanism. To this end, we propose two variants of *fast Transformers* that address the quadratic complexity of self-attention computation. We first present *Linear* attention which is based on Katharopoulos et al. (2020). Linear attention uses the kernelization trick and the associativity property of matrix products to reduce the complexity from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}(N)$, where N is the sequence length.

We next present *Clustered* attention which instead of computing the attention for every query, groups them into clusters and uses the centroids for attention computation. To further improve this approximation, we use the cluster centroids to identify the keys with the highest attention per query and compute the exact key/query dot products. This results in a model with linear complexity with respect to the sequence length for a fixed number of clusters.

In Chapter 6, we present a practical method for compute and data efficient ASR with W2V2 models. We introduce a stochastic pooling mechanism that can be independently applied to queries and keys-values to reduce the context length for the attention computation without introducing significant overheads and any additional learnable parameters. We show that our method enables training a single model that can support a number of operating points with a smooth trade-off between WER and inference latency.

Finally, in Chapter 7, we summarize our findings and propose new directions for future research.

2 Background on Automatic Speech Recognition

This chapter provides a brief background on the Lattice-Free Maximum Mutual Information (LFMMI), and Connectionist Temporal Classification (CTC) based ASR models that we use as baselines in this thesis. We start by providing an overview of the key components of a typical ASR system. We then introduce Hidden Markov Models (HMM), which forms the backbone for LFMMI ASR models. We then introduce the LFMMI proposed by Povey et al. (2016) which uses multiple stages of training. We also briefly discuss the end-to-end (E2E) flat-start LFMMI (E2E-LFMMI) extension proposed by Hadian et al. (2018). Finally, we present the CTC (Graves et al., 2006a) model. We end this chapter with a discussion on the databases and metrics used to evaluate the methods proposed in this thesis.

2.1 Key Components in ASR

A typical ASR system consists of five major components as highlighted in Figure 2.1. The *feature extraction* module takes as input an audio signal and extracts relevant features that are input to the Acoustic Model (AM). The AM computes the likelihood of an acoustic feature belonging to a phonetic state. The *Language Model (*LM*)* estimates the probability of a word sequence by modeling the correlations between words from the training text. The *decoder* starts by constructing a search graph by replacing the word tokens of the n-gram LM with the corresponding phone sequences provided by the *lexicon* (pronunciation dictionary). The decoder then outputs the most probable word hypothesis by combining the acoustic model and language model scores.

This thesis focuses on reducing the data and computation requirements for the Acoustic Model (AM). In the following sections, we discuss the key components of acoustic model training.

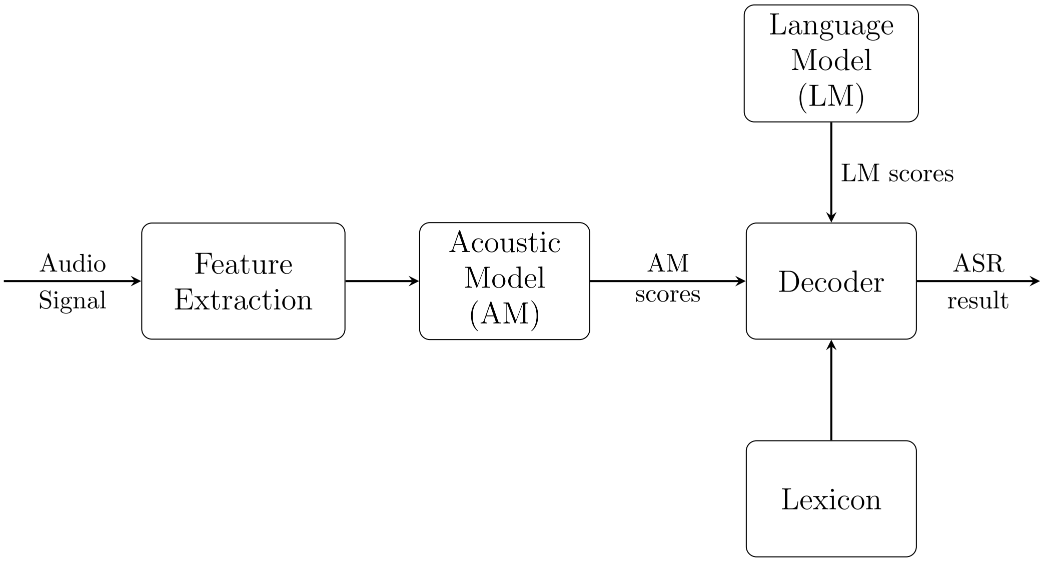


Figure 2.1: ASR System Architecture

2.2 Hidden Markov Models

Hidden Markov Model (HMM) is a statistical model that provides an elegant means to model sequential data. HMM has served as the core component of almost all large vocabulary ASR systems over the last several decades (Jelinek, 1976; Bourlard and Morgan, 1993; Povey et al., 2016). Below, we provide a short introduction to the HMMs. For more detailed reading, we refer the readers to the HMM tutorial with a focus on speech recognition by Rabiner (1989).

2.2.1 Discrete Markov Chain

A discrete Markov chain is a stochastic model that describes the evolution of a discrete-time system. At any time, the system is in one of R distinct states: $\{q_1, q_2, \dots, q_R\}$ and the system dynamics satisfy the Markov property. Markov property states that the current state of the system Q_t depends only on a limited number of past states. HMMs are based on the first-order Markov chain, where the current state only depends on the predecessor state. In terms of random variables, the state random variable Q_t at time t satisfies

$$P(Q_t = q_i | Q_{t-1} = q_j, Q_{t_2} = q_k, \dots, Q_1 = q_m) = P(Q_t = q_i | Q_{t-1} = q_j). \tag{2.1}$$

The joint probability of a sequence of T random variables $\mathcal{Q} =< Q_1, Q_2, \dots, Q_T >$ following the first-order Markovian property is given by

$$P(\mathcal{Q}) = P(Q_1) \prod_{t=2}^{T} P(Q_t | Q_{t-1}), \tag{2.2}$$

where Q_t can take value from the set $\{q_1, q_2, \dots, q_R\}$.

The first-order Markov chain can be defined using the two sets of probabilities, namely

- *Prior probabilities* refer to the probability of the Markov chain starting from a particular state. This is denoted as Π_k and is given by

  $$\Pi_k = P(Q_1 = q_k) \quad \forall\, k \in \{1, 2, \ldots, R\}. \tag{2.3}$$

  Note that the prior probabilities need to sum to one, i.e. $\sum_{k=1}^{R} \Pi_k = 1$.

- *Transition probabilities* refer to the probability of transitioning from a given state q_i at time t to another state q_j at time $t+1$. This is denoted as A_{ik} and is given by

  $$A_{ij} = P(Q_{t+1} = q_j | Q_t = q_i) \quad \forall\, i, j \in \{1, 2, \ldots, R\}. \tag{2.4}$$

  Note that $\forall\, i \in \{1, 2, \ldots, R\}$, A_{ij} satisfies the stochastic constraint $\sum_{j=1}^{R} A_{ij} = 1$.

The first-order Markov chain described above is also referred to as an observable Markov process (Rabiner, 1989) since the state Q_t at any time t also corresponds to an actual observable event. As described by Rabiner (1989), this process can be too restrictive for many practical problems of interest. In the following section, we describe how HMM extends this idea for practical applications such as ASR.

2.2.2 Hidden Markov Model

In the following, we restate the HMM description from (Dighe, 2019) with only minor changes to the notations. A Hidden Markov Model extends the observable Markov process by making the observation random variable X_t a probabilistic function of the state random variable Q_t. The observation themselves could be a discrete symbol or a continuous-valued vector. Below, we only consider first-order HMM which is an extension of the first-order Markovian chain. Moreover, we only consider continuous densities HMM which means that the observation is a continuous-valued vector.

Similar to the first-order Markov chain, at any time t, the HMM state Q_t can take values from the set: $\{q_1, \ldots, q_R\}$. Given the state $Q_t = q_i$, the model can generate the observation X_t governed by the probability density function $p(X_t | Q_t = q_i)$. This is referred to as *Emission* probability. Only the emitted observation is *visible* to an observer, the actual state remains *hidden*. Given the observation sequence, the state sequence is non-deterministic and can only be probabilistically estimated from the observation sequence.

Let us define the random variable denoting the observed sequence with T time steps as $\mathcal{X} =< X_1, X_2, \ldots, X_t >$. The HMM can now be completely defined by the following components:

- State set: $\mathbb{Q} = \{q_1, q_2, \ldots, q_R\}$. The state random variable Q_t at time t takes values from this set.

- Observation set: $\mathbb{R}^m$. The observation random variable X_t at time t takes the value $\mathbf{x}_t \in \mathbb{R}^m$.
- Prior probabilities: Π_k. The probability of starting from the state q_k as defined in (2.3).
- Transition probabilities: A_{ij}. The probability of transitioning from a given state q_i at time t to another state q_j at time $t+1$ as defined in (2.4).
- Emission probabilities: $b_k(\mathbf{x})$: The probability of emitting an observation $\mathbf{x} \in \mathbb{R}^m$ from the state q_k.

Besides the first-order Markovian assumption, the second assumption for HMM is referred to as *HMM conditional-independence* assumption which states that the observation emitted at time t is dependent only on the hidden state at time t, and is conditionally independent of all other past states and observations, i.e.

$$p(X_t|X_1,\dots,X_{t-1},Q_1,\dots,Q_t) = p(X_t|Q_t). \tag{2.5}$$

2.3 HMM based ASR System

In a typical HMM based ASR framework, we find the word sequence $\mathbf{W}^*$ that maximizes the probability of a word sequence $\mathbf{W} = [w_1, w_2, \dots, w_N]$ given the sequence of acoustic features $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]$. Here $\mathbf{x}_t \in \mathbb{R}^m$ denotes the acoustic feature at time t. Formally, this can be written as follows:

$$\begin{aligned} \mathbf{W}^* &= \arg\max P(\mathbf{W}|\mathbf{X}) && (2.6) \\ &= \frac{p(\mathbf{X}|\mathbf{W}) P_{\mathrm{LM}}(\mathbf{W})}{p(\mathbf{X})}, && (2.7) \end{aligned}$$

where $P_{\mathrm{LM}}(\mathbf{W})$ refers to the probability of the word sequence $\mathbf{W}$ estimated from a Language Model (LM) and $p(\mathbf{X}|\mathbf{W})$ is the likelihood of the acoustic features given the word sequence, estimated from an acoustic model. We can ignore the denominator $p(\mathbf{X})$ as it does not change the optimal word sequence $\mathbf{W}^*$.

To estimate $p(\mathbf{X}|\mathbf{W})$, we assume the acoustic features to be generated by a HMM model for the word sequence $\mathbf{W}$. We can then estimate $p(\mathbf{X}|\mathbf{W})$ by marginalizing over all possible hidden state sequences $\mathcal{Q}$. This is done using the Forward-Backward algorithm (Chang and Hancock, 1966; Baum et al., 1970). Thus, the likelihood $p(\mathbf{X}|\mathbf{W})$ can be computed as

$$\begin{aligned} p(\mathbf{X}|\mathbf{W}) &= \sum_{\mathcal{Q}} p(\mathbf{X}, \mathcal{Q}|\mathbf{W}) && (2.8) \\ &= \sum_{\mathcal{Q}} \Pi_{Q_1}\, p(x_t|Q_1) \prod_{t=2}^{T} A_{Q_{t-1}Q_t}\, p(x_t|Q_t). && (2.9) \end{aligned}$$

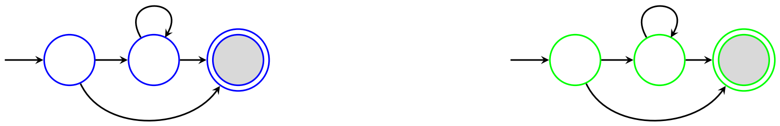
During decoding, we use the Viterbi algorithm (Rabiner, 1989) which uses the most probable state sequence to approximate marginalization over all possible state sequences $\mathcal{Q}$. When combined with beam search, this significantly reduces the search complexity for decoding. Viterbi decoding can approximate the likelihood as follows:

$$p(\mathbf{X}|\mathbf{W}) = \sum_{\mathcal{Q}} p(\mathbf{X}, \mathcal{Q}|\mathbf{W}) \tag{2.10}$$

$$= \sum_{\mathcal{Q}} \Pi_{Q_1}\, p(x_t|Q_1) \prod_{t=2}^{T} A_{Q_{t-1}Q_t}\, p(x_t|Q_t) \tag{2.11}$$

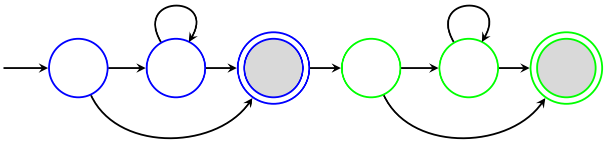
$$\approx \max_{\mathcal{Q}} \Pi_{Q_1}\, p(x_t|Q_1) \prod_{t=2}^{T} A_{Q_{t-1}Q_t}\, p(x_t|Q_t). \tag{2.12}$$

Composing Word HMMs: We previously stated that the acoustic features are generated by a HMM model for the word sequence $\mathbf{W}$. Choosing the right speech unit is essential towards building a robust ASR model that balances model complexity and available training data. A typically ASR system uses a HMM model for each character, phone, bi-phone or tri-phone unit. The HMM for a word sequence can then be composed by concatenating the individual HMMs as shown in Figure 2.2. In the example shown in Figure 2.2, we use a two-state hmm topology for each of the phone /h/ and /e/. The shaded states do not have an emission probability $b_k(\mathbf{x})$ associated and correspond to the non-emitting states (Povey et al., 2011). They make it easy to connect individual phone HMMs to make acoustic model training easy.



(a) HMM for the phone /h/

(b) HMM for the phone /e/



(c) HMM for the word *he*

Figure 2.2: The HMM corresponding to the word *he* can be obtained by concatenating the HMMs for the corresponding phones: /h/ and /e/.

2.4 Sequence Discriminative Training

In this section, we briefly go over two of the most popular sequence discriminative training criteria used in modern ASR, namely, Lattice-Free Maximum Mutual Information (LFMMI) and Connectionist Temporal Classification (CTC). For more details on Maximum Mutual

Information (MMI) and LFMMI, we refer the readers to (Yu and Deng, 2014; Povey et al., 2016; Povey, 2003). For a detailed reading on CTC, we refer the readers to (Graves et al., 2006a).

We first introduce the following notations that we use in subsequent discussions. Training dataset with R utterances is referred to as $\mathcal{D}$. The u-th utterance is denoted by $\{\mathbf{X}^u, \mathbf{W}^u\}$ with $\mathbf{X}^u$ being the feature sequence of length T and $\mathbf{W}^u$ being the reference transcription.

2.4.1 Maximum Mutual Information

The MMI criterion (Bahl et al., 1986) is a sequence discriminative training criteria that takes into account the entire utterance instead of frame-level objectives like cross-entropy loss. Formally, the MMI objective is given by

$$\begin{aligned}
\mathcal{F}_{\text{MMI}} &= \sum_{u=1}^{R} \log P\left(\mathbf{W}^u | \mathbf{X}^u, \boldsymbol{\theta}, \text{LM}\right) && (2.13)\\
&= \sum_{u=1}^{R} \log \frac{p\left(\mathbf{X}^u, \mathbf{W}^u | \boldsymbol{\theta}, \text{LM}\right)}{\sum_{\hat{\mathbf{W}}} p\left(\mathbf{X}^u, \hat{\mathbf{W}} | \boldsymbol{\theta}, \text{LM}\right)} && (2.14)\\
&= \sum_{u=1}^{R} \log \frac{p\left(\mathbf{X}^u | \mathbf{W}^u, \boldsymbol{\theta}\right) P_{\text{LM}}\left(\mathbf{W}^u\right)}{\sum_{\hat{\mathbf{W}}} p\left(\mathbf{X}^u | \hat{\mathbf{W}}, \boldsymbol{\theta}\right) P_{\text{LM}}\left(\hat{\mathbf{W}}\right)} && (2.15)\\
&= \sum_{u=1}^{R} \log \frac{p\left(\mathbf{X}^u | \mathcal{M}_{\mathbf{W}^u}, \boldsymbol{\theta}\right) P_{\text{LM}}\left(\mathbf{W}^u\right)}{\sum_{\hat{\mathbf{W}}} p\left(\mathbf{X}^u | \mathcal{M}_{\hat{\mathbf{W}}}, \boldsymbol{\theta}\right) P_{\text{LM}}\left(\hat{\mathbf{W}}\right)}, && (2.16)
\end{aligned}$$

where R refers to the total number of utterances, $\mathcal{M}_{\mathbf{W}^u}$ corresponds to the numerator HMM corresponding to a word sequence $\mathbf{W}^u$, $P_{\text{LM}}(\mathbf{W})$ denotes the language model probability for any word sequence $\mathbf{W}$, and $\boldsymbol{\theta}$ is the model parameter.

Given any word sequence $\mathbf{W}$, we can compute the $p(\mathbf{X}^u | \mathcal{M}_{\mathbf{W}}, \boldsymbol{\theta})$ as follows:

$$p(\mathbf{X}^u | \mathcal{M}_{\mathbf{W}}, \boldsymbol{\theta}) = \sum_{\mathcal{Q} \in \mathcal{M}_{\mathbf{W}}} p(\mathcal{Q}, \mathbf{X}^u | \boldsymbol{\theta}), \tag{2.17}$$

where $\mathcal{Q}$ corresponds to all possible state sequences corresponding to the HMM model for the word sequence $\mathbf{W}$ denoted as $\mathcal{M}_{\mathbf{W}}$. This can be computed effectively using the Forward-Backward algorithm for the HMM (Chang and Hancock, 1966; Baum et al., 1970).

Thus, in (2.16), the numerator computes the log-likelihood of the observation sequence $\mathbf{X}^u$ given the transcription $\mathbf{W}^u$ and the denominator computes the log-likelihood over all possible word sequences. This can also be seen as a contrastive loss where we are trying to maximize the observation likelihood given the true word sequence and minimize the likelihood for all other word sequence weighted by the LM probabilities.

Computing the denominator in (2.16) requires summation over all possible word sequences which is not practically feasible. This is because the HMM graph representing the denominator is too large for computation. The typical approach to remedy this is to use a lattice representing the most competing word sequences. A path through the lattice represents a possible word sequence. The denominator lattice with competing word sequences is obtained by using a another model that is usually trained with cross-entropy objective.

2.4.2 Lattice-Free Maximum Mutual Information

LFMMI model was proposed by Povey et al. (2016) as an alternate to the Lattice-based MMI systems. This is achieved by switching from a word LM to a phone LM. The number of phones are much smaller than the number of words resulting in significant reduction in the number of states for the phone LM. We can now represent denominator as a graph with no approximation. The denominator can be computed using the GPU hardware for fast training.

Formally, denoting the denominator graph as $\mathcal{M}_{\text{den}}$, the LFMMI cost function is given as follows:

$$\mathcal{F}_{\text{LFMMI}} = \sum_{u=1}^{R} \log \frac{p\left(\mathbf{X}^u \,|\, \mathcal{M}_{\mathbf{W}^u}, \boldsymbol{\theta}\right) P_{\text{LM}}\left(\mathbf{W}^u\right)}{p\left(\mathbf{X}^u \,|\, \mathcal{M}_{\text{den}}, \boldsymbol{\theta}\right)}. \tag{2.18}$$

The standard implementation of LFMMI computes the numerator in (2.18) using alignments from another acoustic model; typically a monophone HMM/GMM model. The resulting ASR system requires multiple stages for acoustic model training. Alternatively, Hadian et al. (2018) proposed the numerator estimation in a single stage end-to-end (E2E) manner. We refer to this as the flat-start LFMMI (E2E-LFMMI) model.

2.4.3 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) (Graves et al., 2006a) is another sequence discriminative training criterion that does not require any frame-level alignments between the input feature sequence $\mathbf{X}^u$ of length T and the target label sequence $\mathbf{y}^u$ of length U. For ASR, the target label sequence $\mathbf{y}^u$ can be obtained from the word sequence $\mathbf{W}^u$ by using a lexicon mapping that maps each word to a unique label sequence. The labels are typically characters, phonemes, or sub-word unit.

Analogous to the state sequence defined in HMM, CTC starts by defining the concept of a path between $\mathbf{X}^u$ and $\mathbf{y}^u$. A valid path is generated by extending the label sequence $\mathbf{y}^u$ such that it matches the input length T. The target label sequence can be extended by either repeating any label and/or inserting the *blank* symbol. This extended representation is referred to as the CTC path. The *blank* symbol represents the probability of not emitting any label at the particular time step. CTC objective computes the probability of the target labels given the input features by summing over the probability of all valid paths between the two.

The CTC objective is given by

$$\mathcal{F}_{\text{ctc}} = \sum_{u=1}^{R} \log P\left(\mathbf{W}^u | \mathbf{X}^u, \boldsymbol{\theta}\right), \tag{2.19}$$

where $\boldsymbol{\theta}$ is the model parameter.

Given the input features $\mathbf{X}^u$ and the word sequence $\mathbf{W}^u$, we first use the lexical mapping to obtain the label sequence $\mathbf{y}^u$. We then compute $P\left(\mathbf{W}^u | \mathbf{X}^u, \boldsymbol{\theta}\right)$ as follows:

$$P\left(\mathbf{W}^u | \mathbf{X}^u, \boldsymbol{\theta}\right) = \sum_{\pi \in \mathbf{y}^u} P\left(\pi | \mathbf{X}^u, \boldsymbol{\theta}\right) \tag{2.20}$$

$$= \sum_{\pi \in \mathbf{y}^u} \prod_{t=1}^{\mathbb{T}_u} P\left(\pi_t | \mathrm{x}^u, \boldsymbol{\theta}\right), \tag{2.21}$$

where π corresponds to a valid CTC path.

The conditional probability of the label at each timestep, $p(\pi_t|\mathrm{x}^u,\boldsymbol{\theta})$ is estimated using a neural network conditioned on the whole input sequence. The model can be trained to maximize (2.21) using gradient descent. The required gradients are computed using the Forward-Backward algorithm as described by Graves et al. (2006a).

Relation between CTC and HMM

As discussed by Zeyer et al. (2017), the CTC objective function can be thought of as maximum likelihood training over a composite HMM, where each label (e.g. character) has a special 2-state HMM topology (Figure 2.3). The composite HMM can be obtained by concatenating a blank state (with a self-loop and a forward null transition) with the label HMMs. A single blank state is inserted between each repetitive label. This composite HMM ensures that all possible state sequences are identical to all possible paths $\pi \in \mathbf{y}^u$. For more details connecting the CTC and the LFMMI criteria, we refer the readers to (Zeyer et al., 2017).

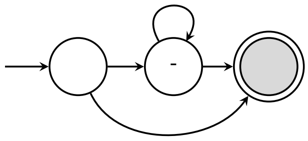


Figure 2.3: HMM corresponding to the CTC label

2.5 Deep Neural Networks

The interest in *deep learning* has exploded in the last decade after its pivotal success in the Imagenet competition in 2012 (Krizhevsky et al., 2017).

The hardware advancements in computing with Graphical Processing Units (GPUs) (Nickolls et al., 2008) coupled with the open source deep learning toolkits (Collobert et al., 2002; Bergstra et al., 2011; Povey et al., 2011; Jia et al., 2014; Abadi et al., 2016; Paszke et al., 2019) have played a key role in the democratization and success of deep learning.

A deep neural network is a composition of differentiable layers that can be structured to form any Directed Acyclic Graph (DAG). Modern deep learning frameworks support any differentiable function $f : \mathbb{R}^P \to \mathbb{R}^Q$ to be expressed as a neural network layer. The modularity and composability of the deep learning stack has resulted in a wide variety of deep learning architectures such as VGG (Simonyan and Zisserman, 2015), ResNet (He et al., 2016), Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), and Transformers (Vaswani et al., 2017).

In the following, we first provide a brief introduction to the convolutional neural networks (CNNs) and recurrent neural networks (RNNs). We then discuss in detail the Transformer architecture which is a key topic of focus in this thesis.

2.5.1 Convolutional Neural Networks

Modern Convolutional Neural Networks (CNNs) (LeCun et al., 1989) uses one or more convolution layers to successively build higher-level abstraction from an input signal. Convolutional Neural Network (CNN) models were inspired from time-delay neural network (TDNN) (Waibel, 1989) to efficiently process the input signal using sliding filters.

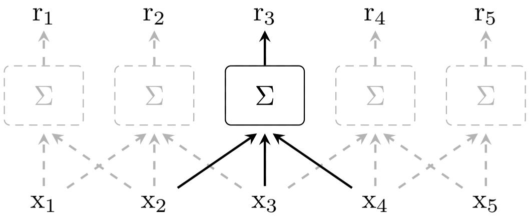


Figure 2.4: Illustration of a convolution operation over a 1D input signal. The output at any time-step is a weighted combination of the input signal with the kernel weights over the receptive field of size three.

As shown in the Figure 2.4, a CNN or TDNN model uses the convolution operation to generate the output at each time step. A convolution operation consists of an element-wise multiplication of the kernel weights with corresponding inputs followed by a summation of the multiples. Multiple layers of convolutions and non-linear activation functions are stacked to build more complex representations using a larger receptive field.

2.5.2 Recurrent Neural Networks

Previously discussed CNN models can only model fixed size context and are often used for applications with fixed input lengths. In contrast to this, Recurrent Neural Network (RNN)s (Hopfield, 1982; Rumelhart et al., 1986) allow dealing with sequences of arbitrary length.

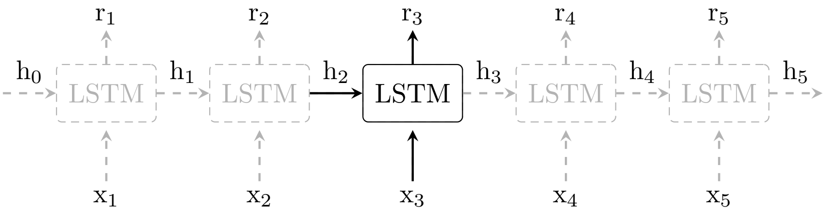


Figure 2.5: Example of a LSTM based RNN model. The hidden state, h_t stores the information of the past inputs till the time t. The output representation r_t at any time t is computed using the hidden state h_{t-1} and the input x_t. The hidden state is also updated to compute output at the next time step

As shown in Figure 2.5, RNN models consists of a time-dependent *hidden state* that stores information about the past inputs. At any time t, a RNN model uses the hidden state, h_{t-1} and the current input, x_t to produce the output as well as to update the hidden state. While the vanilla RNN models are able to make use of the entire context, vanishing or exploding gradients is a key issue that hinders effective their training (Hochreiter, 1991; Bengio et al., 1993). Gated variants such as LSTM (Hochreiter and Schmidhuber, 1997) and Gated recurrent units (GRU) (Cho et al., 2014) allow for better gradient propagation and are often used together with convolutional layers for automatic speech recognition (Amodei et al., 2016a).

2.5.3 Transformers

Transformer models were introduced by Vaswani et al. (2017) in the context of neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015a). Since their introduction they have demonstrated impressive results on a variety of supervised and self-supervised training tasks dealing with natural language (Devlin et al., 2019; Radford et al., 2019), audio (Sperber et al., 2018b; Baevski et al., 2020), and images (Parmar et al., 2019; Dosovitskiy et al., 2021).

We show the Transformer model architecture in Figure 2.6. The transformer model maps an input sequence, $x \in \mathbb{R}^{N \times F}$ with N token vectors of dimensions F to an output sequence with same number of tokens. The transformer is defined as the composition of L transformer layers $T_1(\cdot), \ldots, T_L(\cdot)$. Given an input sequence x, the output $T_l(x)$ for the l-th transformer layer can be obtained as follows:

$$y = \mathrm{LN}(\mathrm{SA}_l(x) + x), \tag{2.22}$$

$$T_l(x) = \mathrm{LN}(\mathrm{FFN}(y) + y), \tag{2.23}$$

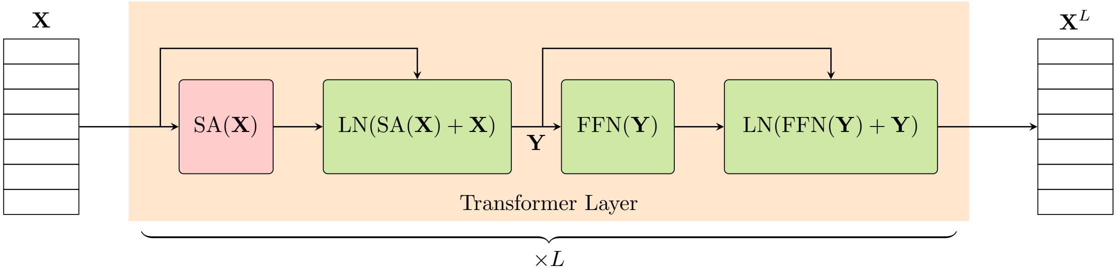


Figure 2.6: The Transformer model architecture. The Transformer model maps an input sequence **X** with N tokens to another sequence with the same number of tokens. The core component of the Transformer model is the self-attention function indicated by SA(·). Self-attention is the only function acts accross sequences. FFN(·) refers to the feed-forward module, and LN(·) refers to the layer normalization.

where LN(·) refers to the *layer normalization* function (Ba et al., 2016), FFN(·) refers to the *feedforward network* that transforms each feature independently of the others and is usually implemented with a small two-layer multilayer perceptron. $\mathrm{SA}_l(\cdot)$ is the self attention function and is the only part of the transformer that acts across sequences.

The self attention function $\mathrm{SA}_l(\cdot)$ computes, for every position, a weighted average of the feature representations of all other positions with a weight proportional to a similarity score between the representations. Formally, given the input sequence x, we first obtain the "queries" $Q \in \mathbb{R}^{N\times D}$, "keys" $K \in \mathbb{R}^{N\times D}$, and "values" $V \in \mathbb{R}^{N\times M}$ by linearly projecting x as follows:

$$Q = xW_Q, \tag{2.24}$$

$$K = xW_K, \tag{2.25}$$

$$V = xW_V, \tag{2.26}$$

where the matrices $W_Q \in \mathbb{R}^{F\times D}$, $W_K \in \mathbb{R}^{F\times D}$ and $W_V \in \mathbb{R}^{F\times M}$.

The queries, keys, and values are then used to compute the attention weights $A \in \mathbb{R}^{N\times N}$ and the self-attention output $\mathrm{SA}_l(x) = \hat{V}$ as follows,

$$A = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{D}}\right), \tag{2.27}$$

$$\hat{V} = AV. \tag{2.28}$$

Note that in the previous equation, the softmax function is applied rowwise to QK^T.

Comparison with convolutional and recurrent networks

Figure 2.7 illustrates the output computation for the third time-step using self-attention mechanism. Comparing this to Figure 2.4, we can see that both convolution and self-attention express the output as the weighted combination of input features. However, self-attention uses the entire context as opposed to a fixed-length input context used by the convolutions. Furthermore, the attention weights are not fixed and are determined by the input sequence. Comparing with the RNN in Figure 2.5, we can also see that self-attention weights explicitly model the contributions from each time step. This alleviates gradient vanishing issues by making direct connections between each pair of time-step.

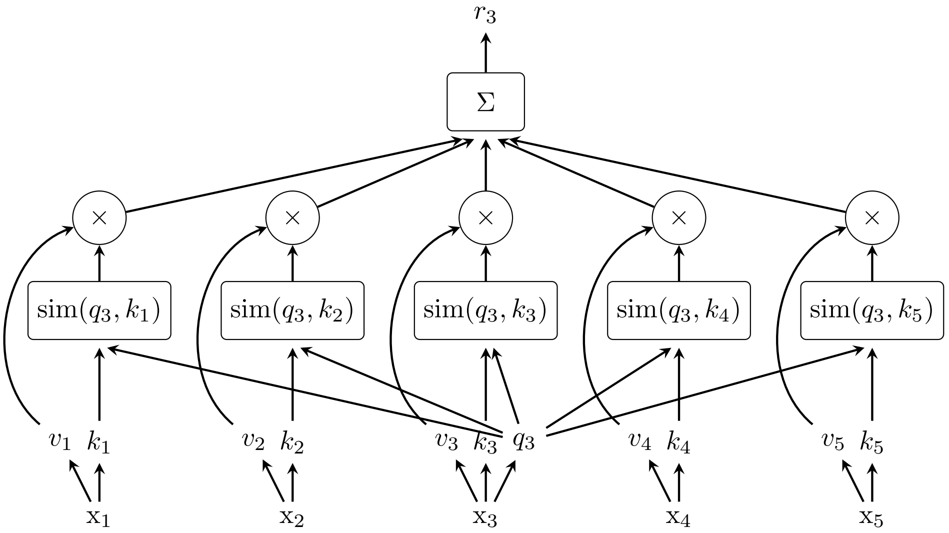


Figure 2.7: Showing the self-attention output computation for the third time-step. The input x_t is first projected to obtain the queries, keys, and values. We then obtain the attention weights by computing the similarity scores between the third query, q_3 and all the keys. The attention weights are normalized to sum to 1. The output representation r_3 is simply the weighted combination of values with the previously computed attention scores. Note that we skip the normalization step for clarity. Furthermore, the dot-product attention can be obtained by substituting $\text{sim}(q,k)$ by $\exp(q^T k)$.

Self Attention: Computation Complexity

Figure 2.8 shows the self attention computation described previously. From Figure 2.8 and the (2.27) it is evident the computing attention weights requires $\mathcal{O}\left(N^2 D\right)$ operations and the weighted average of (2.28) requires $\mathcal{O}\left(N^2 M\right)$. This results in an asymptotic complexity of $\mathcal{O}\left(N^2 D + N^2 M\right)$.

Transformers for Speech Recognition

Following their success on natural language processing tasks, transformers currently also achieve state-of-the-art performance on a number of ASR benchmarks. Karita et al. (2019) compared the performance of the transformer model to the LSTM based conventional recurrent neural networks. Their findings reveal that the transformer model achieves superior

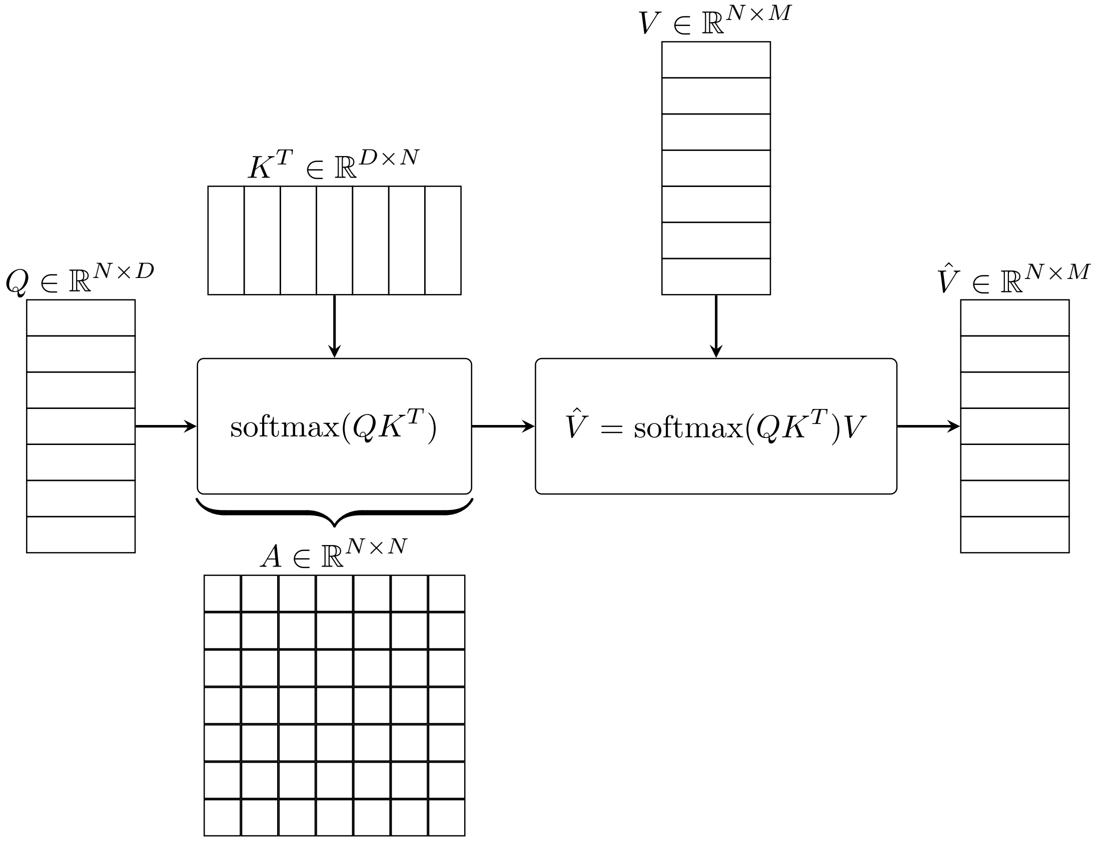

$V \in \mathbb{R}^{N\times M}$
$K^T \in \mathbb{R}^{D\times N}$
$Q \in \mathbb{R}^{N\times D}$
$\hat{V} \in \mathbb{R}^{N\times M}$
$\mathrm{softmax}(QK^T)$
$\hat{V} = \mathrm{softmax}(QK^T)V$
$A \in \mathbb{R}^{N\times N}$

Figure 2.8: Flow-chart demonstrating the computation for *vanilla* attention. Given matrices Q, K, V, we first compute the attention weights $A \in \mathbb{R}^{N\times N}$. The output value $\hat{V}_i$ is simply sum of values V_j weighted by the attention weights A_{ij}.

performance on 13/15 benchmarks. Furthermore, *Conformer* (Gulati et al., 2020), which is a recently proposed variant of the Transformer model, further improves the ASR performance over the vanilla transformers (Guo et al., 2021).

Besides supervised training, transformers have also become the de-facto choice of architecture for recent self-supervised pre-training methods that are trained on a large amount of unlabelled utterances. The most popular of these are (1) wav2vec 2.0 (W2V2) model and its variants (Baevski et al., 2020; Chung et al., 2021; Sadhu et al., 2021) that are trained using the contrastive loss and (2) masked acoustic model (MAM) variants (Liu et al., 2021; Wang et al., 2020) that are based on the reconstruction of masked segments. The transformer-based pre-trained models are subsequently fine-tuned on the supervised data to improve the ASR acoustic model. Baevski et al. (2020) report a relative WER improvement of 41% and 56% on the clean and other portions of the Librispeech dataset when the pre-trained W2V2 model is fine-tuned on the 100 hours of supervised data.

Given their wide success, this thesis focuses on analyzing and improving different aspects of transformer-based ASR models. First, in Chapter 4, we analyze the out-of-domain generalization for MAM and W2V2 based self-supervised pre-trained models. Next, in Chapter 5, we present two novel variants of fast transformers that improve the computational complexity of self-attention to efficiently process long sequences. Finally, in Chapter 6, we present a practical method for compute and data efficient ASR with W2V2 models.

2.6 ASR Evaluation Metric

The performance of an ASR system is most commonly evaluated using the *word error rate (*WER*)* metric. Given the reference transcripts for the test dataset and the outputs of an ASR system, the word error rate is defined as:

$$\text{WER (in \%)} = 100 \times \frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\text{Number of words in reference}}, \tag{2.29}$$

where the number of substitutions, insertions, and deletions are obtained by aligning ASR output and reference with minimum edit distance. WER is measured in percentage and signifies the number of errors ASR system makes relative to the reference length. Thus a lower ASR is desired.

For semi-supervised learning experiments, we assume some supervised training data $\mathcal{D}_S$ and some untranscribed data $\mathcal{D}_U$. In this setup, we also report WER Recovery Rate (WRR) that measures the WER improvements from the semi-supervised learning. WRR is given by

$$\text{WRR} = \frac{\text{BaselineWER} - \text{SemisupWER}}{\text{BaselineWER} - \text{OracleWER}}. \tag{2.30}$$

Here, BaselineWER refers to the WER for the baseline system trained only on $\mathcal{D}_S$, SemisupWER refers to ASR system that exploits $\mathcal{D}_U$ using semi-supervised training. Finally OracleWER refers to the system that uses $\mathcal{D}_S$ and true transcriptions for $\mathcal{D}_U$ to build the oracle model.

We leave the discussion of any other experiment specific metric to the corresponding chapter.

2.7 Databases

In the following, we discuss the databases considered for this thesis. Given that the objective of the thesis is to improve the ASR performance for languages with limited resources, we experiment with databases between a few tens of hours to a couple of hundreds of hours of supervised training data. Below, we present the details of all the databases used in the thesis.

2.7.1 Fisher Corpus

Fisher English Corpus (Cieri et al., 2004) was collected with Fisher telephone conversation protocol where a large number of participants were asked to make three telephone calls lasting 10 minutes. For each call, a participant discussed an assigned topic with another participant, whom they typically do not know. This resulted in a dataset with large inter-speaker variation and vocabulary usage. Unlike previous data collection protocols such as Switchboard-II (Godfrey et al., 1992) that were originally developed for language and speaker identification but were later adapted for ASR, the Fisher Corpus was developed to address a critical need of speech researchers working on robust ASR systems.

The corpus is released in two parts. Part 1 Transcripts contains 5850 telephone conversations (984 hours) in English. Part 2 contains 5849 audio records, each containing an entire conversation of up to ten minutes. The two parts contain roughly 1960 hours of transcribed speech for training with 6813 female speakers, and 5104 male speakers. The corpus also contains 3.3 hours of data for development and 3.2 hours of data for testing.

Similar to (Manohar et al., 2018), we evaluate the proposed dropout based self-supervised approach on the Fisher English corpus. We randomly chose a subset of speakers (250 hours) from the corpus to be used as unsupervised data. We use a 50 hours subset from the corpus as the supervised data. The results are reported on separately held-out development and test sets, which are part of the standard Kaldi (Povey et al., 2011) recipe.

2.7.2 Librispeech Corpus

Librispeech corpus introduced by Panayotov et al. (2015) consists of 1000 hours of read English speech recorded at 16 KHz sampling rate. The corpus is derived from audiobooks that are part of the LibriVox project. The training portion of the corpus is split into three subsets, with about 100, 360 and 500 hours of data, respectively. The *clean* and *other* portions of the dataset correspond to the higher and noisy quality of the recordings, respectively. In Table 2.1, we provide the statistics on the LibriSpeech corpus from Panayotov et al. (2015).

In this thesis, we always use only the 100h subset of the dataset for training ASR models. Our models are evaluated on both the *clean* and *other* portions of the dev and test sets.

2.7.3 Switchboard Corpus

The Switchboard Corpus is a collection of about 2400 two-sided telephone conversations consisting of about 260 hours of telephonic conversations recorded at 8 KHz. This is also referred to as Switchboard (300h) setup in literature. There are 543 speakers in total, with 302 male and 241 female speakers from all areas of the United States. Switchboard corpus does not contain a development set so we split the dataset into a train (255 hours) and a dev set (5 hours). We use the 2000 HUB5 evaluation set, which comprises about 11 hours of data.

Table 2.1: Statistics of the LibriSpeech database.

Subset	Total data (in hours)	Female speakers	Male speakers
dev-clean	5.4	20	20
test-clean	5.4	20	20
dev-other	5.3	16	17
test-other	5.1	17	16
train-clean-100	100.6	125	126
train-clean-360	363.6	439	482
train-other-500	496.7	564	602

The data is collected by a computer-driven robot operator that selects a participant and dials another person to participate in a conversation. The participants are given a topic for discussion, and the speech is recorded from the two subjects into separate channels. There are about 70 topics, of which about 50 were most frequently used. The topics and callees were selected such that no two speakers would converse together more than once and no participant spoke more than once on a given topic.

2.7.4 Babel Corpus

The BABEL speech corpus (Gales et al., 2014) is undertaken as a part of IARPA Babel program, which aims to advance speech recognition for a diverse set of languages with limited resources. Each language in the Babel language pack contains conversational speech data with transcriptions for upto 80 hours.

In this thesis, we use the *Swahili* and *Tagalog* languages from the Babel dataset to study the cross-lingual generalization performance for different self-supervised pre-training approaches. Swahili and Tagalog comprise 38.5 and 84 hours of supervised training data, respectively. For both languages, we report the results on the *dev10h* development part due to the lack of a separate evaluation set. We use a 2 hour development set for model selection.

2.7.5 Wall Street Journal Corpus

The Wall Street Journal (WSJ) corpus introduced by Paul and Baker (1992) contains read English data from a large number of speakers. The speakers are asked to read out newspaper text paragraphs. The training set consists of 80 hours of transcribed data from 284 speakers which is usually referred to as SI-284. The development data labeled as *dev92* consists of 503 sentences of development data from ten different speakers. The evaluation data, *eval*93, consists of 333 sentences of test data from another eight different speakers. Detailed statistics is shown in Table 2.2

Table 2.2: Statistics of the WSJ dataset.

Subset	#Speakers	#Utterance	Total data (in hours)
Train	284	37416	81
Dev	10	503	1.1
Eval	8	333	1.1

3 Uncertainty Aware Semi-supervised Learning with Dropout

3.1 Introduction

Semi-supervised learning is an approach that improves the performance of the ASR models trained with limited transcribed data but with abundant untranscribed audios. This is achieved by first training a seed model using the transcribed data for supervision. The seed model is then used for automatically transcribing the unlabelled audios. However, the improvements greatly depend on the quality of the seed model and can cause the model to reinforce its errors.

To this end, we present a novel framework that uses "Dropout" for uncertainty aware semi-supervised learning. Given the seed model, we use dropout during inference to generate a decoding hypothesis for the test utterance. For any utterance, we generate N decoded hypotheses selecting different random active neurons each time. As shown in (Gal and Ghahramani, 2016), this can lead to an approximate Bayesian inference over the acoustic model weights and approximates sampling from the posterior predictive distribution over word sequences. We then combine all the N hypotheses to form an unbiased supervision lattice for the corresponding unlabeled utterance. As discussed in Section 3.3, the variations in different decoded hypotheses are often highly localized and depict locations where the ASR decoding might be inaccurate. This allows the model to learn and take advantage of the correct decoding segments while the gradients corresponding to the erroneous words are obfuscated.

In the rest of the chapter, we first present our dropout based framework for ASR uncertainty estimation and quantification in Section 3.2. Then, in Section 3.3, we conduct experiments to show that the proposed frameworks can identify ASR errors without the need for ground truth transcripts. In Section 3.5, we extend this framework for semi-supervised learning and discuss related works. Finally, in Section 3.7, we evaluate our proposed method on the Fisher-English corpus (Cieri et al., 2004).

We show that the variations in different dropout hypotheses are often highly localized and indicate the locations of erroneous words. We also demonstrate that the localized uncertainties can be used to accurately estimate the WER without the need for oracle transcriptions. Furthermore, our experiments on the Fisher English dataset show that we can combine the dropout hypotheses to improve the quality of supervision lattice for semi-supervised learning.

3.2 ASR Uncertainty estimation with Dropout

In this section, we present the details of our approach to estimating uncertainty in the output of an ASR system. Our approach is based on the seminal work by Gal (2016). In this work, we model uncertainty at the word level and explore its effectiveness for localizing ASR errors and estimating WER without the need for transcripts.

Dropout-based training (Srivastava et al., 2014) is a standard regularization technique often used to improve the generalization properties of state-of-the-art deep learning models used in computer vision, speech processing, and natural language applications. While dropout is typically used during training to prevent overfitting of DNNs, Gal and Ghahramani (2016) show that dropout can also be used during inference to compute the model's uncertainty on its predictions.

Given a DNN model trained with dropout, the *predictive uncertainty* for a test sample can be estimated using Monte Carlo dropout. During inference, a test sample is forward propagated N times with a randomly generated random mask. Formally, let $\mathcal{Y}_x$ be the random variable denoting the output of the DNN given the input $\mathbf{x}$. Also, let $\hat{\mathbf{y}}_i(x)$ denote the output for the i-th forward pass with a randomly generated dropout mask. The estimates for the mean and the variance for $\mathcal{Y}_x$ are given by

$$\mathbb{E}[\mathcal{Y}_x] \approx \sum_{i=1}^{N} \frac{\hat{\mathbf{y}}_i(x)}{N} \tag{3.1}$$

$$\mathbb{V}[\mathcal{Y}_x] \approx \tau^{-1}\mathbf{I}_D + \frac{1}{N}\sum_{i=1}^{N} \hat{\mathbf{y}}_i(x)^\top \hat{\mathbf{y}}_i(x) - \mathbb{E}[\mathbf{y}]^\top \mathbb{E}[\mathbf{y}], \tag{3.2}$$

where, τ is constant to be determined by the model structure.

In this work, we use Monte Carlo dropout during inference to model word level uncertainties. As shown in Figure 3.1, for each utterance, we forward-pass it N times through the DNN acoustic model with random dropout masks. The frame-level state posteriors corresponding to the N acoustic model outputs are then processed through the decoding pipeline to generate N dropout-hypotheses. As shown by Gal and Ghahramani (2016), this process leads to a Bayesian inference over the acoustic model weights.

We also obtain a hypothesis by keeping the dropout off during test time, as is done traditionally. The resulting $N+1$ hypotheses can then be used to estimate the word-level confidences, localize ASR errors, and estimate WER for a given utterance.

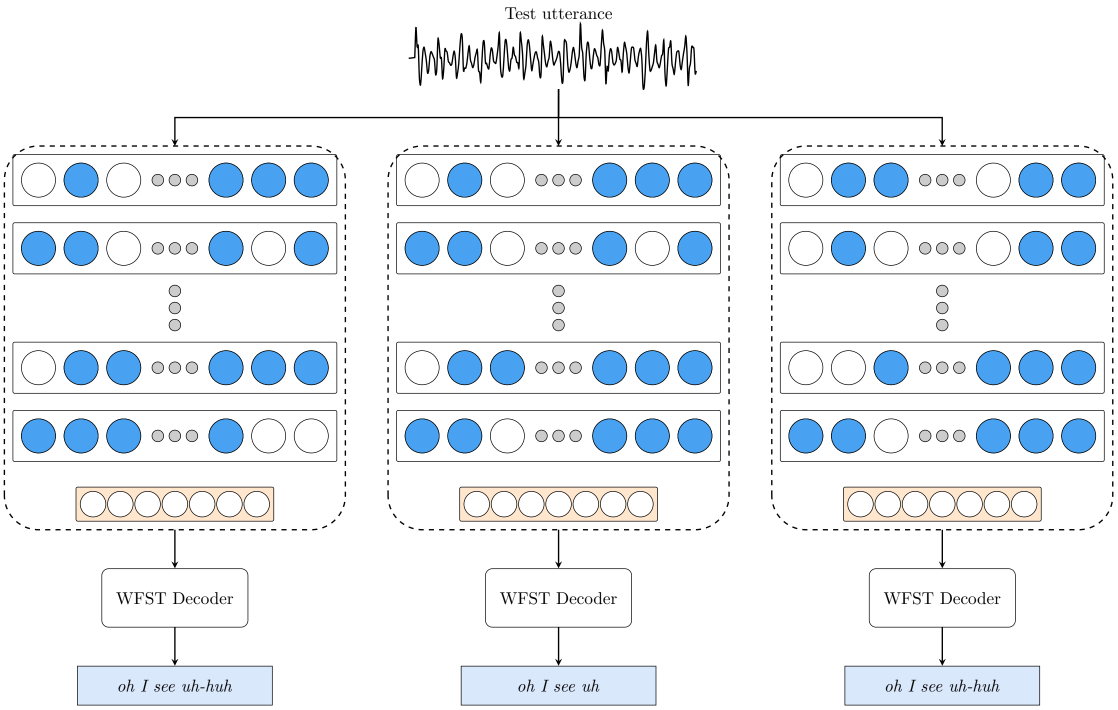
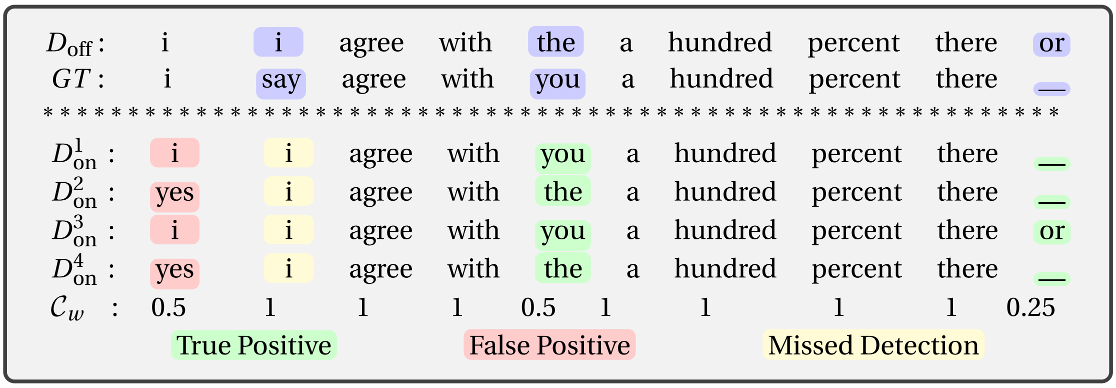


Figure 3.1: Decoding with dropout on at test time. Each network represents a different random selection of the active nodes. The white nodes denote dropped out units. As shown in the figure, dropout usage during inference can change the best hypothesis.



D_{off} :	i	i	agree	with	the	a	hundred	percent	there	or
GT :	i	say	agree	with	you	a	hundred	percent	there	—
* *										
D^1_{on} :	i	i	agree	with	you	a	hundred	percent	there	—
D^2_{on} :	yes	i	agree	with	the	a	hundred	percent	there	—
D^3_{on} :	i	i	agree	with	you	a	hundred	percent	there	or
D^4_{on} :	yes	i	agree	with	the	a	hundred	percent	there	—
$\mathcal{C}_w$:	0.5	1	1	1	0.5	1	1	1	1	0.25

True Positive False Positive Missed Detection

Figure 3.2: Example demonstrating the use of dropout for word-level confidence estimation and subsequent error localization given a confidence threshold. We first align the $N = 4$ dropout turned-on hypotheses $\{D^i_{\text{on}}\}^N_{i=1}$ against the hypotheses with the dropout turned-off D_{off}. Then, for each word, we use the mean agreement between all the hypotheses to estimate its confidence. Each word with confidence below the set threshold is considered to be an error.

Figure 3.2 shows an example of an utterance decoded $N = 4$ times with dropout turned on. Here, D_{off} refers to the decoded output with dropout turned off and D^1_{on}-D^4_{on} refers to the decoded output with dropout on. GT refers to the ground truth transcript and $\mathcal{C}_w$ is the word level estimated confidence. The blue boxes refer to the errors between the D_{off} case and GT which are used in the traditional WER computation. The red and green boxes show the word positions where D^1_{on}-D^4_{on} hypotheses disagree with each other. The differences in decoded hypotheses are due to the acoustic model's uncertainty in predicting the posterior probabilities for acoustic features in some segments of the utterance. For the utterance shown below, we observe that two uncertain word positions overlap with the mismatches between D_{off} and GT (true positive detections), one uncertainty is a false positive, and one mismatch is not detected.

3.2.1 Word Error Rate Estimation

The word error rate (WER) metric is used as a straightforward way to evaluate and compare the performance of ASR systems. Word errors are typically caused by the inputs which are noisy or exhibit a mismatch with the training conditions. Such inputs also lead to a higher predictive uncertainty of the ASR model, which indicates potential speech recognition errors. We exploit dropout-based uncertainty to estimate the WER for an utterance i in an unsupervised manner.

We start by calculating the edit distances between all $\binom{N}{2}$ pairs corresponding to the N dropout hypotheses. The mean edit distance, ${}^iE_\mu$, is obtained as the mean of the top-K edit distances, where K is the hyper-parameter tuned on the development data. Similarly, we then obtain the mean length, ${}^iL_\mu$, of the decoding corresponding to the top-K edit distances. The WER for the utterance is then estimated as the ratio of ${}^iE_\mu$ and ${}^iL_\mu$. Let D denote the whole dataset with $|D|$ utterances. The WER of this dataset is calculated as the ratio of total mean edit distance and total mean length summed over all the utterances.

Let ${}^i_kE$ and ${}^i_kL$ denote the top-K edit distances and corresponding lengths for the utterance i. The WER for the utterance i is given by:

$$ {}^iE_\mu = \frac{\sum_{j=1}^{K} {}^i_kE_j}{K} \tag{3.3} $$

$$ {}^iL_\mu = \frac{\sum_{j=1}^{K} {}^i_kL_j}{K} \tag{3.4} $$

$$ {}^iWER = \frac{{}^iE_\mu}{{}^iL_\mu} \tag{3.5} $$

$$ WER_{data} = \frac{\sum_{i=1}^{|D|} {}^iE_\mu}{\sum_{i=1}^{|D|} {}^iL_\mu} \tag{3.6} $$

3.2.2 Error Localization Using Word-Level Confidence

In addition to estimating WER, we also exploit dropout uncertainty to localize the errors in ASR. To achieve this, we propose a method that relies on computing the word-level confidences of the ASR hypotheses, i.e., the D_{off} case. Word confidences represent the word-by-word reliability of the D_{off} decoding. If the confidence $\mathcal{C}_w$ of a word w in the decoding D_{off} is less than a pre-defined threshold τ, our system predicts a *potential location of error*. If the predicted error locations are the same as the mismatch positions between D_{off} and ground truth transcription GT, we can claim that the error localization is accurate. Our error localization method then works as follows.

To estimate the word level confidences, similar to ROVER (Fiscus, 1997), we first align the N dropout turned-on hypotheses $\{D^i_{\text{on}}\}^N_{i=1}$ against the hypotheses with the dropout turned-off D_{off}. Then, for each word, we use the mean agreement between all the hypotheses to estimate its confidence. Formally, for an utterance, the confidence for its w^{th} word in ASR decoding D_{off} is given by

$$C_w = \frac{\sum_{k=1}^{N} I(D^k_{\text{on}}[w] = D_{\text{off}}[w])}{N}, \tag{3.7}$$

where, $D^k_{\text{on}}[w]$ and $D_{\text{off}}[w]$ denote w^{th} words in decoding D^k_{on} and D_{off} respectively and function $I(\dot{})$ is the indicator function. The confidences C_w lie in the range $[0,1]$ and we predict an error location wherever C_w is lower than a threshold τ. Note that while ROVER (Fiscus, 1997) iteratively updates the reference word transition network (WTN) starting from a seed hypothesis, we always use the dropout turned-off D_{off} as the reference hypothesis for alignment. This is because we are interested in the word level confidences for the output (D_{off}) of a single ASR system while ROVER tries to combine outputs of multiple ASR systems to improve the WER.

3.3 Error localization and WER Estimation Experiments

We evaluate our dropout uncertainty-based WER estimation approach on the Switchboard database (Godfrey et al., 1992). A 110h subset *train_100k* of the actual 300h training data is used for training the acoustic models. We tune the hyperparameters K and τ for better WER estimation on an 11.5h subset *dev* taken from the rest of the training data. Finally, we evaluate our approach on a *test* subset that has 8.4h of speech. We ensure that there are no common speakers in our data partitions.

To evaluate the efficacy of our error localization method, we use an “Intersection over Union” (IoU) metric where “Intersection” refers to the intersection between the true errors and the predicted errors and “Union” refers to the set union of the true errors and predicted errors. Note that true errors refer to mismatches between D_{off} and GT as shown in Figure 3.2. IoU lies between 0 and 1, and is highest when the predicted errors exactly match the true errors. It penalizes for both false negatives (when a true error is not detected) and false positives (when a correct word is predicted as an error). In the example presented in Figure 3.2, if $\tau = 0.6$, there are 3 error locations, their intersection with true error locations is 2, and the union is 4. Hence, the IoU will be 0.5. Similarly, if $\tau = 0.4$, the IoU will be 0.33 (intersection=1, union=3). In this work, we use the mean IoU over all the sentences as the indicator of localization performance.

We investigate two different acoustic models, namely DNN-HMM and CTC-based model. We use the Kaldi (Povey et al., 2011) *nnet1* recipe for training a feedforward DNN-HMM ASR system. Kaldi *tri4* setup based on LDA+MLLT+SAT system is used for generating senone alignments (8564 units) and fMLLR transformed MFCC features (1320 dimensional, after appending delta features and context of 11). DNN acoustic model has 6 hidden layers having 2048 neurons each. We set a dropout rate of 0.2 for all hidden layers during training and the same is used during testing.

The phoneme-based CTC model is trained in Pytorch using Baidu's CTC implementation for Deepspeech 2 (Amodei et al., 2016b). It has 4 layers of Bidirectional Long Short-Term Memory (BLSTM), with 320 cells in each layer and direction. We use 40-dimensional log-mel filterbank coefficients as acoustic features together with their first and second-order derivatives. Dropout was applied for all BLSTM layers. The dropout rate was set to 0.2. A trigram LM is used for decoding for both DNN-HMM and CTC models, and no further LM-based rescoring of lattices is done.

To compute the oracle WERs, we use the basic scoring script `compute-wer` provided with Kaldi instead of NIST sclite tool, which involves text normalization. Although the oracle WERs computed this way are usually high, they pose as a more suitable ground truth to compare with the estimated WERs using our approach.

Table 3.1: Hyperparameter values tuned on the *dev* set of the Switchboard dataset.

	Dropout			N-best list			MBR
ASR Model	N	K	τ	N	K	τ	τ
DNN-HMM	100	5	1.0	60	542	0.8	0.9
CTC	24	119	0.9	60	530	0.8	0.9

3.4 Error localization and WER Estimation Results

As a baseline for WER estimation, we replace the N dropout-on hypotheses by the N-best hypotheses of the decoding lattice and use (3.3) to (3.6) to estimate WER.

Similarly, for word error localization baseline, we use the N-best list hypotheses in (3.7) to estimate word-level confidences. Another baseline based on (Xu et al., 2011) evaluates the word-level confidence from the word posterior probabilities computed using forward-backward likelihood computation on a lattice.

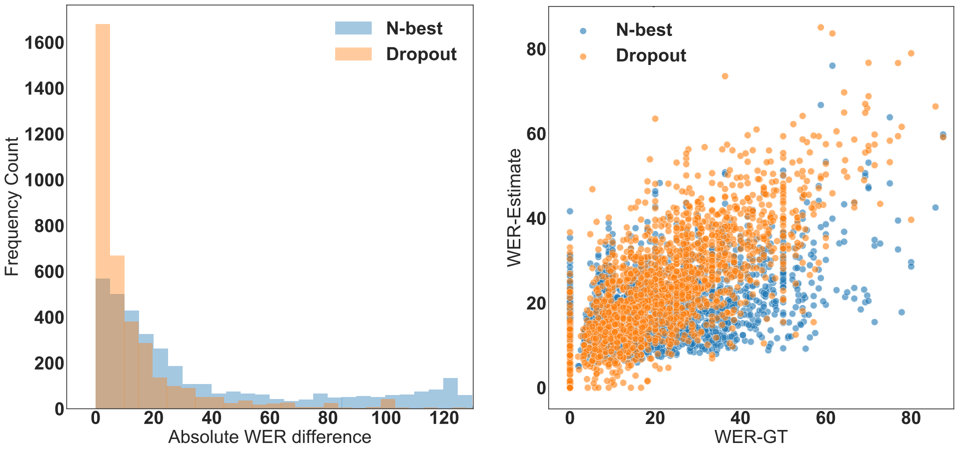
3.4.1 Word Error Rate Estimation

Table 3.2 compares the dropout and N-best list based WER estimation on the Switchboard database. The results are split according to the length of the ground truth transcription for better analysis. We observe that dropout WER estimate is better across all ground truth sentence lengths and on both DNN-HMM and CTC ASR systems. Although the WER estimate using the N-best list over all the sentence lengths (last column) is reasonable, it severely overestimates on shorter sentence and underestimates the WER on longer sentences. One remedy to this could be use a smaller N for shorter sequences and a larger N for longer sequences. However, that would require tuning N for a range of possible sequence lengths. Furthermore, each hypothesis in the N-best list is different, and thus, it cannot give a WER of 0. For small length sentences, this results in overestimation of the WER. For example, if we consider sentences with only one word in the ground truth, the n-best list would still contain all different hypotheses resulting in a WER >= 100. In contrast to this, dropout outputs change only at word locations where the acoustic model is uncertain. Therefore, if the acoustic model's uncertainty is very low along the whole utterance being decoded, then it is possible that each of N hypotheses is identical. As a result, it does not suffer from the problem of overestimating WER.

Table 3.2: Results on estimating WER on the Switchboard *test* set using dropout uncertainty. *Dr* refers to dropout based estimation, *Nb* refers to N-best list based estimation, and *Gt* refers to the Ground truth WER. While the N-best list reasonably estimated the WER over all the sentences (last column), it significantly overestimates the WER on shorter sentence and underestimates on longer sentences. In contrast, dropout-based WER estimation consistently performs well over all sentence lengths.

	Sentence Lengths				
ASR System	[1-3]	[5-6]	[7-10]	>= 11	>= 1
DNN-*test*-Gt	35.5	28.8	25.1	22.3	23.3
DNN-*test*-Dr	33.2	31.0	25.2	21.5	22.7
DNN-*test*-Nb	73.6	42.1	30.2	13.7	19.7
CTC-*test*-Gt	30.6	31.2	25.3	23.3	24.0
CTC-*test*-Dr	34.4	35.5	29.7	23.9	25.2
CTC-*test*-Nb	93.1	49.0	34.3	15.9	22.7

Figure 3.3(a) shows a histogram of the utterance-wise absolute difference between true WER and estimated WER based on dropout and N-best list. For a perfect estimator, the absolute difference for every sentence will be 0. We notice that for the dropout-based estimation, the number of utterances with a very small absolute difference (~0) is much higher than those



(a) Histogram of absolute difference between estimated WER and true WER

(b) Correlation plot between estimated WER and true WER. For Dropout, correlation coefficient, $R = 0.75$ and for N-Best, $R = 0.43$

Figure 3.3: Comparison between dropout and N-best list based WER estimation on the Switchboard *test* set for the CTC system. (a) Histogram of absolute difference between estimated WER and true WER. (b) Correlation between estimated WER and true WER.

for the N-best list based estimation. For a randomly picked utterance, the dropout-based WER estimate is much more likely to be close to the true WER than the N-best list estimation. Figure 3.3(b) shows the correlation between the ground truth-based true WER and estimated WER. Each point on the scatter plot is an utterance. We observe a high correlation for dropout-based estimation (0.75) as compared to the N-best list based estimation (0.43). This is also evident from the dropout scatter plot being more dense in the diagonal region of the plot. As expected from the discussion above, the estimated WER for N-best list is always greater than 0.

3.4.2 Error Localization Using Word Confidences

Table 3.3 shows the performance of word confidence-based ASR error localization in terms of the IoU metric. We compare the proposed dropout approach against N-best list and lattice-based word posterior probability based approaches. We observe that the IoU metric is higher using our dropout method as compared to both of the lattice-based approaches. For DNN-HMM as well as CTC-based ASR systems, dropout achieves an IoU of ~0.6 which depicts that a significant number of error locations are accurately identified. Note that the results in Table 3.3 are averaged over all the utterance lengths. In our experiments, we noticed that IoU metric could be as high as ~0.7-0.8 for very short utterances (≤ 6 words long).

Table 3.3: Error localization performance on the *dev* and *test* parts of the Switchboard dataset. We use the IoU metric as it penalizes both false errors and missed detections. Dropout achieves an IoU of ~ 0.6 indicating that a significant number of errors are correctly identified.

	Approach		
ASR System	Dropout	N-best	Lattice Confidence
DNN-*dev*	0.55	0.45	0.54
DNN-*test*	0.59	0.50	0.58
CTC-*dev*	0.56	0.41	0.51
CTC-*test*	0.61	0.41	0.58

3.5 Uncertainty Aware Semi-supervised Learning

In the previous section, we demonstrated how dropout could effectively capture the ASR model's uncertainties at inference time. In this section, we propose a novel way to exploit the dropout uncertainty to improve the performance of the ASR models in the context of semi-supervised learning with LFMMI criterion.

Semi-supervised learning uses a seed model trained on the transcribed data to decode the unlabelled transcripts. The unlabelled utterances and the decoded transcripts are then used to train the model with LFMMI loss or, with any other sequence discriminative criterion.

In this thesis, instead of using the decoding transcripts directly, we propose to maximize the expected LFMMI objective for unlabelled data by sampling target word sequences from the posterior-predictive distribution over word sequences given an utterance and the seed model. Our proposed loss for semi-supervised training is given as follows:

$$\mathcal{F}_{\text{mmi}} = \max_{\theta} \sum_{u \in \mathcal{D}_U} \log\left(\mathop{\mathrm{E}}_{W_t \sim P(W|O^u, \theta_S)} P(W_t|\mathbf{O}^{(u)}, \theta) \right), \tag{3.8}$$

where $\mathbf{O}^{(u)}$ is the sequence of acoustic observations for utterance u, θ_S refer to the seed model parameters estimated using the supervised training data $\mathcal{D}_S$. $\mathcal{D}_U$ is the unlabelled data. W_t is the target word sequence sampled from the posterior-predictive distribution over words.

In contrast, the regular semi-supervised LFMMI objective (Manohar et al., 2018) is

$$\mathcal{F}_{\text{mmi}} = \max_{\theta} \sum_{u \in \mathcal{D}_U} \log\left(\sum_{W_t \in \mathcal{G}_{\text{num}}^{(u)}} P(W_t|\mathbf{O}^{(u)}, \theta) \right) \tag{3.9}$$

$$\approx \max_{\theta} \sum_{u \in \mathcal{D}_U} \log\left(\mathop{\mathrm{E}}_{W_t \sim \mathcal{G}_{\text{num}}^{(u)}} P(W_t|\mathbf{O}^{(u)}, \theta) \right) \tag{3.10}$$

where $\mathcal{G}_{\text{num}}^{(u)}$ is the decoding lattice for the utterance u generated using the seed model trained using the supervised dataset $\mathcal{D}_S$.

This can be seen an approximation to the proposed loss in (3.8) where the expectation is taken over the word sequences in the decoding lattice and each output word sequence in the lattice is assumed to be equally likely. Using the decoded lattice for supervision improves the performance in comparison to using only the best path. However, the supervision lattice contains the hypotheses corresponding to the Maximum Likelihood estimate of weights and not those sampled from the posterior predictive distribution over word. As a result, the semi-supervised training might be affected / biased by the incorrect hypotheses in the lattice even when the best path has no mistakes.

Given that a fixed beam search is typically used for decoding, we observe that the shorter utterances (with one or two words) are affected more as they contain a large number of potentially incorrect hypotheses even after pruning the lattices. In Figure 3.4a, we show one such example of a decoding lattice containing a number of hypotheses even when the example utterance is quite clean. Although the model is quite confident on the sentence, the decoding lattice still contains many incorrect paths which will deteriorate the supervision quality. Note that the effect can be contained if the correct hypothesis appears more times in the decoding lattice than its competitors. However, as shown later in Table 3.4 we find that combining dropout decodings results in a smaller sentence error rate compared to using lattices.

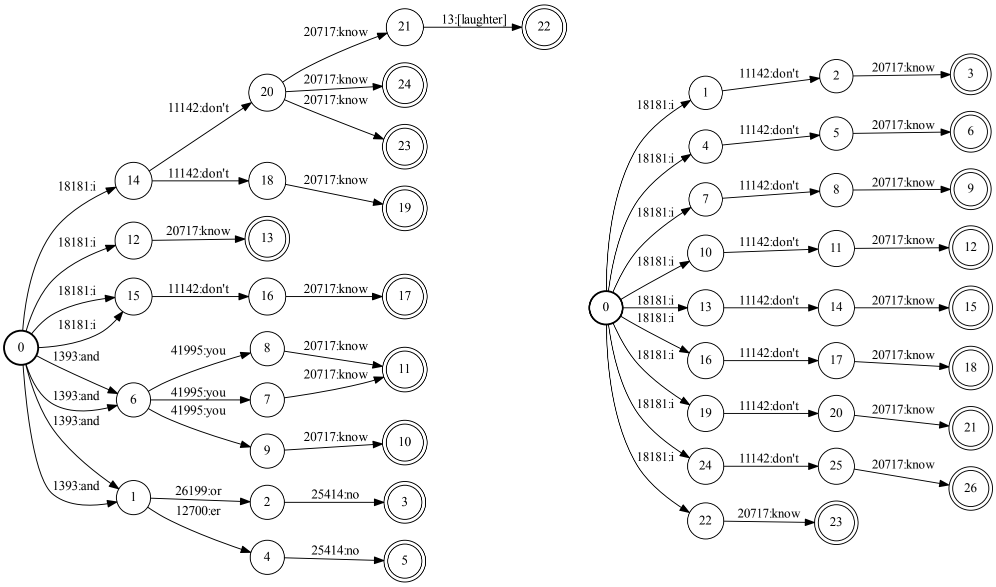
In the following, we discuss how dropout provides a simple way to approximate the loss in (3.8) by sampling from the posterior-predictive distribution.

Sampling from posterior-predictive distribution

As discussed previously, we use dropout at test time to decode the same utterance multiple times. As shown in (Gal and Ghahramani, 2016), this approximates Bayesian inference over the seed model parameters. This allows to sample from the approximate posterior-predictive distribution $P(W|O^{(u)}, \theta_S)$.

We propose to employ dropout during testing to generate N hypotheses that are combined to obtain the supervision lattice. This is motivated by our previous observations that the same hypothesis gets sampled when the acoustic model is confident. Moreover, we also observe local variations in the dropout hypothesis that correlate with errors made by the acoustic model. Decoding with dropout corresponds to sampling from the approximate posterior-predictive distribution over words and leads to an unbiased estimate to (3.8). We investigate dropout sampling for both acoustic and language model.

Dropout Sampling from Acoustic Model: Figure 3.5 presents the different steps of uncertainty aware semi-supervised training with dropout. Given some labelled data $\mathcal{D}_S$ and unlabelled data $\mathcal{D}_U$, we start by training a seed model using only the labelled data. The seed model is then



(a) pruned decoding lattice from the dropout-off acoustic model

(b) unbiased lattice combined from multiple dropout decoding samples

Figure 3.4: Example of lattices for a clearly spoken utterance. (a) represents the conventional decoding lattice generated with dropout-off acoustic model used in (Manohar et al., 2018). (b) denotes the proposed unbiased lattice generated by combining multiple decoding hypothesis with dropout activated at test time.

used to decode each unlabelled utterance N times with dropout activated during inference. We also generate the hypothesis with dropout off as is done traditionally. The $N+1$ hypotheses are combined to obtain the supervision lattice for an unlabelled utterance. Finally, the seed model is trained using both the supervised and supervision lattices for the untranscribed data.

More specifically, we generate an unbiased supervision lattice for each unlabeled utterance by pruning the dropout lattices with a very small beam and composing them together. We keep the rest of the training steps the same as proposed in (Manohar et al., 2018).

Figure 3.4b shows the lattice for the same example utterance, generated using the proposed approach. We see that most paths in this lattice correspond to the correct transcription since the model is confident (high $P(W|\mathbf{O}^{(u)}, \theta_s)$) on this clearly spoken utterance. When the model is uncertain about an utterance, more variations appear in the combined lattice that allows model to take advantage of the alternative paths. We hypothesize that the lattice combined from different dropout-based decoding samples reflects the uncertainty of the acoustic model and is able to foster the more likely word sequences, while keeping variations for uncertain utterances, thus improving the semi-supervised training performance.

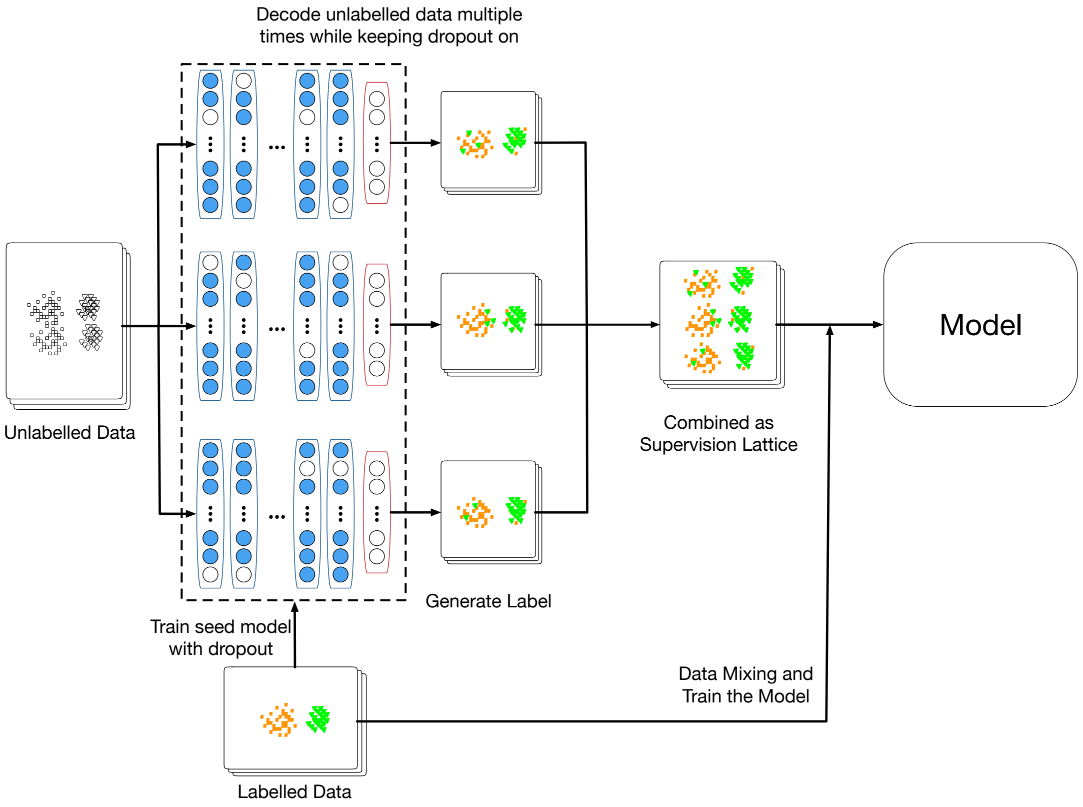

Decode unlabelled data multiple times while keeping dropout on
...
Unlabelled Data
Combined as Supervision Lattice
Model
Generate Label
Train seed model with dropout
Data Mixing and Train the Model
Labelled Data

Figure 3.5: Flow-chart showing the steps for uncertainty aware semi-supervised learning with dropout. We start by training a seed model on the transcribed data. We then decode untranscribed utterances using the seed model with dropout activated. The decoded outputs are combined to obtain the supervision lattices. Each network represents a random selection of the nodes with the white nodes denoting the dropped out units. Finally, the seed model is trained using both the supervised and untranscribed data.

Dropout Sampling from Language Model: The unlabelled utterances are decoded using a WFST based decoder (Mohri et al., 2002) which uses a N-gram language model. As a result of this, incorporating language model uncertainty is non-trivial in comparison to the acoustic model uncertainty.

To incorporate the language model uncertainty, we use the same framework by re-scoring the N-gram based decoded lattice with a DNN-based language model. For each unlabeled utterance, we first obtain the decoding lattice using the acoustic model with dropout off as is done conventionally. We then re-score this lattice N times by using a dropout enabled language model. Similar to the acoustic model case, the re-scored lattices are pruned and combined to generate the supervision lattice reflecting the language model uncertainty. We also investigate the combination of the dropout sampling from both the acoustic and the language model to further improve the performance of the semi-supervised learning models.

3.6 Related Work Discussion

Semi-supervised learning (Zavaliagkos et al., 1998; Wessel and Ney, 2005) uses the seed model trained on transcribed data to automatically generate the transcripts for the unlabelled data. However, the generated transcripts are likely to contain errors as the seed model is trained with limited amount of supervised data. Directly training on erroneous transcriptions can cause the model to further reinforce its own biases and be stuck in local minima as the model parameters are already tuned to have a high likelihood for generated transcripts.

Xu et al. (2020) overcome this limitation by incorporating LM into decoding and using data augmentation to avoid local minima. Furthermore, it was shown in (Wessel and Ney, 2005; Khonglah et al., 2020; Xu et al., 2020) that iterative decoding and training results in better performance than using the entire unsupervised data in one-shot. A complementary technique to iterative training is incorporating the ASR uncertainty or confidence measures to selectively use the unsupervised training data. In prior research, a prominent method for quantifying ASR uncertainty has been computing lattice-based confidence measures. The posterior probability of a recognized word can be estimated from a word lattice (Kemp and Schaaf, 1997; Wessel et al., 1998; Evermann and Woodland, 2000) or a word confusion network (Mangu et al., 1999; Evermann and Woodland, 2000) without any additional training. For semi-supervised learning, confidence measures can be applied at a frame level (Veselý et al., 2013), word level (Wessel and Ney, 2005; Wessel et al., 2001; Thomas et al., 2013; Veselý et al., 2017) or utterance level (Novotney et al., 2009; Grézl and Karafiát, 2013; Zhang et al., 2014).

More recently, Manohar et al. (2018) combine lattice-based supervision with LFMMI objective for semi-supervised training. As opposed to using only the best decoding path for any untranscribed utterance, they use the entire decoding lattice generated by the Kaldi (Povey et al., 2011) decoder based on WFST (Mohri et al., 2002). This allows the model to learn from alternative hypotheses when the best path is not accurate. Although lattice-based supervision significantly improves over the best path training, it can also degrade the performance when the best path hypothesis has much lower WER than the alternate hypotheses. The decoding lattice contains the most competitive hypothesis for the Maximum Likelihood weights and can bias the training towards incorrect hypotheses. In contrast, we propose to use dropout to sample alternate hypothesis from the approximate posterior-predictive distribution over words. The lattice generated with dropout hypotheses implicitly takes into account the word level confidences. More generally, we can also explicitly model word-level confidences by aligning the dropout hypothesis against the reference hypothesis with dropout turned off and counting word agreements similar to ROVER (Fiscus, 1997) but using the same acoustic model.

The proposed approach has similarities to Negative Conditional Entropy (NCE) (Manohar et al., 2015) for semi-supervised training where the authors minimize the expected risk over the uncertain decoding of the unsupervised data. However, in contrast to (Manohar et al., 2015), where the decoding lattice with forward-backward likelihood computation estimates the likelihood of word-sequence, in this work, we directly sample from the approximate posterior-predictive distribution using dropout to generate the supervision lattice. The approach proposed in (Li et al., 2017) also shares some similarities in the sense that the labels of unlabeled data are the decoding output from multiple seed models to incorporate the diversity. An ensemble of models is trained in parallel using these diverse labels, and then averaged as the final model. In the context of our framework, these multiple seed models can be considered as the dropout-based neural network samples and all the diverse labels are combined into one supervision lattice used for LF-MMI training. Thus, the proposed method is simpler and more rigorous.

In this thesis, we present experiments using the LFMMI training criterion. The extension to other end-to-end training is straightforward, for example, please refer to our colleagues work (Dey et al., 2019) on using dropout based semi-supervised learning for attention-based E2E models.

3.7 Semi-supervised Learning Experiments

Like (Manohar et al., 2018), we report our results on the Fisher English corpus (Cieri et al., 2004). A randomly chosen subset of speakers (250 hours) from the corpus is used as unsupervised data. The transcripts from the remaining 1250 hours are used to train the language models for decoding and re-scoring the unsupervised data. We use a 50 hours subset from the corpus as the supervised data to train the seed model. The results are reported on separately held-out development and test sets (about 5 hours each), which are part of the standard Kaldi (Povey et al., 2011) recipe for Fisher English. In addition to the WER, we also report WER Recovery Rate (WRR) (Ma and Schwartz, 2008). WRR measures the WER improvements relative to the oracle model trained using true transcripts for unlabelled data. It is given by

$$\mathrm{WRR} = \frac{\mathrm{BaselineWER} - \mathrm{SemisupWER}}{\mathrm{BaselineWER} - \mathrm{OracleWER}},$$

where the BaselineWER refers to the WER obtained using the seed model trained only using the transcribed data.

Following the standard Kaldi recipe, we first train a TDNN (Waibel et al., 1989) seed model using only the supervised data and the LFMMI criterion. The TDNN model consists of 8 hidden layers, with 450 hidden units in each layer. We apply dropout to each of the layer. We use i-vector (Dehak et al., 2011) for speaker adaptation of the neural network. The i-vector extractor is trained using the combined supervised and unsupervised datasets. The phone LM used for creating the denominator graph is estimated using phone sequences from both supervised and unsupervised data. Following (Manohar et al., 2018), a higher weight is assigned to the phone sequences from supervised data (1.5 for the 50 hours supervised dataset and 1 for the unsupervised data).

We also train a DNN-based language model train on the same data to incorporate the language

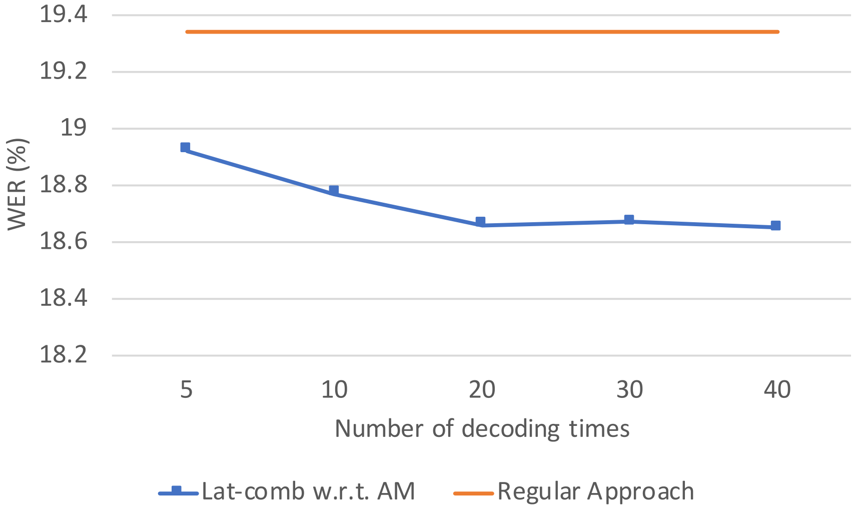


Figure 3.6: WER (%) obtained on the Fisher-English *dev* set for semi-supervised setups trained using a different number of dropout samples, N. The dropout-based sampling is only applied to the acoustic model. The red line denotes the regular semi-supervised training approach (Manohar et al., 2018) which only uses the dropout-off decoded lattice for semi-supervised training.

model uncertainty. This network consists of 3 temporal convolutional layers (Bai et al., 2018), with 600 units in each layer. The size of the word embeddings is fixed to 600 and the kernel size is taken to be 3. Dropout is applied to each of the layer.

3.8 Semi-supervised Learning Results

3.8.1 Effect of Dropout Sample Numbers from Acoustic Model

In this section, we investigate the number of acoustic model dropout hypothesis N needed to represent the posterior-predictive distribution over word sequences. Although increasing N allows for a better representation of the distribution, it is time consuming to decode multiple times. Therefore, it is important to investigate appropriate value of N for a good trade-off. We vary N from 5 to 40 and generate supervision lattices for the unsupervised data. As a baseline, we use the decoding lattice generated from the same acoustic model in a standard way (with dropout off), following (Manohar et al., 2018). We evaluate the performance on the development set without language model re-scoring.

As shown in Figure 3.6, the performance of the proposed method improves as N increases due to the better representation of the posterior-predictive distribution. We do not observe any improvements for N greater than 20. Therefore, we keep using $N = 20$ for the following experiments except when explicitly stated.

Quality Analysis of the Supervision Lattices

From Figure 3.6, we see that the proposed unbiased lattices yield better WER than the regular semi-supervised training that uses the decoding lattices generated with dropout off. We compare the averaged WER and the sentence error rate (SER) of the proposed unbiased lattices ($N = 20$) and the regular decoding lattice to verify our previous hypothesis that unbiased lattice are less affected when the model produces correct outputs. We evaluate the WER of each lattice by averaging the WER of the N-best hypotheses for each untranscribed utterance. The regular decoding lattice was generated from the dropout-off model and was pruned before this evaluation.

Table 3.4: We compare the averaged WER(%) and SER (%) for the combined lattice and regular decoding lattice on the Fisher-English *dev* set. We note that the dropout-based lattice combination achieves much lower sentence error rate (SER). This reduces the effect of incorrect hypotheses when the model is confident and improves the performance of semi-supervised training.

	avg. WER	SER
Regular Lattice	23.6	87.8
Lattice combination	23.1	75.7

Table 3.4 shows that the unbiased lattice has a better WER and a much better SER than the regular lattice. The better WER and SER confirms our hypothesis that the lattice combination from different dropout samples reduces the effect of incorrect hypotheses when the acoustic model is confident on the unlabeled sentence, while keeping alternative paths to be exploited when the acoustic model is uncertain. The much better SER also indicates that dropout hypotheses are particularly useful when the model decodes perfectly thereby providing better supervision quality and improved performance on the development set.

3.8.2 Effect of Number of Dropout Samples from Language Model

Similarly to Section 3.8.1, we analyze here the effect of N with respect to language model. To generate unbiased supervision with respect to the language model, we first decode an utterance in the standard way (keeping dropout off) and obtain the corresponding lattice. The lattice is then re-scored N times using the DNN-based language model while keeping dropout on. We vary N from 5 to 40. As shown in Figure 3.7, the performance on the development set does not change much with different values of N and the proposed approach yields very slight improvement. One of our previous hypotheses was that the Dropout-based Monte Carlo sampling can help reduce the confusion in the supervision lattice especially for shorter sentences. However, language model re-scoring for sentences with one or two words wouldn't make much difference by its nature. We found there are around one-third of the unsupervised

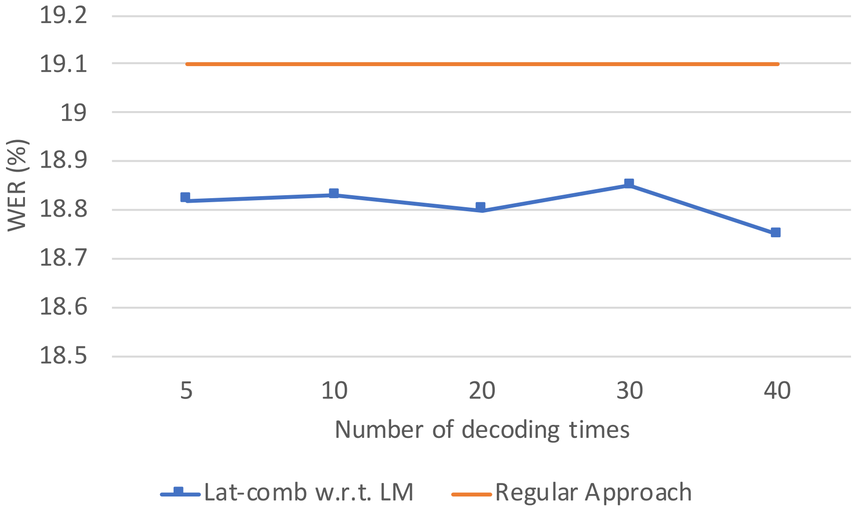


Figure 3.7: WER (%) obtained on the Fisher-English *dev* set for semi-supervised setups trained using a different number of dropout samples, N. The dropout-based sampling is only applied to the language model. The red line denotes the regular semi-supervised training approach (Manohar et al., 2018) where the supervision lattices were also re-scored using DNN LM.

utterances containing only 3 words or less. Furthermore, in terms of total hours this constitutes an even smaller percentage of the total training data. Therefore, applying dropout sampling on the language model only slightly improves the performance.

3.8.3 Complete Comparison

Table 3.5: Comparison between combined lattice and regular decoding lattice in WER(%) obtained on the *dev* and *test* sets of the Fisher-English dataset. The 50h supervised system is used as baseline to calculate WRR.

	System	Dev	Test	WRR
(a)	50h supervised	21.0	20.9	-
(b)	Regular Approach	19.1	19.2	53.7%
(c)	Lattice combination w.r.t. AM	18.5	18.3	76.1%
(d)	Lattice combination w.r.t. LM	18.8	18.7	65.7%
(e)	Lattice combination w.r.t. AM+LM	**18.5**	**18.2**	77.6%
(f)	Oracle	17.7	17.5	

Table 3.5 presents the results of uncertainty aware training on the test set. Row (a) shows the performance of supervised training only using 50 hours of transcribed data. Row (f) shows the results for supervised training using the oracle transcripts for all utterances combining super-

vised and unsupervised data. Rows (b)-(e) present the results for different semi-supervised training configurations. For each configuration, we re-score the supervision lattices for unlabeled data using the DNN language model. For re-scoring the unbiased acoustic lattice in the proposed framework, we first generate the decoding lattice samples by keeping dropout activated for the acoustic model. Then, each decoding lattice is re-scored before pruning and combination. Row (c) presents the results when dropout is applied only for the acoustic model. Similarly, row (d) presents results when dropout is applied only with the language model. In row (e), we explore their combination to verify whether the dropout sampling from both acoustic model and language model can further improve the performance.

As we can see in the Table 3.5, the semi-supervised training approach as proposed in (Manohar et al., 2018) (b) yields around 8.6% relative WER reduction. Incorporated with uncertainty information from only the acoustic model, the unbiased supervision lattice improves over the supervised system by around 12.2%. Dropout sampling from language model also brings improvement, although the improvement is not as much as the one from acoustic model. The combination cannot further improve the performance significantly. Most of the gains come from the acoustic part. In total, the proposed semi-supervised training approach yields approximately 12.4% relative improvement over the supervised setup. Compared with the regular LFMMI semi-supervised training, the proposed approach gives 4.2% relative WER reduction and 51.6% WER recovery rate.

3.9 Conclusions

We have proposed a novel way to exploit dropout uncertainty in the context of semi-supervised training for DNN-based ASR systems. We show that the variations in different decoded hypotheses with dropout are often highly localized at certain word positions and depict locations where the ASR decoding might be inaccurate. Experiments on the Switchboard dataset with 2 different acoustic models show that the dropout uncertainty enables accurate word error rate estimation without the need of ground truth transcripts. We further show that word level confidences estimated from the dropout hypotheses are more robust to the length of the sentences when compared to lattice-based approaches.

We then extend the use of dropout-based uncertainty estimation to semi-supervised learning with LFMMI. We show that the unbiased lattice combined from different dropout decoding samples reduces the incorrect hypotheses for correct decodings, while keeping variations for uncertain unlabeled utterances. Experiments on the Fisher English dataset show that using dropout to generate supervision lattices further improves the WER over the regular semi-supervised training framework. While this thesis primarily focused on LFMMI training, the idea has also been investigated in the context of end-to-end frameworks for semi-supervised learning (Dey et al., 2019).

In this chapter, we conducted experiments using the TDNN and LSTM neural network architectures. We think that dropout-based semi-supervised learning can also help improve the performance of the modern Transformer-based architectures. We leave this experimentation for future work.

4 Robust Self-Supervised Pre-training

4.1 Introduction

In the previous chapter, we looked at semi-supervised training, where a seed model trained on labelled data is used to transcribe unlabelled utterances. These transcribed utterances are then used together with supervised data to improve the ASR system performance. More recently, self-supervised learning methods to learn powerful representations directly from unlabelled data have received much attention (Baevski et al., 2020; Chung et al., 2019; Liu et al., 2021; van den Oord et al., 2018; Wang et al., 2020; Baevski et al., 2022).

In contrast to semi-supervised learning (Chapter 3), self-supervised learning aims to improve the seed model by exploiting unlabelled data before adaptation on supervised data. Semi-supervised learning and self-supervised learning are complimentary as using the self-supervised pre-trained network as the starting point for supervised adaptation improves the seed model, enhancing the quality of transcriptions generated for the unlabelled data (Xu et al., 2021).

Self-supervised training approaches can be broadly grouped into two classes: (1) autoregressive models that try to predict the future representations conditional on the past inputs (van den Oord et al., 2018; Chung et al., 2019) and (2) bidirectional models that learn to predict masked parts of the input (Baevski et al., 2020; Wang et al., 2020; Liu et al., 2021).

Currently, bidirectional models outperform autoregressive self-supervised models for ASR (Baevski et al., 2020; Liu et al., 2021). In (Baevski et al., 2020; Liu et al., 2021), the authors pre-train acoustic models with transformer architecture on 1000 hours of unsupervised Librispeech (Panayotov et al., 2015) data. The pre-trained models are later adapted on a 100 hour supervised subset of Librispeech data to achieve state-of-the-art performance for ASR. However, in both cases, the authors only consider cross-entropy based HMM-DNN systems or Connectionist Temporal Classification (Graves et al., 2006b) for supervised training. Moreover, they evaluate the supervised adaptation when unlabelled and transcribed data belong to the same domains.

In this chapter, we examine the choice of sequence discriminative training criterion and the robustness to domain mismatch for supervised adaptation for models pre-trained using masked acoustic model (MAM) (Liu et al., 2021) and wav2vec 2.0 (W2V2) (Baevski et al., 2020) approaches. In sections 4.1.1 and 4.1.2, we first provide a brief overview of MAM and W2V2 pre-training. Next in sections 4.2 and 4.3, we present the details of the datasets, data preparation, and supervised training for automatic speech recognition. Finally, in Section 4.4, we present the results of supervised adaptation on the Librispeech, Switchboard, and Babel datasets.

We show that for acoustic models pre-trained with MAM, the E2E-LFMMI criterion outperforms the cross-entropy based hybrid HMM-DNN model adaptation used in (Liu et al., 2021). In contrast, fine-tuning the W2V2 model with E2E-LFMMI or CTC criterion yields similar performances with neither consistently better than the other. Our results on the adapting the MAM and W2V2 pre-trained models on out-of-domain conversational speech (Switchboard) and cross-lingual data (Babel) show that both self-supervised pre-training methods provide significant gains over the models trained only with supervised data.

4.1.1 Masked Acoustic Modeling

The masked acoustic model (MAM) is introduced by Liu et al. (2021) for self-supervised pre-training with a bidirectional transformer model. Figure 4.1 provides an overview of MAM pre-training. The input to the network is a sequence of acoustic features with a percentage of the input frames masked or noise corrupted. The model attempts to reconstruct the original input given the corrupted input and is trained with L_1 loss.

Liu et al. (2021) use fMLLR features to pre-train the transformer acoustic model. In contrast, we use 80-dimensional filterbank (FBank) energy features for pre-training. We do not use fMLLR features as they require alignments from a previously trained ASR model for feature extraction. This makes them unsuitable in general unsupervised settings. We pre-train the model on the publicly available Librispeech (Panayotov et al., 2015) dataset that comprises 960 hours of read speech data.

In the following, we briefly describe the input perturbations used for the MAM task, the model architecture, and the training details.

Input Perturbations

As described in (Liu et al., 2021), we apply the following three perturbations to the input acoustic features.

First, we apply time alterations, where we randomly select starting time indexes and then mask 7 consecutive frames. Time alteration blocks can overlap each other resulting in altered blocks with more than 7 frames. The selected frames are set to zero with a probability of 0.8,

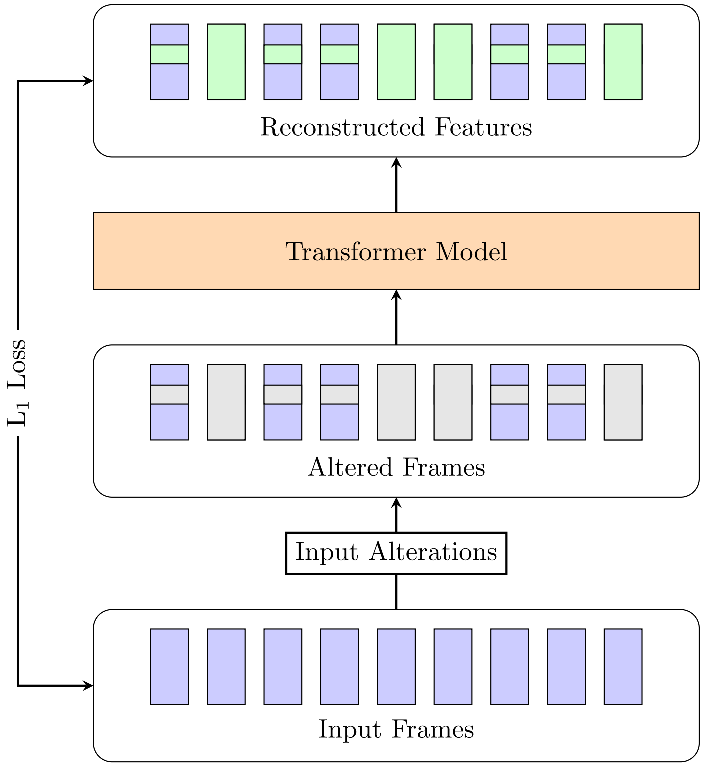


Figure 4.1: Illustrating masked acoustic modeling based self-supervised pre-training approach.

replaced with random segments of frames with a probability of 0.1, or left unaltered with a probability of 0.1. The total number of masked frames is roughly 15% for any input sequence.

We then apply frequency alteration where we randomly mask a block of consecutive channels to zero for all time steps across the input sequence. The number of channels to be masked is selected in the range of 0 to W_c with equal probability. We set W_c to 16 in our experiments.

Finally, we apply magnitude alteration where we add noise to each element of the acoustic features. We sample each element in the noise matrix from a Gaussian distribution with mean zero and variance 0.2. The magnitude alteration is applied with a probability of 0.15.

Model Architecture

We train two transformer architectures that we refer to as MAM-B and MAM-S. The base model (MAM-B) comprises 12 encoder layers, each with 6 attention heads. The embedding dimension is set to 64 for each head and the feed-forward dimension is set to 1536. We use all 960 hours of data to pre-train this model.

The small model (MAM-S) has 3 encoder layers, each with 12 attention heads. We set embedding dimension to 64 for each head and the feed-forward dimension to 3072. We only use the *train-clean-100* subset to pre-train MAM-S.

Training

Both transformer models are trained with the Adam optimizer (Kingma and Ba, 2015) with a mini-batch size of 36 utterances with maximum input feature sequence length of 1500. The learning rate is warmed up over the first 7% of total training steps to a peak value of 0.0002 and then linearly decayed. We train for a total of 200 000 steps. This results in a total of ~ 35 and ~ 353 epochs for the MAM-B and MAM-S models pre-trained on 960 and 100 hour subsets of the Librispeech dataset, respectively.

4.1.2 wav2vec 2.0

wav2vec 2.0 (W2V2) is a self-supervised learning framework introduced by Baevski et al. (2020) to learn powerful representations from raw audio data using contrastive learning (van den Oord et al., 2018; Kawakami et al., 2020).

Figure 4.2 shows different steps for W2V2 self-supervised pre-training. The raw audio speech is first encoded using a multi-layer convolutional neural network. The encoded latent representations are forwarded to a masking module that masks about 49% of the total time steps. Each span of masked features has 10 or more consecutive time steps. The *unmasked* latent representations are also quantized to be later used for the contrastive task.

The masked latent representations are then input to a transformer model to learn contextualized representations c_t by minimizing the contrastive loss $\mathcal{L}_m$. Given a contextualized output c_t for a masked time step, the model needs to identify the true quantized latent representation q_t in a set of $K+1$ candidates $\tilde{q}$ that comprises q_t and the $K = 100$ distractors. The contrastive loss is given below:

$$\mathcal{L}_m = -\log \frac{\exp\left(\text{sim}(c_t, q_t)\right)}{\sum_{\tilde{q}\sim Q_t} \exp\left(\text{sim}(c_t, \tilde{q})\right)}. \tag{4.1}$$

In (4.1), cosine similarity, $\text{sim}(a, b) = \frac{a^T b}{\|a\|\|b\|}$ is used. As shown in Figure 4.2, the distractors are uniformly sampled from other masked time steps of the same utterance.

In this work, we use the *BASE* W2V2 model pre-trained on 1000 hours of Librispeech data. The pre-trained model is provided by Baevski et al. (2020). In the following, we restate the details of model architecture and training for completeness.

Model Architecture

The BASE wav2vec 2.0 model (W2V2-B) consists of a convolutional front end with seven blocks, each with 512 channels. The strides are set to (5,2,2,2,2,2,2) and the kernel widths are set to (10,3,3,3,3,2,2). The convolutional encoder downsamples the raw audio from 16 KHz to an output frequency of 50 Hz. The resulting output has a stride of about 20ms between each sample, and a receptive field of 400 input samples or 25ms of audio.

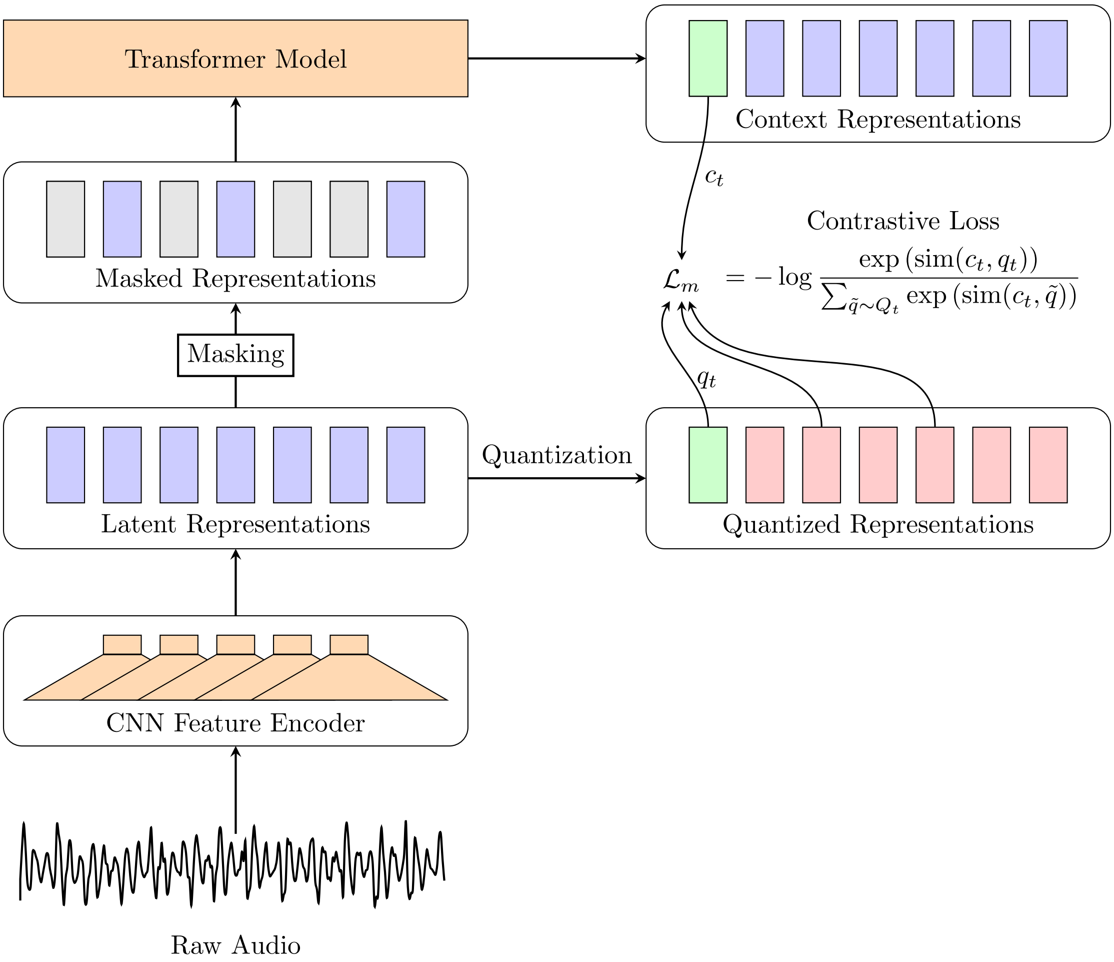


Figure 4.2: Illustrating different steps of wav2vec 2.0 self-supervised pre-training approach. A convolutional front end first transforms raw speech into a sequence of latent features which are then forwarded to a quantization and masking module. The masked representations are fed to a transformer model that learns contextualized representations $\mathbf{c_t}$ by solving a contrastive task. Given $\mathbf{c_t}$ for a masked time step, the model has to identify the true quantized representation $\mathbf{q_t}$ among a set of candidates that includes $\mathbf{q_t}$ and some distractors.

The transformer encoder consists of 12 layers, each with 8 attention heads. The embedding dimension is set to 96 for each head and the feed-forward dimension is set to 3072. Another convolutional layer with a kernel size of 128 and 16 groups is used for modeling the relative positional embeddings.

Training

The model is trained for 400K updates with Adam optimizer (Kingma and Ba, 2015) using a learning rate schedule with peak learning rate set to 5×10^{-4}. The learning rate is linearly increased for the first 8% of updates followed by a linear decay. Each input batch consists of about 1.6h of raw audio inputs, with each audio cropped to a maximum of 15.6 sec.

4.2 Acoustic Model Adaptation

This section provides details for fine-tuning the MAM and W2V2 models pre-trained on the 960h of Librispeech dataset.

4.2.1 Model Training

Given an input utterance, we input the filter bank (FBank) features to the MAM model or raw audio to the W2V2 model. The output of the transformer encoder of the pre-trained models is passed as input to a seven-layered factorized time-delay neural network (TDNNF) which is fine-tuned model together with pre-trained model weights using E2E-LFMMI and/or CTC criteria. For TDNNF models, we set hidden layer dimension to 1024 and bottleneck dimension to 128.

For supervised training with E2E-LFMMI, we use full biphones to enable flat-start training without the need of prior alignments. Our CTC model is trained using character units. We apply speed and volume perturbations to increase the dataset to three times.

All our models are trained with PyTorch (Paszke et al., 2019). For fine-tuning with CTC, we use the Fairseq toolkit (Ott et al., 2019), and for E2E-LFMMI, we use the Espresso toolkit (Wang et al., 2019b) which uses PyChain (Shao et al., 2020) for the implementation of LFMMI loss. We use the PyTorch implementation for natural gradient update from (Madikeri et al., 2020). We describe MAM and W2V2 specific training details below:

Masked acoustic model: We fine-tune the pre-trained models 15 epochs with a batch size of 32 utterances. We use Adam (Kingma and Ba, 2015) optimizer with a learning rate that is decayed from 0.001 to 0.00003 using a polynomial decay. When fine-tuning the pre-trained model, we set the learning rate for the pre-trained network to be 0.00003 and use the same learning rate policy for the TDNNF network. We fine-tune the MAM models only using the E2E-LFMMI criterion. We also compare against the cross-entropy based adaptation to as *hybrid* in later experiments.

wav2vec 2.0: Following the setup from (Baevski et al., 2020), we train for a maximum of 30000 and 75000 updates for E2E-LFMMI and CTC criterion, respectively, where each update is over $\sim$ 1500 seconds of speech input. In terms of total epochs, E2E-LFMMI and CTC models are trained for 38 and 88 epochs on Librispeech, 10 and 27 epochs on Switchboard, 100 and 205 epochs on Swahili, 57 and 112 epochs on Tagalog, respectively.

Table 4.1 presents the total number of trainable parameters for different acoustic models. For both E2E-LFMMI and CTC, we update the BASE model parameters with a learning rate that is linearly increased to 3e-5 over 10% of the updates, then held constant for 40% of the updates, and linearly decreased for 50% of the updates. For TDNNF model parameters, we use a learning rate that is 20 times the current learning rate for BASE model updates. We also use the natural gradient update (Povey et al., 2015) for training with the E2E-LFMMI objective.

Table 4.1: Comparing the number of parameters for different choices of acoustic models.

	Architecture	Parameters
(a)	TDNNF-L	12M
(b)	TDNNF-S	9M
(c)	MAM-S	22M
(d)	MAM-B	36M
(e)	W2V2-B	95M

4.2.2 Decoding

For all models trained with E2E-LFMMI, we use the WFST decoder from (Povey et al., 2011) with a beam width of 15. For the models trained with CTC, we use the decoder from (Pratap et al., 2019) with a beam width of 500. We always use the language model from Kaldi recipes (Povey et al., 2011) which are trained with SRILM (Stolcke, 2002). We found this to give better results than KenLM (Heafield, 2011).

4.3 Experimental Setup

We evaluate the W2V2 and MAM pre-trained models by fine-tuning them on three different datasets, each with a few hundred hours of transcribed audios. We fine-tune the pre-trained model weights together with a seven layered factorized time-delay neural network (TDNNF) architecture which we refer to as TDNNF-S. We also consider the setting where the pre-trained model weights are frozen for MAM models, and the model is used as a feature extractor. In this setting, the extracted features are input to a twelve layered TDNNF model (TDNNF-L) whose weights are fine-tuned on the supervised dataset. The results for this are provided in the Appendix A.1.

4.3.1 Datasets

We evaluate the performance of the pre-trained models on three different datasets with increasing order of difficulties. The first dataset we consider is the 100 hour subset of Librispeech (Panayotov et al., 2015) called *train-clean-100*. This is the easiest setting as there is no domain shift with respect to the pre-training data. We next consider the Switchboard (Godfrey et al., 1992) dataset with 300 hours of supervision data. Both Switchboard and Librispeech (pre-training data) consist of utterances spoken in English. However, Switchboard has conversational speech recorded at 8KHz while Librispeech is read speech recorded at 16KHz. We finally fine-tune on two of the Babel (Gales et al., 2014) languages: Tagalog (84h) and Swahili (38.5h). This is the most challenging setting due to the language and acoustic conditions mismatch.

4.3.2 Baselines

We compare the performance of pre-trained MAM and W2V2 models against the baseline that is trained only using the supervised data. Our baseline is a twelve-layered TDNNF model with the hidden layer dimension of 1024 and the bottleneck dimension of 128. It is trained with E2E-LFMMI objective using 80-dimensional filterbank (FBank) features.

4.4 Results

In the following, we compare the Word Error Rate (WER) achieved by supervised adaptation of pre-trained models to the models trained from scratch. The number of hours denotes the transcribed data for adaptation. Unless specified, we use the pre-trained model trained on 960 hours of Librispeech data.

4.4.1 Librispeech (100h)

In this experiment, we discuss the setting when the pre-trained data and labelled data for ASR come from the same domain. For all experiments, we report results using the model that achieves lowest WER on the *dev-clean* set. We first decode using a trigram language model and then rescore using a 4-gram language model.

Comparing E2E-LFMMI and Cross-Entropy Criteria

Table 4.2 compares the effect of training criterion used for adaptation of the model pre-trained with MAM. From rows (a) and (b), it can be seen that using LFMMI for adaptation outperforms hybrid models when FBANK features are used for self-supervised pre-training. Moreover, compared to training from scratch using only the labelled data (a), using MAM-S (b) performs slightly worse on the clean data. This is because MAM-S was pre-trained using the same 100 hour subset which is used for training (a). Thus both models are exposed to exactly the same amount of data during training. Interestingly (b) performs better than (a) on the other portion of the test data. We think that these gains are a result of noise robustness due to the perturbations added during pre-training.

Rows (d) and (e) compare the LFMMI and cross-entropy based adaptation for the case of base models (MAM-B) pre-trained with 960h data. Note that the MAM-B model used by Liu et al.

---

[1] pre-trained with fMLLR features

Table 4.2: Comparing the performance of MAM pre-trained model fine-tuning with E2E-LFMMI against cross entropy based hybrid model. All models are fine-tuned on the 100 hour subset of the Librispeech dataset. We report the results on the clean and other portions of the test set. Fine-tuning the pre-trained model with LFMMI (E2E-LFMMI) significantly outperforms the cross-entropy based hybrid model training. Note that the base (MAM-B) model in (Liu et al., 2021) are pre-trained with fMLLR features. For a fair comparison, we also report the result for adaptation on MAM-S models pre-trained with FBANK features. For MAM-S, the pre-trained models are used as feature extractors to compare against the results reported for hybrid model. TDNNF-L and TDNNF-S refer to the 12 and 7 layered TDNNF architectures. liGRU refers to light Gated Recurrent Unit. E2E-LFMMI refers to flat-start training with LFMMI and hybrid refers to cross-entropy based HMM-DNN models.

			Word Error Rate			
			3-gram		**4-gram**	
	Architecture	Supervision	clean	other	clean	other
		Supervised Only				
(a)	TDNNF-L	e2e-lfmmi	8.6	26.3	5.9	20.0
		Pre-training + Supervised				
(b)	MAM-S + TDNNF-L	e2e-lfmmi	8.9	25.3	6.1	18.8
(c)	MAM-S + liGRU (Liu et al., 2021)	hybrid	11.8	-	9.4	-
(d)	MAM-B + TDNNF-S	e2e-lfmmi	7.8	-	5.3	-
(e)	MAM-B + liGRU [1] (Liu et al., 2021)	hybrid	8.2	-	5.8	-

(2021) is pre-trained using fMLLR features. We find that even in this case, the LFMMI criterion outperforms the cross-entropy adaptation. Furthermore, comparing (c) and (e), we observe that the cross-entropy based fine-tuning is sensitive to the pre-training features. In contrast, fine-tuning with LFMMI leads to much less performance difference.

Main Results

We present our main results in Table 4.3. We observe that fine-tuning either of the W2V2 and MAM models pre-trained with 960 hours of untranscribed data results in better performance than the baseline trained with only supervised data (a). Furthermore, we can see that W2V2 pre-training gives significantly better WER compared to the MAM pre-trained model. However, please note that the W2V2 model is pre-trained for twice as many steps with larger batch sizes.

In rows (d) and (e), we compare the performance of fine-tuning the wav2vec 2.0 BASE model with E2E-LFMMI and CTC loss. It can be seen that both models reach a similar level of performance, providing ~ 12.7% and ~ 11.5% absolute WER improvements over the supervised *TDNNF* baseline on the noisy portion of the test set. Note that we did not apply any additional regularization or changes to train with the CTC loss. In contrast, comparing rows

(a) and (b) on the *dev* set, it can be seen that CTC training requires additional regularization and modifications to the deep neural network training to reach a similar level of performance.

From tables 4.2 and 4.3, we find that LFMMI training improves performance for fine-tuning models pre-trained with MAM. However, fine-tuning W2V2 models with either CTC or LFMMI give a similar level of performance. We also find that while fine-tuning the pre-trained models gives the most improvements, using the pre-trained model as a feature extractor also outperforms the baseline trained only on supervised data (please see Appendix A.1.1).

Table 4.3: Comparison of word error rates (WER) (in %) on the clean and other parts of the Librispeech test set with and without 4-gram language model rescoring. Fine-tuning the pre-trained models significantly outperforms the baseline trained using only supervised data. MAM-S, MAM-B refers to small, and base transformer models used for pre-training. TDNNF-S and TDNNF-L refer to the 7 and 12 layered TDNNF architectures used for fine-tuning. e2e-lfmmi refers to flat-start training with LFMMI and ctc refers to Connectionist Temporal Classification criterion.

			Word Error Rate				
			dev	**test**			
			3-gram	**3-gram**		**4-gram**	
	Architecture	Supervision	clean	clean	other	clean	other
			Supervised Only				
(a)	TDNNF-L	e2e-lfmmi	8.3	8.6	26.3	5.9	20.0
(b)	Bi-LSTM (Billa, 2017)	ctc	11.1	-	-	-	-
	+ max perturbation	ctc	9.8	-	-	-	-
	+ cascade dropout	ctc	7.9	8.7	26.1	-	-
			Pre-training + Supervised				
(c)	MAM-B + TDNNF-S	e2e-lfmmi	-	7.8	20.2	5.3	14.7
(d)	W2V2-B + TDNNF-S	e2e-lfmmi	-	**4.4**	**8.9**	3.5	**7.3**
(e)	W2V2-B + TDNNF-S	ctc	-	-	-	**3.3**	8.5

4.4.2 Switchboard (300h)

In this experiment, we explore the case when the pre-training data and labelled data for ASR belong to the same language but are different with respect to content, speakers, and acoustic conditions. Switchboard has conversational speech recorded at 8 KHz while Librispeech has read speech at 16 KHz. To be compatible with the pre-trained models, we resample the Switchboard recordings at 16 KHz before extracting the features. For the *TDNNF-L* baseline trained only with labelled data, we use the 8 KHz recordings.

Table 4.4 compares the WER for the models trained from scratch to those pre-trained on Librispeech data. In both cases, the pre-trained models outperform the model trained from scratch. Once again, the fine-tuning W2V2 model with CTC or LFMMI loss gives a similar performance. Similar to the case of Librispeech, we again find that while using the pre-trained model as the feature extractor improves upon the baseline model, fine-tuning provides the most gains (please see Appendix A.1.2).

Table 4.4: Comparison of word error rates (WER) (in %) on eval2000 test set portion of the Switchboard dataset. The 3-gram language model is based on Switchboard, whereas the 4-gram employs Switchboard+Fisher training set transcripts. Fine-tuning either of the pre-trained W2V2 or MAM model outperforms the baseline. Fine-tuning W2V2 model with E2E-LFMMI or CTC significantly outperforms MAM model.

			Word Error Rate			
			3-gram		**4-gram**	
	Architecture	Criterion	swbd	ch	swbd	ch
		Supervised Only				
(a)	TDNNF-L	e2e-lfmmi	11.6	22.5	10.3	20.5
(b)	TDNN-LSTM (Hadian et al., 2018)	e2e-lfmmi	11.3	21.5	9.8	19.3
(c)	Bi-LSTM (Audhkhasi et al., 2019)	ctc	-	-	12.2	21.8
		Pre-training + Supervised				
(d)	MAM-B + TDNNF-S	e2e-lfmmi	10.9	20.4	9.4	18.2
(e)	W2V2-B + TDNNF-S	e2e-lfmmi	**7.3**	**14.5**	6.7	13.7
(f)	W2V2-B + TDNNF-S	ctc	-	-	**6.6**	**13.2**

4.4.3 Babel: Swahili (38h) and Tagalog (84h)

In our final experiment, we consider the scenario when the pre-training data and labelled data for ASR do not share the same language or the acoustic conditions. We consider Swahili and Tagalog which are two low-resource languages from the Babel database. Similar to the Switchboard setup, we resample the recordings at 16 KHz to be compatible with the pre-trained model. Once again, for the *TDNNF-L* model trained from scratch only on the supervised data, we use the 8 KHz recordings.

Due to the lack of a separate evaluation set, we report results on the *dev10h* development part of both languages. We remove 1000 utterances from the training set to be used as the development set for model selection. We use trigram language model for decoding.

Section 4.4.3 compares the WER for the models trained from scratch to those that make use of Librispeech pre-training data. Consistent with the previous results, we find that fine-tuning pre-trained models outperform the supervised only baseline. We also find that for both Swahili and Tagalog, W2V2 model fine-tuned with E2E-LFMMI and CTC obtained similar performance. Both models also outperform the model pre-trained with MAM.

Table 4.5: Comparison of word error rates (WER) (in %) on dev10h set for the Swahili and Tagalog languages of the Babel dataset. Fine-tuning the pre-trained wav2vec 2.0 BASE model significantly outperforms the monolingual and MAM baselines. Note that while we use SRILM, XLSR-10 model uses KenLM for decoding and does not use speed or volume perturbation.

			Word Error Rate	
	Model	Criterion	Swahili	Tagalog
	Supervised Only			
(a)	TDNNF	e2e-lfmmi	39.5	44.9
(b)	BLSTM-HMM (Inaguma et al., 2019)	hybrid	38.3	46.3
	Pre-training + Supervised			
(c)	XLSR-10 (Large) (Conneau et al., 2021)	ctc	35.5	37.3
(d)	MAM-B + TDNNF-S	e2e-lfmmi	36.7	43.4
(e)	W2V2-B + TDNNF-S	e2e-lfmmi	**29.4**	**36.9**
(f)	W2V2-B + TDNNF-S	ctc	30.4	37.3

We additionally report the WER for the XLSR-10 model from Conneau et al. (2021). This is a large wav2vec 2.0 multilingual model pre-trained on 10 languages. As can be seen, we get a much better word error rate on Swahili and a comparable performance on Tagalog. We think that the results might not be directly comparable because we use speed and volume perturbation for data augmentation and do not score on non-language symbols. Additionally, XLSR-10 uses KenLM for decoding while we use SRILM. In our experiments, we noticed a significant degradation in WER using KenLM. Despite these differences, it is clear from our results that wav2vec 2.0 BASE model pre-trained on Librispeech still offers a very competitive baseline to the large multilingual model.

In contrast to the results on Switchboard and Librispeech datasets, we find that using pre-trained model as a feature extractor performs worse than the models trained from scratch. This indicates the representations learned by the pre-trained model on Librispeech data removes some vital information specific to these languages resulting in worse performance than the baseline. However, on fine-tuning, the model adjusts its parameters to recapture this information and outperforms the baseline. The full results can be found in Appendix A.1.3.

4.5 Conclusions

This chapter investigates the effects of the sequence discriminative training criteria and out-of-domain robustness for the supervised adaptation of pre-trained wav2vec 2.0 (W2V2) and MAM models. We show that the LFMMI criterion performs better for fine-tuning the MAM model. In contrast, fine-tuning the W2V2 model with either E2E-LFMMI or CTC gives a similar performance with no additional regularization needed for CTC training. We further show that fine-tuning MAM or W2V2 model outperforms models trained with only supervised data even under strong distributional shifts.

In the future, we intend to combine self-supervision based approaches with multi-lingual training and iterative decoding based semi-supervised training approaches to further improve the performance in low resource settings.

5 Efficient Attention Models

5.1 Introduction

Sequence modelling is a fundamental task of machine learning, integral in a variety of applications such as neural machine translation (Bahdanau et al., 2015b), image captioning (Xu et al., 2015), summarization (Maybury, 1999), automatic speech recognition (Dong et al., 2018) and synthesis (van den Oord et al., 2016), etc. The Transformer architecture (Vaswani et al., 2017) has been proven a powerful tool, significantly advancing the state-of-the-art for most aforementioned tasks. In particular, transformers employ self-attention to handle long sequences without the vanishing-gradient problem inherent in RNNs (Hochreiter et al., 2001; Arjovsky et al., 2016).

Nonetheless, despite their impressive performance, self-attention comes with computational and memory requirements that scale quadratic to the sequence length, limiting their applicability to long sequences. The quadratic complexity becomes apparent if we consider the core mechanism of self-attention, namely splitting the input sequence into queries and keys and then each query attending to all keys. To this end, recently, there has been an increasing interest in developing methods that address this limitation (Dai et al., 2019; Sukhbaatar et al., 2019; Child et al., 2019; Kitaev et al., 2020).

These methods can be broadly categorized into two distinct lines of work. Those that focus on improving the asymptotic complexity of the self-attention computation (Child et al., 2019; Lee et al., 2019; Kitaev et al., 2020; Roy et al., 2021) and those that aim at developing techniques that make transformers applicable to longer sequences without addressing the quadratic complexity of self-attention (Dai et al., 2019; Sukhbaatar et al., 2019). The former reduces the asymptotic complexity by limiting the number of keys that each query attends. The latter increases the length of the sequence that a transformer can attend to without altering the underlying complexity of the self-attention mechanism.

This thesis proposes two efficient attention alternatives to vanilla self-attention that scale linearly with sequence length. Section 5.4 introduces the *linear transformer* model that uses a kernel-based formulation of self-attention and the associative property of matrix products to calculate the self-attention weights (Section 5.4.1). Using our linear formulation, we also express causal masking with linear complexity and constant memory (Section 5.4.2). This formulation reveals the relation between transformers and RNNs, which enables us to perform autoregressive inference orders of magnitude faster (Section 5.4.3).

Next, Section 5.5 presents *clustered attention*, a fast approximation of self-attention. Clustered attention makes use of similarities between queries and groups them in order to reduce the computational cost. In particular, we perform fast clustering using locality-sensitive hashing and K-Means and only compute the attention once per cluster. This results in linear complexity for a fixed number of clusters (Section 5.5.1). In addition, we showcase that we can further improve the quality of our approximation by separately considering the keys with the highest attention per cluster (Section 5.5.2). Finally, we provide theoretical bounds of our approximation quality with respect to the full attention (Section 5.5.1 and Appendix B.2.2) and show that our model can be applied for inference of pre-trained transformers with minimal loss in performance.

We evaluate our models on automatic speech recognition and image generation tasks to showcase that the proposed efficient attention can reach the performance levels of vanilla transformer while being significantly faster during training and inference. Moreover, we demonstrate that our proposed clustered attention can approximate pre-trained wav2vec 2.0 (W2V2), BERT, and BigGAN models on the popular speech recognition, natural language, and image generation tasks with only a few clusters and without loss in performance.

5.2 Related Work

In this section, we discuss the most relevant works on scaling transformers to long sequences. We start by presenting approaches that looked at speeding up the attention computation before the Transformer architecture was introduced. Subsequently, we discuss approaches that speed up transformers without changing self-attention computation complexity. Finally, we summarize the related works on improving the asymptotic complexity of the attention layer in transformer models.

5.2.1 Attention Improvements Before Transformers

Attention has been an integral component of neural networks for sequence modelling for several years (Bahdanau et al., 2015b; Xu et al., 2015; Chan et al., 2016). However, its quadratic complexity with respect to the sequence length hinders its applicability on large sequences.

Among the first attempts to address this was the work of Britz et al. (2017) that proposes aggregating the information of the input sequence into fewer vectors and performing attention with these fewer vectors, thus speeding up the attention computation and reducing the memory requirements. However, the input aggregation is performed using a learned but fixed matrix that remains constant for all sequences, significantly limiting the model's expressivity. Similarly, Chiu and Raffel (2018) limit the number of accessible elements to the attention by attending monotonically from the past to the future. Namely, if timestep i attends to position j then timestep $i+1$ cannot attend to any of the earlier positions. Note that in order to speed up the attention computation, the above methods are limiting the number of elements that each layer attends to. Recently, some of these approaches have also been applied in the context of transformers (Ma et al., 2020).

5.2.2 Non-asymptotic Improvements

Below, we summarize techniques that seek to apply transformers to long sequences without focusing on improving the quadratic complexity of self-attention. The most important are Adaptive Attention Span Transformers (Sukhbaatar et al., 2019) and Transformer-XL (Dai et al., 2019).

In (Sukhbaatar et al., 2019), it was proposed to limit the self-attention context to the closest samples (attention span), in terms of relative distance with respect to the time step, thus reducing both the time and memory requirements of self-attention computation. This is achieved using a masking function with learnable parameters that allows the network to increase the attention span if necessary. Transformer-XL (Dai et al., 2019), on the other hand, seeks to increase the effective sequence length by introducing segment-level recurrent training, namely splitting the input into segments and attending jointly to the previous and the current segment. The above, combined with a new relative positional encoding results in models that attend to more distant positions than the length of the segment used during training.

Although both approaches have been proven effective, the underlying limitations of self-attention still remain. Attending to an element that is N timesteps away requires $\mathcal{O}\left(N^2\right)$ memory and computation. In contrast, our model trades off a small error in the computation of the full attention for an improved *linear* asymptotic complexity. This makes processing long sequences possible.

5.2.3 Improvements in Asymptotic Complexity

Sparse Transformer (Child et al., 2019) factorizes the self-attention mechanism into local and strided attention. The local attention is computed between the C nearest positions and the strided attention is computed between positions that are C steps away from each other. When C is set to $\sqrt{N}$ the total asymptotic complexity becomes $\mathcal{O}\left(N\sqrt{N}\right)$ both in terms of memory and computation time. The above factorization requires two self-attention layers for any position to attend to any other position. In addition, the factorization is fixed and data independent. This makes it intuitive for certain signals (e.g. images), however in most cases it is arbitrary.

More recently, Kitaev et al. (2020) proposed Reformer. This method further reduces complexity to $\mathcal{O}\left(N\log N\right)$ by using locality-sensitive hashing (LSH) to perform fewer dot products. Note that to be able to use LSH, Reformer constrains the keys to be identical to the queries. As a result, this method cannot be used for tasks where the keys need to be different from the queries.

In contrast to Sparse Transformer and Reformer, the proposed linear and clustered attention variants impose no constraints on the queries and keys and scale linearly with respect to the sequence length. Furthermore, *clustered* attention automatically groups the input queries that are similar without the need for a manually designed factorization or constraints. Finally, for our proposed models information flows always from every position to every other position.

Most related to the *clustered transformers* is Set Transformers (Lee et al., 2019) which reduce the computational complexity of attention by introducing a set of trainable parameters I with M vectors, called inducing points. Set Transformers compute attention between the input sequence X, of length N and the M inducing points I to get a new sequence H, of length $M << N$. The new sequence H is then used to compute the attention with X to get the output representation. For a fixed M, the asymptotic complexity becomes linear with respect to the sequence length. Inducing points are expected to encode some global structure that is task-specific. However, this introduces additional model parameters for each attention layer. In contrast to this, we use clustering to project the input to a fixed sequence of smaller length without any increase in the number of parameters. Moreover, we show that not only our method has the same asymptotic complexity, it can also be used with pre-trained models without additional training.

Most related to the *linear transformers* are the works that try to linearize softmax. Softmax has been the bottleneck for training classification models with a large number of categories (Goodman, 2001; Morin and Bengio, 2005; Mnih and Hinton, 2008). Recent works (Blanc and Rendle, 2018; Rawat et al., 2019), have approximated softmax with a linear dot product of feature maps to speed up the training through sampling. Inspired by these works, we linearize the softmax attention in transformers. Concurrently with this work, Shen et al. (2021) explored the use of linearized attention for the task of object detection in images. In comparison, we not only linearize the attention computation, but also develop an autoregressive transformer model with linear complexity and constant memory for both inference and training. Moreover, we show that every transformer can be seen as a recurrent neural network through the lens of kernels.

In the following, we first recap the Transformer model introduced in Section 2.5.3. We then present the proposed efficient attention variants with linear complexity of attention computation with respect to the sequence length.

5.3 Transformers

The transformer function (Vaswani et al., 2017) maps an input sequence $x \in \mathbb{R}^{N \times F}$ with N feature vectors of dimensions F to another sequence with the same number of features. More formally, a transformer is a function $T : \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times F}$ defined by the composition of L transformer layers $T_1(\cdot), \ldots, T_L(\cdot)$ as follows,

$$\begin{aligned} y &= \mathrm{LN}(\mathrm{SA}_l(x) + x), \\ T_l(x) &= \mathrm{LN}(\mathrm{FFN}(y) + y). \end{aligned} \tag{5.1}$$

In (5.1), LN(·) refers to the *layer normalization* function , FFN(·) refers to the *feedforward network* that transforms each feature independently of the others and is usually implemented with a small two-layer multilayer perceptron. $\mathrm{SA}_l(\cdot)$ is the self attention function and is the only part of the transformer that acts across sequences.

The self attention function $\mathrm{SA}_l(\cdot)$ computes, for every position, a weighted average of the feature representations of all other positions with a weight proportional to a similarity score between the representations. Formally, the input sequence x is projected by three matrices $W_Q \in \mathbb{R}^{F \times D}$, $W_K \in \mathbb{R}^{F \times D}$ and $W_V \in \mathbb{R}^{F \times M}$ to corresponding representations Q, K and V. Given these, we define the attention matrix $A \in \mathbb{R}^{N \times N}$ as,

$$Q = xW_Q, \tag{5.2}$$

$$K = xW_K, \tag{5.3}$$

$$V = xW_V, \tag{5.4}$$

$$A = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{D}}\right), \tag{5.5}$$

where $Q \in \mathbb{R}^{N \times D}$ denotes the *queries* and $K \in \mathbb{R}^{N \times D}$ denotes the *keys*. Note that softmax(·) is applied row-wise. Using the attention weights A and the *values* $V \in \mathbb{R}^{N \times M}$, we compute the new values $\mathrm{SA}_l(x) = \hat{V}$ as follows,

$$\mathrm{SA}_l(x) = \hat{V} = AV. \tag{5.6}$$

Computation Complexity of Self Attention: From (5.5) it is evident the computing attention weights requires $\mathcal{O}\left(N^2 D\right)$ operations and the weighted average of (5.6) requires $\mathcal{O}\left(N^2 M\right)$. This results in an asymptotic complexity of $\mathcal{O}\left(N^2 D + N^2 M\right)$.

5.4 Linear Transformers

In this section, we formalize our proposed *linear transformer*. We present that changing the attention from the traditional *softmax* attention to a feature map based dot product attention results in better time and memory complexity as well as a causal model that can perform sequence generation in linear time, similar to a recurrent neural network.

Initially, in Section 5.4.1 we present our proposed *linear transformers* in the setting of non-autoregressive models. Subsequently, in Section 5.4.2 we extend *linear* attention to the case of auto-regressive/causal models. Finally, in Section 5.4.3 we rewrite the transformer as a recurrent neural network.

5.4.1 Linearized Attention

The self attention in (5.6) implements a specific form of self-attention called softmax attention where the similarity score is the exponential of the dot product between a query and a key. Given that subscripting a matrix with i returns the i-th row as a vector, we can write a generalized attention equation for any similarity function as follows,

$$\hat{V}_i = \frac{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{N} \text{sim}\left(Q_i, K_j\right)}. \tag{5.7}$$

The self-attention in (5.7) is equivalent to (5.6) if we substitute the similarity function with $\text{sim}\left(q, k\right) = \exp\left(\frac{q^T k}{\sqrt{D}}\right)$.

The definition of attention in (5.7) is generic and can be used to define several other attention implementations such as polynomial attention or RBF kernel attention (Tsai et al., 2019). Note that the only constraint we need to impose to $\text{sim}\left(\cdot\right)$, in order for (5.7) to define an attention function, is to be non-negative. This includes all kernels $k(x, y) : \mathbb{R}^{2\times F} \to \mathbb{R}_+$.

Given such a kernel with a feature representation $\phi\left(x\right)$ we can rewrite the output value computation in (5.6) as follows,

$$\hat{V}_i = \frac{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right) V_j}{\sum_{j=1}^{N} \phi\left(Q_i\right)^T \phi\left(K_j\right)}, \tag{5.8}$$

and then further simplify it by making use of the associative property of matrix multiplication to

$$\hat{V}_i = \frac{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right) V_j^T}{\phi\left(Q_i\right)^T \sum_{j=1}^{N} \phi\left(K_j\right)}. \tag{5.9}$$

The above equation is simpler to follow when the numerator is written in vectorized form as follows,

$$\left(\phi\left(Q\right) \phi\left(K\right)^T\right) V = \phi\left(Q\right) \left(\phi\left(K\right)^T V\right). \tag{5.10}$$

Note that the feature map $\phi\left(\cdot\right)$ is applied rowwise to the matrices Q and K.

From (5.5), it is evident that the computational and memory requirements of softmax attention scales with $\mathcal{O}\left(N^2\right)$, where N represents the sequence length. In contrast, it can be seen from Figure 5.1 and (5.9) that our proposed *linear transformer* has time and memory complexity

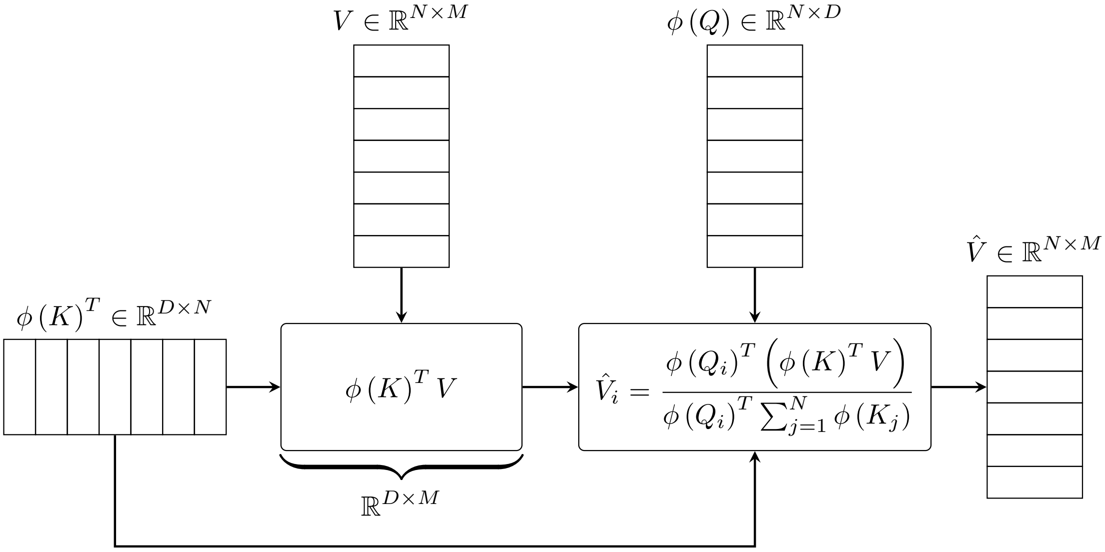

$V \in \mathbb{R}^{N\times M}$
$\phi(Q) \in \mathbb{R}^{N\times D}$
$\hat{V} \in \mathbb{R}^{N\times M}$
$\phi(K)^T \in \mathbb{R}^{D\times N}$
$\phi(K)^T V$
$\hat{V}_i = \frac{\phi(Q_i)^T \left(\phi(K)^T V\right)}{\phi(Q_i)^T \sum_{j=1}^{N} \phi(K_j)}$
$\mathbb{R}^{D\times M}$

Figure 5.1: Flow-chart demonstrating the computation for *linear* attention. Given the queries Q and key K, we first apply the kernel feature function $\phi()$ to obtain $\phi(Q)$ and $\phi(K)$. We then exploit the matrix product associativity property to change the order of computation with linear complexity and memory requirements.

$\mathcal{O}(N)$ because we can compute $\sum_{j=1}^{N} \phi\left(K_j\right) V_j^T$ and $\sum_{j=1}^{N} \phi\left(K_j\right)$ once and reuse them for every query.

Feature Maps and Computational Cost

For softmax attention, the total cost in terms of multiplications and additions scales as $\mathcal{O}\left(N^2 \max(D, M)\right)$, where D is the dimensionality of the queries and keys and M is the dimensionality of the values. On the contrary, for linear attention, we first compute the feature maps of dimensionality C. Subsequently, computing the new values requires $\mathcal{O}(NCM)$ additions and multiplications.

The previous analysis does not take into account the choice of kernel and feature function. Note that the feature function that corresponds to the exponential kernel is infinite dimensional, which makes the linearization of exact softmax attention infeasible. On the other hand, the polynomial kernel, for example, has an exact finite dimensional feature map and has been shown to work equally well with the exponential or RBF kernel (Tsai et al., 2019). The computational cost for a linearized polynomial transformer of degree 2 is $\mathcal{O}\left(ND^2M\right)$. This makes the computational complexity favorable when $N > D^2$. Note that this is true in practice since we want to be able to process sequences with tens of thousands of elements.

For our experiments, that deal with smaller sequences, we employ a feature map that results in a positive similarity function as defined below,

$$\phi(x) = \text{elu}(x) + 1, \tag{5.11}$$

where elu(·) denotes the exponential linear unit (Clevert et al., 2016) activation function. We prefer elu(·) over relu(·) to avoid setting the gradients to 0 when x is negative. This feature map results in an attention function that requires $\mathcal{O}(NDM)$ multiplications and additions. In our experimental section, we show that the feature map of (5.11) performs on par to the full transformer, while significantly reducing the computational and memory requirements.

5.4.2 Causal Masking

The transformer architecture can be used to efficiently train autoregressive models by masking the attention computation such that the i-th position can only be influenced by a position j if and only if $j \leq i$, namely a position cannot be influenced by the subsequent positions. Formally, this causal masking changes (5.7) as follows,

$$\hat{V}_i = \frac{\sum_{j=1}^{i} \text{sim}\left(Q_i, K_j\right) V_j}{\sum_{j=1}^{i} \text{sim}\left(Q_i, K_j\right)}. \tag{5.12}$$

Following the reasoning of Section 5.4.1, we linearize the masked attention as described below,

$$\hat{V}_i = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi\left(K_j\right)}. \tag{5.13}$$

By introducing S_i and Z_i as follows,

$$S_i = \sum_{j=1}^{i} \phi\left(K_j\right) V_j^T, \tag{5.14}$$

$$Z_i = \sum_{j=1}^{i} \phi\left(K_j\right), \tag{5.15}$$

we can simplify (5.13) to

$$\hat{V}_i = \frac{\phi(Q_i)^T S_i}{\phi(Q_i)^T Z_i}. \tag{5.16}$$

Note that, S_i and Z_i can be computed from S_{i-1} and Z_{i-1} in constant time hence making the computational complexity of linear transformers with causal masking linear with respect to the sequence length.

In the following, we discuss the gradient computation and the difference in behaviour during training and inference for autoregressive transformers.

Gradient Computation

A naive implementation of (5.16), in any deep learning framework, requires storing all intermediate values S_i in order to compute the gradients. This increases the memory consumption by $\max(D, M)$ times; thus hindering the applicability of causal linear attention to longer sequences or deeper models. To address this, we derive the gradients of the numerator in (5.13) as cumulative sums. This allows us to compute both the forward and backward pass of causal linear attention in **linear time** and **constant memory**. A detailed derivation is provided in the supplementary material.

Given the numerator $\bar{V}_i$ and the gradient of a scalar loss function with respect to the numerator $\nabla_{\bar{V}_i}\mathcal{L}$, we derive $\nabla_{\phi(Q_i)}\mathcal{L}$, $\nabla_{\phi(K_i)}\mathcal{L}$ and $\nabla_{V_i}\mathcal{L}$ as follows,

$$\nabla_{\phi(Q_i)}\mathcal{L} = \nabla_{\bar{V}_i}\mathcal{L}\left(\sum_{j=1}^{i}\phi\left(K_j\right)V_j^T\right)^T, \tag{5.17}$$

$$\nabla_{\phi(K_i)}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)V_i\,, \tag{5.18}$$

$$\nabla_{V_i}\mathcal{L} = \left(\sum_{j=i}^{N}\phi\left(Q_j\right)\left(\nabla_{\bar{V}_j}\mathcal{L}\right)^T\right)^T\phi\left(K_i\right). \tag{5.19}$$

The cumulative sum terms in (5.13), (5.17) to (5.19) are computed in linear time and require constant memory with respect to the sequence length. This results in an algorithm with computational complexity $\mathcal{O}(NCM)$ and memory $\mathcal{O}(N\max(C, M))$ for a given feature map of C dimensions. A pseudocode implementation of the forward and backward pass of the numerator is given in algorithms 1 and 2 respectively in the Appendix B.1.1.

Training and Inference

When training an autoregressive transformer model the full ground truth sequence is available. This makes layerwise parallelism possible for the attention, layer normalization, and the feed-forward computation of (5.1). As a result, transformers are more efficient to train than recurrent neural networks. On the other hand, during inference the output for timestep i is the input for timestep $i+1$. This makes autoregressive models impossible to parallelize. Moreover, the cost per timestep for transformers is not constant; instead, it scales with the square of the current sequence length because attention must be computed for all previous timesteps.

Our proposed *linear transformer* model *combines the best of both worlds.* When it comes to training, the computations can be parallelized and take full advantage of GPUs or other accelerators. When it comes to inference, the cost per time and memory for one prediction is constant for our model. This means we can simply store the $\phi\left(K_j\right)V_j^T$ matrix as an internal state and update it at every time step like a recurrent neural network. This results in inference **thousands of times faster** than other transformer models.

5.4.3 Transformers are RNNs

In literature, transformer models are considered to be a fundamentally different approach to recurrent neural networks. However, from the causal masking formulation in Section 5.4.2 and the discussion in the previous section, it becomes evident that any transformer layer with causal masking can be written as a model that, given an input, modifies an internal state and then predicts an output, namely a Recurrent Neural Network (RNN). Note that, in contrast to Universal Transformers (Dehghani et al., 2019), we consider the recurrence with respect to time and not depth.

In the following equations, we formalize the transformer layer of (5.1) as a recurrent neural network. The resulting RNN has two hidden states, namely the attention memory s and the normalizer memory z. We use subscripts to denote the timestep in the recurrence.

$$s_0 = 0, \tag{5.20}$$

$$z_0 = 0, \tag{5.21}$$

$$s_i = s_{i-1} + \phi\left(x_i W_K\right)\left(x_i W_V\right)^T, \tag{5.22}$$

$$z_i = z_{i-1} + \phi\left(x_i W_K\right), \tag{5.23}$$

$$y_i = f_l\left(\frac{\phi\left(x_i W_Q\right)^T s_i}{\phi\left(x_i W_Q\right)^T z_i} + x_i\right). \tag{5.24}$$

In the above equations, x_i denotes the i-th input and y_i the i-th output for a specific transformer layer. Note that our formulation does not impose any constraint on the feature function and it can be used for representing *any transformer* model, in theory even the ones using softmax attention. This formulation is a first step towards better understanding the relationship between transformers and popular recurrent networks (Hochreiter and Schmidhuber, 1997) and the processes used for storing and retrieving information.

5.5 Clustered Transformers

This section introduces the clustered attention variants for approximating the vanilla softmax attention. In Section 5.5.1, we first show that for queries close in the Euclidean space, the attention difference can be bounded by the distance between the queries. *Clustered attention* exploits this property to reduce the computational complexity by clustering the queries and computing the attention using only the query centroids. We then introduce *improved-clustered* attention (Section 5.5.2) that improves the approximation by first extracting the top-k keys with the highest attention per cluster and then computing the attention on these keys separately for each query that belongs to the cluster.

5.5.1 Clustered Attention

The core idea of clustered attention is to group queries into C clusters and use only the cluster centroids for attention computation

More formally, let us define $S \in \{0,1\}^{N \times C}$, a partitioning of the queries Q into C non-overlapping clusters, such that, $S_{ij} = 1$, if the i-th query Q_i belongs to the j-th cluster and 0 otherwise. Using this partitioning, we can now compute the *clustered attention*. First, we compute the cluster centroids as follows,

$$Q_j^c = \frac{\sum_{i=1}^{N} S_{ij} Q_i}{\sum_{i=1}^{N} S_{ij}}, \tag{5.25}$$

where Q_j^c is the centroid of the j-th cluster. Let us denote $Q^c \in \mathbb{R}^{C \times D}$ as the centroid matrix. Now, we can compute the clustered attention as if Q^c were the queries. Namely, we compute the clustered attention matrix $A^c \in \mathbb{R}^{C \times N}$

$$A^c = \text{softmax}\left(\frac{Q^c K^T}{\sqrt{D}}\right) \tag{5.26}$$

and the new values $\hat{V}^c \in \mathbb{R}^{C \times M}$

$$\hat{V}^c = A^c V. \tag{5.27}$$

Finally, the value of the i-th query becomes the value of its closest centroid, namely,

$$\hat{V}_i = \sum_{j=1}^{C} S_{ij} \hat{V}_j^c. \tag{5.28}$$

In Figure 5.2 we show the *clustered* attention computation for an example sequence with 8 queries and using 3 clusters. Figure 5.2 and (5.26) shows that we use the same attention weights for queries that belong to the same cluster. As a result, the attention computation now becomes $\mathcal{O}(NCD)$, where $C \ll N$. Furthermore, we only need to compute the attention weights and the weighted average of the values *once per cluster*. Then, we can broadcast the same value to all queries belonging to the same cluster. This allows us to reduce the number of dot products from N for each query to C for each cluster, which results in an asymptotic complexity of $\mathcal{O}(NCD) + \mathcal{O}(CNM)$.

Note that in practice, we use multi-head attention, this means that two queries belonging to the same cluster can be clustered differently in another attention head. Moreover, residual connection in the output of the attention layer can cause two queries belonging to the same cluster to have different output representations. The combined effect of residual connections and multi-head attention allows new clustering patterns in subsequent layers. Further note that the clustered attention groups queries from different time steps to compute the centroids for attention computation. This makes it unsuitable for auto-regressive modeling where the i-th output value can only be influenced by a position j if and only if $j \leq i$.

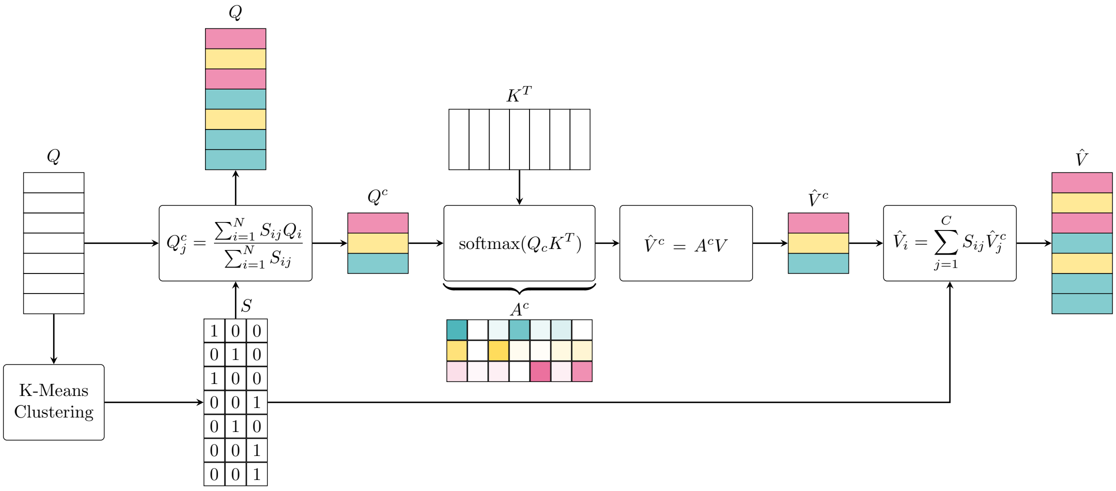


Figure 5.2: Flow-chart demonstrating the computation for *clustered* attention. We use different colors to represent the query groups and the computed centroids. The same colors are then used to show the attention weights A^c, new values for the centroids $\hat{V}^c$, and the resulting values $\hat{V}$ after broadcasting.

Quality of the approximation

From the above, we show that grouping queries into clusters can speed up the self-attention computation. However, in the previous analysis, we do not consider the effects of clustering on the attention weights A. To address this, we derive a bound for the approximation error. In particular, we show that the difference in attention can be bounded as a function of the Euclidean distance between the queries.

Proposition 1. *Given two queries* Q_i *and* Q_j *such that* $\left\|Q_i - Q_j\right\|_2 \leq \epsilon$*,*

$$\left\| \mathit{softmax}\left(Q_i K^T\right) - \mathit{softmax}\left(Q_j K^T\right) \right\|_2 \leq \epsilon \left\|K\right\|_2 , \tag{5.29}$$

where $\|K\|_2$ *denotes the spectral norm of* K*.*

Proof. Given that softmax$(\cdot)$ has Lipschitz constant less than 1 (Gao and Pavel, 2017),

$$\begin{aligned} &\left\| \text{softmax}\left(Q_i K^T\right) - \text{softmax}\left(Q_j K^T\right) \right\|_2 \\ &\leq \left\| Q_i K^T - Q_j K^T \right\|_2 \\ &\leq \epsilon \left\|K\right\|_2 \end{aligned} \tag{5.30}$$

□

Theorem 1 shows that queries that are close in Euclidean space have similar attention distributions. As a result, the error in the attention approximation for the i-th query assigned to the j-th cluster can be bounded by its distance from the cluster centroid Q_j^c.

Grouping the Queries

We have shown that given a representative set of queries, we can approximate the attention with fewer computations. K-Means clustering minimizes the sum of squared distances between the cluster members and would be an ideal choice to find the representative set of queries given our analysis from theorem 1. However, for a sequence of length N one iteration of Lloyd's algorithm for the K-Means has an asymptotic complexity $\mathcal{O}(NCD)$. To speed up the distance computations, we propose to use *Locality-Sensitive Hashing* (LSH) on the queries and then K-Means in Hamming space. We use the sign of random projections (Shrivastava and Li, 2014) to hash the queries followed by K-Means clustering with hamming distance as the metric. This results in an asymptotic complexity of $\mathcal{O}(NCL + CBL + NDB)$, where L is the number of Lloyd iterations and B is the number of bits used for hashing.

5.5.2 Improving clustered attention

In the previous section, we show that clustered attention provides a fast approximation for softmax attention. In this section, we discuss how this approximation can be further improved by considering separately the keys with the highest attention. To intuitively understand the importance of the above, it suffices to consider a scenario where a key with low attention for some query gets a high attention as approximated with the cluster centroid. This can happen when the number of clusters is too low or due to the convergence failure of K-Means. For the clustered attention, described in Section 5.5.1, this introduces a significant error in the computed value. The variation discussed below addresses such limitations.

After having computed the clustered attention A^c from (5.26), we find the k keys with the highest attention for each cluster. The main idea then is to improve the attention approximation on these top-k keys for each query that belongs to the cluster. To do so, we first compute the dot product attention as defined in (5.5) on these top-k keys for all queries belonging to this cluster. For any query, the computed attention on these top-k keys will sum up to one. This means that it cannot be directly used to substitute the clustered attention on these keys. To address this, before substitution, we scale the computed attention by the total probability mass assigned by the clustered attention to these top-k keys.

More formally, we start by introducing $T \in \{0,1\}^{C \times N}$, where $T_{ji} = 1$ if the i-th key is among the top-k keys for the j-th cluster and 0 otherwise. We can then compute the probability mass, let it be $\hat{m}_j$, of the top-k keys for the j-th cluster, as follows

$$\hat{m}_j = \sum_{i=1}^{N} T_{ji} A^c_{ji}. \tag{5.31}$$

Now we formulate an improved attention matrix approximation $A^t \in \mathbb{R}^{N\times N}$ as follows

$$A^t_{il} = \begin{cases} \frac{\hat{m}_j \exp(Q_i K_l^T)}{\sum_{r=1}^{N} T_{jr} \exp(Q_i K_r^T)} & \text{if } T_{jl} = 1 \\ A^c_{jl} & \text{otherwise} \end{cases}. \tag{5.32}$$

Note that in the above, i denotes the i-th query belonging to the j-th cluster and $\sqrt{D}$ is omitted for clarity. In particular, (5.32) selects the clustered attention of (5.26) for keys that are not among the top-k keys for a given cluster. For the rest, it redistributes the mass $\hat{m}_j$ according to the dot product attention of the queries with the top-k keys. The corresponding new values, $\hat{V} \in \mathbb{R}^{N\times D_v}$, are a simple matrix product of A^t with the values,

$$\hat{V} = A^t V. \tag{5.33}$$

We can decompose (5.33) into clustered attention computation and two sparse dot products, one for every query with the top-k keys and one for the top-k attention weights with the corresponding values. This adds $\mathcal{O}(Nk \max(D, M))$ to the asymptotic complexity of the attention approximation of (5.26). A graphical illustration of the improved clustered attention is provided in the Appendix B.2.1.

Quality of the approximation

In the following, we provide proof that improved clustered attention in (5.32) is a direct improvement over the clustered attention in (5.26), in terms of the L_1 distance from the attention matrix A.

Proposition 2. *For the i-th query belonging to the j-th cluster, the improved clustered attention A^t_i and clustered attention A^c_j relate to the full attention A_i as follows,*

$$\left\| A^t_i - A_i \right\|_1 \leq \left\| A^c_j - A_i \right\|_1 \tag{5.34}$$

The proof of the above proposition is presented in Appendix B.2.2 in the appendix. From (5.34) it becomes evident that improved clustered attention will always approximate the full attention better compared to clustered attention.

5.6 Automatic Speech Recognition Experiments

In this section, we analyze experimentally the performance of the proposed *linear* and *clustered* transformers on non-autoregressive Automatic speech recognition (ASR) task in Section 5.6. We evaluate different transformer models on two datasets, the Wall Street Journal dataset (Section 5.6.1) and the Switchboard dataset (Section 5.6.2).

We compare our model with the vanilla transformers (Vaswani et al., 2017), which we refer to as **softmax** and the Reformer (Kitaev et al., 2020), which we refer to as **lsh-X**, where X denotes the rounds of hashing. The *linear attention* introduced in Section 5.4.1 is referred to as **linear**. For training the *linear transformers*, we use the feature map of (5.11). We refer to *clustered attention*, introduced in Section 5.5.1, as **clustered-X** and to *improved clustered attention*, introduced in Section 5.5.2, as **i-clustered-X**, where X denotes the number of clusters. Unless mentioned otherwise we use $k = 32$ for the top-k keys with improved clustered attention.

For Reformer we use a PyTorch port of the published code. We do not use reversible layers since it is a technique that could be applied to all methods. All experiments are conducted using NVidia GTX 1080 Ti with 11GB of memory and all models are implemented in PyTorch (Paszke et al., 2019). Our PyTorch code for *linear* and *clustered* attention can be found at https://linear-transformers.com/ and https://clustered-transformers.github.io, respectively.

5.6.1 Evaluation on Wall Street Journal (WSJ)

In this experiment, we use the 80 hour WSJ dataset (Paul and Baker, 1992) with 40-dimensional mel-scale filterbanks without temporal differences as features to the transformer models. We train using Connectionist Temporal Classification (CTC) (Graves et al., 2006b) loss with phonemes as ground-truth labels. The approximate average and maximum sequence lengths for the training inputs are 780 and 2500, respectively.

For this task, we additionally compare with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with 3 layers of hidden size 320. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3} which is reduced when the validation error stops decreasing. For the transformer models, we use 9 layers with 6 heads and embedding dimension of 32 for each head. As an optimizer, we use RAdam with an initial learning rate of 10^{-4} that is divided by 2 when the validation error stops decreasing.

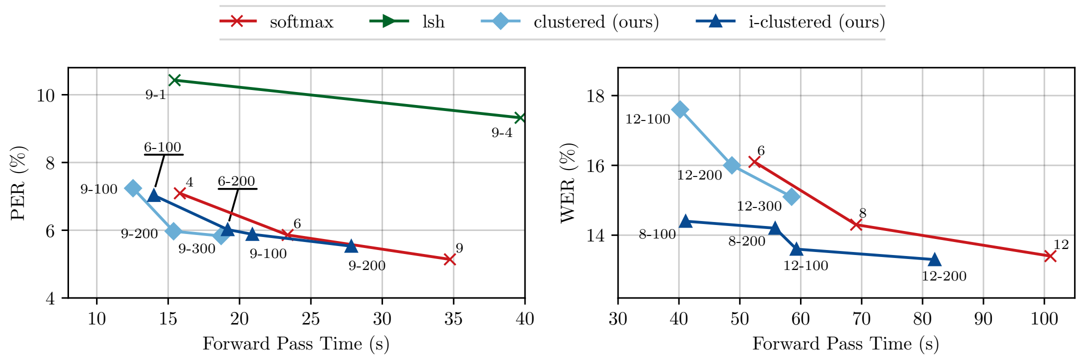
Table 5.1: Performance comparison in automatic speech recognition on the WSJ dataset. The results are given in the form of phoneme error rate (PER) and training time per epoch. Our models outperform the LSTM and Reformer while being faster to train and evaluate. Details of the experiment can be found in Section 5.6.1.

Method	Test PER	Time/epoch (s)	Training time (h)	Epochs
Bi-LSTM	10.94	1047	15.0	50
Softmax	5.12	2514	88.0	127
LSH-1	9.43	1004	189.7	680
LSH-4	8.59	2350	210.1	321
Linear (ours)	8.01	**754**	145.1	692
clustered-100 (ours)	7.50	803	102.2	458
i-clustered-100 (ours)	5.61	1325	**72.1**	195

Convergence Behaviour: In Table 5.1, we report the required phoneme error rate (PER), time per epoch, and total training time for all transformer variants with 9 layers. We observe that linear and clustered attention are about 3× faster than softmax (per epoch) and achieve lower PER than both Reformer variants (lsh-1 and lsh-4) and the recurrent network baseline. We also notice that improved clustered is the only method that is not only faster per epoch but also in total wall-clock time required to converge. We provide training evolution plots in the supplementary.

Speed Accuracy Trade-off: We start by comparing the performance of our proposed model with various transformer variants under an equalized computational budget. To this end, we train *softmax* with 4, 6 and 9 layers to get a range of the required computation time and achieved *phone error rate* (PER). Similarly, we train *i-clustered* with 6 and 9 layers. Both models are trained with 100 and 200 clusters. We also train *clustered* with 9 layers, and 100, 200 and 300 clusters. Finally, we train Reformer with 9 layers, and 1 and 4 hashing rounds. We refer the reader to Appendix B.3.5 for the specifics of all transformer architectures as well as their training details.

In Figure 5.3a, we plot the achieved PER on the validation set with respect to the required time to perform a full forward pass. Our *i-clustered* achieves lower PER than all other baselines for a given computational budget.



(a) Wall Street Journal

(b) Switchboard

Figure 5.3: We compare the achieved performance of various transformer models under an equalized computational budget. The numbers near the datapoints denote the number of layers and number of clusters or hashing rounds where applicable. i-clustered is consistently better than all baselines for a given computational budget both in WSJ and Switchboard datasets. The details can be found in Section 5.6.1 and Section 5.6.2 respectively.

Approximation Quality: To assess the approximation capabilities of our proposed *clustered* attention, we train different transformer variants on the aforementioned task and evaluate them using other self-attention implementations during inference. As the Reformer requires the queries to be identical to the keys to evaluate its approximation ability we also train a softmax attention model with shared queries and keys, which we refer to as **shared-softmax**. Note that both clustered attention and improved clustered attention can be used for approximating shared-softmax, simply by setting keys to be equal to queries. Table 5.2 summarizes the results. Improved clustered attention (7-8 rows) achieves the lowest phone error rate in every comparison. This implies that it is the best choice for approximating pre-trained models. In addition, we also note that the approximation improves as we increase the number of clusters.

Table 5.2: We report the validation phone error rate (PER) on the WSJ dataset (Section 5.6.1). We train with one model and evaluate with another to assess the approximation abilities of different models. Underline denotes training and testing with the same model. Improved cluster (rows 7-8) approximates the softmax and the shared-softmax significantly better than all the other fast attention methods.

		Train with					
		softmax	shared-softmax	lsh-1	lsh-4	clustered-100	i-clustered-100
Evaluate with	softmax	5.14	-	-	-	7.10	5.56
	shared-softmax	-	6.57	25.16	41.61	-	-
	lsh-1	-	71.40	10.43	13.76	-	-
	lsh-4	-	64.29	9.35	9.33	-	-
	clustered-100	44.88	40.86	68.06	66.43	7.06	18.83
	clustered-200	21.76	25.86	57.75	57.24	6.34	8.95
	i-clustered-100	9.29	13.22	41.65	48.20	8.80	5.95
	i-clustered-200	6.38	8.43	30.09	42.43	7.71	5.60
	oracle-top	17.16	77.18	43.35	59.38	24.32	6.96

Furthermore, to show that the top keys alone are not sufficient for approximating *softmax*, we also compare with an attention variant, that for each query only keeps the 32 keys with the highest attention. We refer to the latter as **oracle-top**. We observe that oracle-top achieves significantly larger phone error rate than improved clustered in all cases. This implies that improved clustered attention also captures the significant long tail of the attention distribution.

5.6.2 Evaluation on Switchboard

From the previous ASR evaluation experiment, we observe that only improved clustered attention converges faster than vanilla softmax attention with minimal degradation in performance. In this experiment, we evaluate the *clustered* attention variants on more difficult Switchboard dataset (Godfrey et al., 1992), which is a collection of 2,400 telephone conversations on common topics among 543 strangers. We additionally report results for the twelve-layered factorized time-delay neural network (TDNNF) model from Section 4.4.2. All transformers are trained with lattice-free MMI loss (Povey et al., 2016) and as inputs we use 80-dimensional filter-bank features with fixed positional embeddings. The average input sequence length is roughly 534 and the maximum sequence length is approximately 3850. Details regarding the transformer architectures as well as their training details are provided in Appendix B.3.5.

Convergence Behaviour: Table 5.3 summarizes the computational cost of training the transformer models with 12 layers on the Switchboard dataset as well as the WER on the test set. We observe that the vanilla softmax attention gives the smallest average WER. However, it is the slowest to train. In contrast, improved clustered attention is only ~ 3.3% relative worse while being about ~ 1.8× faster to train. Due to the larger sequences in this dataset both clustered and i-clustered are faster to train per epoch and with respect to the total required wall-clock time.

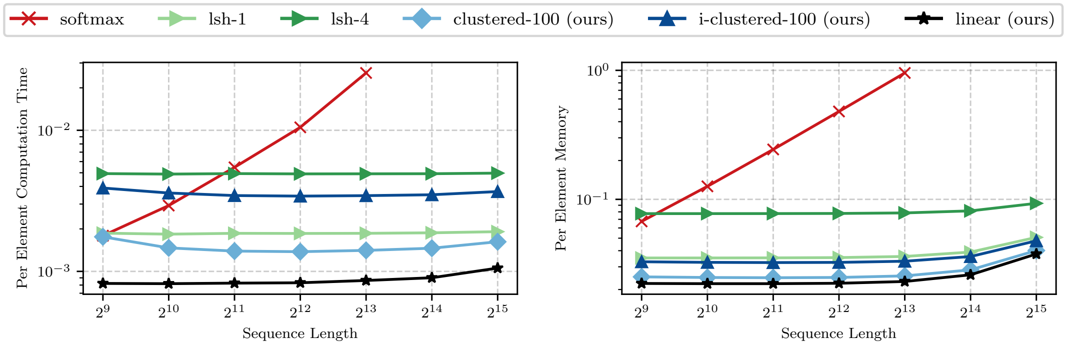
Table 5.3: Performance comparison in automatic speech recognition on the eval2000 test set portion of the Switchboard dataset. The results are given in the form of word error rate (wer) (WER) and training time per epoch. We use a 4-gram language model that is trained on the Switchboard+Fisher training set transcripts. improved clustered attention model converges 2× faster than the vanilla softmax attention while being only 0.5% worse in WER. We additionally report results for the 12 layered TDNNF model from Section 4.4.2. More details of the experiment can be found in Section 5.6.2.

	Word Error Rate				
	swbd	ch	average	Time/epoch (h)	Training time (h)
TDNNF-L Section 4.4.2	10.3	20.5	15.4	-	-
Softmax	10.3	19.6	15.0	3.84	228.1
clustered-100 (ours)	13.7	23.3	18.5	1.91	132.1
i-clustered-100 (ours)	10.7	20.3	15.5	2.57	127.4

Speed Accuracy Trade-off: Similar to Section 5.6.1, we compare the performance of various transformer models given a specific computational budget. To this end, we train *softmax* with 6, 8 and 12 layers. Similarly, we train *i-clustered* with 8 and 12 layers; both with 100 and 200 clusters. Finally, we also train *clustered* with 12 layers, and 100, 200 and 300 clusters. In Figure 5.3b, we plot the achieved word error rate (WER) in the validation set of Switchboard with respect to the required time to perform a full forward pass. Our *i-clustered* is consistently better than *softmax* for a given computational budget. In particular, for a budget of approximately 50 seconds, improved clustered achieves more than 2 percentage points lower WER. Furthermore, we note that it is consistently better than clustered attention for all computational budgets.

5.7 Experiments on Other Modalities

In this section, we analyze experimentally the performance of the proposed *linear* and *clustered* transformers. First, in Section 5.7.1, we benchmark different self-attention models in terms of computational cost and memory consumption on synthetic data. We then evaluate



(a) Per Element Time

(b) Per Element Memory

Figure 5.4: Per element GPU time and memory consumption for a forward/backward pass. All models, except softmax, scale linearly with respect to the sequence length since they have constant time and memory per element.

linear transformer on the autoregressive task of image generation in Section 5.7.2. Finally in Section 5.7.3, we demonstrate that *clustered* attention can approximate transformer models pre-trained on speech, image, and natural language tasks with minimal loss in performance. We show that our models achieves competitive performance with respect to the state-of-the-art transformer architectures, while requiring significantly less GPU memory and computation.

5.7.1 Synthetic Tasks

In the following, we compare the memory consumption and computation time for different attention types on synthetically generated sequences. We also examine the convergence properties on controlled tasks whose details can be found in Appendix B.3.1.

Time and Memory Benchmark

In this experiment, we measure the required memory and GPU time *per single sequence element* to perform a forward/backward pass for the various self-attention models. We compare the memory consumption and computation time on artificially generated sequences of various lengths. For clustered attention we use 100 clusters, 63 bits for the LSH, and 10 Lloyd iterations for the K-Means. For the improved clustered attention, we use the same configuration with $k = 32$. For Reformer, we evaluate on two variants using 1 and 4 rounds of hashing. All models consist of 1 layer with 6 attention heads, embedding dimension of 64 for each head, and a feed-forward dimension of 1536.

Figure 5.4 illustrates how these metrics evolve as the sequence length increases from $N = 2^9$ to $N = 2^{15}$. For a fair comparison, we use the maximum possible batch size for each method and we divide the computational cost and memory with the number of samples in each batch and the sequence length.

We note that, in contrast to all other methods, vanilla transformer scales quadratically with respect to the sequence length and does not fit in GPU memory for sequences longer than 2^{13} elements. All other methods scale linearly. Linear attention is both the fastest and consumes the least amount memory. Clustered attention becomes faster than the vanilla transformer for sequences with 1000 elements or more, while improved clustered attention surpasses it for sequences with 2000 elements. Note that with respect to per sample memory, our proposed linear and clustered attention variants perform better than all other methods. This can be explained by the fact that our method does not require storing intermediate results to compute the gradients from multiple hashing rounds as Reformer does. It can be seen, that lsh-1 is faster than the improved clustered attention, however, as also mentioned by (Kitaev et al., 2020) Reformer requires multiple hashing rounds to generalize.

5.7.2 Image Generation

Transformers have shown great results on the task of conditional or unconditional autoregressive generation (Radford et al., 2019; Child et al., 2019), however, sampling from transformers is slow due to the task being inherently sequential and the memory scaling with the square of the sequence length. In this section, we train causally masked transformers to predict images pixel by pixel.

We evaluate our models on two different datasets. We first consider the MNIST dataset (LeCun and Cortes, 2010) consisting of 60000 images of handwritten digits between 0 and 9. We then evaluate on the more difficult CIFAR-10 dataset (Krizhevsky, 2009) which consists of 60000 colour images from 10 different classes including animals and vehicles. Our achieved performance in terms of bits per dimension is on par with *softmax* attention while being able to generate images **more than 1,000 times faster** and with **constant memory per image** from the first to the last pixel. We refer the reader to appendices B.3.2 to B.3.4 for comparisons in terms of training evolution, quality of generated images and time to generate a single image. In addition, we also compare with a faster softmax transformer that caches the keys and values during inference, in contrast to the PyTorch implementation.

MNIST

First, we evaluate our model on image generation with autoregressive transformers on the widely used MNIST dataset (LeCun and Cortes, 2010). The architecture for this experiment comprises 8 attention layers with 8 attention heads each. We set the embedding size to 256 which is 32 dimensions per head. Our feed forward dimensions are 4 times larger than our embedding size. We model the output with a mixture of 10 logistics as introduced by Salimans et al. (2017). We use the RAdam optimizer with a learning rate of 10^{-4} and train all models for 250 epochs. For the reformer baseline, we use 1 and 4 hashing rounds. Furthermore, as suggested in Kitaev et al. (2020), we use 64 buckets and chunks with approximately 32 elements. In particular, we divide the 783 long input sequence to 27 chunks of 29 elements each. Since the sequence length is relatively small, namely only 784 pixels, to remove differences due to different batch sizes we use a batch size of 10 for all methods.

Table 5.4: Comparison of autoregressive image generation of MNIST images. Our linear transformers achieve almost the same bits/dim as the full softmax attention but more than 300 times higher throughput in image generation. Full details in Section 5.7.2.

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

Table 5.4 summarizes the results. We observe that linear transformers achieve almost the same performance, in terms of final perplexity, as softmax transformers while being able to generate images more than 300 times faster. This is achieved due to the low memory requirements of our model, which is able to simultaneously generate 10,000 MNIST images with a single GPU. In particular, the memory is constant with respect to the sequence length because the only thing that needs to be stored between pixels are the s_i and z_i values as described in (5.22) to (5.23). On the other hand, both softmax and Reformer require memory that increases with the length of the sequence.

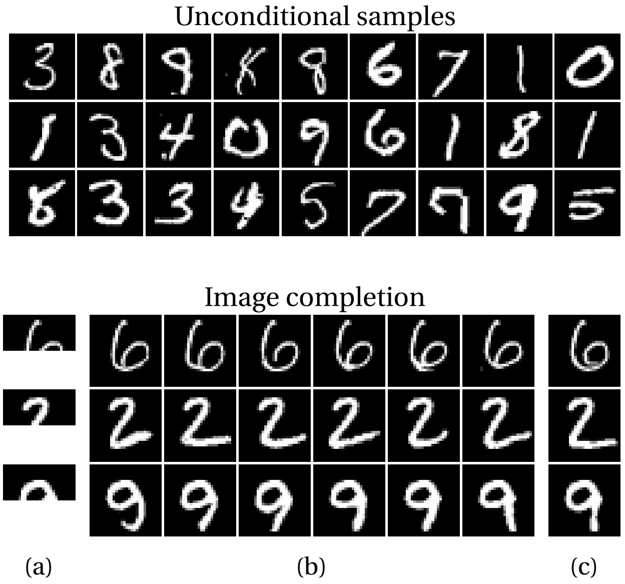


Figure 5.5: Unconditional samples and image completions generated by our method for MNIST. (a) depicts the occluded original images, (b) the completions and (c) the original. Our model achieves comparable bits/dimension to softmax, while having more than **300 times** higher throughput, generating **142 images/second**. For details see Section 5.7.2.

Image completions and unconditional samples from our MNIST model can be seen in Figure 5.5. We observe that our linear transformer generates very convincing samples with sharp boundaries and no noise. In the case of image completion, we also observe that the transformer learns to use the same stroke style and width as the original image effectively attending over long temporal distances. Note that as the achieved perplexity is more or less the same for all models, we do not observe qualitative differences between the generated samples from different models.

CIFAR-10

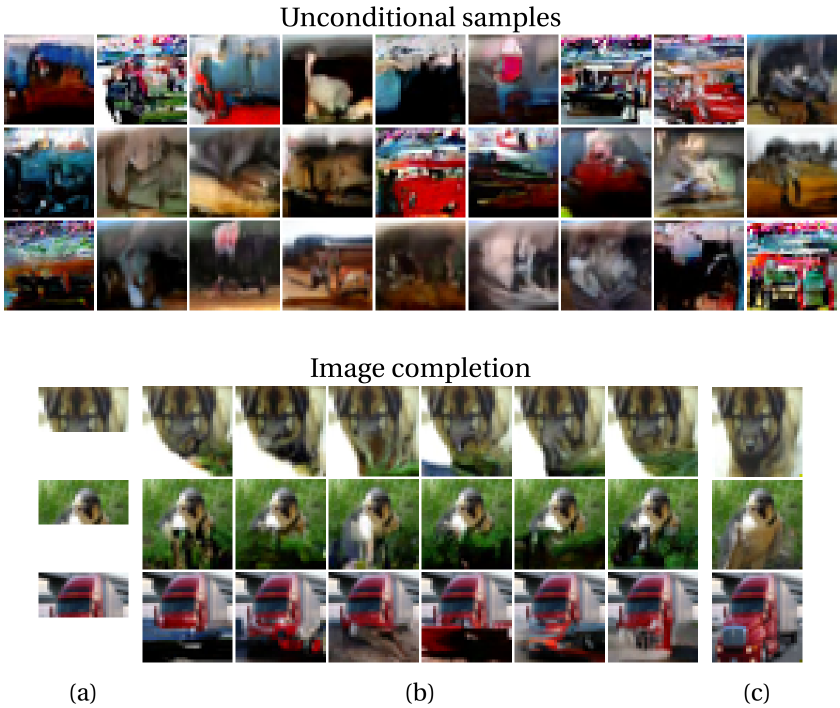


Figure 5.6: Unconditional samples and image completions generated by our method for CIFAR-10. (a) depicts the occluded original images, (b) the completions and (c) the original. As the sequence length grows linear transformers become more efficient compared to softmax attention. Our model achieves more than **4,000 times** higher throughput and generates **17.85 images/second**. For details see Section 5.7.2.

The benefits of our linear formulation increase as the sequence length increases. To showcase that, we train 16 layer transformers to generate CIFAR-10 images (Krizhevsky, 2009). For each layer we use the same configuration as in the previous experiment. For Reformer, we use again 64 buckets and 83 chunks of 37 elements, which is approximately 32, as suggested in the paper. Since the sequence length is almost 4 times larger than for the previous experiment, the softmax transformer can only be used with a batch size of 1 in the largest GPU that is available to us, namely an NVidia P40 with 24GB of memory. For both the linear transformer and reformer, we use a batch size of 4. All models are trained for 7 days. We report results in terms of bits per dimension and image generation throughput in Table 5.5. Note that although the main point of this experiment is not the final perplexity, it is evident that as the sequence length grows, the fast transformer models become increasingly more efficient per GPU hour, achieving better scores than their slower counterparts.

Table 5.5: We train autoregressive transformers for 1 week on a single GPU to generate CIFAR-10 images. Our linear transformer completes 3 times more epochs than softmax, which results in better perplexity. Our model generates images 4,000× faster than the baselines. The full details of the experiment are in Section 5.7.2.

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

As the memory and time to generate a single pixel scales quadratically with the number of pixels for both Reformer and softmax attention, the increase in throughput for our linear transformer is even more pronounced. In particular, **for every image generated** by the softmax transformer, **our method can generate 4,460 images**. Image completions and unconditional samples from our model can be seen in Figure 5.6. We observe that our model generates images with spatial consistency and can complete images convincingly without significantly hindering the recognition of the image category. For instance, in Figure 5.6b, all images have successfully completed the dog's nose (first row) or the windshield of the truck (last row).

5.7.3 Pre-Trained Model Approximation

To highlight the ability of our *clustered* attention model to approximate arbitrarily complicated attention distributions, we evaluate it for approximating the models pre-trained on speech, image, and natural language processing tasks. For speech, we consider the previously discussed wav2vec 2.0 (W2V2) model (Baevski et al., 2020). For vision, we consider the pre-trained BigGAN model (Brock et al., 2019). For text, we approximate the RoBERTa model (Liu et al., 2019a).

wav2vec 2.0 Approximation for Speech Recognition

For this task, we evaluate the pre-trained W2V2 Base model that has been fine-tuned on the 100h Librispeech dataset. We use the model provided by (Baevski et al., 2020). In Table 5.6, we present the word error rate on the clean and other parts of the test set decoded using a greedy decoder and no language model. We also report the inference for the greedy decoder.

Table 5.6: We measure the word error rate and inference times for wav2vec 2.0 model approximation. We report the WER on the clean and other portion of the Librispeech test sets. We use a greedy decoder without any language model to measure the WER and the inference times. We additionally report the WER obtained using 4-gram LM (in parenthesis). We find that *improved clustered* attention with only 100 clusters can approximate the base model with minimal loss in performance. The slower inference time is due to the short sequence lengths. In contrast to the FBank features extracted at 100Hz, the wav2vec 2.0 convolutional front end downsamples the input to the transformer encoder to 50Hz.

		Word Error Rate		
	Model	test-clean	test-other	Inference Time (s)
(a)	softmax	6.1 (3.4)	13.3 (8.2)	52.2
(b)	clustered-100	70.4 (42.5)	72.2 (48.0)	60.0
(c)	i-clustered-100	7.5 (3.6)	14.7 (8.6)	74.1
(d)	i-clustered-200 (topk=16)	6.7 (3.5)	13.9 (8.3)	77.2
(e)	i-clustered-200	6.2 (3.4)	13.5 (8.3)	80.5

Additionally, we report the WER obtained with a beam search decoder from (Pratap et al., 2019) using a beam width of 500. For this, we use the 4-gram language model from Kaldi recipes (Povey et al., 2011). We find that improved clustered attention can approximate the pre-trained model with a small loss in performance. We also see that the approximation improves as we increase the number of clusters. Note that the *softmax* attention is faster for the inference as the W2V2 model has a convolutional front-end that downsamples the audio to 50Hz before they are input to the transformer encoder.

We also note that the wav2vec 2.0 model with vanilla attention in row (a) gives results similar to our CTC model from row (e) of Table 4.3. The minor difference is because we use a seven-layered TDNNF model on top of the transformer encoder for the model (e) in Section 4.4.1. In contrast, W2V2 model from (Baevski et al., 2020) uses a linear layer.

RoBERTa Approximation for Natural Language Processing

We evaluate on 10 different tasks on the GLUE (Wang et al., 2019a) and SQuAD (Rajpurkar et al., 2018) benchmarks. These involve tasks such as question answering (SQuAD) and textual entailment (RTE), which exhibit arbitrary and sparse attention patterns. We refer the reader to Wang et al. (2019a); Rajpurkar et al. (2018) for a detailed analysis of all tasks.

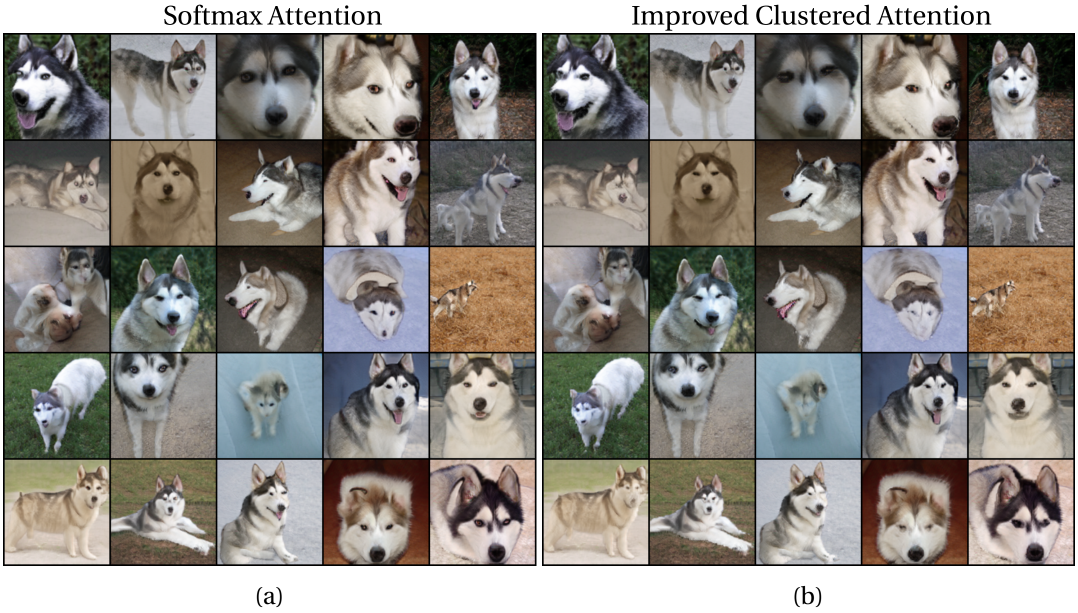
For the GLUE tasks, the maximum sequence length is 128 while for SQuAD, it is 384. For each task, we use 25 clusters for approximation which is less than 20% and 10% of the input sequence length for GLUE and SQuAD tasks respectively. In Table 5.7, we summarize the performance per task. We observe that improved clustered performs as well as the softmax transformer in all tasks but SQuAD, in which it is only marginally worse. Moreover, we note that clustered performs significantly worse in tasks that require more complicated attention

Table 5.7: We report the performance of model approximation on GLUE and SQuAD benchmarks involving natural language processing tasks such as textual entailment (RTE) and question answering (SQuAD). More details about the tasks can be found in (Wang et al., 2019a; Rajpurkar et al., 2018). Following common practice, we report accuracy for all tasks except STS-B and SQuAD, where we report Pearson correlation and F1-score respectively. For all metrics higher is better.

	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI	SQuAD
softmax	0.601	0.880	0.868	0.929	0.915	0.682	0.947	0.900	0.437	0.904
clustered-25	0.598	0.794	0.436	0.746	0.894	0.498	0.944	0.789	0.437	0.006
i-clustered-25	0.601	0.880	0.873	0.930	0.915	0.704	0.947	0.900	0.437	0.876

patterns such as SQuAD and RTE. For inference time, *softmax* was faster than the *clustered* attention variants due to short sequence lengths.

BigGAN Approximation for Image Generation



(a) (b)

Figure 5.7: Samples generated by BigGAN model using softmax attention and improved clustered attention with 128 clusters. We observe that using only a few clusters and no fine-tuning, improved clustered attention can synthesize images that are almost indistinguishable from those generated by the softmax attention.

For this task, we take the pre-trained BigGAN model (Brock et al., 2019) that is trained on the Imagenet dataset to generate images of size 64 × 64. In Figure 5.7, we present images synthesized using the softmax attention and improved-clustered attention with only 128 clusters. We can see that the improved clustered attention can generate high fidelity images that are almost indistinguishable from the softmax attention with no fine-tuning. Though the images look very similar, there are minor artifacts that remain due to approximation error. For example, if we compare the images in first column of second row in (a) and (b), we can see that eyes seem to be missing for image generated with the improved clustered attention. We believe some of this performance can be recovered with few steps of fine-tuning.

5.8 Conclusions

In this work, we presented two efficient alternatives to vanilla transformer to reduce the computational and memory requirements for self-attention. We first presented *linear transformer*, a model that exploits the associativity property of matrix products together with softmax kernelization to compute self-attention in time and memory that scales linearly with respect to the sequence length. We show that our model can be used with causal masking and still retain its linear asymptotic complexities. Finally, we express the transformer model as a recurrent neural network, which allows us to perform inference on autoregressive tasks thousands of time faster.

This property opens a multitude of directions for future research regarding the storage and retrieval of information in both RNNs and transformers. Another line of research to be explored is related to the choice of feature map for linear attention. For instance, using the Taylor series expansion for the softmax function could allow us to use feature maps that can approximate the models pre-trained with softmax attention.

We then presented *clustered attention* a method that approximates vanilla transformers with significantly lower computational requirements. In particular, we have shown that on the ASR task on Switchboard dataset, our model can be up to 2× faster during training and inference while being only $\sim 3\%$ relatively worse in WER. In contrast to recent fast variations of transformers, we have also shown that our method can efficiently approximate pre-trained models with full attention while retaining the linear asymptotic complexity.

The proposed linear and clustered attention variants open several research directions towards applying transformers on long sequence tasks such as music generation, scene flow estimation etc. We consider masked language modeling for long texts to be of particular importance, as it will allow fine-tuning for downstream tasks that need a context longer than the commonly used 512 tokens.

6 Towards Data and Compute Efficient ASR

6.1 Introduction

In the previous chapters, we discussed two popular self-supervised pre-training approaches, W2V2 and MAM, that employ the self-attention based Transformer architecture to learn useful representations from untranscribed audio and improve the data efficiency for ASR models. However, the quadratic complexity of self-attention computation makes the Transformers hard to scale for long sequence modeling.

In Chapter 5, we presented *linear* and *clustered* attention as two efficient variants of the Transformer that improve the compute efficiency for long sequence modeling. We find that the *linear* attention significantly improves inference time and consumes much less memory. However, it results in WER degradation for the ASR task. In contrast, *improved clustered* attention results in minimal WER degradation and has lower memory requirements. However, it has a slower inference when dealing with small input sequences. Slower inference is further exacerbated in the case of W2V2 model which extracts acoustic features at 50Hz instead of the traditional features extracted at 100Hz.

In this chapter, we present a practical method for compute and data efficient ASR[1]. We build on the recent Squeeze and Efficient Wav2vec (SEW) architecture (Wu et al., 2022) for efficient training and inference with the W2V2 model. The SEW architecture modifies the convolutional feature extractor to improve inference latency. Moreover, they introduce a squeezing mechanism that downsamples the convolutional front-end output from 50Hz to 25Hz. This reduces the computation for the transformer encoder. The output of the encoder is upsampled with a linear layer to produce output at 50Hz. Note that while the asymptotic complexity for self-attention computation remains quadratic in the sequence length, in practice, we observe upto 2× faster inference. However, one downside to SEW is the WER degradation for sensitive applications.

We propose to overcome this limitation with stochastic pre-training and fine-tuning for W2V2

---

[1] work done during the internship at Meta AI

models. Deep neural networks with stochastic depth using layerdrop and block drops have previously been explored in (Wu et al., 2018; Fan et al., 2020; Huang et al., 2016; Liu et al., 2019b) for efficient inference. In this chapter, we introduce stochastic compression for on-demand compute reduction for W2V2 model. Stochastic compression enables training a single model that can support a number of operating points with a smooth trade-off between WER and inference latency.

As opposed to training with a fixed squeezing factor, at each iteration, we sample a squeezing factor S that takes a value from 1 to S_f. Given the squeezing factor S, we compress the input to the transformer encoder by this factor. Similar to SEW, the output of the transformer encoder is upsampled to the original sequence length using a linear layer. In addition to this, for each transformer layer, we also introduce a stochastic pooling mechanism that could be independently applied to queries and keys-values in a decoupled fashion. In contrast to the squeezing mechanism, we do this without introducing any additional learnable parameters.

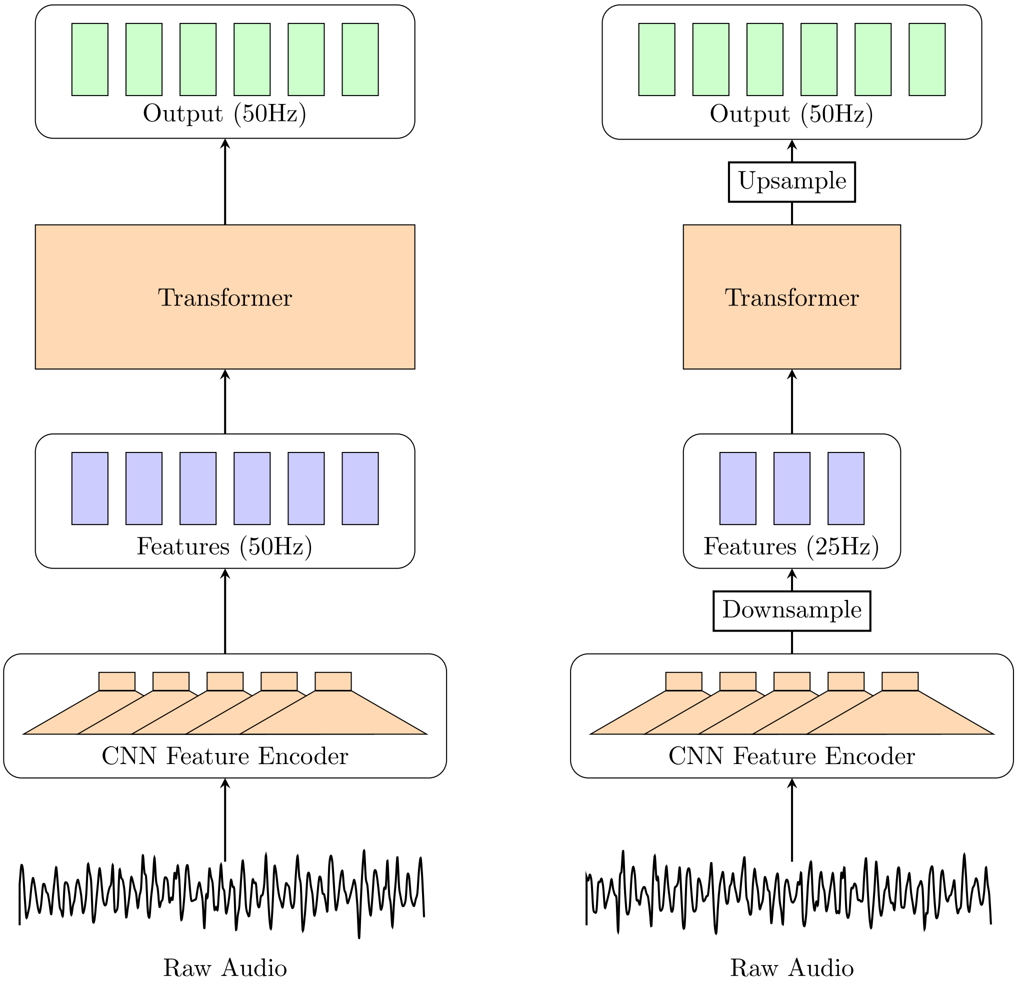
We show that stochastic pre-training and fine-tuning provides a smooth trade-off between WER and inference time with only marginal performance degradation compared to the non-stochastic variants. Thus allowing for a range of operating points for applications with varying requirements. We further show that by fine-tuning the same stochastically pre-trained model to a specific configuration (operating point), we can get the same accuracy as the corresponding non-stochastic model. This removes the need for pre-training multiple models, resulting in significant computational savings.

6.2 Related Works

There have been several works for improving the efficiency of W2V2 models for ASR. In (Peng et al., 2021b), the authors propose model distillation and quantization techniques to reduce the computational requirements and inference latency for the W2V2 model. While quantization substantially improves the CPU inference time with no WER degradation, the impact on more common GPU inference is not discussed. In contrast, model distillation improves the inference time for both CPU and GPU; however, it substantially degrades the WER from 2.6% to 9.5% on the clean portion of the Librispeech development dataset. Moreover, both techniques do not reduce the pre-training time.

In (Lai et al., 2021), the authors present Prune, Adjust and Re-Prune (PARP), a method for discovering sparse subnetwork from a pre-trained W2V2 model that gives the same or better WER for downstream ASR task. While the proposed method discovers subnetworks with better WER for dataset, the training time for both self-supervised pre-training and supervised fine-tuning remain the same due to the use of unstructured sparsity masks.

Most relevant to this work is the Squeeze and Efficient Wav2vec (SEW) architecture proposed by Wu et al. (2022). The SEW model introduces two important architectural changes to the W2V2 architecture. First, they introduce compact wave feature extractor (WFE-C) that



(a) Original wav2vec 2.0
(b) Squeeze and Efficient Wav2vec (SEW)

Figure 6.1: Comparing original wav2vec 2.0 and SEW model architecture with a squeezing factor of 2. SEW architecture introduces a downsampling layer to reduce the input sequence length by a factor of 2.

replaces the original W2V2 convolutional feature extractor (WFE-O). WFE-O uses the same number of channels in all layers of its convolutional extractor. WFE-C starts with a small number of channels c and doubles the channel when the sequence length is downsampled by 4 times. WFE-C distributes the forward and backward pass computation more evenly across layers resulting in a similar WER as WFE-O while being much faster. In this work, we always use WFE-C-c64-l1 as the compact feature extractor. We refer the readers to the Section 4.4 of (Wu et al., 2022) for more details.

Next, they introduce *Squeezed Context Networks* to reduce the length of input sequence to the transformer encoder. As shown in Figure 6.1b, they introduce a *squeezing* mechanism via a downsampling layer at the output of the convolutional encoder to reduce the output rate from 50Hz to 25Hz. The downsampled sequence reduces the memory and computational time for the transformer encoder. The upsampling layer at the output of transformer encoder produces outputs at 50Hz for contrastive loss computation.

6.3 Stochastic wav2vec 2.0

In the following, we describe the proposed method for training the W2V2 model with stochastic compression.

6.3.1 Basic Notations

We start by introducing basic definitions that we will later use to define query and key-value mean pooled attention. Let us denote the queries as $Q \in \mathbb{R}^{N \times D}$, keys as $K \in \mathbb{R}^{S \times D}$, and values as $V \in \mathbb{R}^{S \times M}$, where N and S denotes the query and key sequence lengths respectively. D and M denote the embedding dimensions for keys and values respectively. The attention output $V' \in \mathbb{R}^{N \times M}$ is given as follows:

$$A(Q,K,V) = V' = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V. \tag{6.1}$$

Let us also define the mean-pooling or downsampling operator $D(X, S_p)$ that takes as input a sequence $X \in \mathbb{R}^{N \times D_x}$ and a pooling factor S_p to output another sequence $X^p \in \mathbb{R}^{N_p \times D_x}$ where $N_p = \lceil \frac{N}{S_p} \rceil$. Here $\lceil a \rceil$ denotes the *ceil* operation. Note that we may need to pad the signal appropriately. The i-th output X_i^p is given by:

$$X_i^p = \sum_{j=1}^{S_p} \frac{X_{(i*S_p)+j}}{S_p} \quad \forall i \in \{1, \ldots, N_p\}. \tag{6.2}$$

Finally, let us define the upsampling operator $U(X^p, S_u)$ that takes as input a sequence $X^p \in \mathbb{R}^{N_p \times D_x}$ and an upsampling factor S_u to output another sequence $X \in \mathbb{R}^{N \times D_x}$ where $N = (N_p * S_u)$. The i-th output X_i is given by:

$$X_i = X^p_{\lfloor \frac{i}{S_u} \rfloor} \quad \forall i \in \{1, \ldots, N\}, \tag{6.3}$$

where $\lfloor a \rfloor$ denotes the *floor* operation.

6.3.2 Query and Key-Value Mean Pooled Attention

The upsampling layer in the SEW architecture introduces some additional parameters to the W2V2 model. In contrast to this, we introduce query and key-value mean pooling during self-attention computation which can be applied independently to each layer with no additional parameters. This allows for finer control over the compression at each transformer layer.

Let us denote the query pooling and key-value pooling factors as S_q and S_k respectively. Given S_q and S_k, we first compute the mean pooled queries Q_p, keys K_p, and values V_p using the previously defined downsampling operator $D(\cdot)$ as follows:

$$Q_p = D(Q, S_q), \tag{6.4}$$

$$K_p = D(K, S_k), \tag{6.5}$$

$$V_p = D(V, S_k). \tag{6.6}$$

The output $V' \in \mathbb{R}^{N\times M}$ for the pooled attention is:

$$V'_p = A(Q_p, K_p, V_p), \tag{6.7}$$

$$V' = U(V'_p, S_q). \tag{6.8}$$

6.3.3 Stochastic Compression

In contrast to SEW models, where the squeezing factor remains fixed during pre-training and fine-tuning, at each iteration, we sample the squeezing factor uniformly from the set $\{1, \ldots, S_f\}$. Similarly, for each transformer layer we also sample the query and key-value mean pooling factors from the sets $\{1, \ldots, S_q\}$ and $\{1, \ldots, S_k\}$ respectively.

6.4 Experimental Setup

6.4.1 Models

We conduct experiments with W2V2, SEW, and our proposed stochastically compressed (St-SEW) models with 12 and 24 encoder layers. For all models, we use WFE-C-c64-l1 as the feature extractor. In Table 6.1, we describe the main hyper-parameters for different classes of models. Note that we can view W2V2 and SEW models as special cases of the stochastic models with a specific setting of squeezing and pooling factors.

6.4.2 Pre-training

We pre-train models on 960h of Librispeech (Panayotov et al., 2015) dataset. We use the same hyperparameters as W2V2 base (Baevski et al., 2020). To reduce the computational requirements, our models are trained with 8 NVIDIA V100 GPUs using Fairseq (Ott et al., 2019) and PyTorch (Paszke et al., 2019). We double the maximum tokens per batch and set gradient accumulation steps to 4 to simulate 64 GPUs as used in (Baevski et al., 2020).

Table 6.1: Model hyper-parameters and pre-training times in hours for base (B) and large (L) models. S_f refers to possible squeeze factors, S_q and S_k refer to the possible query and key-value pooling factors. E and D refer to the model dimension and the number of layers (depth) in transformer respectively. We estimate the pre-training times (PT) for 400K steps for models trained using 960 hour of Librispeech data on 8 NVIDIA V100 GPUs.

Model	S_f	S_k	S_q	E	D	PT (hr)
W2V2-B	1	1	1	768	12	117
W2V2-L	1	1	1	1024	24	172 [2]
SEW-B	2	1	1	768	12	83
SEW-L	2	1	1	1024	24	115
St-SEW-B	{1,2}	{1,2}	{1,2}	768	12	100
St-SEW-L	{1,2}	{1,2}	{1,2}	1024	24	140

6.4.3 Fine-tuning

We add a linear layer to the top of the transformer encoder and fine-tune the model for 20K steps using the CTC objective (Graves et al., 2006b) on the 10h subset of the Librispeech dataset. We use the *dev-other* for model selection during fine-tuning. For stochastically pre-trained models, we consider the following two strategies for fine-tuning:

Stochastic Fine-tuning: In this setting, we fine-tune stochastically by sampling the squeezing and pooling factors similar to pre-training. During validation, we use randomly selected values for S_f, S_k, and S_q. This model allows for a smooth trade-off between WER and inference time for different settings of squeeze and pooling factors used during inference.

Deterministic Fine-tuning: In this setting, we fine-tune the model for a fixed configuration of squeeze and pooling factors. In contrast to stochastic fine-tuning, this model can only be inferred with the selected configuration. However, we find that this gives better WER. Note that, for each configuration, we fine-tune the same stochastically pre-trained model resulting in significant computational savings as pre-training typically requires more computational resources and time.

6.4.4 Evaluation

Similar to (Wu et al., 2022), we consider pre-training time (Table 6.1), inference time, and word error rate (WER) (tables 6.3 and 6.4) as metrics to measure model efficiency. We report inference times (in seconds) on *dev other* split using CTC greedy decoding on NVIDIA V100 with FP32 operations. We use the 4-gram language model (LM) and wav2letter decoder (Pratap

---

[2] Estimated time. The model is unstable and pre-training diverged.

et al., 2019) for decoding with language model (LM). Similar to (Wu et al., 2022), we do not tune hyper-parameters when decoding with LM. We use the default LM weight 2, word score -1, and beam size 50.

Inference with Stochastic Models

For our proposed stochastically pre-trained models fine-tuned in stochastic or deterministic settings, we provide inference time and WER for various configurations of squeeze, query and key-value pooling factors.

6.5 Results

In the following, we first analyze the performance of stochastic W2V2 model for different query and key-value mean-pooling choices. We then discuss the trade-offs for the stochastically trained W2V2 model against the original W2V2 and SEW models. Finally, we present the WER results for Base and Large on the *clean* and *other* parts of the Librispeech test sets.

6.5.1 How much query and key-value pooling?

We consider (a) $\{1,2,3\}$ and (b) $\{1,2\}$ as the two choices of key-value and query pooling sets to analyze the WER and inference time trade-offs for base models pre-trained for 100K steps.

In Table 6.2, we present the pre-training time (PT) for each of these models. We can see that the pre-training time for both (a) and (b) is quite similar. Both of these are faster than the W2V2-B model and slower than the SEW-B model. This is expected because we occasionally sample the squeeze and pooling factors as 1 in which case the computation time for our model would be higher than that of SEW-B model.

Table 6.2: Comparing pre-training time for different choices of query and key-pooling factors for St-SEW-B models trained for 100K steps.

Model	$\{S_f, S_k, S_q\}$	PT (hours)
W2V2-B	{1},{1},{1}	29.2
SEW-B	{2},{1},{1}	20.7
St-SEW-B-1-2	{1,2},{1,2},{1,2}	24.9
St-SEW-B-1-2-3	{1,2},{1,2,3},{1,2,3}	24.6

In Figure 6.2, we present the inference time and WER trade-off for different models. We plot the WER obtained on *dev-other* split when the same model is then inferred with different values for squeeze (S_f), key-value (S_k) and query pooling (S_q) pooling factors indicated on the figures as triplet in the order S_f-S_k-S_q . We also train the W2V2-B and SEW-B models for comparison. We can see that the model trained with pooling factors sampled from $\{1,2\}$ outperforms the model

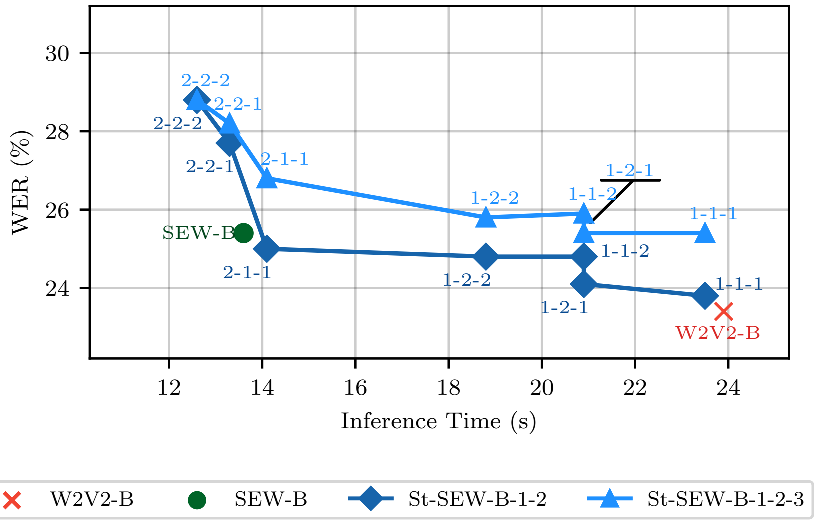


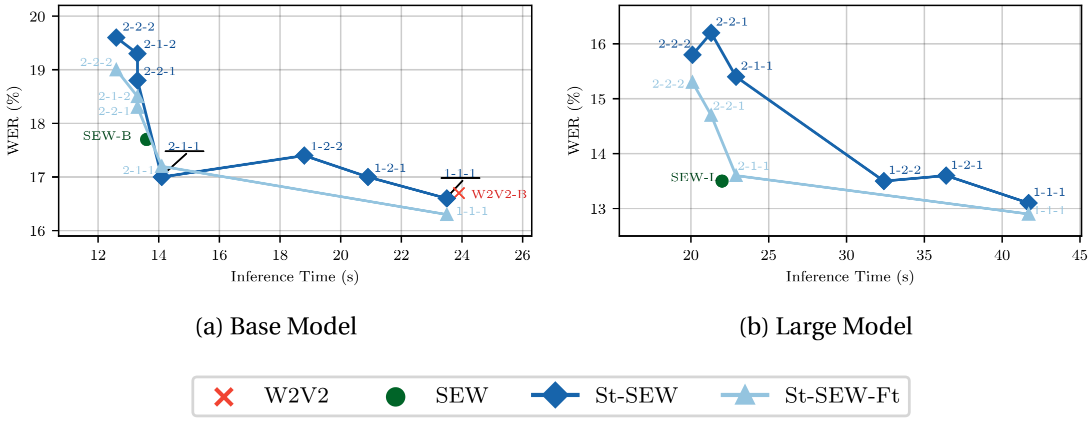
Figure 6.2: We compare the different query and key-value pooling options for St-SEW-B models trained for 100K steps. The numbers near the datapoints denote the inference configuration in the order: squeeze (S_f), key-value pooling (S_k), and query pooling (S_q) factors. We can see that using pooling factors of $\{1,2\}$ results in a better performance trade-off.

that samples pooling factors from $\{1,2,3\}$. We also see that the performance for this model is similar to the non-stochastic W2V2-B and SEW-B models. In all subsequent experiments, we always choose the query and key-value pooling factors from $\{1,2\}$.

6.5.2 On-demand compute reduction inference

In this section, we compare the WER and inference time trade-offs for Base and Large models. We pre-train each model for 400K steps followed by fine-tuning on the 10h transcribed subset of the Librispeech dataset. We use *St-SEW-B-Ft* and *St-SEW-L-Ft* to denote the stochastic pre-trained models fine-tuned in the deterministic setting as discussed before. We evaluate the inference time and WER for the following configurations of (S_f,S_k,S_q) factors: (a) (1,1,1), (b) (2,1,1) (c) (2,2,1), and (d) (2,2,2). We select (1,1,1) and (2,1,1) to compare against the W2V2 and SEW models respectively. We then increase the query and key pooling to further increase the overall compression resulting in faster inference. We skip results for the (2,1,2) as it is very similar to (2,2,1).

Figure 6.3 shows the WER and inference-time for Base and Large models evaluated on the *dev-other* portion of the Librispeech dataset. For Base models, the same stochastically fine-tuned (St-SEW) model performs only marginally worse than W2V2 and SEW models when inferred using the corresponding configurations. There is a bigger performance difference in the case of Large models especially when $S_f = 2$ during inference. However, the difference in performance can be recovered, if we fine-tune the models to specific configurations (St-SEW-L-Ft). Depending on the application requirements, we can choose different operating points



(a) Base Model

(b) Large Model

Figure 6.3: WER and inference time tradeoff for different Base and Large models. The numbers near the datapoints denote the configuration used during inference in the order (S_f-S_k-S_q). We see that the stochastically trained model provides a smooth trade-off between WER and inference times. Fine-tuning the pre-trained to a specific configuration of interest improves the WER significantly.

(configurations) for an on-demand compute reduction for a smooth trade-off in WER.

For Large models, we find that W2V2-L pre-training is unstable and diverges quickly. In contrast to this the stochastic model pre-training and fine-tuning is stable. We always use post-layer norm for Transformer encoder. In (Baevski et al., 2020), for stable pre-training, Large models are trained with pre-layer norm and convolutional front-end with extra normalization layers resulting in significantly higher training time and inference latency.

6.5.3 Test Set Evaluation

We present the WER obtained by Base and Large models on the *clean* and *other* portion of the Librispeech test sets.

Base Model Evaluation: In Table 6.3, we present the results for the Base models. Rows (a)-(c) present the inference times for different configurations of squeeze (S_f, key pooling (S_k), and query pooling (S_q) factors. Comparing configuration (1, 1, 1) and (2, 1, 1), we see that changing the squeeze factor to 2 improves the inference time by more than 40%. Increasing the pooling factors to 2 decreases the latency further by about 10%.

Rows (d)-(g) and (h)-(k) present the WER for the clean and other portion of the Librispeech test set, respectively. Comparing W2V2-B and SEW-B models, we first note that the squeeze factor to 2 only degrades the greedy decoder WER by 7% and 5% relatively for the clean and other test sets, respectively. From rows (f) and (j), the WER for *St-SEW-B* model is very close to W2V2 and SEW models in the corresponding configurations. From (g) and (k), we also see that fine-tuning to specific configurations (*St-SEW-B-Ft*) gives slightly better WER.

Table 6.3: Inference times and WER result for base models on *other* and *clean* portions of the Librispeech test sets. We fine-tune the wav2vec 2.0 base model pre-trained on 960 hours of Librispeech data on the 10 hour transcribed subset of the Librispeech dataset. We report the WER obtained using greedy decoding as well as 4-gram LM (in parenthesis).

		Inference (S_f,S_k,S_q)			
	Model	1,1,1	2,1,1	2,2,1	2,2,2
Inference times (dev other)					
(a)	W2V2-B	23.9	-	-	-
(b)	SEW-B	-	13.6	-	-
(c)	St-SEW-B	23.5	14.0	13.3	12.6
Word Error Rate Results (test clean)					
(d)	W2V2-B	9.5 (5.0)	-	-	-
(e)	SEW-B	-	10.2 (4.9)	-	-
(f)	St-SEW-B	9.7 (5.1)	9.8 (5.2)	11.0 (5.4)	11.4 (5.4)
(g)	St-SEW-B-Ft	9.4 (5.0)	10.0 (4.9)	10.8 (5.1)	11.2 (5.2)
Word Error Rate Results (test other)					
(h)	W2V2-B	16.7 (10.7)	-	-	-
(i)	SEW-B	-	17.6 (10.8)	-	-
(j)	St-SEW-B	16.8 (11.0)	17.3 (11.3)	19.0 (11.6)	19.9 (11.8)
(k)	St-SEW-B-Ft	16.5 (10.8)	17.3 (10.8)	18.5 (11.3)	19.3 (11.7)

Large Model Evaluation: Table 6.4 presents the results for Large models. From row (b), we can note that similarly to the Base model, increasing the squeeze factor to 2 reduces the inference latency by ~ 45%. Furthermore, increasing the pooling factors to 2 improves the latency by another ~ 12%.

In contrast to the Base model in Table 6.3, we find that *St-SEW-L* does not perform as well when the squeeze factor is set to 2 (row d). Specifically, we see that the greedy decoding WER on the clean test set increases from 8.8% (row c) to 10.2% (row d) when the proposed St-SEW-L is inferred in the configuration (2, 1, 1). From rows (f) and (g), we also observe a significant degradation on other test set for this configuration. We suspect that the randomly selected values for S_f, S_k, and S_q during validation may cause the selected model to be better for some configurations than other. However, from rows (e) and (h), we again see that fine-tuning the stochastic models to the configuration of interest (*St-SEW-L-Ft*) bridges the performance gap. Note that WER improvement is also reflected for the case of beam search decoder that uses 4-gram language model.

Table 6.4: Inference times and WER result for large models on *other* and *clean* portions of the Librispeech test sets. We fine-tune the wav2vec 2.0 base model pre-trained on 960 hours of Librispeech data on the 10 hour transcribed subset of the Librispeech dataset. We report the WER obtained using greedy decoding as well as 4-gram LM (in parenthesis).

		Inference (S_f,S_k,S_q)			
	Model	1,1,1	2,1,1	2,2,1	2,2,2
		Inference times (dev other)			
(a)	SEW-L	-	22.0	-	-
(b)	St-SEW-L	41.7	22.9	21.3	20.1
		Word Error Rate Results (test clean)			
(c)	SEW-L	-	8.8 (4.6)	-	-
(d)	St-SEW-L	8.5 (4.8)	10.2 (5.1)	10.4 (5.4)	9.8 (5.1)
(e)	St-SEW-L-Ft	8.4 (4.7)	8.9 (4.5)	9.6 (4.7)	9.8 (4.8)
		Word Error Rate Results (test other)			
(f)	SEW-L	-	13.9 (9.1)	-	-
(g)	St-SEW-L	13.6 (9.3)	16.2 (10.1)	16.6 (10.1)	16.3 (10.2)
(h)	St-SEW-L-Ft	13.4 (9.1)	14.1 (9.1)	15.2 (9.5)	15.7 (9.9)

6.6 Conclusions

We proposed a stochastic compression technique for compute reduction during W2V2 pre-training as well as for on-demand compute reduction during inference. We show that stochastically pre-trained and fine-tuned models provide multiple operating points with smooth performance trade-off for different applications. We further show that fine-tuning the stochastically pre-trained model to a specific configuration provides the same WER as a model pre-trained and fine-tuned from scratch for the same configuration.

Currently, we use the same squeeze and pooling factors for all utterances. In future, we will explore techniques to adaptively choose the compression factors depending on the input utterance to improve the WER and inference time trade-off.

7 Conclusions and Future Work

7.1 Conclusions

This thesis focused on developing compute-efficient Automatic speech recognition (ASR) models for low-resource languages. Specifically, we looked at techniques to reduce the *transcribed data* and *computation* required for deep neural networks (DNN) based ASR models.

First, in Chapter 3, we proposed a novel framework that uses dropout at the test time to model uncertainty in prediction hypotheses. We systematically exploited this uncertainty to estimate WER without needing the ground-truth transcripts. We further demonstrated that the predictive uncertainty could be used to accurately localize the ASR system's errors as well as for uncertainty-aware semi-supervised learning. We have shown that the supervision lattice generated by combining the dropout decodings retains alternate hypotheses for uncertain utterances and reduces the incorrect paths when the model is confident. Our experiments showed that the dropout-based uncertainty aware supervision lattices improve the WER over the regular semi-supervised training framework.

Next, in Chapter 4, we looked at self-supervised pre-training methods, which aim to improve the seed model by exploiting unlabelled data *before* supervised adaptation. We investigated the domain generalization performance and the choice of sequence discriminative training criterion for two popular pre-training methods: masked acoustic model (MAM) and wav2vec 2.0 (W2V2), which are trained by predicting masked parts of the input. Our experiments on out-of-domain conversational speech (Switchboard) and cross-lingual data (Babel) concluded that both self-supervised pre-training methods provide significant gains over the models trained only with supervised data. Furthermore, we have shown that W2V2 pre-training helps mitigate the overfitting issues with Connectionist Temporal Classification (CTC) training and reduces the performance gap to flat-start LFMMI (E2E-LFMMI) criterion. In contrast, E2E-LFMMI performs better when fine-tuning the MAM pre-trained model.

In Chapter 5, we shifted our attention from reducing the amount of transcribed data to improving the computational requirements for the ASR models based on the Transformer architecture. Specifically, we tackled the quadratic complexity of the self-attention computation with respect to the input sequence. We first presented *linear* attention that generalizes self-attention formulation by expressing the key-query similarity as a linear dot-product of kernelized feature maps. The kernelized formulation together with the associativity property of matrix products reduces the computation and memory complexity of self-attention to linear with respect to sequence lengths. We further showed that this formulation enables autoregressive inference with linear complexity and constant memory.

We then presented *clustered* attention as a method to approximate vanilla attention. We showed that the queries close in Euclidean space also have similar attention distributions. We utilize this property to cluster the queries and use the centroids for attention computation. We also proposed *improved clustered* attention that improves the attention approximation by recomputing attention between queries and a small subset of keys. Our experiments demonstrated that the improved clustered attention outperforms vanilla attention and other efficient attention models for a given computational budget. Our approximation experiments with pre-trained models showed that improved attention could approximate complex attention patterns without fine-tuning and minimal accuracy loss.

Our experimental findings also revealed the limitations of the proposed attention mechanisms. We found that while *linear* attention improves inference speed, it results in WER degradation. On the other hand, *improved clustered* attention does not suffer from WER degradation but has slower inference for shorter sequences due to clustering overheads. To overcome this limitation, in Chapter 6, we proposed a practical method for compute and data efficient ASR with W2V2 models. We used mean-pooling to reduce the number of queries and keys-values for the attention computation without introducing large overheads or additional parameters. Furthermore, instead of using a fixed pooling/compression factor, we proposed sampling these factors uniformly from a set of possible compression factors. We demonstrated that our method of stochastic compression enables training a single model that can support a number of compression factors at inference time, resulting in a smooth trade-off between WER and inference latency.

To conclude, this thesis found that wav2vec 2.0 model pre-trained with stochastic compression enables the development of computationally efficient ASR models that can be easily adapted to new languages using only a limited amount of data.

7.2 Future Work

We believe the methods and techniques proposed in this thesis open up a number of research directions. Below we discuss a few possible research directions.

The first half of the thesis focused on reducing the amount of transcribed data. We investigated the dropout-based semi-supervised training in the context of supervised adaptation on a monolingual training. A possible direction to improve the seed model would be to combine the transcribed data from multiple languages and utilize multilingual training approaches. Multilingual training using language adaptive modeling (Madikeri et al., 2021; Tong et al., 2017) or universal phone set (Vu et al., 2014) have been shown to perform better than the monolingual models. Thus, using the improved seed model with our uncertainty-aware semi-supervised learning framework would be a promising direction to further boost the performance on low-resource languages. Similarly, the combination of self-supervised pre-training on unlabelled utterances and uncertainty-aware semi-supervised learning would be another research direction of interest.

In the second part, we focussed our attention on efficient transformer architectures. Our proposed *linear* and *clustered* attention mechanisms enable the application of these models to tasks and modalities involving long sequences. One particular use case of interest would be to see if self-supervised learning using audio inputs with longer context improves the performance for downstream ASR task. W2V2 pre-training restricts the maximum duration for input utterance to 15 seconds due to the quadratic computational and memory complexity of vanilla self-attention. Given the improvements in natural language pre-training using longer context (Beltagy et al., 2020; Zaheer et al., 2020), it would be of interest to investigate the performance benefits of pre-training with longer context for automatic speech recognition. Moreover, clustered attention groups queries from different timesteps, which makes it unsuitable for streaming ASR or autoregressive modeling as it would leak information from the future time-steps. One way to address this limitation would be to use cluster centroids to identify the top-k keys for the query members and only compute attention on those keys. This can allow masking for autoregressive modeling. In future research, we would like to conduct experiments on autoregressive and streaming ASR tasks. Finally, the proposed *linear* attention has opened several research directions. The first set of approaches introduces relative positional embeddings for linear transformers (Qin et al., 2022; Su et al., 2021) to improve the expressivity. The second line of work focuses on improving the kernel feature maps by gating mechanisms (Peng et al., 2021a; Schlag et al., 2021) or through softmax kernel approximation using random fourier features (Choromanski et al., 2021). We think a promising direction would be constructing feature maps with structured sparsity. This would increase the expressivity of feature maps without incurring significant overheads for sparse dot products.

A Appendix for Chapter 4

A.1 Pre-trained Models as Feature Extractors

In this section, we compare the performance of using the pre-trained model as feature extractor to fine-tuning the pre-trained model weights. When using the model for feature extraction, we freeze the weights of pre-trained model. We pass the output of the encoder to a twelve layered factorized time-delay neural network (TDNNF) architecture referred to as *TDNNF-L*. We denote this setting as FE in the later experiments. In the second setting, we fine-tune the pre-trained model together with a seven layered TDNNF architecture *TDNNF-S*. We conduct these experiments for the MAM pre-trained model. For baseline, we use the previously discussed *TDNNF-L* trained only on the supervised data.

A.1.1 Librispeech (100h)

Table A.1 shows that the MAM pre-trained model outperforms the baseline model in row (a) even when the pre-trained models is used a feature extractor (b). The most notable difference can be seen in the case of the *noisy* portion of the test dataset where we find ≈ 4.7% absolute WER improvements. From row (c), we can see that the fine-tuning model results in most improvements.

A.1.2 Switchboard (300h)

Table A.2 presents the results for the Switchboard dataset. Similar to the Librispeech experiment, we find that using the pre-trained model as a feature extractor or fine-tuning the weights outperforms the baseline model trained from scratch. Once again, the fine-tuned model results in most improvements in terms of WER.

Table A.1: Comparing the performance of MAM pre-trained model when used as a feature extractor to fine-tuning the pre-trained model weights. Note that in both settings, we obtain better WER compared to the baseline model trained only on the supervised data. Fine-tuning the pre-trained model weights in significant improvements for the case of noisy portion of the test set. MAM-B refers to the base transformer model used for pre-training. TDNNF-L and TDNNF-S refer to the 12 and 7 layered TDNNF architectures, respectively. E2E-LFMMI refers to flat-start training with LFMMI.

			Word Error Rate			
			3-gram		**4-gram**	
	Architecture	Supervision	clean	other	clean	other
		Supervised Only				
(a)	TDNNF-L	e2e-lfmmi	8.6	26.3	5.9	20.0
		Pre-training + Supervised				
(b)	MAM-B + TDNNF-L (FE)	e2e-lfmmi	7.98	22.14	5.52	16.32
(c)	MAM-B + TDNNF-S	e2e-lfmmi	**7.78**	**20.19**	**5.35**	**14.75**

Table A.2: Comparison of word error rates (WER) (in %) on eval2000 test set for the 300 hours Switchboard task. The 3-gram language model is based on Switchboard, whereas the 4-gram employs Switchboard+Fisher training set transcripts. Using a pre-trained model as a feature extractor or fine-tuning outperforms the baseline. Fine-tuning achieves the best WER on both callhome and switchboard part of eval set.

			Word Error Rate			
			3-gram		**4-gram**	
	Architecture	Criterion	swbd	ch	swbd	ch
		Supervised Only				
(a)	TDNNF-L	e2e-lfmmi	11.8	22.5	10.3	20.3
(b)	TDNN-LSTM (Hadian et al., 2018)	e2e-lfmmi	11.3	21.5	9.8	19.3
		Pre-training + Supervised				
(d)	MAM-B + TDNNF-L (FE)	e2e-lfmmi	11.3	22.0	9.9	19.7
(e)	MAM-B + TDNNF-S	e2e-lfmmi	10.9	20.4	9.4	18.2

	Model	Criterion	**Word Error Rate**	
			Swahili	Tagalog
	Supervised Only			
(a)	TDNNF	e2e-lfmmi	39.5	44.9
(b)	BLSTM-HMM (Inaguma et al., 2019)	hybrid	38.3	46.3
	Pre-training + Supervised			
(c)	MAM-B + TDNNF-L (FE)	e2e-lfmmi	40.4	46.6
(d)	MAM-B + TDNNF-S	e2e-lfmmi	36.7	43.4

Table A.3: Comparison of word error rates (WER) (in %) on dev10h set for the Swahili and Tagalog languages of the Babel dataset. Using the pre-trained model performs worse than the baseline trained only with supervised data. Fine-tuning the pre-trained MAM model recovers the performance and significantly outperforms the monolingual and other baselines.

A.1.3 Babel: Swahili (38h) and Tagalog (84h)

Table A.3 compares the WER for the models trained from scratch to the models pre-trained on Librispeech. It can be seen that for both Swahili and Tagalog, using pre-trained model as feature extractor performs worse than the models trained from scratch indicating that the pre-trained model removes important language specific information. However, on fine-tuning, the model adjusts its parameters to recapture this information and outperforms the baseline.

B Appendix for Chapter 5

B.1 Linear Transformers

B.1.1 Forward and Backward Pass

Algorithm 1 Forward pass for linear transformers with causal masking

```
Inputs $\phi(Q), \phi(K), V$
$\bar{V} \leftarrow 0$
$S \leftarrow 0$
for $i = 1, \ldots, N$ do
    $S \leftarrow S + \phi(K_i)\, V_i^T$                (5.14)
    $\bar{V} \leftarrow \phi(Q_i)\, S$
end for
Return $\bar{V}$
```

B.1.2 Gradient Derivation

In the first section of our supplementary material, we derive in detail the gradients for causally masked linear transformers and show that they can be computed in linear time and constant memory. In particular, we derive the gradients of a scalar loss with respect to the numerator of the following equation,

$$\hat{V}_i = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)\, V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)}. \tag{B.1}$$

The gradient with respect to the denominator and the fraction are efficiently handled by autograd. Without loss of generality, we can assume that Q and K already contain the vectors mapped by $\phi(\cdot)$, hence given the numerator

$$\bar{V}_i = Q_i^T \sum_{j=1}^{i} K_j V_j^T, \tag{B.2}$$

Algorithm 2 Backward pass for linear transformers with causal masking

```
Inputs $\phi(Q), \phi(K), V$ and $G$
$G$ is the gradient of the loss with respect to the output of algorithm 1
$S \leftarrow 0$
$\nabla_{\phi(Q)}\mathcal{L} \leftarrow 0$
for $i = 1, \ldots, N$ do
    $S \leftarrow S + \phi(K_i) V_i^T$
    $\nabla_{\phi(Q_i)}\mathcal{L} \leftarrow G_i S^T$                    (5.17)
end for
$S \leftarrow 0$
$\nabla_{\phi(K)}\mathcal{L} \leftarrow 0$
$\nabla_V \mathcal{L} \leftarrow 0$
for $i = N, \ldots, 1$ do
    $S \leftarrow S + \phi(Q_i) G_i^T$
    $\nabla_{V_i}\mathcal{L} \leftarrow S^T \phi(K_i)$                   (5.19)
    $\nabla_{\phi(K_i)}\mathcal{L} \leftarrow S V_i$                     (5.18)
end for
Return $\nabla_{\phi(Q)}\mathcal{L}$, $\nabla_{\phi(K)}\mathcal{L}$, $\nabla_V \mathcal{L}$
```

and $\nabla_{\bar{V}}\mathcal{L}$ we seek to compute $\nabla_Q \mathcal{L}$, $\nabla_K \mathcal{L}$ and $\nabla_V \mathcal{L}$. Note that $Q \in \mathbb{R}^{N \times D}$, $K \in \mathbb{R}^{N \times D}$ and $V \in \mathbb{R}^{N \times M}$. To derive the gradients, we first express the above equation for a single element without using vector notation,

$$\bar{V}_{ie} = \sum_{d=1}^{D} Q_{id} \sum_{j=1}^{i} K_{jd} V_{je} = \sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je}. \tag{B.3}$$

Subsequently we can start deriving the gradients for Q by taking the partial derivative for any Q_{lt}, as follows

$$\frac{\partial \mathcal{L}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \frac{\partial \bar{V}_{le}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \left(\sum_{j=1}^{l} K_{jt} V_{je} \right). \tag{B.4}$$

If we write the above equation as a matrix product of gradients it becomes,

$$\nabla_{Q_i} \mathcal{L} = \nabla_{\bar{V}_i} \mathcal{L} \left(\sum_{j=1}^{i} K_j V_j^T \right)^T, \tag{B.5}$$

proving (5.17). In (B.4) we made use of the fact that Q_{lt} only affects $\bar{V}_l$ hence we do not need to sum over i to compute the gradients. However, for K and V this is not the case. In particular, K_j affects all $\bar{V}_i$ where $i \geq j$. Consequently, we can write the partial derivative of the loss with respect to K_{lt} as follows,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial K_{lt}} &= \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \bar{V}_{ie}}{\partial K_{lt}} = \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \left(\sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je} \right)}{\partial K_{lt}} \\ &= \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} Q_{it} V_{le}.\end{aligned} \tag{B.6}$$

As for Q we can now write the gradient in vectorized form,

$$\nabla_{K_i} \mathcal{L} = \left(\sum_{j=i}^{N} Q_j \left(\nabla_{\bar{V}_j} \mathcal{L} \right)^T \right) V_i, \tag{B.7}$$

proving (5.18). Following the same reasoning, we can compute the partial derivative of the loss with respect to V_{lt} and prove (5.19). Note that the cumulative sum matrices for the gradient with respect to Q and K have the same size, however one is computed in the forward direction (summing from 1 to N) similarly to the forward pass and the other is computed in the backward direction (summing from N to 1) similar to backpropagation through time done in RNNs.

B.2 Clustered Transformers

B.2.1 Improved clustered attention

In this section, we first describe how we can efficiently compute *i-clustered* attention using sparse dot products with the top-k keys and values. We then present a flow chart demonstrating the same in Figure B.1.

As discussed in Section 5.5.2, the improved attention matrix approximation A^t_i for the query, Q_i belonging to the cluster j is computed as follows:

$$A^t_{il} = \begin{cases} \frac{\hat{m}_j \exp\left(Q_i K_l^T\right)}{\sum_{r=1}^{N} T_{jr} \exp\left(Q_i K_r^T\right)} & \text{if } T_{jl} = 1 \\ A^c_{il} & \text{otherwise} \end{cases}, \tag{B.8}$$

where, $T \in \{0,1\}^{C \times N}$, stores the top-$k$ keys for each cluster. $T_{ji} = 1$ if the i-th key is among the top-k keys for the j-th cluster and 0 otherwise. In addition, $\hat{m}_j$ is the total probability mass on the top-k keys for the j-th cluster given by:

$$\hat{m}_j = \sum_{r=1}^{N} T_{jr} A^c_{jr}. \tag{B.9}$$

Note that we can compute the attention weights A^t_i on the top-k keys by first taking sparse dot-product of Q_i with the top-k keys followed by the softmax activation and rescaling with the total probability mass m_j. For the rest of the keys, the attention weight is the clustered-

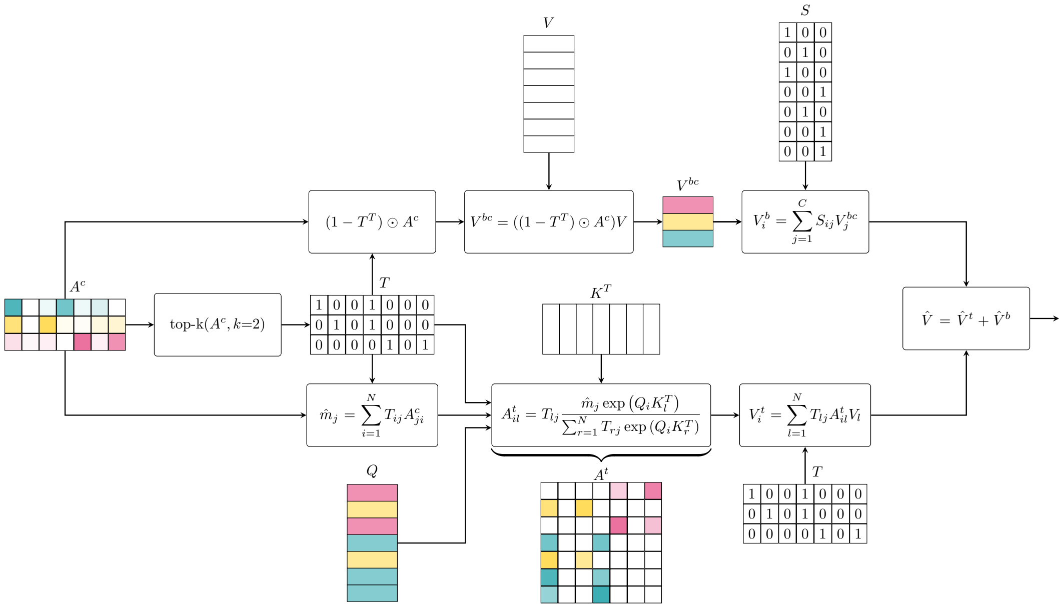

V
S
V^{bc}
$(1 - T^T) \odot A^c$
$V^{bc} = ((1 - T^T) \odot A^c)V$
$V_i^b = \sum_{j=1}^{C} S_{ij} V_j^{bc}$
$\hat{V}$
A^c
top-k(A^c, k=2)
T
K^T
$\hat{V} = \hat{V}^t + \hat{V}^b$
$\hat{m}_j = \sum_{i=1}^{N} T_{ij} A_{ji}^c$
$A_{il}^t = T_{lj} \frac{\hat{m}_j \exp(Q_i K_l^T)}{\sum_{r=1}^{N} T_{rj} \exp(Q_i K_r^T)}$
$V_i^t = \sum_{l=1}^{N} T_{lj} A_{il}^t V_l$
Q
A^t

Figure B.1: Flow-chart demonstrating the computation for *i-clustered* attention. The lower half of the figure shows the new value $\hat{V}^t$, computed by sparse dot-products with the keys K and values V corresponding to the top-k keys in T. The top half of the figure shows the computation for $\hat{V}^b$, which is the weighted average of the rest of the values with weights coming from the clustered attention A^c. The resulting values $\hat{V}$ is the sum of $\hat{V}^b$ and $\hat{V}^t$. Further details are provided in Section 5.5.2 and Appendix B.2.1.

attention weight A_i^c.

Similarly, the new values $\hat{V}_i$ can be decomposed into the following two terms,

$$\hat{V}_i = \hat{V}_i^t + \hat{V}_i^b, \tag{B.10}$$

where $\hat{V}_i^t$ is the weighted average of the values corresponding to the top-k keys with weights being the improved attention on the top-k keys. On the other hand, $\hat{V}_i^b$ is the weighted average of the rest of the values with weights being the clustered attention A_i^c. The following equations show how we compute $\hat{V}_i^t$ and $\hat{V}_i^b$,

$$\hat{V}_i^t = \sum_{l=1}^{N} T_{jl} A_{il}^t V_l, \tag{B.11}$$

$$\hat{V}_i^b = \sum_{l=1}^{N} (1 - T_{jl}) A_{il}^c V_l, \tag{B.12}$$

Note that $\hat{V}_i^t$ is weighted average of k values for each query and thus requires $\mathcal{O}(NkM)$ operations. $\hat{V}_i^b$ only needs to be computed once per-cluster centroid and thus requires $\mathcal{O}(NCM)$ operations.

In Figure B.1 we present the *i-clustered* attention computation for the same example sequence with 8 queries and the number of clusters and top-k keys set to 3. The lower half of the figure shows the new value $\hat{V}^t$, computed by first taking sparse dot-products with the top 3 keys to get the attention weights. This is followed by taking the weighted average of the 3 corresponding values. The top half of the figure shows the $\hat{V}^b$ computation. This is same as clustered attention computation but with attention weights corresponding to top 3 keys set to 0 for A^c. The resulting values $\hat{V}$ is the sum of $\hat{V}^b$ and $\hat{V}^t$.

B.2.2 Quality of the approximation

Proposition 3. *For the i-th query belonging to the j-th cluster, the improved clustered attention A_i^t and clustered attention A_j^c relate to the full attention A_i as follows,*

$$\left\| A_i^t - A_i \right\|_1 \leq \left\| A_j^c - A_i \right\|_1 \tag{B.13}$$

Proof. As discussed before, the improved attention matrix approximation A_i^t for the query, Q_i is computed as follows:

$$A_{il}^t = \begin{cases} \frac{\hat{m}_j \exp\left(Q_i K_l^T\right)}{\sum_{r=1}^N T_{jr} \exp\left(Q_i K_r^T\right)} & \text{if } T_{jl} = 1 \\ A_{il}^c & \text{otherwise} \end{cases}, \tag{B.14}$$

where, $T \in \{0,1\}^{C \times N}$, stores the top-$k$ keys for each cluster, $T_{ji} = 1$ if the i-th key is among the top-k keys for the j-th cluster and 0 otherwise. $\hat{m}_j$ is the total probability mass on the top-k keys for the j-th cluster, computed as follows:

$$\hat{m}_j = \sum_{r=1}^N T_{jr} A_{jr}^c. \tag{B.15}$$

Given the full attention A_i, (B.14) can be simplified to

$$A_{il}^t = \begin{cases} \frac{\hat{m}_j}{m_i} A_{il} & \text{if } T_{jl} = 1 \\ A_{il}^c & \text{otherwise} \end{cases}, \tag{B.16}$$

where, m_i is the total probability mass on the same top-k keys for the i-th query, computed using the true attention A_i, as follows:

$$m_i = \frac{\sum_{r=1}^N T_{jr} \exp\left(Q_i K_r^T\right)}{\sum_{r=1}^N \exp\left(Q_i K_r^T\right)} \tag{B.17}$$

$$= \sum_{r=1}^N T_{jr} A_{ir}. \tag{B.18}$$

Without loss of generality, let us assume, $T_{jl} = 1 \quad \forall \quad l \in \{1, \ldots, k\}$ and $T_{jl} = 0 \quad \forall \quad l \in \{k+$

$1,\dots,N\}$.

In this case, (B.16) can be written as:

$$A^t_{il} = \begin{cases} \frac{\hat{m}_j}{m_i} A_{il} & \text{if} \quad l \leq k \\ A^c_{il} & \text{if} \quad l \geq k+1 \end{cases}. \tag{B.19}$$

The total probability masses on the top-k keys, m_i and $\hat{m}_j$ can now be expressed as:

$$m_i = \sum_{r=1}^{k} A_{ir}. \tag{B.20}$$

$$\hat{m}_j = \sum_{r=1}^{k} A^c_{jr}. \tag{B.21}$$

From (B.19) it is clear that the clustered attention, A^c_i, and the improved clustered attention, A^t_i, only differ on the keys $\{1,\dots,k\}$. Thus, it suffices to show that A^t_i has lower approximation error on these keys. The approximation error on the top-k keys $\{1,\dots,k\}$, let it be e_t, between the *i-clustered* attention and the *full* attention is as follows:

$$\begin{aligned} e_t &= \sum_{l=1}^{k} \left| A_{il} - A^t_{il} \right| && \text{(B.22)} \\ &= \sum_{l=1}^{k} \left| A_{il} - A_{il} \frac{\hat{m}_j}{m_i} \right| && \text{(B.23)} \\ &= \sum_{l=1}^{k} A_{il} \left| 1 - \frac{\hat{m}_j}{m_i} \right| && \text{(B.24)} \\ &= \left| 1 - \frac{\hat{m}_j}{m_i} \right| \sum_{l=1}^{k} A_{il} && \text{(B.25)} \\ &= m_i \left| 1 - \frac{\hat{m}_j}{m_i} \right| && \text{(B.26)} \\ &= \left| m_i - \hat{m}_j \right| && \text{(B.27)} \\ &= \left| \sum_{l=1}^{k} A_{il} - A^c_{jl} \right| && \text{(B.28)} \\ &\leq \sum_{l=1}^{k} \left| A_{il} - A^c_{jl} \right| && \text{(B.29)} \end{aligned}$$

Therefore,

$$\left\|A_i - A_i^t\right\|_1 = \sum_{l=1}^{k} \left|A_{il} - A_{il}^t\right| + \sum_{l=k+1}^{N} \left|A_{il} - A_{il}^t\right| \tag{B.30}$$

$$= \sum_{l=1}^{k} \left|A_{il} - A_{il}^t\right| + \sum_{l=k+1}^{N} \left|A_{il} - A_{jl}^c\right| \tag{B.31}$$

$$\leq \sum_{l=1}^{k} \left|A_{il} - A_{jl}^c\right| + \sum_{l=k+1}^{N} \left|A_{il} - A_{jl}^c\right| \tag{B.32}$$

$$\leq \left\|A_i - A_i^c\right\|_1 \tag{B.33}$$

□

B.3 Experiments

B.3.1 Synthetic Tasks

Sequence Duplication

To examine the convergence properties of *linear transformers* we train on an artificial copy task with causal masking. Namely, the transformers have to copy a series of symbols similar to the sequence duplication task of Kitaev et al. (2020). We use a sequence of maximum length 128 with 10 different symbols separated by a dedicated separator symbol. For all three methods, we train a 4 layer transformer with 8 attention heads using a batch size of 64 and the RAdam optimizer (Liu et al., 2020a) with a learning rate of 10^{-3} which is reduced to 10^{-4} after 3000 updates. Figure B.2 depicts the loss with respect to the number of gradient steps. We observe that linear converges smoothly and reaches a lower loss than lsh due to the lack of noise introduced by hashing. In particular, it reaches the same loss as softmax.

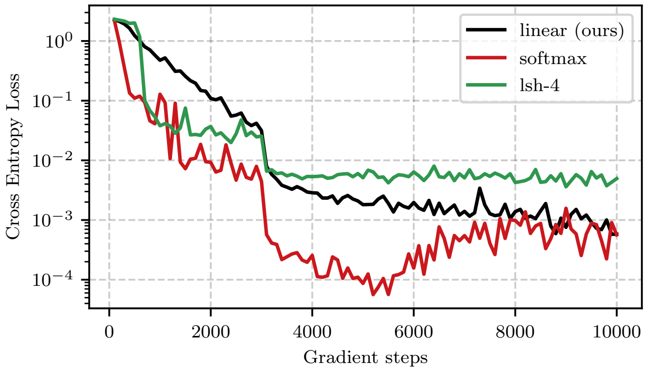
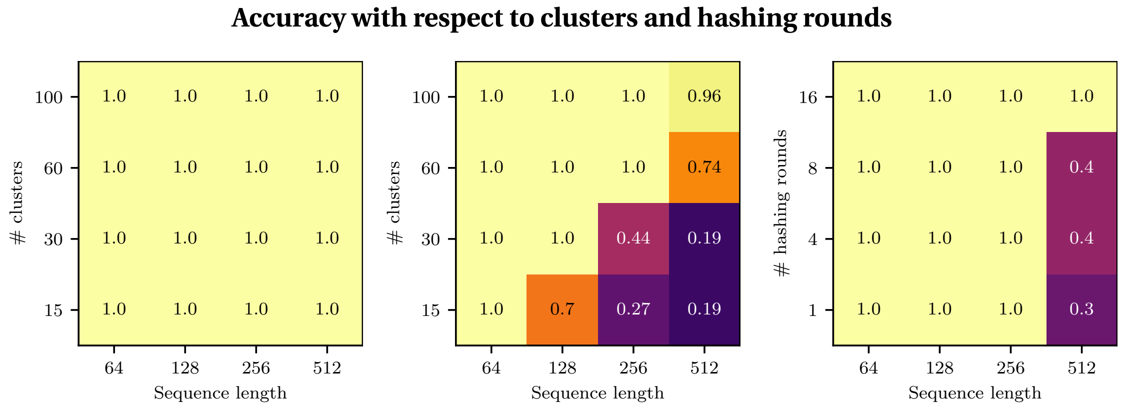


Figure B.2: Convergence comparison of *softmax*, *linear* and *reformer* attention on a sequence duplication task. *linear* converges stably and reaches the same final performance as softmax. The details of the experiment are in Appendix B.3.1.



(a) Improved clustered (b) Clustered (c) Reformer

Figure B.3: The heatmaps depict the achieved accuracy on an artificial copy task (Appendix B.3.1) as the sequence length, the number of clusters and the number of hashing rounds varies. Improved clustered (Figure B.3a) is the only fast transformer variant that can solve the task perfectly for any sequence length and number of clusters combination.

Ablation on clusters and sequence length

Following (Kitaev et al., 2020), we introduce a synthetic task to analyze the relationship between the number of clusters and sequence length. In our task, the transformer models need to copy some symbols that are masked out from either the first or second half of the sequence. In particular, we generate a random sequence of tokens and we prepend a unique separator token, let it be 0. The sequence is then copied to get a target of the form $0w0w$, where $w \in \{1,\dots,C\}^L$, C is the number of possible symbols and L is the sequence length. To generate the input, we replace some symbols from the first half of the sequence and some different symbols from the second half, such that the target sequence can be reconstructed from the input. An example of an input output pair with $L = 4$ can be seen in Figure B.4. Note that to solve this task, transformers simply need to learn to attend to the corresponding tokens in the two identical halves of the sequence.

Input	0	4	M	2	2	0	4	5	M	2
Output	0	4	5	2	2	0	4	5	2	2

Figure B.4: Example of an input and output pair for the masked copy task. M denotes the masked out tokens.

We set the sequence length L to one of $\{31, 63, 127, 255\}$ which means the input length varies between $N = 2^6$ and $N = 2^9$. For each sequence, we sample tokens uniformly from $\{1,\dots,10\}$ and randomly mask out 20% of the tokens. To analyze the impact of number of clusters on performance, we train full transformer as well as clustered variants with different number of clusters and Reformer with different number of hashing rounds.

All transformer variants consist of 4 layers, 6 attention heads, embedding dimension of 32 for each head, and feed-forward dimension of 768. For both clustered and improved clustered attention, we set the number of bits for LSH to 63 and the number of Lloyd iterations for the K-Means to 10. Both clustered and improved clustered attention are trained with 15, 30, 60 and 100 clusters. We also train Reformer with 1, 4, 8 and 16 hashing rounds. Finally, all models are trained using R-Adam optimizer (Liu et al., 2020b) with a learning rate of 0.0002, batch size of 32 for 5000 iterations.

In Figure B.3, we illustrate the results of this experiment as heatmaps depicting the achieved accuracy for a given combination of number of clusters and sequence length for clustered transformers and number of hashing rounds and sequence length for Reformer. Note that the vanilla transformer solves the task perfectly for all sequence lengths. We observe that both clustered (Figure B.3b) and Reformer (Figure B.3c) require more clusters or more rounds as the sequence length increases. However, improved clustered achieves the same performance as vanilla transformers, namely *perfect accuracy*, for every number of clusters and sequence length combination. This result increases our confidence that the required number of clusters for our method is not a function of the sequence length but of the task at hand.

B.3.2 Autoregressive Image Generation: Training Evolution

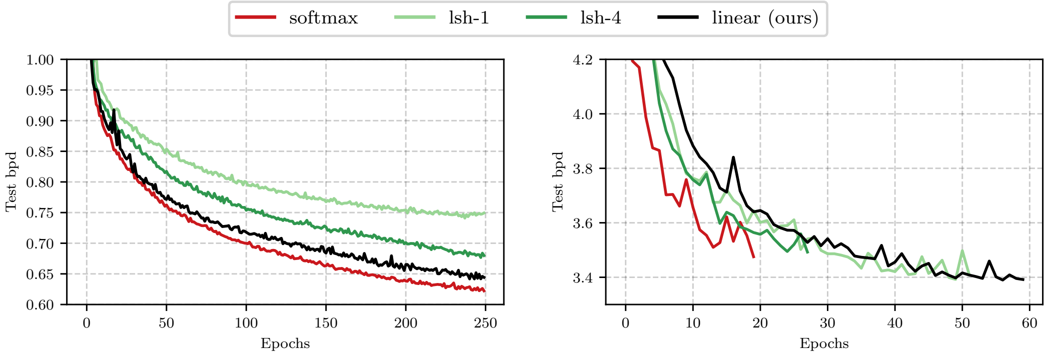
In Figure B.5 we present the training evolution of all transformer models in our experiments. For the MNIST experiment (Figure B.5a) we train all methods for 250 epochs. The sequence length is small enough so that the training time does not vary significantly for all methods. We observe that our method converges on par with softmax attention outperforming significantly both reformer variants.

On the other hand, for CIFAR-10 (Figure B.5b) we train all methods for a fixed amount of time, namely 7 days. We observe that *lsh-1* and *linear* complete significantly more epochs than softmax and lsh-4 and achieve better performance. This gap is expected to increase with a further increase in sequence length.

B.3.3 Autoregressive Image Generation: Throughput Discussion

Stateful softmax attention

In Section 5.7.2, we report the image generation throughput and we compare with **softmax** transformer and **lsh**. In this section we create another baseline, denoted as **stateful-softmax**, that implements a softmax autoregressive transformer as a recurrent model. Namely, all the keys and values are saved and then passed to the model again when predicting the next element of the sequence. The state of this recurrent model is the set of keys and values which has size proportional to the sequence length. This is qualitatively different to our proposed model that has a state with fixed dimensions and computing the i-th state given the previous one has fixed computational cost regardless of i.



(a) MNIST

(b) CIFAR-10

Figure B.5: Training evolution of transformers for our autoregressive image generation experiments. It can be observed that *linear transformers* converge consistently faster than Reformer and in the autoregressive experiments on par with softmax. For MNIST all methods are trained for 250 epochs while for CIFAR we train for 7 days. The details of the experiments can be found in Section 5.7.2.

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
Stateful-softmax	0.621	7.56	(16.8×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

(a) Image generation on MNIST

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
Stateful-softmax	3.47	0.32	(80×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

(b) Image generation on CIFAR-10

Table B.1: Comparison of autoregressive image generation throughput of MNIST and CIFAR-10 images. The experiment can be found in Section 5.7.2. For stateful-softmax we save the keys and values and reuse them for predicting the next element. A detailed description of this extra baseline can be found in Appendix B.3.3.

Table B.1 summarizes the results. We observe that stateful-softmax is significantly faster than vanilla transformers. However, its complexity is still quadratic with respect to the sequence length and our formulation is more than 50× faster for CIFAR-10. Moreover, we would like to point out that implementing a similar stateful attention for Reformer is not a trivial task as the sorting and chunking operations need to be performed each time a new input is provided.

Equalizing the batch size

In the previous sections we evaluate the throughput of all transformer variants for the task of autoregressive image generation. However, another important factor to consider is latency, namely the total time required to produce a single image. To this end, we use a batch size of 1 and measure the time required by all methods to generate a single image. In addition to running the inference on the GPU, we also evaluate the time required on CPU. The results are reported in Table B.2.

Method	Seconds (CPU)		Seconds (GPU)	
Softmax	72.6	(13.2×)	10.2	(1.4×)
Stateful-softmax	7.4	(1.3×)	10.4	(1.42×)
LSH-1	46.0	(8.3×)	19.2	(2.6×)
LSH-4	112.0	(20×)	55.8	(7.6×)
Linear (ours)	**5.5**	(1×)	**7.3**	(1×)

(a) Image generation on MNIST

Method	Seconds (CPU)		Seconds (GPU)	
Softmax	8651.4	(191.8×)	300.1	(4.9×)
Stateful-softmax	71.9	(1.6×)	70.4	(1.14×)
LSH-1	2318.9	(51.4×)	221.6	(3.6×)
LSH-4	5263.7	(116.7×)	683.9	(11.1×)
Linear (ours)	**45.1**	(1×)	**61.3**	(1×)

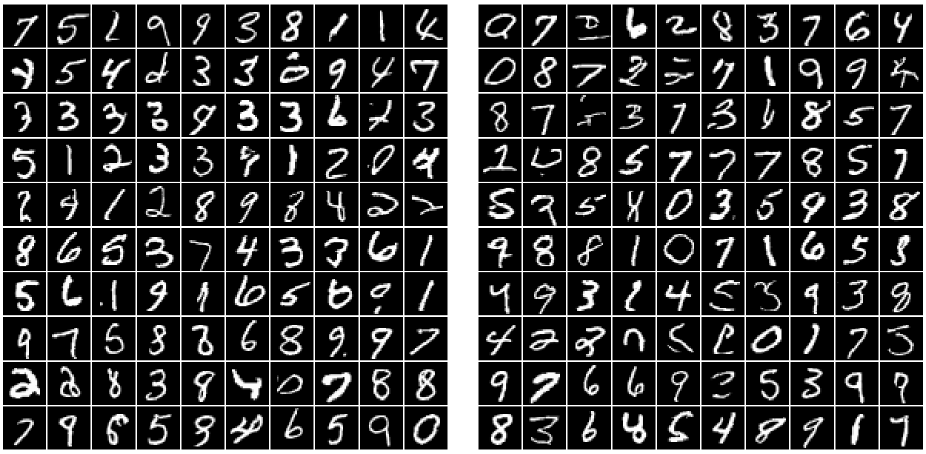
(b) Image generation on CIFAR-10

Table B.2: Comparison of the time required to generate a single image with autoregressive transformers on MNIST and CIFAR-10. We run all methods with a batch size of 1 both on CPU and GPU and report the total time in seconds. For all numbers in the table, lower is better.

We observe that all methods underutilize the GPU and achieve significantly smaller image generation throughput than the one shown in Table B.1. The proposed linear transformer is faster than all the methods and in particular it is almost 6.6× faster than softmax transformers for generating an image on CIFAR-10. Note that our linear autoregressive transformer is the only method faster on the CPU than on the GPU in every case. This is due to the fact that computing the attention as an RNN has such a low cost that the main computational bottleneck becomes the inevitable outer loop over the sequence.

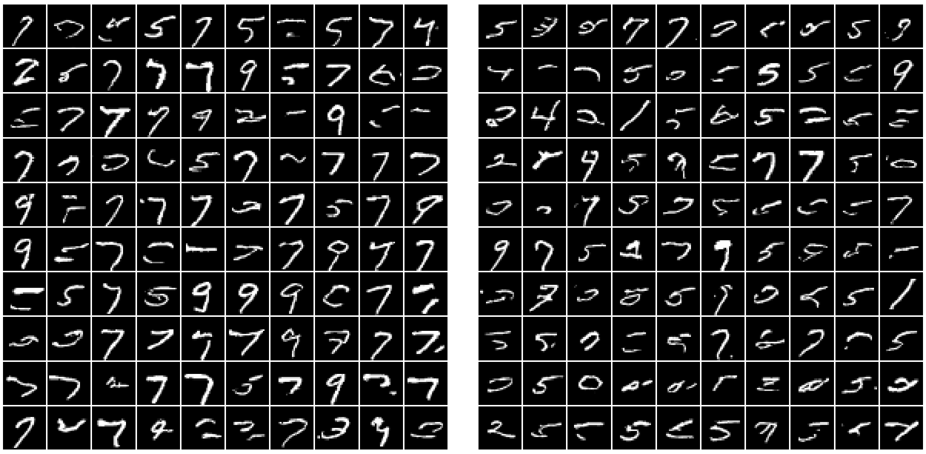
B.3.4 Autoregressive Image Generation: Qualitative Results

In this section, we provide qualitative results for our image generation experiments. Since the perplexity of all models is approximately the same, as expected, the qualitative differences are not significant. A rather interesting observation, however, is that the Reformer models provide significantly fewer variations in their unconditional samples. Moreover, we observe that image completion is a considerably easier task than unconditional generation as all models perform substantially better.



(a) Softmax

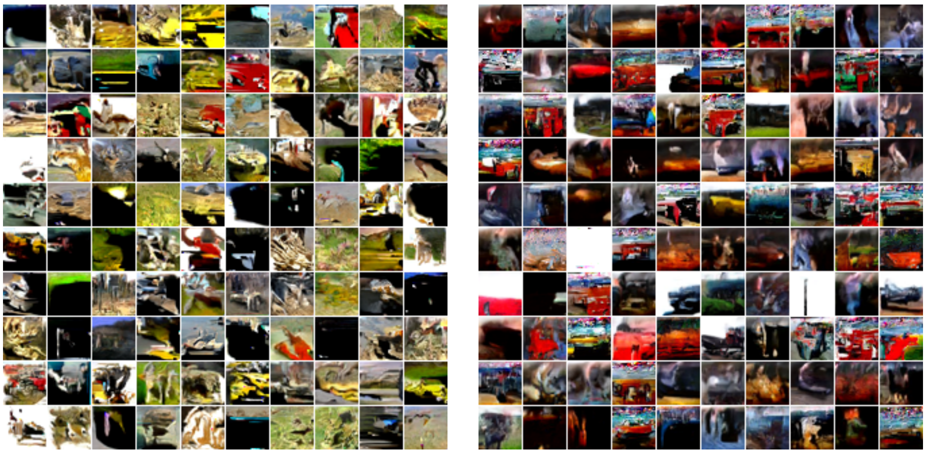
(b) Linear (ours)



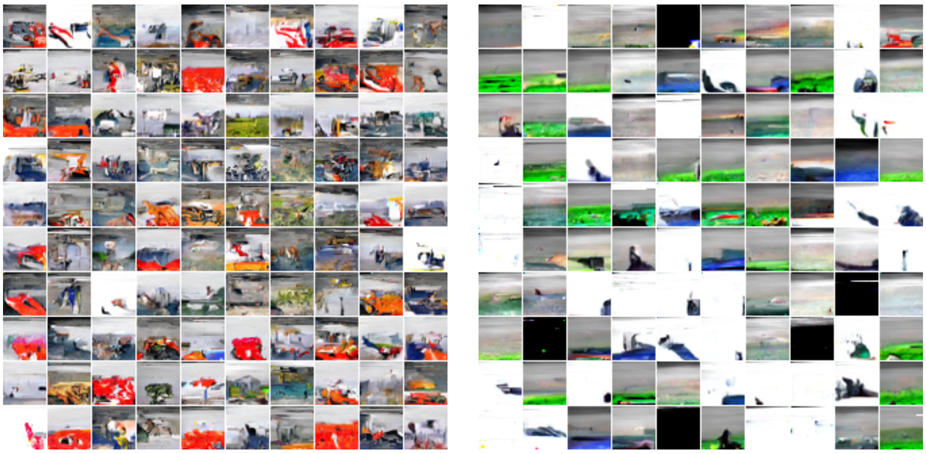
(c) LSH-1

(d) LSH-4

Figure B.6: Unconditional samples from the transformer models trained with MNIST. See Section 5.7.2 for more details.

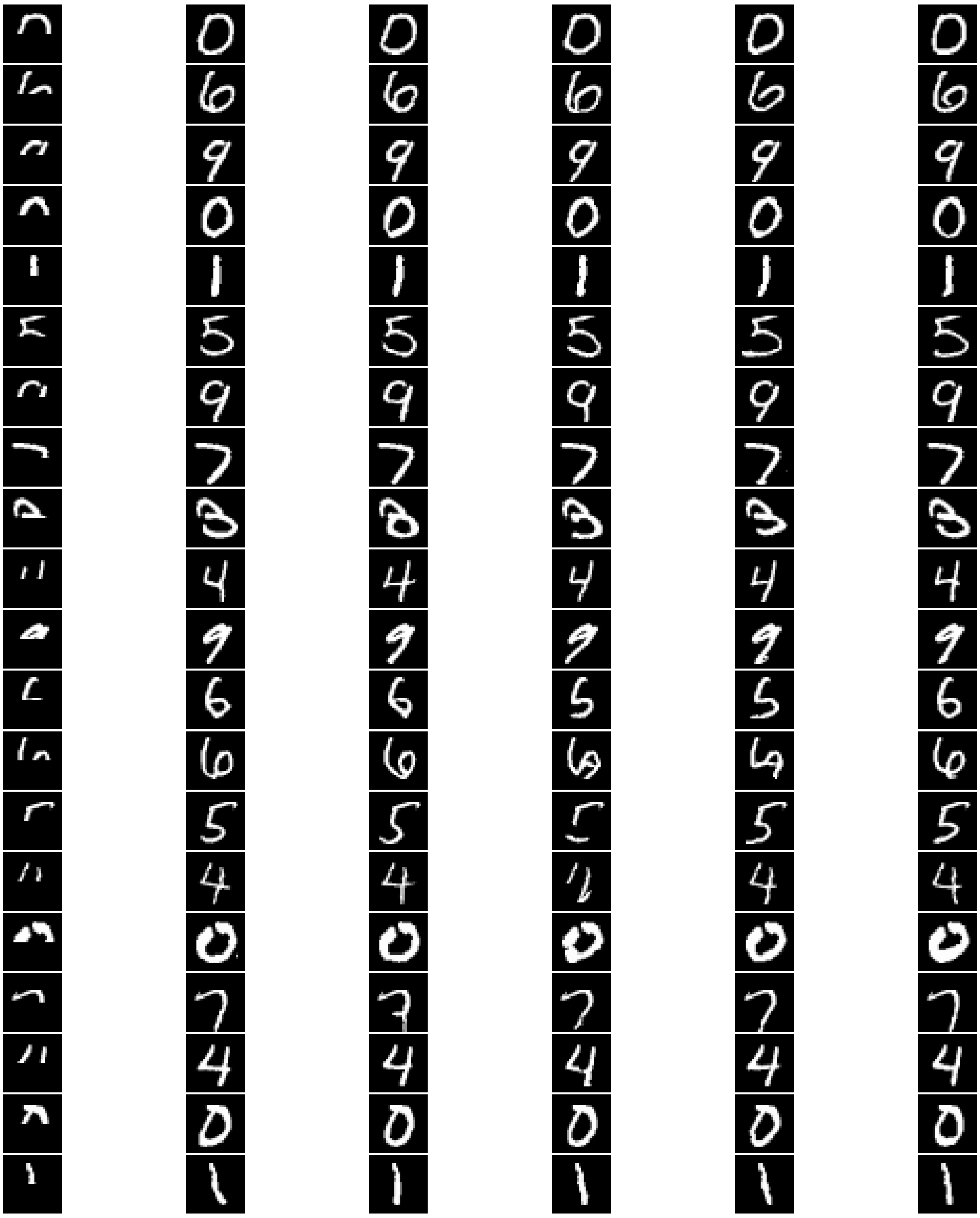


(a) Softmax (b) Linear (ours)



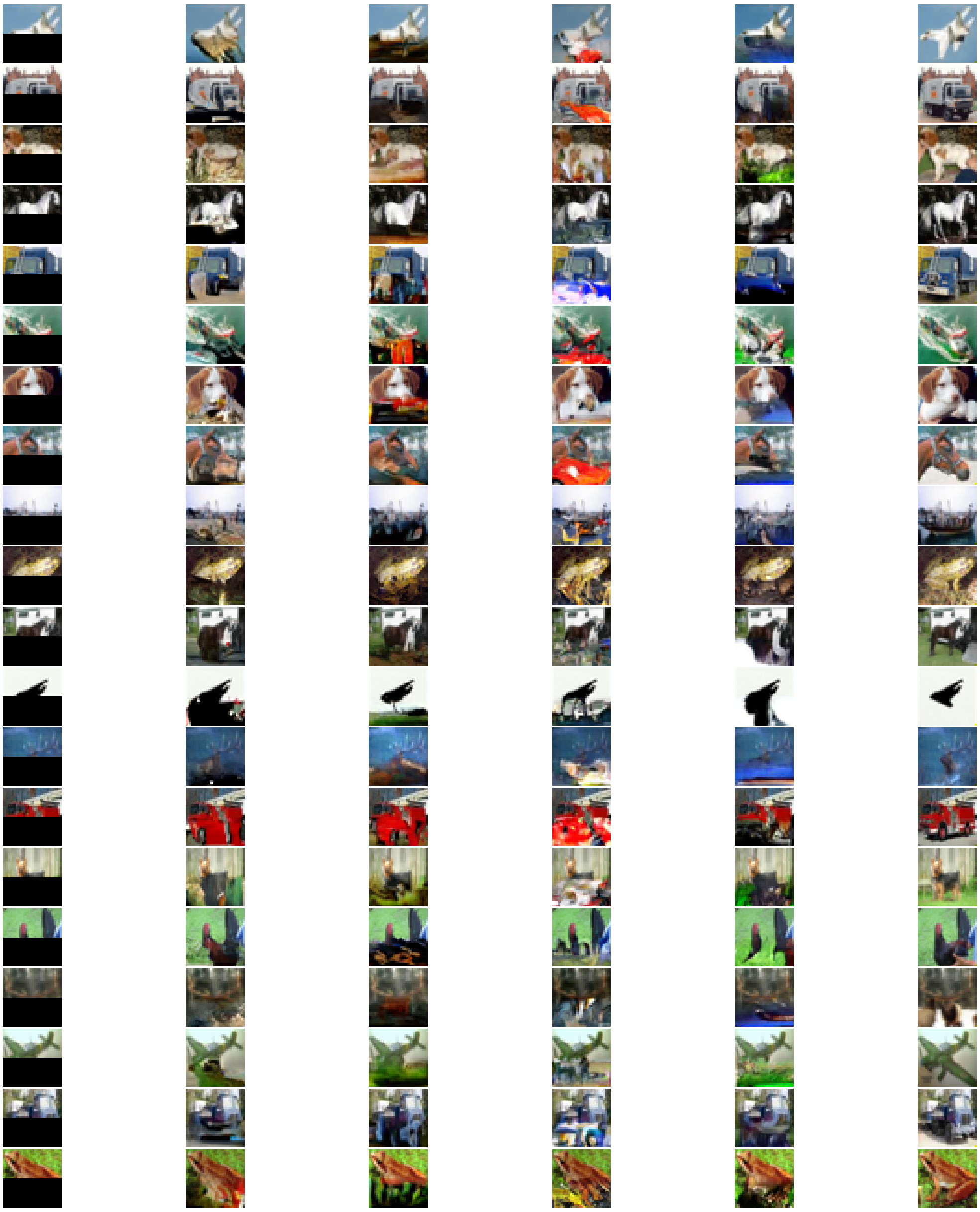
(c) LSH-1 (d) LSH-4

Figure B.7: Unconditional samples from the transformer models trained with CIFAR-10. See Section 5.7.2 for more details.



(a) Occluded (b) Softmax (c) Linear (ours) (d) LSH-1 (e) LSH-4 (f) Original

Figure B.8: MNIST digit completion from all trained models. See Section 5.7.2 for more details.



(a) Occluded (b) Softmax (c) Linear (ours) (d) LSH-1 (e) LSH-4 (f) Original

Figure B.9: CIFAR-10 image completions from all trained transformer models. See Section 5.7.2 for more details.

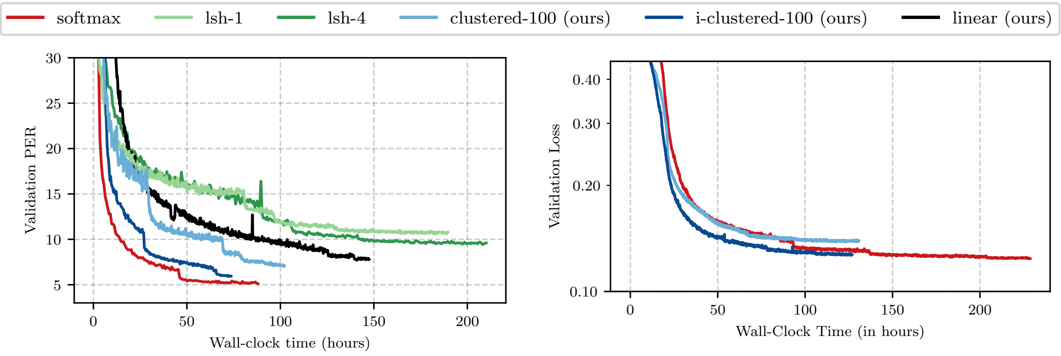
B.3.5 Automatic Speech Recognition

In this section, we present the details for the ASR experiments such as transformer architecture, optimizer, and learning rate schedule. As mentioned in Section 5.7, for *i-clustered*, unless specified otherwise, k is set to 32. Furthermore, all transformers have 6 heads with an embedding dimension of 32 for each head and the feed-forward dimension of 768. Other architectural details specific to each experiment are described in the corresponding section.

Wall Street Journal

Convergence Behaviour:

For this experiment, we train a transformer with the softmax, linear, clustered, and Reformer attention variants. All models consist of 9 layers. For Reformer, we train two variants with 1 and 4 rounds of hashing with chunk size fixed to 32 as suggested. For clustered and improved clustered attention, we set the number of clusters to 100. We also set the number of Lloyd iterations for K-Means to 10 and the bits for LSH to 63. All models are trained to convergence using the R-Adam optimizer (Liu et al., 2020b) with a learning rate of 0.0001, max gradient norm set to 10.0, and weight decay of 0.01. The learning rate is dropped when the validation loss plateaus. For each model, we select the largest batch size that fits the GPU.



(a) Wall Street Journal (b) Switchboard

Figure B.10: We show training/validation loss convergence for different transformer variants. Only *i-clustered* has a faster or comparable wall-clock convergence to softmax attention. Both the linear and clustered variants have a significantly better convergence than both *lsh*-1 and *lsh*-4. Note that due to a smaller batch size *softmax* makes many more updates than all other transformer variants. More details can be found in Appendix B.3.5 and Appendix B.3.5.

In Figure B.10a, we show the training loss convergence for different transformer variants. It can be seen that softmax significantly outperforms Reformer, clustered, and linear in terms of convergence. Furthermore, we see that *i-clustered* has a much faster convergence than the *clustered* attention. This shows that the improved clustered attention indeed approximates the softmax attention better. More importantly, only the *i-clustered* attention has a comparable wall-clock convergence. Since *softmax* has a much smaller batch size, it makes many more updates per epoch. We think that a slightly smaller batch size with more updates would have been a better choice for the clustered transformers w.r.t. the wall-clock convergence. This is reflected in the Switchboard experiments, where the batch sizes for the clustered variants were smaller due to more layers. Finally, as seen from the wall-clock convergence, the linear and the clustered transformers significantly outperform Reformer.

Speed-Accuracy Tradeoff:

As described in Section 5.6.1, for this task we additionally train *full* with 4 and 6 layers. Similarly, we train *clustered* with 9 layers, and 200 and 300 clusters. We also train an *i-clustered* model with 9 layers and 200 clusters, and smaller models with 6 layers, and 100 and 200 clusters.

For *clustered* and *i-clustered* variants with 9 layers, we fine-tuned the previously described models trained with 100 clusters. We fine-tuned for 15 epochs with a learning rate of 0.00001. We train *full* with 4 and 6 layers to convergence in a similar fashion to the *full* with 9 layers described previously. Finally, for *i-clustered*, we first trained a model with 6 layers and 100 clusters using the training strategy used for 9 layers and 100 clusters. We then fine-tuned this model for 15 epochs using 200 clusters and a learning rate of 0.00001.

Switchboard

Convergence Behaviour:

For this experiment, we train a transformer with the full and clustered attention variants. All models consist of 12 layers. For the clustered and improved clustered attention, we set the number of clusters to 100. We also set the number of Lloyd iterations for K-Means to 10 and the bits for LSH to 63.

Following standard practice for flat-start lattice-free MMI training, we train over multiple GPUs with weight averaging for synchronization as described in (Povey et al., 2015). Specifically, we modify the *e2e* training recipe for the **Wall Street Journal** in Kaldi (Povey et al., 2011) with the following two key differences: first, the acoustic model training is done in PyTorch and second, we use RAdam optimizer instead on natural stochastic gradient descent.

All models are trained using the R-Adam optimizer with a learning rate of 0.0002, max gradient norm set to 10.0, and weight decay of 0.01. The learning rate is dropped when the validation loss plateaus. We use the word error rate (WER) on the validation set for early stopping and model selection. The *full* attention model is trained with a batch size of 2 while the clustered variants: *clustered* and *i-clustered* are trained with a batch size of 6.

In Figure B.10b, we show the training loss convergence for different transformer variants. It can be seen that *i-clustered* has the fastest convergence for this setup. Note that the overall training time for *clustered* attention is still less than that of *full* as it starts to overfit early on the validation set WER.

Speed-Accuracy Tradeoff:

For this task, we additionally train *full* with 6 and 8 layers. Similarly, we train *clustered* with 12 layers, and 200 and 300 clusters. We also train *i-clustered* with 12 layer and 200 clusters, and smaller models with 8 layers, and 100 and 200 clusters.

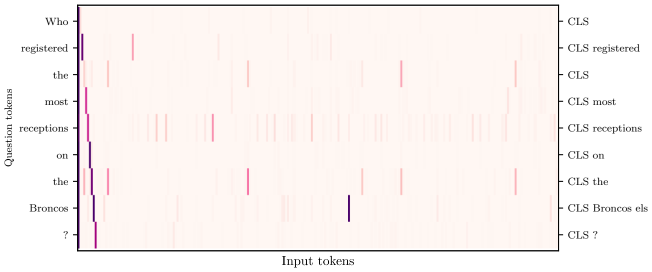
For the *clustered* and *i-clustered* variants with 12 layers, we fine-tuned the previously described models trained with 100 clusters. We fine-tuned for 5 epochs with a learning rate of 0.00001. Once again, *full* with 6 and 8 layers were trained to convergence similarly to *full* with 12 layers described previously. Finally, for *i-clustered* with 8 layers, we first train a model with 100 clusters using the training strategy used for 12 layers and 100 clusters. We then fine-tuned this model for 5 epochs using 200 clusters and a learning rate of 0.00001.

B.3.6 RoBERTa Approximation

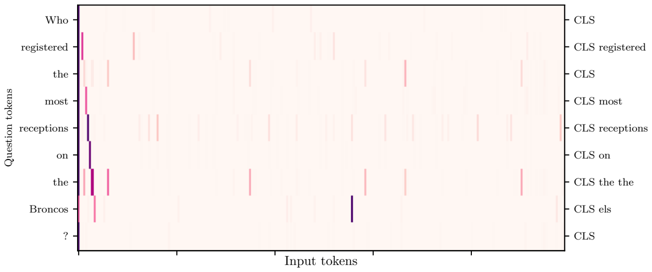
In this section, we provide a qualitative comparison between the *full* attention, and the clustered attention variants *clustered* and *i-clustered* used for approximation. As described in Section 5.7.3, we use 25 clusters for both attention variants. In Figure B.11 we show the attention distribution for the question tokens for a randomly selected question-context tuple from the SQuAD dataset. For each token in the question we show the attention distribution over the input sequence formed by concatenating the question and context tokens with *CLS* and *SEP* tokens appended. It can be seen that with only a few clusters, improved clustered attention approximates the softmax attention very closely even when the attention distribution has complicated and sparse patterns. In contrast, clustered attention fails to approximate such attention distributions. Moreover, we can further see that for almost all question tokens; both full and improved clustered have the same tokens with the highest attention weights. This further strengthens our belief that improved clustered attention can approximate a wide range of complicated attention patterns.

Manning finished the year with a career-low 67.9 passer rating, throwing for 2,249 yards and nine touchdowns, with 17 interceptions. In contrast, Osweiler threw for 1,967 yards, 10 touchdowns and six interceptions for a rating of 86.4. Veteran receiver Demaryius Thomas led the team with 105 receptions for 1,304 yards and six touchdowns, while Emmanuel Sanders caught 76 passes for 1,135 yards and six scores, while adding another 106 yards returning punts. Tight end Owen Daniels was also a big element of the passing game with 46 receptions for 517 yards. Running back C. J. Anderson was the team's leading rusher 863 yards and seven touchdowns, while also catching 25 passes for 183 yards. Running back Ronnie Hillman also made a big impact with 720 yards, five touchdowns, 24 receptions, and a 4.7 yards per carry average. Overall, the offense ranked 19th in scoring with 355 points and did not have any Pro Bowl selections.

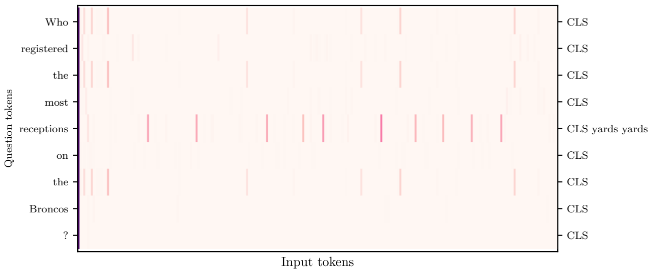
(a) *context*



(b) *full*



(c) *improved-clustered*



(d) *clustered*

Figure B.11: Attention matrices for question-context tuples for *full* attention, and *clustered* and *i-clustered* attention used for approximation. Figure B.11a shows the context for the question with answer highlighted in red. Figure B.11b shows the attention distributions for *full*, Figure B.11c and Figure B.11d show the approximation using *i-clustered* and *clustered* respectively. Note that *i-clustered* has attention patterns very similar to *full* while *clustered* shows qualitatively different attention patterns. For each question token, we also present the tokens with highest attention above a threshold on the right axis. For more information refer to Appendix B.3.6.

Bibliography

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., and Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA. USENIX Association.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J. H., Fan, L., Fougner, C., Hannun, A. Y., Jun, B., Han, T., LeGresley, P., Li, X., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Qian, S., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, C., Wang, Y., Wang, Z., Xiao, B., Xie, Y., Yogatama, D., Zhan, J., and Zhu, Z. (2016a). Deep speech 2 : End-to-end speech recognition in english and mandarin. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 173–182. JMLR.org.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J. H., Fan, L., Fougner, C., Hannun, A. Y., Jun, B., Han, T., LeGresley, P., Li, X., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Qian, S., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, C., Wang, Y., Wang, Z., Xiao, B., Xie, Y., Yogatama, D., Zhan, J., and Zhu, Z. (2016b). Deep speech 2 : End-to-end speech recognition in english and mandarin. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 173–182. JMLR.org.

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1120–1128. JMLR.org.

Asavari and Chhabra, M. (2019). Speech recognition market by deployment mode and end use: Global opportunity analysis and industry forecast, 2019–2026.

Audhkhasi, K., Saon, G., Tüske, Z., Kingsbury, B., and Picheny, M. (2019). Forget a bit to learn better: Soft forgetting for ctc-based automatic speech recognition. In Kubin, G. and Kacic, Z., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2618–2622. ISCA.

Ba, L. J., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

Baevski, A., Hsu, W., Xu, Q., Babu, A., Gu, J., and Auli, M. (2022). data2vec: A general framework for self-supervised learning in speech, vision and language. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 1298–1312. PMLR.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Bahdanau, D., Cho, K., and Bengio, Y. (2015a). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Bahdanau, D., Cho, K., and Bengio, Y. (2015b). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.*

Bahl, L. R., Brown, P. F., de Souza, P. V., and Mercer, R. L. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1986, Tokyo, Japan, April 7-11, 1986*, pages 49–52. IEEE.

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, pages 164 – 171.

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Bengio, Y., Frasconi, P., and Simard, P. Y. (1993). The problem of learning long-term dependencies in recurrent networks. In *Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993*, pages 1183–1188. IEEE.

Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde-Farley, D., Goodfellow, I., Bergeron, A., et al. (2011). Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3.

Billa, J. (2017). Improving LSTM-CTC based ASR performance in domains with limited training data. *CoRR*, abs/1707.00722.

Blanc, G. and Rendle, S. (2018). Adaptive sampled softmax with kernel based sampling. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 589–598. PMLR.

Bourlard, H. A. and Morgan, N. (1993). *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, USA.

Britz, D., Guan, M. Y., and Luong, M. (2017). Efficient attention using a fixed-size memory representation. In Palmer, M., Hwa, R., and Riedel, S., editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 392–400. Association for Computational Linguistics.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). 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 2016, Shanghai, China, March 20-25, 2016*, pages 4960–4964. IEEE.

Chang, R. W. and Hancock, J. C. (1966). On receiver structures for channels having memory. *IEEE Trans. Inf. Theory*, 12(4):463–468.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Chiu, C. and Raffel, C. (2018). Monotonic chunkwise attention. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In Wu, D., Carpuat, M., Carreras, X., and Vecchi, E. M., editors, *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111. Association for Computational Linguistics.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. (2021).

Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Chung, Y., Hsu, W., Tang, H., and Glass, J. R. (2019). An unsupervised autoregressive model for speech representation learning. In Kubin, G. and Kacic, Z., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 146–150. ISCA.

Chung, Y., Zhang, Y., Han, W., Chiu, C., Qin, J., Pang, R., and Wu, Y. (2021). w2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2021, Cartagena, Colombia, December 13-17, 2021*, pages 244–250. IEEE.

Cieri, C., Miller, D., and Walker, K. (2004). The fisher corpus: a resource for the next generations of speech-to-text. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association.

Clevert, D., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Collobert, R., Bengio, S., and Mariéthoz, J. (2002). Torch: a modular machine learning software library. Idiap-RR Idiap-RR-46-2002, IDIAP.

Conneau, A., Baevski, A., Collobert, R., Mohamed, A., and Auli, M. (2021). Unsupervised cross-lingual representation learning for speech recognition. In Hermansky, H., Cernocký, H., Burget, L., Lamel, L., Scharenborg, O., and Motlícek, P., editors, *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 2426–2430. ISCA.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Trans. Speech Audio Process.*, 19(4):788–798.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Dey, S., Motlícek, P., Bui, T., and Dernoncourt, F. (2019). Exploiting semi-supervised training through a dropout regularization in end-to-end speech recognition. In Kubin, G. and Kacic, Z., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 734–738. ISCA.

Dighe, P. (2019). *Sparse and Low-rank Modeling for Automatic Speech Recognition*. PhD thesis.

Dong, L., Xu, S., and Xu, B. (2018). Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 5884–5888. IEEE.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Evermann, G. and Woodland, P. C. (2000). Large vocabulary decoding and confidence estimation using word posterior probabilities. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*, pages 1655–1658. IEEE.

Fan, A., Grave, E., and Joulin, A. (2020). Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Fiscus, J. (1997). A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 347–354.

Fortune Business Insights (2019). Speech and voice recognition market size, share & industry analysis.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org.

Gales, M. J. F., Knill, K. M., Ragni, A., and Rath, S. P. (2014). Speech recognition and keyword spotting for low-resource languages: Babel project research at CUED. In *4th Workshop on Spoken Language Technologies for Under-resourced Languages, SLTU 2014, St. Petersburg, Russia, May 14-16, 2014*, pages 16–23. ISCA.

Gao, B. and Pavel, L. (2017). On the properties of the softmax function with application in game theory and reinforcement learning. *CoRR*, abs/1704.00805.

Godfrey, J. J., Holliman, E., and McDaniel, J. (1992). SWITCHBOARD: telephone speech corpus for research and development. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '92, San Francisco, California, USA, March 23-26, 1992*, pages 517–520. IEEE Computer Society.

Goodman, J. (2001). Classes for fast maximum entropy training. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*, pages 561–564. IEEE.

Graves, A., Fernández, S., Gomez, F. J., and Schmidhuber, J. (2006a). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Cohen, W. W. and Moore, A. W., editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Graves, A., Fernández, S., Gomez, F. J., and Schmidhuber, J. (2006b). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Cohen, W. W. and Moore, A. W., editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Grézl, F. and Karafiát, M. (2013). Semi-supervised bootstrapping approach for neural network feature extractor training. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 470–475. IEEE.

Gulati, A., Qin, J., Chiu, C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. (2020). Conformer: Convolution-augmented transformer for speech recognition. In Meng, H., Xu, B., and Zheng, T. F., editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5036–5040. ISCA.

Guo, P., Boyer, F., Chang, X., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Li, C., Garcia-Romero, D., Shi, J., Shi, J., Watanabe, S., Wei, K., Zhang, W., and Zhang, Y. (2021). Recent developments on espnet toolkit boosted by conformer. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 5874–5878. IEEE.

Hadian, H., Sameti, H., Povey, D., and Khudanpur, S. (2018). Flat-start single-stage discriminatively trained hmm-based models for ASR. *IEEE ACM Trans. Audio Speech Lang. Process.*, 26(11):1949–1961.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O., editors, *Proceedings of the Sixth Workshop on Statistical Machine Translation, WMT@EMNLP 2011, Edinburgh, Scotland, UK, July 30-31, 2011*, pages 187–197. Association for Computational Linguistics.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F., editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

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

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Sciences*, 79:2554–2558.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 646–661. Springer.

Hussein, A., Watanabe, S., and Ali, A. (2022). Arabic speech recognition by end-to-end, modular systems and human. *Comput. Speech Lang.*, 71:101272.

Inaguma, H., Cho, J., Baskar, M. K., Kawahara, T., and Watanabe, S. (2019). Transfer learning of language-independent end-to-end ASR with language model fusion. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6096–6100. IEEE.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In Hua, K. A., Rui, Y., Steinmetz, R., Hanjalic, A., Natsev, A., and Zhu, W., editors, *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 675–678. ACM.

Karita, S., Wang, X., Watanabe, S., Yoshimura, T., Zhang, W., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., and Yamamoto, R. (2019). A comparative study on transformer vs RNN in speech applications. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 449–456. IEEE.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Kawakami, K., Wang, L., Dyer, C., Blunsom, P., and van den Oord, A. (2020). Learning robust and multilingual speech representations. In Cohn, T., He, Y., and Liu, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1182–1192. Association for Computational Linguistics.

Kemp, T. and Schaaf, T. (1997). Estimating confidence using word lattices. In Kokkinakis, G., Fakotakis, N., and Dermatas, E., editors, *Fifth European Conference on Speech Communication and Technology, EUROSPEECH 1997, Rhodes, Greece, September 22-25, 1997*. ISCA.

Khonglah, B. K., Madikeri, S. R., Dey, S., Bourlard, H., Motlícek, P., and Billa, J. (2020). Incremental semi-supervised learning for multi-genre speech recognition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 7419–7423. IEEE.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Lai, C. J., Zhang, Y., Liu, A. H., Chang, S., Liao, Y., Chuang, Y., Qian, K., Khurana, S., Cox, D. D., and Glass, J. (2021). PARP: prune, adjust and re-prune for self-supervised speech recognition. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21256–21272.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

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

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR.

Li, S., Lu, X., Sakai, S., Mimura, M., and Kawahara, T. (2017). Semi-supervised ensemble DNN acoustic model training. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 5270–5274. IEEE.

Liu, A. T., Li, S., and Lee, H. (2021). TERA: self-supervised learning of transformer encoder representation for speech. *IEEE ACM Trans. Audio Speech Lang. Process.*, 29:2351–2366.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020a). On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2020b). On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019a). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. (2019b). Rethinking the value of network pruning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ma, J. Z. and Schwartz, R. M. (2008). Unsupervised versus supervised training of acoustic models. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia, September 22-26, 2008*, pages 2374–2377. ISCA.

Ma, X., Pino, J. M., Cross, J., Puzon, L., and Gu, J. (2020). Monotonic multihead attention. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Madikeri, S. R., Motlícek, P., and Bourlard, H. (2021). Multitask adaptation with lattice-free MMI for multi-genre speech recognition of low resource languages. In Hermansky, H., Cernocký, H., Burget, L., Lamel, L., Scharenborg, O., and Motlícek, P., editors, *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 4329–4333. ISCA.

Madikeri, S. R., Tong, S., Zuluaga-Gomez, J., Vyas, A., Motlícek, P., and Bourlard, H. (2020). Pkwrap: a pytorch package for LF-MMI training of acoustic models. *CoRR*, abs/2010.03466.

Mangu, L., Brill, E., and Stolcke, A. (1999). Finding consensus among words: lattice-based word error minimization. In *Sixth European Conference on Speech Communication and Technology, EUROSPEECH 1999, Budapest, Hungary, September 5-9, 1999*. ISCA.

Manohar, V., Hadian, H., Povey, D., and Khudanpur, S. (2018). Semi-supervised training of acoustic models using lattice-free MMI. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pages 4844–4848. IEEE.

Manohar, V., Povey, D., and Khudanpur, S. (2015). Semi-supervised maximum mutual information training of deep neural network acoustic models. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 2630–2634. ISCA.

Maybury, M. (1999). *Advances in automatic text summarization*. MIT press.

Mnih, A. and Hinton, G. E. (2008). A scalable hierarchical distributed language model. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088. Curran Associates, Inc.

Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Comput. Speech Lang.*, 16(1):69–88.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*. Society for Artificial Intelligence and Statistics.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53.

Novotney, S., Schwartz, R. M., and Ma, J. Z. (2009). Unsupervised acoustic and language model training with small amounts of labelled data. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009, 19-24 April 2009, Taipei, Taiwan*, pages 4297–4300. IEEE.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In Ammar, W., Louis, A., and Mostafazadeh, N., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5206–5210. IEEE.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. (2019). Stand-alone self-attention in vision models. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 68–80.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Paul, D. B. and Baker, J. M. (1992). The design for the wall street journal-based CSR corpus. In *The Second International Conference on Spoken Language Processing, ICSLP 1992, Banff, Alberta, Canada, October 13-16, 1992*. ISCA.

Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N. A., and Kong, L. (2021a). Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Peng, Z., Budhkar, A., Tuil, I., Levy, J., Sobhani, P., Cohen, R., and Nassour, J. (2021b). Shrinking bigfoot: Reducing wav2vec 2.0 footprint. In Moosavi, N. S., Gurevych, I., Fan, A., Wolf, T., Hou, Y., Marasovic, A., and Ravi, S., editors, *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2021, Virtual, November 10, 2021*, pages 134–141. Association for Computational Linguistics.

Povey, D. (2003). *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, K. N., Hannemann, M., Motlíček, P., Qian, Y., Schwarz, P., Silovský, J., Stemmer, G., and Veselý, K. (2011). The kaldi speech recognition toolkit. In *Proceedings of ASRU 2011*, pages 1–4. IEEE Signal Processing Society.

Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., Wang, Y., and Khudanpur, S. (2016). Purely sequence-trained neural networks for ASR based on lattice-free MMI. In Morgan, N., editor, *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 2751–2755. ISCA.

Povey, D., Zhang, X., and Khudanpur, S. (2015). Parallel training of deep neural networks with natural gradient and parameter averaging. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Pratap, V., Hannun, A. Y., Xu, Q., Cai, J., Kahn, J., Synnaeve, G., Liptchinsky, V., and Collobert, R. (2019). Wav2letter++: A fast open-source speech recognition system. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6460–6464. IEEE.

Qin, Z., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., Yan, J., Kong, L., and Zhong, Y. (2022). cosformer: Rethinking softmax in attention. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Rawat, A. S., Chen, J., Yu, F. X., Suresh, A. T., and Kumar, S. (2019). Sampled softmax with random fourier features. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13834–13844.

Roy, A., Saffar, M., Vaswani, A., and Grangier, D. (2021). Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press.

Sadhu, S., He, D., Huang, C., Mallidi, S. H., Wu, M., Rastrow, A., Stolcke, A., Droppo, J., and Maas, R. (2021). wav2vec-c: A self-supervised model for speech representation learning. In

Hermansky, H., Cernocký, H., Burget, L., Lamel, L., Scharenborg, O., and Motlícek, P., editors, *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 711–715. ISCA.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. (2017). Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Schlag, I., Irie, K., and Schmidhuber, J. (2021). Linear transformers are secretly fast weight memory systems. *CoRR*, abs/2102.11174.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shao, Y., Wang, Y., Povey, D., and Khudanpur, S. (2020). Pychain: A fully parallelized pytorch implementation of LF-MMI for end-to-end ASR. In Meng, H., Xu, B., and Zheng, T. F., editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 561–565. ISCA.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. (2021). Efficient attention: Attention with linear complexities. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*, pages 3530–3538. IEEE.

Shrivastava, A. and Li, P. (2014). Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2321–2329.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Sperber, M., Niehues, J., Neubig, G., Stüker, S., and Waibel, A. (2018a). Self-attentional acoustic models. In Yegnanarayana, B., editor, *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 3723–3727. ISCA.

Sperber, M., Niehues, J., Neubig, G., Stüker, S., and Waibel, A. (2018b). Self-attentional acoustic models. In Yegnanarayana, B., editor, *Interspeech 2018, 19th Annual Conference of the International Speech Communication Association, Hyderabad, India, 2-6 September 2018*, pages 3723–3727. ISCA.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In Hansen, J. H. L. and Pellom, B. L., editors, *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*. ISCA.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *CoRR*, abs/2104.09864.

Sukhbaatar, S., Grave, E., Bojanowski, P., and Joulin, A. (2019). Adaptive attention span in transformers. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 331–335. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Synnaeve, G., Xu, Q., Kahn, J., Grave, E., Likhomanenko, T., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. (2019). End-to-end ASR: from supervised to semi-supervised learning with modern architectures. *CoRR*, abs/1911.08460.

Thomas, S., Seltzer, M. L., Church, K., and Hermansky, H. (2013). Deep neural network features and semi-supervised training for low resource speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6704–6708. IEEE.

Tong, S., Garner, P. N., and Bourlard, H. (2017). An investigation of deep neural networks for multilingual speech recognition training and adaptation. In Lacerda, F., editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 714–718. ISCA.

Tong, S., Vyas, A., Garner, P. N., and Bourlard, H. (2019). Unbiased semi-supervised LF-MMI training using dropout. In Kubin, G. and Kacic, Z., editors, *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1576–1580. ISCA.

Tsai, Y. H., Bai, S., Yamada, M., Morency, L., and Salakhutdinov, R. (2019). Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on*

Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, pages 4343–4352. Association for Computational Linguistics.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125. ISCA.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Veselý, K., Burget, L., and Cernocký, J. (2017). Semi-supervised DNN training with word selection for ASR. In Lacerda, F., editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 3687–3691. ISCA.

Veselý, K., Hannemann, M., and Burget, L. (2013). Semi-supervised training of deep neural networks. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 267–272. IEEE.

Vu, N. T., Imseng, D., Povey, D., Motlícek, P., Schultz, T., and Bourlard, H. (2014). Multilingual deep neural network based acoustic modeling for rapid language adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 7639–7643. IEEE.

Vyas, A., Dighe, P., Tong, S., and Bourlard, H. (2019). Analyzing uncertainties in speech recognition using dropout. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6730–6734. IEEE.

Vyas, A., Katharopoulos, A., and Fleuret, F. (2020). Fast transformers with clustered attention. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Vyas, A., Madikeri, S. R., and Bourlard, H. (2021a). Comparing CTC and LFMMI for out-of-domain adaptation of wav2vec 2.0 acoustic model. In Hermansky, H., Cernocký, H., Burget, L., Lamel, L., Scharenborg, O., and Motlícek, P., editors, *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 2861–2865. ISCA.

Vyas, A., Madikeri, S. R., and Bourlard, H. (2021b). Lattice-free mmi adaptation of self-supervised pretrained acoustic models. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 6219–6223. IEEE.

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural Comput.*, 1(1):39–46.

Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech Signal Process.*, 37(3):328–339.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Wang, W., Tang, Q., and Livescu, K. (2020). Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6889–6893. IEEE.

Wang, Y., Khudanpur, S., Chen, T., Xu, H., Ding, S., Lv, H., Shao, Y., Peng, N., Xie, L., and Watanabe, S. (2019b). Espresso: A fast end-to-end neural speech recognition toolkit. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2019, Singapore, December 14-18, 2019*, pages 136–143. IEEE.

Wessel, F., Macherey, K., and Schlüter, R. (1998). Using word probabilities as confidence measures. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98, Seattle, Washington, USA, May 12-15, 1998*, pages 225–228. IEEE.

Wessel, F. and Ney, H. (2005). Unsupervised training of acoustic models for large vocabulary continuous speech recognition. *IEEE Trans. Speech Audio Process.*, 13(1):23–31.

Wessel, F., Schluter, R., Macherey, K., and Ney, H. (2001). Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298.

Wu, F., Kim, K., Pan, J., Han, K. J., Weinberger, K. Q., and Artzi, Y. (2022). Performance-efficiency trade-offs in unsupervised pre-training for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, pages 7667–7671. IEEE.

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., and Feris, R. S. (2018). Blockdrop: Dynamic inference paths in residual networks. In *2018 IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 8817–8826. Computer Vision Foundation / IEEE Computer Society.

Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M. L., Stolcke, A., Yu, D., and Zweig, G. (2017). Toward human parity in conversational speech recognition. *IEEE ACM Trans. Audio Speech Lang. Process.*, 25(12):2410–2423.

Xu, H., Povey, D., Mangu, L., and Zhu, J. (2011). Minimum bayes risk decoding and system combination based on a recursion for edit distance. *Comput. Speech Lang.*, 25(4):802–828.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2048–2057. JMLR.org.

Xu, Q., Baevski, A., Likhomanenko, T., Tomasello, P., Conneau, A., Collobert, R., Synnaeve, G., and Auli, M. (2021). Self-training and pre-training are complementary for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*, pages 3030–3034. IEEE.

Xu, Q., Likhomanenko, T., Kahn, J., Hannun, A. Y., Synnaeve, G., and Collobert, R. (2020). Iterative pseudo-labeling for speech recognition. In Meng, H., Xu, B., and Zheng, T. F., editors, *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 1006–1010. ISCA.

Yu, D. and Deng, L. (2014). *Chapter 8: Deep Neural Network Sequence-Discriminative Training*. Springer, automatic speech recognition — a deep learning approach edition.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020). Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

Zavaliagkos, G., Siu, M., Colthurst, T., and Billa, J. (1998). Using untranscribed training data to improve performance. In *The 5th International Conference on Spoken Language Processing, Incorporating The 7th Australian International Speech Science and Technology Conference, Sydney Convention Centre, Sydney, Australia, 30th November - 4th December 1998*. ISCA.

Zeyer, A., Beck, E., Schlüter, R., and Ney, H. (2017). CTC in the context of generalized full-sum HMM training. In Lacerda, F., editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 944–948. ISCA.

Zhang, P., Liu, Y., and Hain, T. (2014). Semi-supervised DNN training in meeting recognition. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 141–146. IEEE.

Zhang, Y., Park, D. S., Han, W., Qin, J., Gulati, A., Shor, J., Jansen, A., Xu, Y., Huang, Y., Wang, S., Zhou, Z., Li, B., Ma, M., Chan, W., Yu, J., Wang, Y., Cao, L., Sim, K. C., Ramabhadran, B., Sainath, T. N., Beaufays, F., Chen, Z., Le, Q. V., Chiu, C., Pang, R., and Wu, Y. (2021). Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *CoRR*, abs/2109.13226.

Apoorv Vyas

+41-762891921 | apoorv.vyas@idiap.ch | Webpage | Google Scholar | LinkedIn | Github

Research Interests

Deep Learning, Automatic Speech Recognition, and Computer Vision

Education

Indian Institute of Technology, Guwahati 2010 - 2014
B.Tech, Electronics and Electrical Engineering CPI - 8.10 (scale of 10.0)

École Polytechnique Fédérale de Lausanne 2018 - 2022
PhD, Electrical Engineering CPI - 5.33 (scale of 6.0)

Experience

Facebook Artificial Intelligence Research (FAIR) U.S.[*]
Research Intern *Sep. 2021-Dec. 2021*

- Worked on improving the computational efficiency of wav2vec 2.0 pre-training with Transformer models.

Amazon Research Germany
Research Intern *May 2021-July 2021*

- Worked on improving self-supervised training with Transformer models and RNN-Transducers.

Idiap Research Institute Switzerland
Graduate Research Assistant *July 2018-August 2022* [†]

- Working on improving speech recognition for low resource languages with unsupervised learning.
- Working on scaling Transformer architectures to long sequences.

Intel Labs India
Systems Engineer *April 2015–May 2018*

- Developed a method for out of distribution input detection in deep neural networks.
- Developed *low power semantic hashing* for fast and accurate retrieval of similar images.
- Applied compressed sensing techniques for power efficient data gathering in wireless sensor networks.

Oracle India Pvt. Ltd. India
Applications Engineer *July 2014–March 2015*

- Worked on web application development using Oracle's application development framework.

Peer-Reviewed Publications

- Vyas, A., Hsu, W., Auli, M., and Baevski, A. **On-demand Compute Reduction with Stochastic wav2vec 2.0.** *Interspeech, 2022.*

- Vyas, A., Madikeri, S., and Bourlard, H. **Comparing CTC and LFMMI for Out-of-Domain Adaptation of wav2vec 2.0 Acoustic Model.** *Interspeech, 2021.*

- Vyas, A., Madikeri, S., and Bourlard, H. **Lattice-Free MMI Adaptation of Self-supervised Pretrained Acoustic Models.** *International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021.*

---

[*] remotely from Switzerland

[†] expected, with internship breaks

- Vyas, A., Katharopoulos, A., and Fleuret, F. **Fast Transformers with Clustered Attention.** *34th Conference on Neural Information Processing Systems (NeurIPS), 2020.*

- Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. **Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention.** *37th International Conference on Machine Learning (ICML), 2020.*

- Tong, S., Vyas, A., Garner P., and Bourlard, H. **Unbiased Semi-supervised LF-MMI Training Using Dropout.** *Interspeech, 2019.*

- Vyas, A., Dighe, P., Tong, S., and Bourlard, H. **Analyzing Uncertainties in Speech Recognition Using Dropout.** *International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.*

- Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., and Willke, T. **Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers.** *European Conference on Computer Vision (ECCV), 2018.*

- Natarajan, V and Vyas, A., **Power Efficient Compressive Sensing for Continuous Monitoring of ECG and PPG in a Wearable System.** *IEEE World Forum on Internet of Things (WF-IoT), 2016.*

PATENTS

- Vyas, A, Mehta, D. & Srenivasa, V., **Low Power Supervised Semantic-Preserving Shallow Hashing.** US Patent 15792940, Intel

- Baxi, A. & Vyas, A, **Power reduction of optical heart rate sensor in a wearable Cuffless Blood Pressure patch by Local Polynomial Regression of sub-sampled PPG and by ECG synchronized PPG excitation.** US Patent 15492986, Intel

- Vyas, A & Natarajan, V., **Power Efficient Data Gathering by Joint Compressive Sensing and Shortest Path Tree for a IoT Mesh Wireless Sensor Network.** US Patent 15856994, Intel

- Vyas, A & Natarajan, V., **Novel anomaly prediction method for intelligent power-efficient relay scheduling in an IoT Mesh Wireless Sensor Network.** US Patent 10448415, Intel

- Natarajan, V. & Vyas, A , **Novel compressive sensing scheme for power efficient data aggregation in a spatial IoT wireless sensor network.** US Patent 10149131, Intel

TECHNICAL SKILLS

Programming: Python, C/C++, CUDA, Shell Scripting
Frameworks: PyTorch, Kaldi, Keras, LaTex

ACADEMIC SERVICE

Reviewer for TNNLS (2020), NeurIPS (2021, 2022), ICLR (2022)

MISCELLANEOUS

- Divisional Recognition Award at Intel for excellent contributions to the Bio-sensing project to extract heart rate while typing.

- Secured rank 1901 (out of 500K candidates) in the Joint Entrance Examination (JEE) for IITs.