Adapting ASR to Accented Speech using Codebooks

Omkar Nitsure
Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, India
210070057@iitb.ac.in

Priyanshu Yadav
Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
210050125@iitb.ac.in

Pulkit Goyal
Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
210050126@iitb.ac.in

***Abstract*—Speech accents degrade the performance of state-of-the-art end-to-end Automatic Speech Recognition (ASR) systems. Even with self-supervised learning and pre-training of ASR models, the accuracy of less-represented English accents suffers compared to highly represented accents. This work extensively analyzes an accent-aware adaptation technique for self-supervised learning that introduces different manipulations of a trainable set of codebooks. These learnable codebooks enable the model to capture accent-specific information during pre-training, which is further refined during ASR fine-tuning. We also try different approaches to exploit accent information during ASR fine tuning where codebooks act like MOEs and the selection is based on soft labels to enable accent mixing.**

***Index Terms*—Automatic Speech Recognition, Accent Codebooks, Self-Supervised Pretraining, HuBERT, Mixture of Experts**

I. Introduction

Self-supervised learning (SSL) [1] is now an established technique that uses large amounts of unlabelled data for pre-training the models to ensure they learn structures and patterns in images, text, spoken language. We use HuBERT [2] a state-of-the-art architecture that uses a masked language modeling (MLM) objective and generates these pseudo-labels through an initial k-means clustering step. It exploits SSL to capture speech information and is then fine-tuned on small amounts of labeled data for downstream tasks like ASR. This approach showed great performance improvements [3] over the MFCC-based feature extraction but the degree of performance improvement depends on the training data distribution. Some accents like the US accent are predominant (more than 50%) in the training data while some accents like the Scottish accent are much less represented. This skewness in data distribution is reflected in the ASR accuracy for these accents as the performance degrades for less-represented accents.

INTapt [4] tries to take speech representations of different accents closer to the L1 English accent like the US by concatenating the input with trainable prompts. The main feature of this training methodology is that they don't modify the weights of the backbone network during fine tuning. Another approach [5] defines accent-specific, accent-generic weights and uses a gating mechanism to choose the accent-specific weights. Here, they don't restrict the weights to specific accents and allow different weights to be selected by different examples of the same accent thus allowing access to a variety of units which might lead to richer accent information being stored in these units.

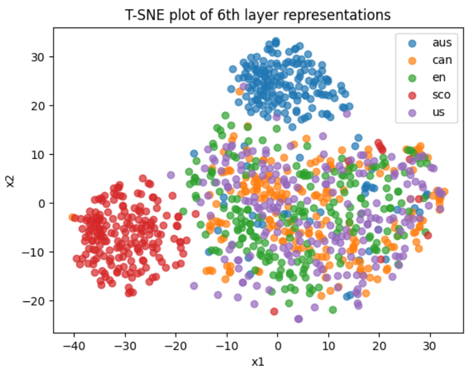
We employ the approach used by [6] for SSL pre-training and aim to improve ASR performance for accented speech by introducing accent information in these speech representations. We show that by using a trainable set of codebooks both during SSL pretraining and fine tuning, ASR performance can be improved for less-represented accents in the training data and completely unseen accents during inference. We also show evidence of accent information being encoded in speech representations through our approach. We analyze codebooks as MOEs to see if they further improve performance on downstream tasks like ASR, which might indicate the help of accent information from a similar accent.

Code is available at https://github.com/omkar-nitsure/Accent-Adaptation-Codebooks/tree/main

II. Analysis

A. Cluster Assignments

The analysis was done for the pretraining architecture defined in [7]. We extracted features from different layers of the SSL pretrained model after incorporating codebook cross-attention to see if we could observe some accent-wise clustering. We observed maximum accent-wise clustering and spacing for features extracted from the 6^{th} layer of the Transformer Encoder. We can see the clustering in the TSNE plot -



We then took some examples that had the same underlying transcript and were recorded for different accents. These examples are important for the analysis and evaluation as they have similar phonemes and the only difference is accent information. Then, we analyzed cluster assignment to see if it is similar for similar accents and widely different for accents belonging to widely different geographic locations. This was done by computing the number of common clusters for each set of pairs of accents and comparing the results for the baseline LibriSpeech [8] model against our approach. We can see that for pairs of similar accents like US-Canada, US-England, and England-Scotland, the number of common cluster assignments has increased in the codebook-based model than the baseline. It indicates that the representations have become more accent-aware.

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	-

Fig. 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	-

Fig. 2: Common labels - codebooks (same text)

B. Beam Search Decoding

We employ modified Beam-search decoding where the accent label is not available during inference and thus cross-attention is performed using each codebook while retaining only top score beams for further steps. We now analyze the codebook-selection distribution during decoding for various seen accents. Here, we consider only those examples where there was no accent confusion at the end of decoding. The filtering criteria was that more than 80% of the selected beams should belong to the true accent for the last 3 decoding steps. For all such examples, we defined a minimum threshold for the true accent and selected only those decoding steps that surpassed the threshold for soft-label generation. This was done to make sure that the distribution didn't become too soft due to the accent confusion in the initial decoding steps. We saw that we get completely different distributions by selecting different thresholds for the extent of true accent in a decoding step. We get closer to one-hot low entropy labels by increasing the threshold and getting high entropy soft labels for lower thresholds. The ideal distribution we choose for training the linear layer affects the extent of codebook mixing across accents and thus limits access to information from other accents. Thus it is important to choose the threshold properly. We found empirically that any threshold in the 0.4 - 0.5 gives a good probability that allows codebooks to be used as a Mixture of Experts (MOEs). The soft labels for thresholds of 0.4 and 0.6 are given below. The highlighted cells are the 2 leading accents picked during decoding.

Accents	AUS	CAN	EN	SCO	US
AUS	0.85	0.03	0.07	0.03	0.02
CAN	0.01	0.85	0.01	0.01	0.12
EN	0.09	0.02	0.83	0.05	0.01
SCO	0.15	0.0	0.0	0.85	0.0
US	0.02	0.12	0.03	0.02	0.81

Fig. 3: soft-distribution for threshold of 0.6

Accents	AUS	CAN	EN	SCO	US
AUS	0.68	0.06	0.13	0.08	0.05
CAN	0.04	0.56	0.08	0.07	0.25
EN	0.16	0.07	0.58	0.1	0.09
SCO	0.19	0.09	0.1	0.45	0.17
US	0.07	0.22	0.07	0.06	0.58

Fig. 4: soft-distribution for threshold of 0.4

III. Methodology

This section covers all experiments done during both encoder pre-training using fairseq [9] and fine tuning on espnet toolkit [10]. We tried a variety of manipulations using codebooks

A. Accent-specific and generic 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 learned by the model through data.

In another experiment, we further split the 500 entry codebook into 6 codebooks having 1 large codebook of 250 entries shared by all accents and 1 codebook of 50 entries each per accent. We assume that speech information that is common to all accents is dominant while the accent information is relatively small and hope the bigger accent-generic codebook captures that common speech information while the corresponding smaller accent-specific codebook captures the information unique to an accent.

B. K-Means centroid pooling

In this approach, we don't make any modifications to the original HuBERT architecture but modify the way k-means cluster labels are computed. Initially, we extract the features from the LibriSpeech checkpoint and train one k-means model 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. Pseudo-labels are then extracted using these cluster centers followed by HuBERT training for 2 epochs of 200k steps each. We performed an analysis of how different targets are distributed among cluster centers. So, for each accent, we analyzed the distribution of cluster label assignments 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. The distribution plot for Scottish accent 5 suggests a strong selection of the corresponding cluster labels while the trend is not so strongly evident for other accents 6.

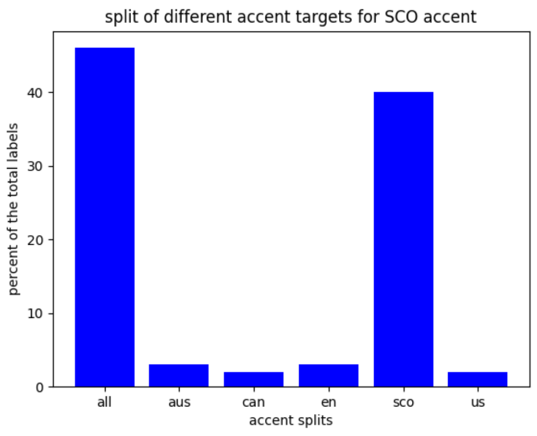


Fig. 5: cluster distribution for Scottish accent

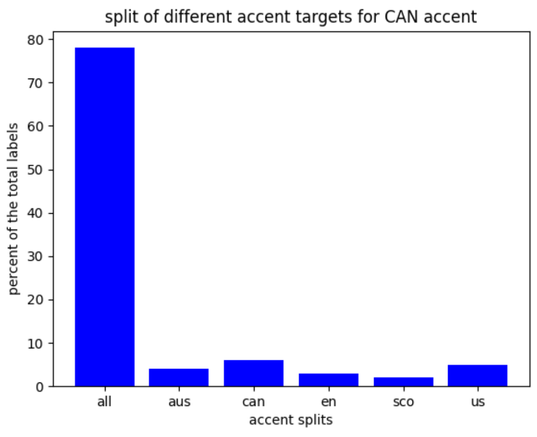


Fig. 6: cluster distribution for Canadian accent

C. Accent-specific and generic targets

HuBERT model pretraining is done using pseudo-labels generated through an offline clustering step. We modified this approach by having 2 pseudo-labels per frame during the pre-training where one of them is generated using the accent-specific k-means model defined earlier while the other uses the accent-generic k-means model. The final output layer now predicts these 2 clusters and 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 had similar pre-training loss indicating that the codebooks had already induced full accent-awareness and having a separate target did not give any additional benefits.

D. Accent-generic finetuning

We start with the checkpoint from Prabhu et al. [6]. We then merge all accent codebooks to form a single codebook and fine-tune the model further, where we use this codebook for all training examples. So it is accent-aware pretraining for 50 epochs followed by an accent-generic fine-tuning for 10 epochs.

E. Linear combination of accent-codebooks

We perform beam-search decoding for the entire training data where we use the checkpoint of Prabhu et al. [6]. We then generate accent distribution much like 3 for each training sample. This distribution is used to form an example-specific codebook which is a linear combination of the different accent-specific codebooks. The normalized probabilities are used as the weights. We experiment with this approach with different thresholds and this linear combination is applied to different HuBERT layers. The best results are achieved for using this approach on interspersed HuBERT layers and we improve over the one-hot selection used in [6].

F. Gating on the codebook selection

We use the same example-specific distribution as in the previous method. We now introduce a gating layer that gets the self-attention output of the 1^{st} Encoder layer as the input. It is supervised by the accent distribution with a cross-entropy loss. We use the output distribution of this layer as the coefficients for the codebook linear combination. The first training step is to do a warm start of 25 epochs where we only train the gating layer using the CE loss. This checkpoint is then used to perform the following experiments.

1) Train only gating layer on CTC loss
2) Train gating layer + codebooks on CTC loss

We also ran an experiment where we had a gating layer on all encoder layers but it did not give any improvements. So, for the following experiments, we have a single gating layer in the 1^{st} Encoder layer. The above experiment also suggests that training codebooks worsens performance. Hence, in the following experiments, we freeze codebooks.

1) Train 1^{st} encoder layer(which includes gating) on CTC
2) Train Encoder layers 1 to 4
3) Train everything else except codebook on CTC loss

1) Gating Layer architecture: The gating layer is an MLP where the first linear layer projects the features to a high dimensional space and then we progressively reduce the feature size till we get a distribution with 5 entries for the 5 accents in training data. The exact mathematical formulation is as follows.

The input to the gating layer is the self-attention output.

$$x = SelfAttention(h_i)$$

$$x = Add\_and\_Norm(h_i, x)$$

These modifications are as follows -

$$y = Conv(x, 1, k)$$

$$y = MeanPool(y)$$

$$y = FFN(y)$$

$$y = Activation(y)$$

$$y = FFN(y)$$

$$y = SoftMax(y)$$

$$C = y \oplus (C_1, C_2, ..., C_n)$$

Followed by cross attention :

$$z = CrossAttention(Q = x, K = C, V = C)$$

G. Modified Cross-attention

Initially, we were using codebooks as the *key* and *value* for the attention (standard cross attention). We now modify the cross attention in the sense that codebook will still be the *key* and *value* but we change the way attention is computed. We try 3 modifications and use the gating layer in each of them.

1) We perform the standard cross-attention for each of the accent codebooks as opposed to only the accent-specific codebook. We then take the linear combination of the attention outputs with the probabilities coming from the gating layer as the coefficients.
2) We combine 100 codebook vectors that we have per accent to form a large codebook with 500 entries. We then compute the attention probabilities and scale probabilities assigned to codeboook-vectors for an accent by the output of the gating layer for that particular accent.
3) Attention probabilities are computed similar to the 2^{nd} case. We now introduce a L1 loss between the sum of probabilities assigned to the codebook vectors of a particular accent and the output of the gating layer for that accent. This way we want the gating layer to control the attention distribution among codebooks.

H. Budget on Codebook Cross-attention coefficients

We now enforce a restriction on the codebooks that can be attended to during cross-attention. We introduce a learnable linear layer and pass the speech representations generated using self-attention through it. We use a single codebook of 50 vectors for cross-attention. This codebook is generated through a linear combination of accent-specific codebooks, the weights of which are outputs of the linear layer. This way, we hope that codebooks of accents in similar geographical locations will help improve ASR accuracy. The ideal probability distribution is computed using the beam search decoding approach discussed earlier and the linear layer is trained using a cross-entropy loss between the output distribution with this ideal distribution. Different settings can be tried for this setting some of which are listed below.

IV. Future Work

We have so far used a single probability distribution, but we can also have gating layers on different codebook layers which are supervised by different probabilities. This will allow us to have more accent mixing in the initial layers and more accent awareness in the later layers of the encoder.

1) Weights across Encoder layers: We can use different probability distributions for different layers of the encoder. We can use a high entropy distribution for initial layers to allow sufficient codebook mixing, and knowledge sharing and use a low-entropy hard-labeling for final layers. This will allow the representations to share knowledge across accents while maintaining the accent specificity induced by cross-attention in the final layers. We can additionally exploit the fact that speech representations in initial encoder layers do not have context information while those in final layers have rich context information through multiple attention computations.

V. Conclusion

Modern Automatic Speech Recognition (ASR) systems are becoming increasingly better but there is always an inherent bias induced by the training data distribution. The training data used for Self-Supervised-Learning (SSL) usually has only 1 or 2 dominant accents. This results in model-generating representations which improve accuracy for these accents 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 dominant accents like the US. 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. In addition to the accent-specific codebooks approach, the number and subsets of codebooks to which a frame can attend 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.

TABLE I: Comparison of the performance (WER %) of different methods. Numbers in bold denote the best WER. Ties are broken using overall WER.

Method	Size	Aggregated			Seen Accents					Unseen Accents								
		All	Seen	Unseen	AUS	CAN	UK	SCT	US	AFR	HKG	IND	IRL	MAL	NWZ	PHL	SGP	WLS
PRABHU ET AL. [6]	76M	18.94	13.6	23.1	12.78	15.07	15.56	11.19	13.4	21.12	26.18	29.08	20.9	32.08	18.58	26.4	34.22	18.34
BUDGET	76M	20.19			15.03	16.13	16.76	14.98	14.03	22.51	27.78	30.54	22.21	33.37	20.52	27.7	35.12	20.78
SINGLE CODEBOOK	76M	19.27			14.43	14.88	15.7	12.45	13.39	22.1	26.35	29.1	21.49	32.85	19.56	26.64	34.24	18.5
ACCENT-GENERIC FT	76M	19.66			14.54	15.48	16.19	15.61	13.4	21.98	27.47	29.46	21.67	33.7	20.11	27.22	34.35	20.16
CB LC (ONLY GATING)		21.6			19.02	16.24	19.26	15.88	14.99	24.08	29.04	31.95	23.63	33.62	23.16	28.43	36.3	22.8
CB LC (GATING + CB)		21.41			18.91	16.32	19.11	15.43	14.79	24.06	28.82	31.95	23.32	33.02	22.81	28.3	35.74	22.85
CB LC ENC LAYER 1-4		20.2			16.05	15.17	17.12	15.97	13.83	22.82	28.44	29.69	22.04	33.82	20.61	27.49	35.4	20.21
EVERYTHING EXCEPT CB		19.8			14.43	15.54	16.46	15.34	13.55	21.93	28.01	29.97	21.63	32.97	20.27	27.76	34.9	20.05
CA SCALING		19.7			14.51	15.4	16.24	14.62	13.58	22.14	27.96	29.67	21.82	33.05	19.95	27.03	34.6	20.36
CA L1-LOSS		19.61			14.59	15.37	16.3	14.62	13.52	22.25	27.81	29.86	21.63	33.17	19.83	26.64	34.24	20.52
DIST FIRST LAYER(0.6)	76M	24.11			16.46	20.15	19.95	18.95	17.71	26.11	33.5	36.64	25.89	38.75	22.64	32.88	43.13	21.14
DIST 1-4 LAYERS(0.6)	76M	22.98			16.21	18.34	18.71	18.41	16.72	25.14	32.35	34.02	25.09	37.06	22.72	31.21	40.64	23.01
DIST ALL LAYERS(0.6)	76M	22.68			17.78	18.01	19.21	17.6	15.63	25.55	31.8	33.62	24.44	38.63	22.9	31.35	39.36	23.11
AVG DIST(0.6)	76M	19.04			12.67	15.62	15.97	11.18	13.5	21.1	26.29	29.15	20.88	31.59	18.67	26.43	34.03	18.96
AVG DIST(0.4)	76M	19.15			13.19	15.03	15.6	11.73	13.55	21.87	27.21	29.29	20.98	32.85	18.72	26.11	34.48	17.72

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

[4] 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.

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

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

[7] D. Prabhu, A. Gupta, O. Nitsure, P. Jyothi, and S. Ganapathy, “Improving self-supervised pre-training using accent-specific codebooks,” in *Interspeech 2024*, pp. 2310–2314, 2024.

[8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, 2015.

[9] 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.

[10] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, “Espnet: End-to-end speech processing toolkit,” in *Interspeech 2018*, pp. 2207–2211, 2018.