Adapting ASR to Accented Speech using Codebooks

Guide: Prof. Preethi Jyothi

Name - Omkar Nitsure
Email - 210070057@iitb.ac.in
Roll No - 210070057

Contents

1 Introduction

With the Advent of Deep Learning, we can see very powerful and complex model architectures, smart optimization techniques, availability of large amounts of pre-training data. It has resulted in unprecedented accuracies for a variety of tasks in areas like **Computer Vision**, **Natural Language Processing** and **Automatic Speech Recognition**. **Self-Supervised Learning** [1] is a new way of making use of large amounts of unlabelled data for pre-training the models to make them learn the structure of **images, text, spoken language**. This technique was used first in NLP and gave an incredible boost in performance. It was first used in Speech Recognition by a model named **HuBERT**[2] which stands for Hidden units like BERT. This technique uses **SSL** for the pretraining stage and then it is further fine-tuned with a small amount of labelled data using the **CTC** [3] loss for Automatic Speech Recognition. The purpose of the pre-training stage is to train the model to output speech representations which are rich in information. These rich representations will be used in fine-tuning to perform the ASR task. It showed great performance improvements [4] over the **MFCC** based features. However, there is an intrinsic bias in the data available for pre-training. The pre-training data has more than **50%** data from the American Accent. It also has a large amount of data for **British** accent. Some accents like **Scottish** accent are much less expressed in the pretraining data and thus the model performance degrades on those accents as compared to the American Accent. Many accents like **Indian**, **New-Zealand**, etc. are sometimes not even part of the pre-training data and thus the performance severely degrades for them. My task as part of this RnD project was to try and modify the representations available at the end of the pre-training stage in such a way that the ASR performance improves for these less-expressed or completely unseen accents. Initially, I started with doing various analysis and ablation studies for an already established technique of introducing accent-specific codebooks in the Hubert architecture. Later, I tried techniques other than the one mentioned above to improve the representation quality.

2 Contributions

The techniques that I implemented are as follows

- Training different k-means models for the accent-generic case and every accent and then pooling all cluster centers to obtain a single set of k-means labels
- having two different targets for the Hubert pretraining stage. One is an accent-specific target, while the other is an accent-generic target.
- Applying budget criteria on the number of codebooks which are utilized in cross-attention to increase their expressibility across different accents.
- Using cross-attention using a single large codebook

3 Literature Review

This section will be about overview of some important papers which were instrumental in designing various different techniques used to improve performance on accented speech. The first paper talks about taking speech representations of low-resource speech accents close to the representations of **native US** accent while the second paper talks about introducing a **budget** on the extent of cross-attention that can be used by a frame.

3.1 INTapt: Information-Theoretic Adversarial Prompt Tuning for Enhanced Non-Native Speech Recognition

The motivation of this paper[5] is to make sure that the speech representations of accented speech are **close enough** to the speech representations of the dominating L1 accent (**US** in our case). It introduces **"Prompts"** concatenated to the original input, which makes the input resemble **(L1)** native English Speech. The key point to note is that it does not update the backbone network (**HuBERT**) weights during fine-tuning and adds some learnable parameters into the input space. The main objective is to train the model to **reduce** accent feature **dependence** between the **original input** and the **prompt-concatenated input**. Then, it is fine-tuned to minimize CTC loss of L2 accents for improving ASR performance for the new concatenated input. The following diagram shows the overall flow and different blocks in the model architecture.

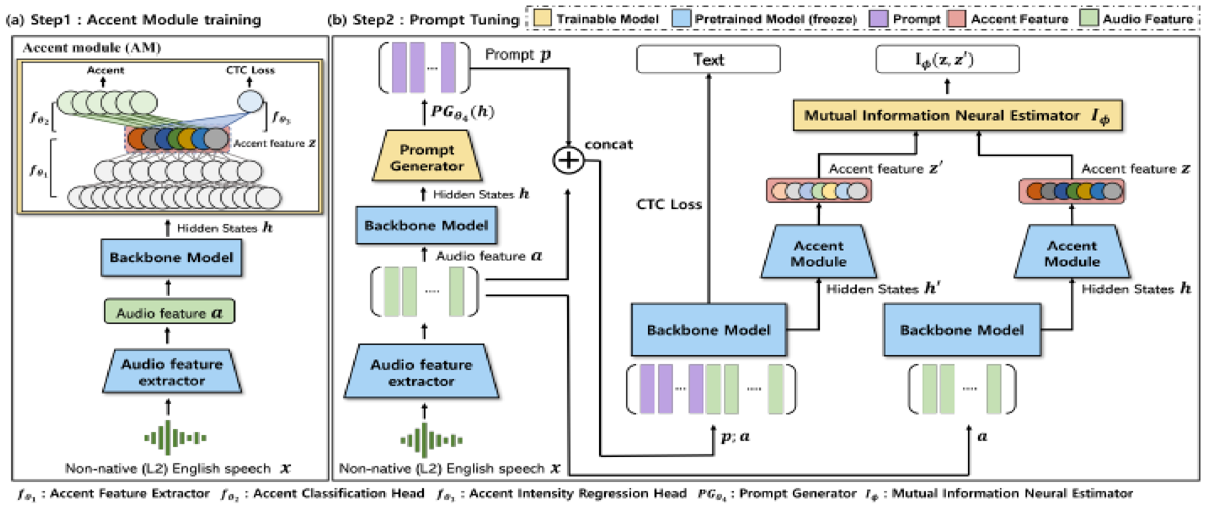

(a) Step1 : Accent Module training
(b) Step2 : Prompt Tuning
Trainable Model
Pretrained Model (freeze)
Prompt
Accent Feature
Audio Feature
Accent module (AM)
Accent
CTC Loss
Accent feature z
Hidden States h
Backbone Model
Audio feature a
Audio feature extractor
Non-native (L2) English speech x
Prompt p
PG(h)
Prompt Generator
concat
Text
CTC Loss
I(z, z')
Mutual Information Neural Estimator
Accent feature z'
Accent Module
Hidden States h'
p; a
a
f1 : Accent Feature Extractor f2 : Accent Classification Head f3 : Accent Intensity Regression Head PG : Prompt Generator I : Mutual Information Neural Estimator

Figure 1: Model Architecture and flow in INTapt

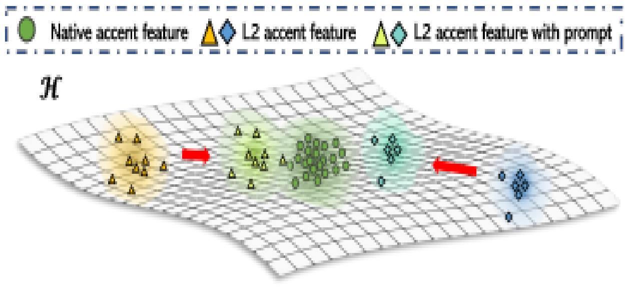


Figure 2: The main objective

The intuition is that the prompt is trained such that the accent of the concatenation is **pushed away** from the input accent while getting closer to the native L1 accent. The concatenation should achieve native CTC loss performance. They showed that there is a higher **Cosine Similarity** between the prompt concatenated input and the L1 accents than the original input. Prompts are generated as input-dependent continuous embeddings from a neural network trained for this specific purpose.

3.1.1 Training the accent module

In the 1st step of training, they train the **accent module**. The purpose of the accent module is to **isolate the accent features** from the audio features of audio. In the 2nd step of training, they trained a **prompt generator** using the before-mentioned objectives. So, we first extract the audio features (MFCC) from the audio and pass it through a pre-trained backbone network (HuBERT) to generate the hidden states. Then, these hidden states are passed through an accented feature extractor (f$_{theta1}$) followed by using another layer (f$_{theta2}$) to label the accents. Also, an **Accent Intensity Regression** layer is trained using the CTC loss to output the intensity of the accents. It will allow the accent features to capture the intensity of accents. The Accent Module is trained using the following loss -

$$\min_{\theta_1,\theta_2} \frac{1}{|B|} \sum_{i \in B} -\log p(y_i | f_{\theta_2}(f_{\theta_1}(h_i))) + \lambda \min_{\theta_1,\theta_3} \frac{1}{|B|} \sum_{i \in B} [f_{\theta_3}(f_{\theta_1}(h_i)) - CTC(x_i)]^2$$

3.1.2 Training the prompt generator

Mutual Information measures the **co-dependence** between 2 Random Variables. The co-dependency between 2 RVs can be the output of a neural network. The objective used to train the prompt generator, which minimizes the mutual information is as follows -

$$\min_{\theta_4} \max_{\phi} \frac{1}{|B|} \sum_{i \in B} CTC(p_{\theta_4}; a) + \lambda I_{\phi}(z_{\theta_4'}, z)$$

3.2 SHARE OR NOT? Learning to schedule language-specific capacity for multilingual translation

This paper[6] comes from the background of NLP where they show that by using a mixture of language-specific and language-generic units in the model architecture, performance on different tasks is improved. They used it in the setting of **Multilingual Neural Machine Translation** (MNMT). They call it Conditional Language-specific Routing (CSLR). The main logic is to use these units and then have a hard binary gating based on the word representations. This paper modifies the standard Transformer architecture to make it better for multilingual translation. The model architecture is as follows -

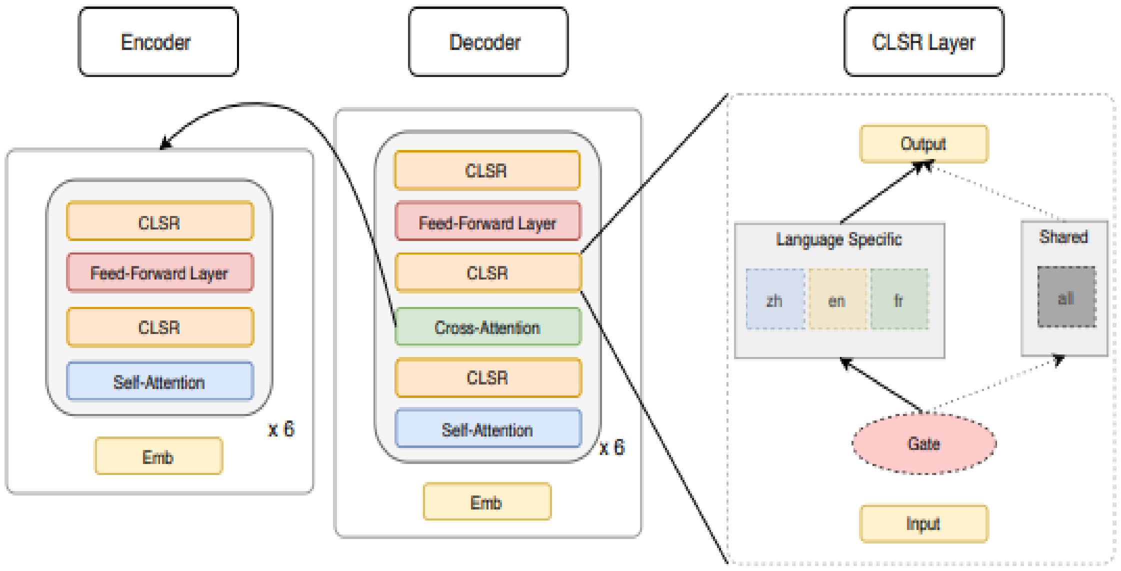


Figure 3: Modified Transformer architecture using CSLR

This approach can be used in the setting of Codebook cross-attention by restricting the number of codebooks that a frame can attend to. Here, we don't restrict which codebooks the frame should attend to like the previous attempt but let the model choose those based on the relative attention coefficient values. The hope is, it will induce more language-specific information in all the codebooks as we are not restricting which codebooks should be used for a specific accent.

4 Codebook-based Cross-Attention for Accented ASR

I will briefly discuss the idea[7] for which I tried different analysis and ablation studies. This technique introduces a fixed set of learnable codebooks per accent available in the pre-training data, namely **Australia, Canada, England, Scotland, US**. The codebooks are a set of **50** vectors per accent. The Hubert architecture consists of the Transformer Encoder Layers, which use **Self-Attention**. We introduced these codebooks and incorporated them in the main flow by doing a **Cross-attention** of them with the frame-wise representations extracted by Self-Attention in different layers of HuBERT. During pre-training, the accent information is given to the model and then it uses only the codebook which corresponds to that particular accent. During inference, the accent information is unavailable, thus beam-search is used to find the best-suited codebook. All this analysis was done on fairseq [8]

4.1 Analysis for cluster assignment

Before training the HuBERT model on the **Common Voice** [9] dataset, we load the **LibriSpeech checkpoint**. Features are extracted from this checkpoint and then a k-means model is trained. Labels are assigned from this k-means model for each frame of an utterance. Following are the histograms of cluster frequencies Vs the number of clusters

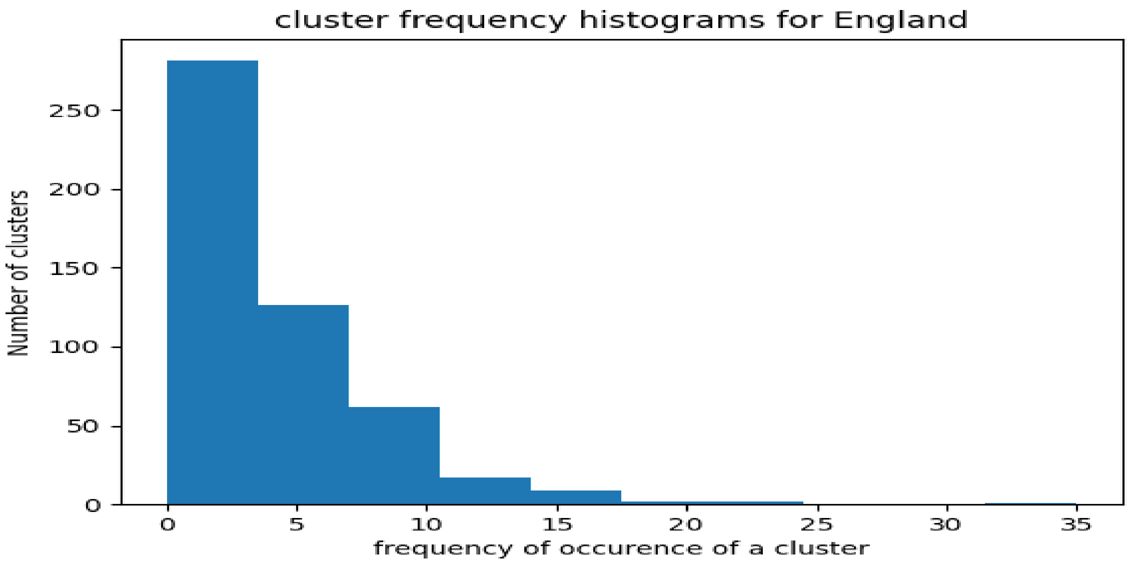


Figure 4: Cluster Assignment Distribution for Baseline Model

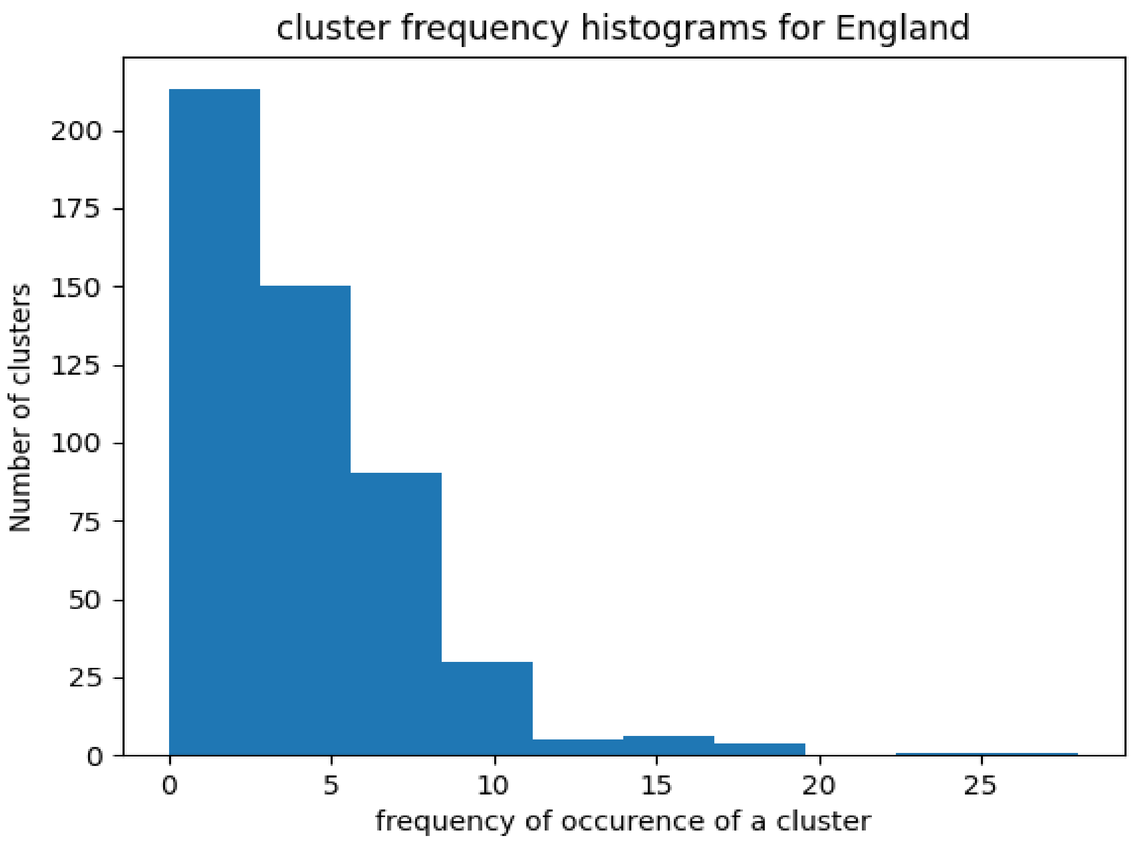


Figure 5: Cluster Assignment Distribution for Codebook based model

In the figure for **baseline** model, we can see the majority of cluster centers not assigned to any example. Here, we can see that more than **250** clusters from the total of **500** clusters are not assigned to any frame at all. Then, in the following figure, we can see the cluster assignments for a model trained using codebooks. Here, we can see that the cluster assignment has increased significantly. More clusters are being utilized indicating much different feature representations for different parts of a particular accent. A similar trend is observed for the other accents and the overall data. The number of less expressed cluster labels is reduced from more than **250** to just above **200** clusters. We then extracted the features from different layers of the trained model to see if we could observe some accent-wise clustering. We observed that the maximum accent-wise clustering and spacing were observed for features extracted from the **6th** layer of the Transformer Encoder. We can see the clustering in the following TSNE plot of the representations. I did some more TSNE visualizations, but they did not give meaningful insights, so I have not shown them here.

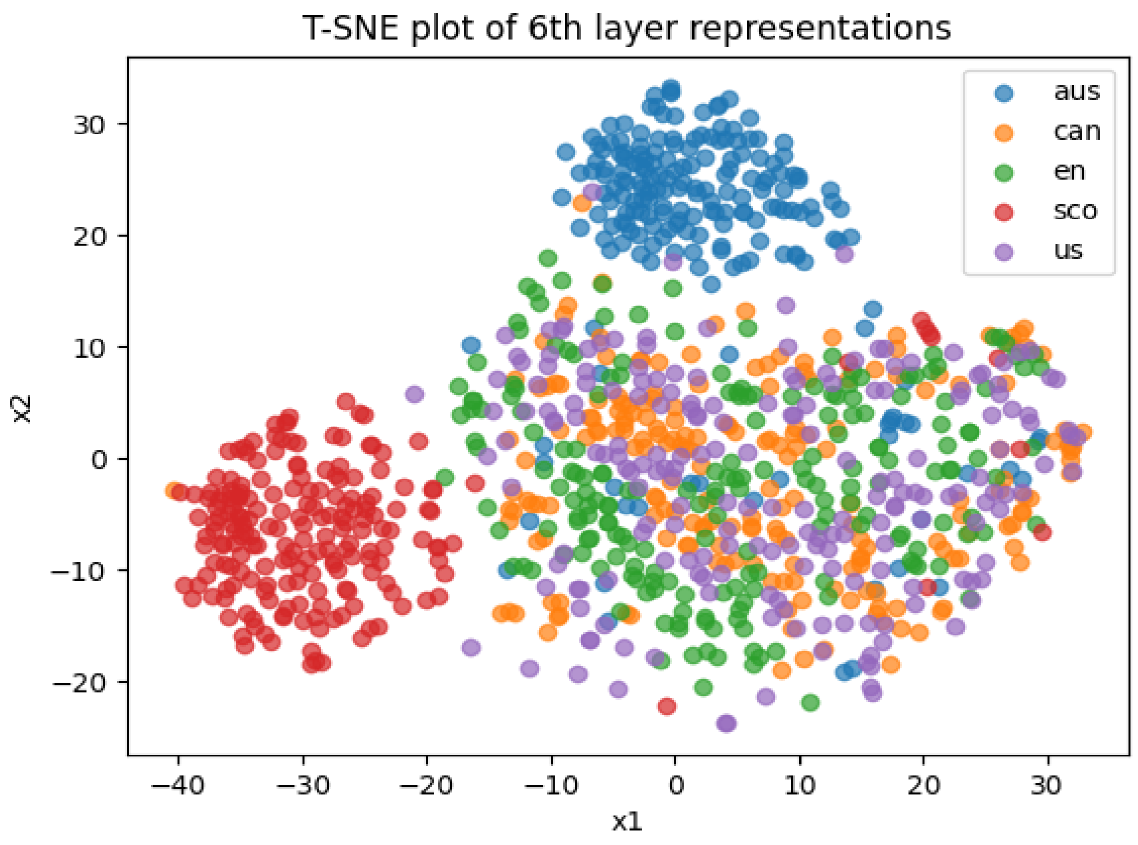


Figure 6: TSNE plot for 6th layer cluster representations

I took some examples, which had the same underlying transcript and were recorded for different accents. These examples are important for the analysis and evaluation as they will have almost the same phonemes and the only difference is the accent which we want to emphasize. Then, I tried to see if the cluster assignment is similar for similar accents and widely different for accents belonging to different geographic locations. I computed the number of common clusters for each set of pairs of accents and compared the results of the baseline Vs our approach of Codebooks. We can see that for pairs of similar accents like **US-Canada**, **US-England**, **England-Scotland**, the number of common cluster assignments have increased in the codebook-based model than the baseline. It indicates that the representations have become more accent-aware. The same approach was repeated on the entire test data. It gave similar results.

Accents	AUS	CAN	EN	SCO	US
AUS	-	45	52	16	45
CAN	45	-	56	15	56
EN	62	56	-	18	53
SCO	16	15	18	-	18
US	45	56	53	18	-

Table 1: common labels - baseline(same text)

Accents	AUS	CAN	EN	SCO	US
AUS	-	48	65	18	52
CAN	48	-	54	17	61
EN	65	54	-	22	60
SCO	18	17	22	-	17
US	52	61	60	17	-

Table 2: common labels - codebooks(same text)

Accents	AUS	CAN	EN	SCO	US
AUS	-	13	21	21	13
CAN	13	-	30	21	34
EN	21	30	-	34	39
SCO	21	21	34	-	21
US	13	34	39	21	-

Table 3: common labels - baseline(full data)

Accents	AUS	CAN	EN	SCO	US
AUS	-	21	27	25	19
CAN	21	-	29	23	35
EN	27	29	-	31	37
SCO	25	23	31	-	23
US	19	35	37	23	-

Table 4: common labels - codebooks(full data)

I also computed how many different cluster labels are common (with high frequency) across different numbers of accents. The analysis was done for both the baseline model and the Codebook-based model.

Model	Buckets(No of clusters with common accents					
	0	1	2	3	4	5
Codebooks (6th)	347	77	35	13	16	12
Baseline	334	91	32	20	15	8

Table 5: Buckets for No of intersecting accents

4.2 Relative attention weights of Codebooks

In the above technique of using **50 codebooks** per accent through cross attention, I tried to visualize the attention coefficients in the cross attention. It is done by passing all examples through the model and then averaging the attention coefficients for all the **50** codebooks. The same analysis was done for different accents and all **12** layers of the Transformer Encoder. One of the matrices for the first **55** frames of examples for the American accent is given below

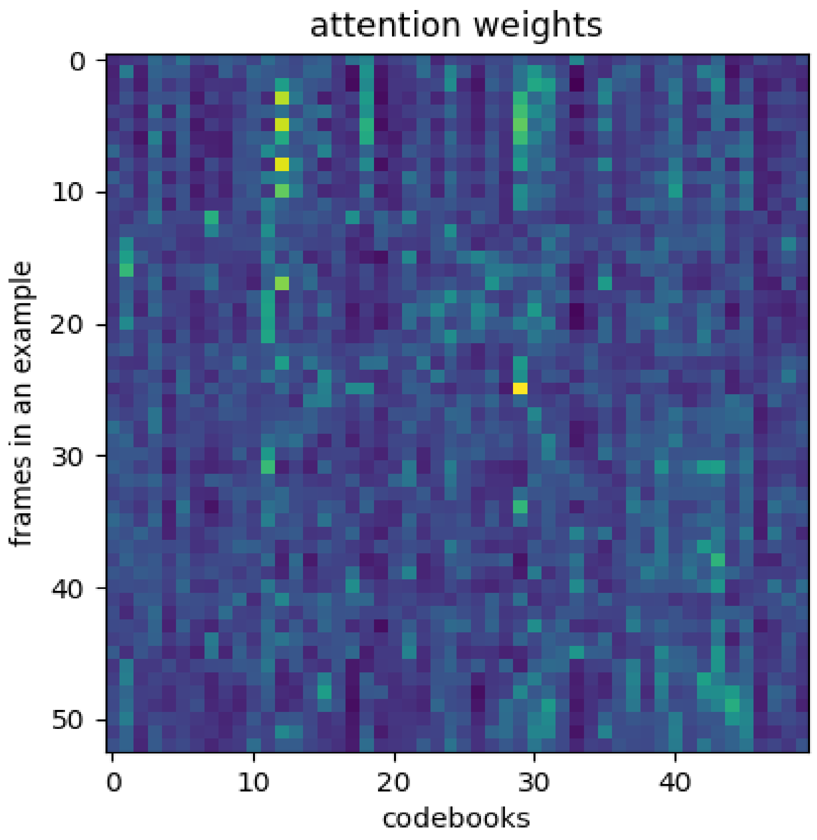


Figure 7: Attention coefficients for the 6th Transformer Layer

We can see that across the first **55** frames of all American accent examples, only some of the **50** codebooks are strongly fired while the firing is uniform for all other codebooks. It is evident because of the presence of vertical **yellow** strips for only some regions.

Now, I will do a literature review, which is essential to understand the contributions/modifications I tried on the HuBERT architecture.

5 Methodology

This section talks about the various methods and modifications tried on the HuBERT architecture for improving ASR performance on accented speech. Some of these methods were inspired by similar techniques used in NLP for low-resource languages -

5.1 Budget on Cross-Attention coefficients of Codebooks

In this approach, we use the same baseline Codebook cross-attention-based model with the only difference that now we don't have accent-specific clusters but have accent-generic ones. We choose to continue with **500** clusters. Now, we make changes in **2** places. First, we modify the way cross-attention with codebooks is computed. Initially, we get the cross-attention logits as before. We pass it through the SoftMax layer to get the probability distribution. Then we use **torch.topk** to choose **50** codebooks with the highest probability and set the coefficients for other codebooks equal to zero. Then, we compute the final **value** of the cross-attention using only these codebooks. The second change is in the loss computation. We want only the selected **50** codebooks to attend to the frame representations heavily. In other words, we want the entire probability distribution to be concentrated in these codebooks. Thus, we want the sum of coefficients of these **50** codebooks to be as close to **1** as possible. Thus, we simply subtract the sum from **1** to get the budget loss. This loss is added to the original loss to get the final loss.

5.2 Class labels through pooling K-Means centroids

This approach doesn't make any modifications to the original HuBERT architecture. It just modifies the way k-means cluster labels are computed. Initially, we extract the features from **LibriSpeech** checkpoint and train k-means models on those. We start with **6** k-means models, one for each accent and one accent-generic model. The accent-specific k-means models have **50** clusters while the accent-generic k-means model has **250** clusters. Then the corresponding cluster centers are extracted from all these models and pooled together to get a set of **500** clusters. Then, the k-means labels are extracted using these cluster centers. This is followed by a standard HuBERT training process for **200k** updates. Then, the features are extracted from the trained model checkpoint and the above procedure is repeated for one more iteration. This approach has shown promising results, with the training loss almost equal to our original codebook cross-attention model.

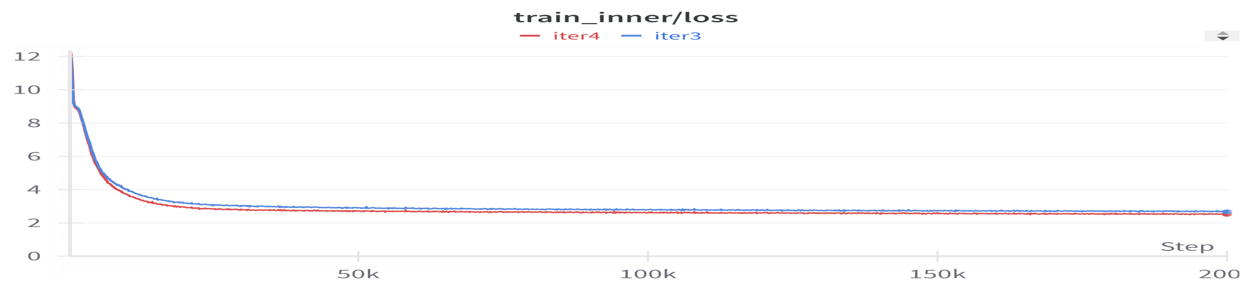
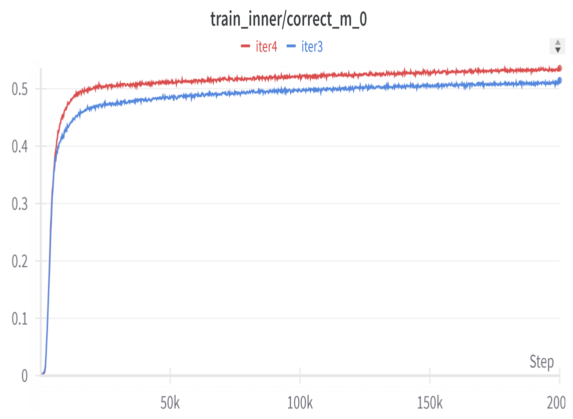
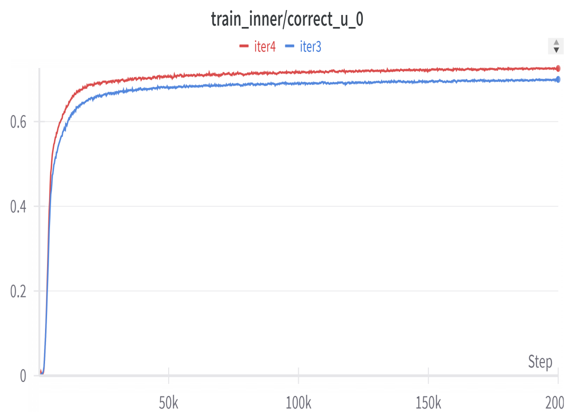


Figure 8: Training loss for both epochs



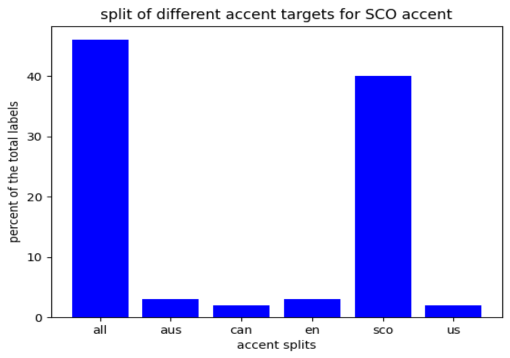
(a) correct for masked embeddings



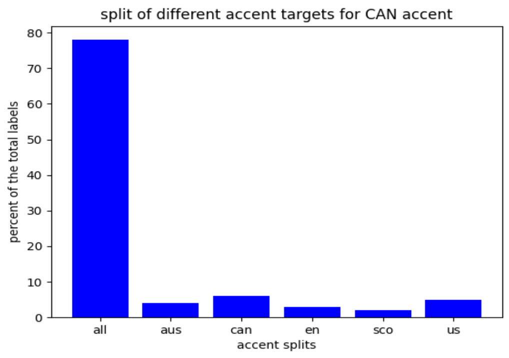
(b) correct for unmasked embeddings

Figure 9: Correct prediction values for both masked and unmasked frames

I did some additional analysis on how the different targets are distributed among different cluster centers. So, for each accent, I checked the distribution of cluster labels assigned among the **6** k-means cluster sets. It shows an evident correlation among similar accents and gives insights into how different accents are phonetically related. Canadian accent is similar to the American accent and thus has maximum label assignment from American clusters among different accents. The Scottish accent is a bit different from the rest of the accents in the training data, and somewhat similar to the British Accent. Thus, we can see most of the label assignments for the Scottish accent come from the accent-generic clusters and clusters for the Scottish accent. Among other accents, British accent labels are the maximum.



(a) distribution of labels for Scottish accent

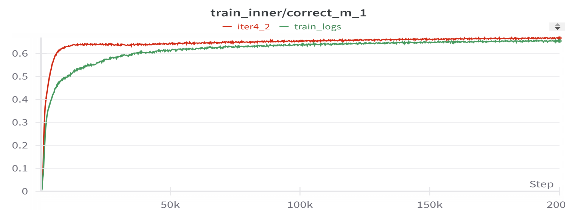


(b) distribution of labels for Canadian accent

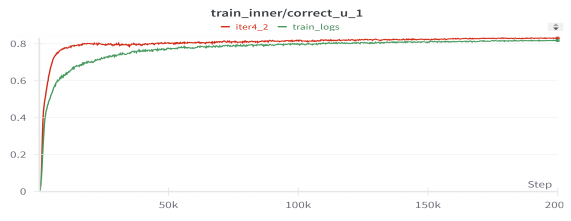
Figure 10: relative label distribution among generic and specific clusters

5.3 Two targets for each frame - Accent-generic and Accent-specific

Before training the HuBERT model on the Common Voice dataset, audio features are extracted from the **LibriSpeech checkpoint**. These features are used to train k-means models one for the accent-generic case and one for each of the accents. Labels are extracted from all of these **6** k-means models and then **2** targets are made for each example. The dimension of the final output layer of HuBERT is doubled and then the output is broken into two equal pieces each having the length of the standard HuBERT model output. These **2** pieces are used as **2** different targets. For every frame, we consider **2** targets. One is an **accent-specific target** coming from **50** k-means cluster labels while the other is an **accent-generic target** coming from **250** k-means cluster labels. The model is trained to predict both these targets together. The hope is that predicting an accent-specific target will act as an auxiliary task and help induce more accent awareness in speech representations. These experiments have shown promising results.



(a) correct for masked embeddings



(b) correct for unmasked embeddings

Figure 11: Correct prediction values for both masked and unmasked frames

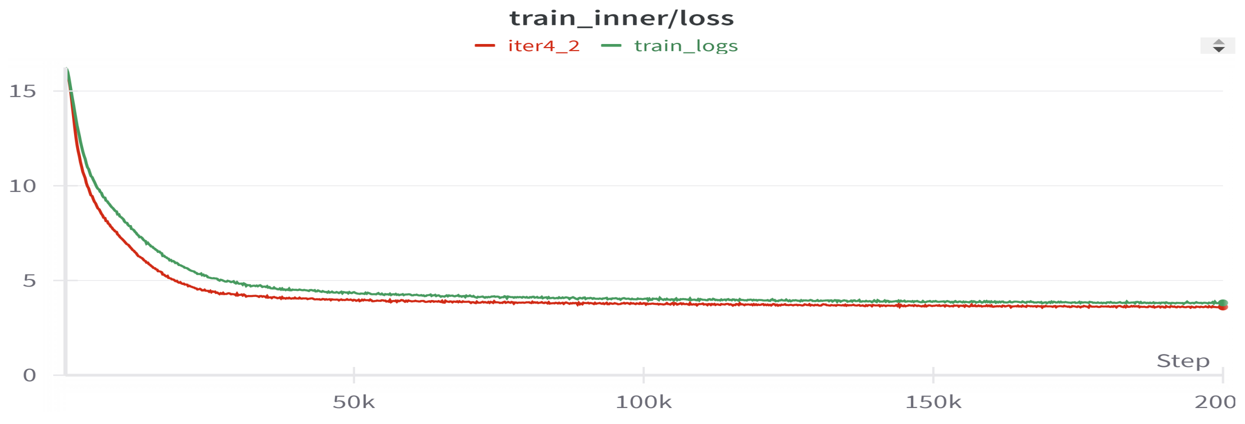


Figure 12: Training loss for both epochs

5.4 Single set of Codebooks for Cross-Attention

This idea is similar to the original accent-specific codebooks approach. Here, instead of defining accent-specific codebooks, we define a single large accent-generic codebook of **500** entries. This codebook is used to do cross-attention in each of the **12** Transformer encoder layers. Here, the hope is that during training, the model will learn which codebooks to attend to for different accents. We are neither forcing the number of codebooks that can be attended to nor the specific codebook entries that are available for cross-attention to a specific example. We let all of this be learnt by the model through data. The training loss is as follows -

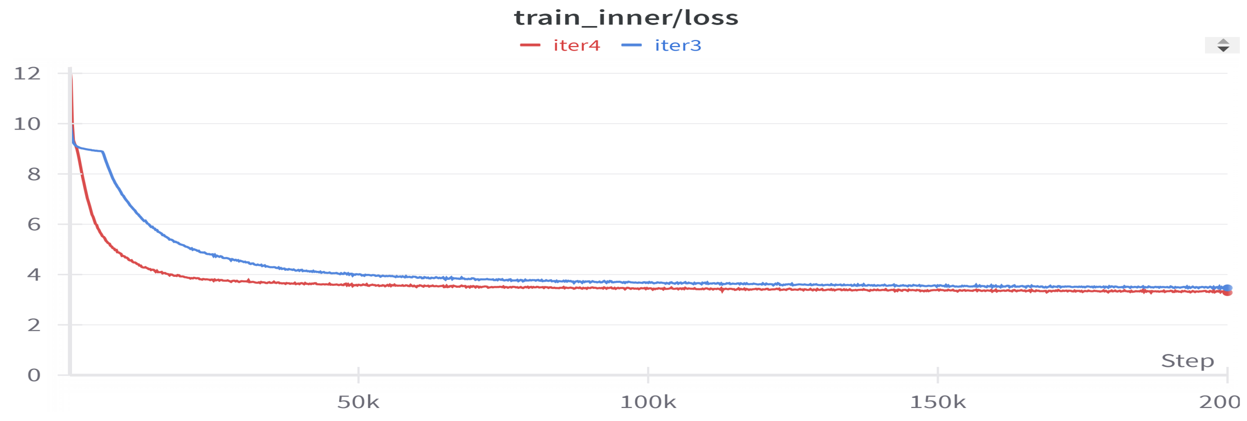
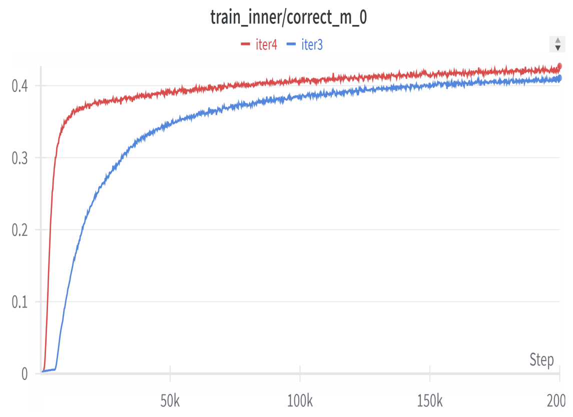
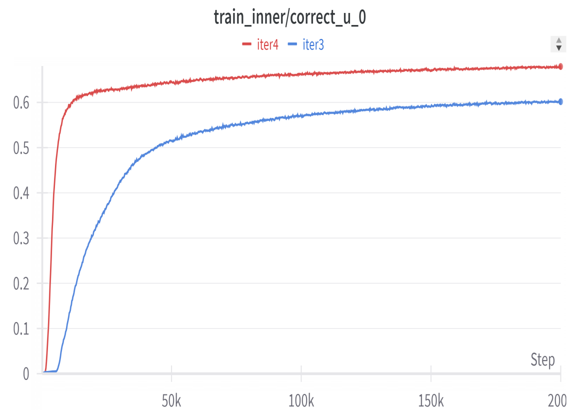


Figure 13: Training loss for both epochs



(a) correct for masked embeddings



(b) correct for unmasked embeddings

Figure 14: Correct prediction values for both masked and unmasked frames

6 Future Work

The idea of pooling **k-means** centroids of each accent for generating the class labels for HuBERT pretraining stage can be explored further. This idea can be complemented by making a change in the HuBERT architecture (like introducing codebooks) to further improve performance. Some hyperparameter tuning might lead to improvement in performance and can give insights on the ideal number of clusters required for fully representing accented speech samples. The idea of introducing budget on the number and subset of codebooks is not completely explored yet. The loss function can be defined in a lot of different ways. Also, the restriction on the number of codebooks selected for cross-attention can be imposed in a number of ways. There is a lot of scope for implementing this technique successfully in speech.

7 Conclusions

Modern Automatic Speech Recognition (ASR) systems are becoming increasingly better but there is always an inherent bias of any of these techniques towards the training data distribution. The training data used for Self-Supervised-Learning (SSL) usually has large amounts of data from only **1** or **2** different accents. This results in model representations favouring those accents over others leading to worse ASR performance on the low-resource accents. We have tried different techniques to make the ASR performance of low-resource accents as good as US accent. The idea of introducing **accent-specific codebooks** results in accent information being encoded in those codebooks. This helps to boost performance on the less-expressed accents. We observed performance boosts on geographically closer **unseen accents** as well. Another approach to tackle this problem would be to induce accent-specific information in the **targets** of the HuBERT pretraining stage. This can be done by either defining **2** targets (one is accent-generic while the other is accent-specific) per frame or by generating labels using **k-means centroids** trained on specific accents. As an addition to the accent-specific codebooks approach, the **number** and **subsets of codebooks** to which a frame can attend to, can be restricted using a budget constraint in the loss function. This was used in NLP and has seen to be giving improvements in similar settings.

References

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

[2] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[4] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv preprint arXiv:1911.03912*, 2019.

[5] E. Yoon, H. S. Yoon, J. Harvill, M. Hasegawa-Johnson, and C. D. Yoo, "Intapt: Information-theoretic adversarial prompt tuning for enhanced non-native speech recognition," *arXiv preprint arXiv:2305.16371*, 2023.

[6] B. Zhang, A. Bapna, R. Sennrich, and O. Firat, "Share or not? learning to schedule language-specific capacity for multilingual translation," 2021.

[7] D. Prabhu, P. Jyothi, S. Ganapathy, and V. Unni, "Accented speech recognition with accent-specific codebooks," *arXiv preprint arXiv:2310.15970*, 2023.

[8] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," *arXiv preprint arXiv:1904.01038*, 2019.

[9] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.