BIASED SELF-SUPERVISED LEARNING FOR ASR

Florian L. Kreyssig[⋆,1]*, Yangyang Shi*[2]*, Jinxi Guo*[2]*, Leda Sari*[2]*,*
Abdelrahman Mohamed[2]*, Philip C. Woodland*[1]

[1]University of Cambridge, UK, [2]Meta AI, USA

flk24@cam.ac.uk

ABSTRACT

Self-supervised learning via masked prediction pre-training (MPPT) has shown impressive performance on a range of speech-processing tasks. This paper proposes a method to bias self-supervised learning towards a specific task. The core idea is to slightly finetune the model that is used to obtain the target sequence. This leads to better performance and a substantial increase in training speed. Furthermore, this paper proposes a variant of MPPT that allows low-footprint streaming models to be trained effectively by computing the MPPT loss on masked and unmasked frames. These approaches are evaluated for automatic speech recognition on the Librispeech corpus, where 100 hours of data served as the labelled data and 860 hours as the unlabelled data. The biased training outperforms the unbiased training by 15.5% after 250k updates and 23.8% after 100k updates on test-other. For the streaming models, the pre-training approach yields a reduction in word error rate of 44.1%.

***Index Terms*—** speech recognition, self-supervised learning, semi-supervised, unsupervised

1. INTRODUCTION

Self-supervised learning (SSL) for speech and audio [1] learns representations of audio by being trained to predict targets that are derived from the input audio itself. Such tasks include learning to predict multiple feature encodings of the input [2, 3], distinguishing near-by future speech features from temporally distant ones [4, 5] or directly predicting future frames [6], and prediction of audio features underneath a mask [7–10]. These self-supervised algorithms are able to leverage large amounts of unlabelled data effectively.

In automatic speech recognition (ASR), after a model is pre-trained using one of these algorithms on the unlabelled data, the model is finetuned using labelled data. Usually, the amount of labelled data is much smaller than the amount of unlabelled data. It is this small amount of labelled data that we will use to bias our self-supervised learning towards ASR.

One of the most successful self-supervised learning algorithms for speech has been Hidden-Unit BERT (HuBERT) [10]. HuBERT clusters speech frames in order to obtain a tokenisation of the input, which is used as the label sequence. During training, portions of the input are masked, and the neural network encoder is trained to predict the corresponding tokens that lie underneath the masks. This style of training will be referred to as masked prediction pre-training (MPPT). For the second iteration of training, hidden embeddings of a model after the first iteration of MPPT are clustered to obtain an improved tokenisation. In order to model the evolution of the speech frames, as needed in the first iteration of training, the model has to learn and represent the underlying phones. Hence, the tokenisation that is used for the second iteration of training has a strong correlation with the underlying phones [10].

Although HuBERT demonstrates competitive word error rates (WERs) after finetuning, there are some problems that cannot be overcome without any supervised signal. ASR performance is not only about obtaining a concise representation of the audio but also about learning which information to ignore (*e.g.* speaker or channel information). The cluster tokens that are obtained for training can represent various phones. However, they represent phones in various contexts in terms of the channel and the speaker. This leads to the MPPT task being much more difficult than it would otherwise need to be, which, in turn, leads to slow training and needing many updates ([10, 11] train for 400k updates, [12] trains for 1m updates).

To address this issue, this paper proposes a method to bias the HuBERT masked prediction task towards ASR using the labelled data used in the finetuning stage. To bias the MPPT towards ASR, the model used to obtain the tokenisation used in the second iteration of training is slightly finetuned using an ASR-loss. This leads to a tokenisation that is specialised for ASR and is also more consistent and predictable. The results will show that this leads to much faster training. Furthermore, by treating the biased tokens as discovered phonetic units, models can be pre-trained by predicting these tokens directly, even for unmasked audio. This paper proposes that this style of training, which is akin to cross-entropy training of DNN-HMM hybrid models [13], can be highly effective for training streaming ASR models with a restricted input window.

The remainder of this paper is organised as follows. Section 2 outlines the biased semi-supervised learning method, a variant of MPPT for low-footprint streaming models and discusses related work. Sections 3 and 4 present the experimental setup and results and Section 5 gives conclusions.

2. METHOD

2.1. HuBERT

This section summarises [10], the model structure, how to obtain the label sequences and how to train the model on this label sequence.

In [10], the model structure uses a convolutional encoder to encode the raw waveform input. This is followed by a convolutional layer with a large kernel size that is used for positional encodings. This is followed by multiple (*e.g.* 12 in HuBERT-Base) Transformer layers [14]. The output embeddings from the final Transformer layer are then used to predict the underlying token label. This paper uses a modified and more efficient model structure that is explained in detail in Section 3.2.

To train the model, a label sequence is obtained by clustering MFCC features or hidden embeddings. Clustering is performed

⋆Work done while Florian Kreyssig was an Intern at Meta AI.

globally by using the entire corpus (or a representative sample of it) and not locally for each sequence. The model is then trained by first masking portions of the output of the convolutional encoder and then predicting the labels that lie underneath those masks.

HuBERT training occurs in multiple iterations. The first iteration of training uses a target sequence obtained by clustering speech features using KMeans. For the succeeding iterations, hidden embeddings from the trained model of the previous iteration are clustered. The correct layer to use for this can be found based on the metrics described in Section 2.4.

Details about the exact training parameters for the baseline models used in this paper are given in Section 3.4. This training protocol will be referred to as *unbiased*.

2.2. Biased Masked Prediction Pre-Training

Clustering is challenging and inherently ambiguous when samples of different clusters are not well separated in the feature space. Without a supervision signal, clustering is not well-defined. This paper introduces a supervised signal into the pipeline of Section 2.1 through a small but important change. The first iteration of training is identical to the unbiased MPPT training. The difference to the unbiased MPPT training comes before the second iteration of MPPT. To bias the MPPT towards the desired task, the trained model of the first iteration is finetuned using a supervised loss for a small number of updates. This biases the model and, thus, the clusters towards the supervised task. Whilst this paper uses a Connectionist Temporal Classification (CTC) [15] loss to make the MPPT ASR specific, this biasing step could be done using any speech processing task or even multi-task training. After the biasing step, the embeddings are clustered using KMeans clustering. Our results will later show that compared to clustering unbiased embeddings, a far smaller number of clusters is necessary and even advantageous. More detail about the exact training parameters is given in Section 3.5. This training protocol will be referred to as *biased*.

In self-training [16–20] supervised data is used to build a seed model, which is then used to label the unlabelled data. The final model is then trained using the combined data. Whilst these methods have a similar operating point to our approach, the difference is that these methods are bootstrapped from a fully supervised model, which can easily overfit the initial data set. Our method is biased to the supervised set using a small number of updates to avoid any overfitting. Furthermore, self-training methods rely on a strong language model as shown by [21]. Our method does not rely on a language model for pre-training the encoder. [11] uses self-training in an MPPT setup where the tokenisation is obtained using a seed hybrid or CTC system.

2.3. Low-footprint streaming models

MPPT is challenging for streaming models that have small input contexts. The reason for this is that too much of the input is masked in order to make accurate predictions. Making the task prohibitively difficult leads to bad representations. On top of this, removing the 1D-Convolutional layer further improves efficiency but would make the MPPT task even harder. This paper solves this issue by computing the classification loss on both the masked and the unmasked frames. By treating the clusters obtained from clustering the embeddings of a second-iteration biased model as acoustic units, the loss obtained from the unmasked frames can be treated as cross-entropy (CE) training for DNN-HMM hybrid models [13, 22]. In practice, the input was masked as before, but the loss-weight on the masked frames and the loss-weight on the unmasked frames are both 0.5. Most of the learning comes from the unmasked frames, and the masking serves as a form of regularisation. [23, 24] train a streaming model from a non-streaming model using teacher-student training. The teacher in [24] is initialised from a self-supervised model.

2.4. Additional Metrics

The training and evaluation pipeline, including pre-training, finetuning and decoding, has many stages. Hence, the final WER should not be the only metric to analyse the examined approaches.

Masked Prediction Accuracy This is the accuracy that the model achieves on the MPPT task. For the same label sequence, this metric is correlated with final WER. For the same model architecture and training steps, this metric is an indication of how "easy" the task is. The task can be made easier simply by reducing the number of clusters K. It can also be easier because the tokenisation is more consistent. This increased consistency can be viewed to be analogous to having less label noise in supervised training. Thus, this is desirable.

Cluster-Purity and Label-purity These two metrics are calculated based on labelling each frame with its corresponding cluster and with a ground-truth label. In this work, the data is aligned with 870 clustered bi-character units. Cluster-purity is the maximum achievable accuracy of predicting the cluster purely based on the label. Label-purity is the maximum achievable accuracy of predicting the ground-truth label purely based on the cluster. These metrics indicate how much the cluster tokens are correlated with the ground-truth labels, and therefore they give an indication towards how similar the MPPT is to ASR training. Hence, these metrics are highly correlated with final decoding WER. It is important to note that when increasing K for the same embeddings, the cluster-purity will go down, and the label-purity will go up. Therefore these statistics are difficult to compare for models with different K.

3. EXPERIMENTAL SETUP

3.1. Data

The proposed methods were evaluated on the Librispeech data [25]. It contains 960 hours of data from audiobooks. A random 100-hour subset serves as the supervised data set that is used for biasing and finetuning. The other 860 hour serves as the unsupervised data set. For the MPPT, the whole 960 hours of data is used for training. The utterances in Librispeech were segmented to be of length $\leq 10s$.

3.2. Model

The model structure used in this paper has several differences from the one used in [10]. The convolutional encoder with an output frequency of 50 Hz, which processes the raw-waveform input, is replaced with a conventional 80dim FBANK analysis operating at 100 Hz with a 25 ms window size. The size of the convolutional kernel of the 1D-Convolution layer used for relative positional encodings is reduced from 127 (at 50 Hz, equiv. to 2.54 seconds) down to 31 (at 100 Hz, equiv. to 0.31 seconds). This layer is followed by stacking 4 frames which reduces the frequency of the encoder outputs down to 25 Hz. Furthermore, the Transformer layers [14] are replaced by the much more efficient Emformer layers [26]. The encoder has 20 Emformer layers with an embedding dimension of 512, a feed-forward dimension of 2048, and 8 attention heads. For most experiments, the Emformer layers use a right-context of 1 and a centre-context of 160. For the low-footprint streaming models (see

Section 2.3), a centre-context of 4 was used, equivalent to 160 ms, and the 1D-Convolution layer was removed. This yields an algorithmic latency induced by the encoder of 120 ms.

3.3. Notation

To make this paper more readable, models pre-trained with MPPT are given a model ID X_Z^Y. X relates to the vectors that are clustered to obtain the label-sequence for training, *e.g.* M for 39dim-MFCC features. Y relates to how long the model was trained *e.g.* 100k for 100k updates. Z relates to the number of clusters used, *e.g.* 100. M_{100}^{100k} is a model trained for 100k updates for which the training labels are obtained by clustering 39dim-MFCC features using K=100.

3.4. Baseline Pipeline

This pipeline is aligned with HuBERT-Base [10] and is entirely unsupervised. The first iteration of MPPT uses a label sequence obtained by clustering 39dim-MFCC[1] features using K=100. These features are at 100 Hz, and the label sequence is sub-sampled by a factor of 4 to align with the 25 Hz output frequency of the model. The models that are trained for 100k and 250k updates have the IDs M_{100}^{100k} and M_{100}^{250k}, respectively and will also be referred to as iteration-1 models. MFCC features are used over FBANK features due to their better match with the feature independence assumption of the squared distance metric of the KMeans algorithm. The second iteration of training uses a label sequence obtained by clustering the embeddings of one of the layers of M_{100}^{250k} using K=500. For this second clustering step, a layer that is in the middle of the model should be used, *i.e.* if 12 layers are used, use layer 6. For this paper's 20-layer model, layer 10 was used. The models that are trained for 100k, 250k and 400k updates have the IDs H_{500}^{100k}, H_{500}^{250k} and H_{500}^{400k}, respectively. These models will also be referred to as iteration-2 models.

3.5. Biased Pipeline

For the biased pipeline, the starting point is the unbiased iteration-1 model that was trained for 250k updates (*i.e.* M_{100}^{250k}). This model is finetuned for 20k updates using a CTC loss on the labelled data. This is the biasing step. These 20k updates come after the linear output layer is trained for 10k updates itself, as done for the finetuning (see Sec. 3.7) as well[2]. Then the embeddings of the 17$^{\text{th}}$ emformer-layer are clustered to obtain the target sequence. The 17$^{\text{th}}$ layer was picked as it maximised the label-purity. The more similar the model is to an ASR model, the closer the ideal layer for the embeddings to the output. For the biased training, both K=500 and K=100 are tested. These pre-trained models are C_β^α.

3.6. Training Parameters

For the MPPT of any iteration (biased or unbiased), around 50% of the input frames are masked by applying overlapping masks of length 20 frames (equivalent to 200 ms; [10] used masks length 10 for 50 Hz features *i.e.* also 200 ms). This is done by choosing 4% of frames as starting frames for the masks of length 20. The classification loss is computed only for the masked frames. The models are trained on 32 GPUs with a batch size of 87.5 seconds per GPU (for 250k updates, this is equivalent to 169 epochs). The learning rate ramps up linearly for the first 8% of updates and then decays linearly down to 0. The peak learning rate is 5e-4.

[1] 13dim plus delta and deltadelta.

[2] Finetuning for only 5k updates also gave good results, but training for too long (*e.g.* 80k updates) started to give worse results, which could be due to overfitting to the supervised subset of the data.

ID	Embeddings		K	T-steps	M-acc
	Model	Layer			
M_{100}^{100k}	39D-MFCC		100	100k	48.4
M_{100}^{250k}	39D-MFCC		100	250k	49.5
H_{500}^{100k}	M_{100}^{250k}	10	500	100k	58.3
H_{500}^{250k}	M_{100}^{250k}	10	500	250k	59.1
H_{500}^{400k}	M_{100}^{250k}	10	500	400k	61.0
C_{500}^{100k}	M_{100}^{250k} + ft.	17	500	100k	67.7
C_{500}^{250k}	M_{100}^{250k} + ft.	17	500	250k	70.1
C_{500}^{400k}	M_{100}^{250k} + ft.	17	500	400k	71.5
C_{100}^{100k}	M_{100}^{250k} + ft.	17	100	100k	77.0
C_{100}^{250k}	M_{100}^{250k} + ft.	17	100	250k	79.1

Table 1. Masked prediction pre-training. ID is the ID of the model that was trained. Embeddings are the vectors clustered into K clusters to obtain the training sequence. M-acc is the model's accuracy on the masked frames on the validation set. T-steps is the number of training updates. '+ft.' indicates the biasing step.

Embeddings		K	cluster-	label-
Model	Layer		purity	purity
39D-MFCC		100	0.074	0.064
M_{100}^{250k}	10	500	0.079	0.162
M_{100}^{250k}	10	100	0.152	0.114
M_{100}^{250k} + ft.	17	500	0.299	0.197
M_{100}^{250k} + ft.	17	100	0.330	0.194

Table 2. Label purity is the label prediction accuracy if cluster is known. Cluster purity is the cluster prediction accuracy if label is known. Labels are a set of 870 clustered bi-characters.

3.7. Supervised Baselines and Finetuning

The pre-trained models are finetuned using the CTC loss [15]. The output units are 5000 sentence pieces [27] with Byte Pair Encoding (BPE) [28] as the segmentation algorithm. For finetuning, the projection and output layer of the pre-trained models are replaced with an output layer mapping to the 5001 output units (incl. the blank unit). The pre-trained encoder is frozen for the first 10k updates, and the entire model was trained for 100k updates (10k+90k). The peak learning rate was 2.5e-5, which is 40x smaller than for the purely supervised models (1e-3). The purely supervised models have the same model structure as the pre-trained models. All models use SpecAugment [29] without time-warping. For decoding, the official Librispeech 4-gram language model was used.

4. EXPERIMENTAL RESULTS

The masked prediction accuracies are given in Table 1. The label-statistics are given in Table 2 and WERs are given in Table 3. WERs in the text will refer to test-other except when explicitly mentioned.

Data	Init.	test-clean	test-other
100h	-	8.68	17.76
100h	M_{100}^{100k}	6.98	14.01
100h	H_{500}^{100k}	6.52	13.00
100h	C_{500}^{100k}	5.28	10.39
100h	C_{100}^{100k}	5.06	9.91
100h	S_{100}^{100k}	4.64	8.93
100h	-Conv1D	4.95	10.30
100h	+K=2000	4.81	9.93
100h	M_{100}^{250k}	6.26	12.68
100h	H_{500}^{250k}	5.27	10.72
100h	C_{500}^{250k}	4.67	9.60
100h	C_{100}^{250k}	4.48	9.06
100h	S_{100}^{250k}	4.50	8.75
100h	H_{500}^{400k}	5.02	10.23
960h	-	3.26	7.37

Table 3. Results on Librispeech test sets after CTC finetuning. Data is hours of finetuning data. Init. is the model used for initialisation.

4.1. Baselines

The WERs of the baseline 100-hour and 960-hour purely-supervised models are 17.76% and 7.37%, respectively. The unbiased models M_{100}^{250k} and H_{500}^{250k} achieve WERs of 12.68% and 10.72%. Thus, two iterations of unbiased pre-training reduce the WER by 39.6% over the supervised baseline and "recover" 67.8% of the reduction in WER obtained from 860 hours of labelled data (17.76% $\rightarrow$ 7.37%). Training for the second iteration for 100k updates (*i.e.* initialised from H_{500}^{100k}) only yields a 26.8% reduction (13.00% WER).

The clusters derived from layer 10 of M_{100}^{250k} give better predictions of the bi-character labels than the clusters from the MFCC features (label-purity of 0.114 vs 0.064) for K=100.

Looking at Table 1, it can be seen that even with the increase in the number of clusters from 100 to 500, the masked-prediction accuracy is higher for H_{500}^{250k} than for M_{100}^{250k} (59.1% vs 49.5%). This is likely due to the tokenisation being more consistent.

4.2. Biased HuBERT training

To recall, before clustering, the iteration-1 model, M_{100}^{250k}, is finetuned for 20k update steps. The embeddings of the 17$^{\text{th}}$ layer are clustered. For K=500, the purity statistics improve significantly (see Table 2) over the unbiased models. The cluster-purity increases from 0.079 to 0.299 and the label-purity from 0.162 to 0.194. The strong increase in cluster-purity confirms the idea that without supervision, many cluster tokens represent the same phonetic unit but in different acoustic contexts (speaker, channel, noise). Part of the challenge of acoustic modelling is to be invariant to changes in context. Thus, a higher cluster-purity is desired. Another effect of the biasing is that K=100 seems to be large enough to model the bi-character labels. For the unbiased embeddings, the label-purity drops from 0.162 down to 0.114 when changing K=500 down to K=100. For the biased embeddings, only a small drop from 0.197 to 0.194 is observed. This observation motivated the decision to try K=100 for the biased second iteration of training and not only K=500 as commonly used [10–12].

From Table 1, it can be observed that the biased labels are much cleaner and more predictable. The masked prediction accuracy, after 250k training steps, increases from 59.1% to 70.1% (K=500 for both). Using K=100 has a masked prediction accuracy of 79.1%.

For 250k updates, the biasing model (init. from C_{500}^{250k} outperforms the unbiased model (init. from H_{500}^{250k}) by 10.4% (10.72% WER vs. 9.60% WER). Using K=100 further reduces the WER to 9.06% (15.5% rel. WER reduction). Therefore the biased training is able to recover 83.7% of the reduction in WER that could be obtained by using 860 hours of labelled data in comparison to the 67.8% achieved by the unbiased pre-training.

The improvements due to the biasing are even stronger when training for only 100k updates. The relative WER reduction over the unbiased baseline is 20.1% for K=500 and 23.8% for K=100. This demonstrates that biased SSL improves not only the final performance but also the convergence speed. The main reason for this is likely the improved consistency of the label sequences and the thus increase in the predictability of the cluster-tokens, which can be seen from the increase in masked-prediction accuracy in Table 1. Label noise in supervised learning, which can be seen as analogous to less consistency in the tokenisation obtained from clustering, hinders convergence as shown by [30].

For an iteration-3 model where the label sequence is obtained by clustering the embeddings of C_{100}^{250k}, the convergence speed increases even more (this model is called S_{100}^{α}). Even though WER for 250k updates is only 3.4% lower (8.75% vs 9.06%) than for the biased iteration-2 model (both with K=100), for 100k updates, the WER is 9.9% lower (8.93% vs 9.91%). This again reinforces the connection between how clean the label-tokens are and the convergence speed.

4.3. Low-footprint streaming models

The labels used for the biased iteration-3 models (S_{100}^{α}) are treated as very good acoustic units and used for the CE-style pre-training for which the loss comes equally from the masked and the unmasked frames. In the first experiment, this style of training is used while at the same time removing the 1D-Convolutional layer. This increases the WER from 8.93 to 10.30%. Using a larger token set of K=2000 reduces the WER down to 9.93%. Here, a larger token set helps because another interpretation of this training method is that the model is distilled from C_{100}^{250k}. A larger token set gives more information about C_{100}^{250k}.

For a low-footprint streaming model (with an algorithmic latency of 120 ms), the 100-hour supervised WER is much worse at 12.45%/24.07% WER on test-clean/test-other. Using our proposed style of training, the WER is reduced to 6.13%/13.34% for the same low-footprint streaming model. A reduction in WER of 50.8%/44.6%.

5. CONCLUSIONS

Biased self-supervised learning is a framework for semi-supervised learning of speech representations by biasing the labels of the masked-prediction pre-training towards ASR. Our experiments show large reductions in WER over the unbiased training mode. Furthermore, a substantial improvement in training speed can be seen such that biased training for 100k updates outperforms unbiased training for 400k updates. This achievement also highlighted the strong connection between label noise and convergence speed, which should be kept in mind for developing future self-supervised learning algorithms. Furthermore, we showed the use of biased SSL for small-footprint streaming ASR models.

6. REFERENCES

[1] A. Mohamed, H.y. Lee, L. Borgholt, J.D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.W. Li, K. Livescu, L. Maaløe, T.N. Sainath, & S. Watanabe, "Self-supervised speech representation learning: A review," *arXiv preprint arxiv:2205.10643*, 2022.

[2] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, & Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," *Proc. Interspeech*, Graz, 2019.

[3] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, & Y. Bengio, "Multi-task self-supervised learning for robust speech recognition," *Proc. ICASSP*, Barcelona, 2020.

[4] A. van den Oord, Y. Li, & O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[5] S. Schneider, A. Baevski, R. Collobert, & M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *Proc. Interspeech*, Graz, 2019.

[6] Y. Chung, W. Hsu, H. Tang, & J.R. Glass, "An unsupervised autoregressive model for speech representation learning," *Proc. Interspeech*, Graz, 2019.

[7] A. Baevski, S. Schneider, & M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *Proc. ICLR*, Addis Abab, 2020.

[8] A. Baevski, Y. Zhou, A. Mohamed, & M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Proc. NeurIPS*, Vancouver, 2020.

[9] W.N. Hsu, Y.H.H. Tsai, B. Bolte, R. Salakhutdinov, & A. Mohamed, "HuBERT: How much can a bad teacher benefit ASR pre-training?," *Proc. ICASSP*, Toronto, 2021.

[10] W.N. Hsu, B. Bolte, Y.H.H. Tsai, K. Lakhotia, R. Salakhutdinov, & 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.

[11] C. Wang, Y. Wang, Y. Wu, S. Chen, J. Li, S. Liu, & F. Wei, "Supervision-guided codebooks for masked prediction in speech pre-training," *Proc. Interspeech*, Incheon, 2022.

[12] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, & others, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–14, 2022.

[13] G.E. Dahl, D. Yu, L. Deng, & A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, & I. Polosukhin, "Attention is all you need," *Proc. NeurIPS*, Long Beach, 2017.

[15] A. Graves, S. Fernández, F.J. Gomez, & J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," *Proc. ICML*, Pittsburgh, 2006.

[16] G. Zavaliagkos & T. Colthurst, "Utilizing untranscribed training data to improve performance," *Proc. Broadcast News Transcription and Understanding Workshop*, Landsdowne, 1998.

[17] T. Kemp & A. Waibel, "Unsupervised training of a speech recognizer using TV broadcasts," *Proc. ICSLP*, Sydney, 1998.

[18] F. Wessel & H. Ney, "Unsupervised training of acoustic models for large vocabulary continuous speech recognition," *Proc. ASRU*, Madonna di Campiglio, 2001.

[19] L. Lamel, J. Gauvain, & G. Adda, "Unsupervised acoustic model training," *Proc. ICASSP*, Orlando, Florida, USA, 2002.

[20] D.S. Park, Y. Zhang, Y. Jia, W. Han, C.C. Chiu, B. Li, Y. Wu, & Q. Le, "Improved noisy student training for automatic speech recognition," *Proc. Interspeech*, Shanghai, 2020.

[21] E. Wallington, B. Kershenbaum, O. Klejch, & P. Bell, "On the learning dynamics of semi-supervised training for asr," *Proc. Interspeech*, Brno, 2021.

[22] H. Bourlard, H.A. Bourlard, & N. Morgan, *Connectionist speech recognition: A hybrid approach*, Springer, 1994.

[23] T. Doutre, W. Han, M. Ma, Z. Lu, C.C. Chiu, R. Pang, A. Narayanan, A. Misra, Y. Zhang, & L. Cao, "Improving streaming automatic speech recognition with non-streaming model distillation on unsupervised data," *Proc. ICASSP*, Toronto, 2021.

[24] X. Yang, Q. Li, & P.C. Woodland, "Knowledge distillation for neural transducers from large self-supervised pre-trained models," *Proc. ICASSP*, Singapore, 2022.

[25] V. Panayotov, G. Chen, D. Povey, & S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *Proc. ICASSP*, Brisbane, 2015.

[26] Y. Shi, Y. Wang, C. Wu, C.F. Yeh, J. Chan, F. Zhang, D. Le, & M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," *Proc. ICASSP*, Toronto, 2021.

[27] T. Kudo & J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *Proc. EMNLP*, Brussels, 2018.

[28] R. Sennrich, B. Haddow, & A. Birch, "Neural machine translation of rare words with subword units," *Proc. ACL*, Berlin, 2016.

[29] D.S. Park, W. Chan, Y. Zhang, C.C. Chiu, B. Zoph, E.D. Cubuk, & Q. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Proc. Interspeech*, Graz, 2019.

[30] C. Zhang, S. Bengio, M. Hardt, B. Recht, & O. Vinyals, "Understanding deep learning requires rethinking generalization," *Proc. ICLR*, Toulon, 2017.