Adaptive Decoding for Efficient Automatic Speech Recognition

Anonymous ACL submission

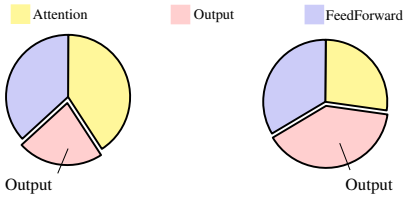
Abstract

The latency and computational demand of End-to-end (E2E) automatic speech recognition (ASR) models hinder their deployment on lightweight devices. We find that, although these models can be tuned for efficiency concerns, the computational burden of large vocabularies remains a challenge. In this paper, we propose an adaptive decoding method (ADD) to speed up E2E ASR systems. It segments the vocabulary based on the inherent characteristics of speech, enabling the models to predict each word with a much smaller vocabulary. Our method significantly reduces the FLOPs required for calculations. We also find that the unit-based methods, developed through self-supervised learning, capture acoustic features well and achieve performance comparable to the phone-based methods.

1 Introduction

End-to-end (E2E) fashion based on neural networks has become the mainstream for automatic speech recognition (ASR) tasks (Dong et al., 2018; Li et al., 2022). Though E2E ASR achieves better performance and generalization, it still has high latency and requires a high-performance computing device. The latency can be much worse, especially on the mobile phone or embedded device that cannot employ GPUs (Shangguan et al., 2021). This is caused by the advanced attention-based model always containing a multitude of parameters and requiring frequent matrix calculations (Vaswani et al., 2017).

Many speedup methods have been proposed to solve this issue (Tay et al., 2022). Generally, there are two ways: 1) model compression, such as the knowledge distillation (KD) method (Hinton et al., 2015) and quantization (Gholami et al., 2021), and 2) attention module acceleration, such as Average Attention (Zhang et al., 2018) and Flash Attention (Dao et al., 2022). However, most of the methods aim for a general domain or are tested under isolated circumstances. There is a lack of analysis on the effect of these methods working together on E2E ASR models deployed on on-device platforms.



(a) Base model (b) Small model

Figure 1: The FLOPs proportion of output layer in the decoder. It rises sharply when the model is compressed.

We carried out an analysis to verify the effect of mainstream methods on lightweight devices. However, the speedup for attention does not bring a significant improvement after applying the model compression method. Unlike the conventional conclusion, the output layer takes a considerable proportion in the highly optimized system as Figure 1 shows. The reason is that the large size of the vocabulary causes a multitude of softmax and multiplication costs (Joulin et al., 2017; Stevens et al., 2021; Banerjee et al., 2020). Thus, an intuitive strategy is cutting off the size of the vocabulary.

We propose a novel adaptive decoding method (called ADD), which is based on the inherent characteristics of voice. We cluster all the words based on the phone, then split the entire vocabulary into small vocabularies from the perspective of pronunciation. At inference time, the model first predicts the most proper vocabulary with minimal cost, then generates the token from the selected vocabularies, which only contain hundreds of words. This method finds the right answer from the most likely candidates rather than a mass of irrelevant words, thus speeding up the decoding process. The contributions of this paper are as follows:

• We find that the output layer poses a computational challenge for ASR, while this issue has

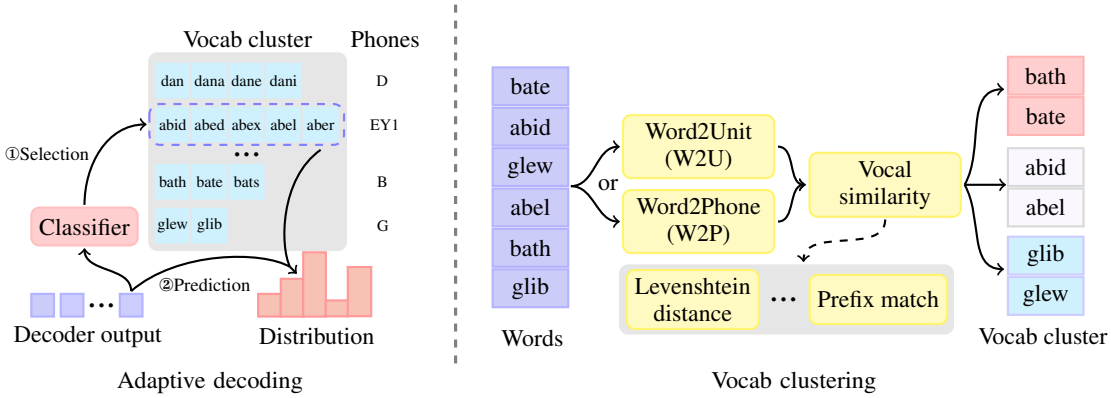


Figure 2: An overview of our method.

received relatively less attention in previous work.

• We propose the ADD method, which adaptively employs a smaller vocabulary to speed up ASR decoding.

• Our analysis shows that the unit-based method captures acoustic features well, providing an alternative to phonetic methods.

2 Acceleration Analysis for E2E ASR

We conducted experiments using two advanced strategies on the state-of-the-art (SOTA) E2E ASR model, Conformer (Gulati et al., 2020), to verify their on-device effects. We apply the KD method to compress the model size and reduce the computing cost. The classic KD method consists of online and offline approaches. The ASR task is prone to overfitting (Park et al., 2019), thus the offline method cannot supply additional information. We choose the online method, which learns the distribution from the teacher model and sets the weight to 0.5 to balance the losses. We find that the KD method still works well as Table 1 shown.

Another attractive strategy is optimizing attention modules. We selected the Average Attention (AAN) (Zhang et al., 2018), which has been widely verified in natural language processing tasks. The method replaces the self-attention with an average distribution in the decoder to achieve decent speedup. Table 1 shows it only achieves a 3.9% improvement in FLOPs based on a small model. This proves that the attention module is not always the key module when the model deploys on a lightweight device with small settings.

Figure 1 shows that the output layer starts to play an important role if the model becomes small. Thus applying a smaller vocabulary can further speed up the inference. However, Table 1 shows a significant decrease in performance, indicating that this method is not a proper solution.

Method	**Param. (M)**	**Test (WER)** Clean	Other	**FLOPs** (M)
Base	45.42	3.11	7.34	426.97
Small	17.52	3.93	8.52	95.46
+KD	17.52	3.72	8.43	95.46
+KD+AAN	17.52	3.61	8.56	91.82
Small *w/* 1k vocab	16.22	3.83	9.57	25.67

Table 1: The acceleration performance comparison

3 Method

3.1 The Adaptive Decoding Strategy

We show an overview of adaptive decoding in Figure 2. For each step, the model first selects the vocab that has been built by cluster method (refer the Figure 2 right). The model predicts the word based on the small vocab rather than the whole.

Training We first use the output feature $\mathbf{o}$ of the decoder to select the corresponding vocabulary. Since the next stage requires the argmax of this prediction, we utilize the Gumbel softmax (Jang et al., 2016). It enables choosing the discrete vocabulary in a fully differentiable way and keeps training and decoding consistent. We convert the $\mathbf{o}$ with linear layer to classification logits $\mathbf{c} \in \mathbb{R}^N$ and N is the number of prepared vocabs. We can get the probability for n-th as following

$$p_n = \frac{\exp(c_n + g_n)/\tau}{\sum_{i=1}^{N} \exp(c_i + g_i)/\tau} \quad (1)$$

where τ is the temperature and we gradually decrease it to control distribution from smooth to sharp. The noise $g = -\log(-\log(u))$ and u is sampled from the uniform distribution $\mathcal{U}(0, 1)$.

Method	Param. (M)	Test (WER) Clean	Other	Avg.	Speed (tokens/s)	FLOPs (M)	Rate of Acceleration
Wenet $w/$ LM (Zhang et al., 2022)	34.76	3.09	7.40	5.25	–	394.97	–
Base	45.42	3.18	7.54	5.36	60.00	426.97	100.00%
Base+Random		3.83(↓ 0.65)	8.37(↓ 0.83)	6.10			
Base+W2P	47.98	3.36(↓ 0.18)	7.93(↓ 0.39)	5.65	65.00	369.37	108.33%
Base+W2U		3.03(↑ 0.15)	7.93(↓ 0.39)	5.48			
Small*	17.52	3.62	8.68	6.15	103.67	91.82	172.77%
Small*+Random		3.82(↓ 0.20)	9.34(↓ 0.66)	6.58			
Small*+W2P	18.92	3.70(↓ 0.08)	9.12(↓ 0.44)	6.41	118.33	25.67	197.22%
Small*+W2U		3.71(↓ 0.09)	8.89(↓ 0.21)	6.30			

Table 2: The clustering effect on real performance. *small** denotes the small model with KD and AAN methods.

We calculate the possibility of all the words to train the model stably. For the n-th vocab, the corresponding output layer converts the $\mathbf{o}$ to the predicted logits $\mathbf{l}$. Then we get the normalize $\mathbf{l}$ by the softmax with static temperature 1.0:

$$p_{n,k} = \frac{\exp(l_k)}{\sum_{i=1}^{V_n} \exp(l_i)} \quad (2)$$

where V_n is the size of n-th vocab. The predicted probability of the model is $p_n \times p_{n,k}$ for k-th token in n-th vocab. Thus the final cross-entropy loss can be denoted as following for each word:

$$\mathcal{L} = \sum_{n}^{N} \sum_{k}^{V_n} y_{n,k}(\log(p_n) + \log(p_{n,k})) \quad (3)$$

where $y_{n,k}$ is the gold value of k-th position in the n-th vocab.

Inference During the forward pass, we take the i-th vocab which is chosen by $i = \underset{i}{\operatorname{argmax}}\ p_n$. We then search for the best candidate from the selected vocab. We only use the selected vocab when applying the beam search and achieve a better speedup effect.

3.2 Word Clustering

The performance of adaptive decoding relies on the well-set vocab sets. We cluster words with the prior knowledge (e.g. phone) as the vocab according to the vocal feature of speech.

Word2Phone cluster The Word2Phone (**W2P**) belongs to the Text analysis sub-task (Mehl, 2006) during text-to-speech processing. We convert all the words to phonetic level according to the dictionary. Then the phone can be viewed as a feature to cluster words. Due to the order of the phone also differing the word, we use the prefix match to select the cluster. We set the cluster rule that the word in one cluster should contain m same prefix phones where m is a super-parameter. To control the size of the vocabulary, we merge the clusters to N vocabs.

Word2Unit cluster We turn to use the speech unit to cluster words for languages with no phonetic information (called **W2U**). The speech unit is obtained by the HuBert (Hsu et al., 2021) which is pre-trained by a self-supervisor method. Then we reverse the ASR dataset and replace the audio with unit as the text-to-unit corpus. After training a text-to-unit model, we convert each word to units. Differing from the phone, the unit for word has a much longer length and it is hard to find the proper cluster center. For the former problem, we apply the metric to evaluate the similarity of two sequences. For the latter problem, we sample N words as the cluster center (called policy π). For every other unit of words, we compute the similarity with the cluster center by negative Levenshtein distance and put the word into the cluster with max similarity. To balance the size of vocabularies, the score of π is computed by the Entropy as follows:

$$e_\pi = \sum_{n}^{N} \left(\log(\frac{V_n + 10}{V + 10N}) \frac{V_n + 10}{V + 10N} \right) \quad (4)$$

We repeat this process hundreds of times and use the policy with the maximum Entropy.

4 Experiments

4.1 Data and Model Settings

We selected LibriSpeech (Panayotov et al., 2015) as the training and evaluation dataset. The default vocabulary size is 10, 000, which is generated by sentencepiece (Kudo and Richardson, 2018). About the phonetic dictionary for word2phone, we use The CMU Pronouncing Dictionary. The N is set to 10 by default. The m for prefix match is set to 1.

Method	N	Epoch	Test (WER) Clean	Other
Random	10	100	19.37	34.80
Frequency (Joulin et al., 2017)	5	28	26.14	34.18
	10	89	14.38	24.84
W2P	6	100	10.53	22.68
	11	79	**10.22**	**22.62**
W2U	5	100	19.23	38.47
	10	96	12.99	26.17
	20	100	15.69	30.04

Table 3: The comparison of clustering approaches

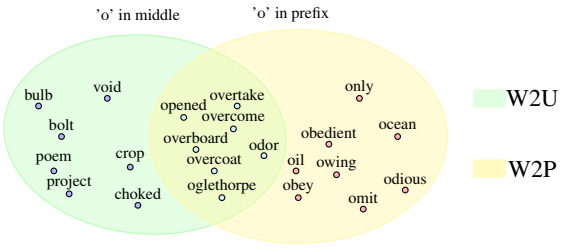


Figure 3: A case of W2P and W2U clustered vocabs.

The *base* model is a 12-layer Conformer as the encoder and a 6-layer vanilla Transformer (Vaswani et al., 2017) decoder to reproduce the subsampling method to establish a baseline (Zhang et al., 2023), with a hidden layer dimension of 256. For the *small* model, we used a 16-layer Conformer as the Encoder and a 3-layer Transformer Decoder, with a hidden layer dimension of 144. All models are implemented in Fairseq toolkit (Ott et al., 2019; Wang et al., 2020). Data augmentation (Park et al., 2019) is adopted in our training. During inference, we average the last 10 checkpoints. We test with beam 1 on a 10-core Processor. More details can be found in the Appendix.

4.2 Results

The results of our method on different model parameters are shown in Table 2. We find our method achieves real speed increases and FLOPs descend on both base and small models. Especially on the small model which aims to deploy on lightweight devices, our method reduces 39.34% FLOPs computation compared with the small model, which is very effective. Differing from directly applying a small vocab that suffers a significant degradation, our ADD method obtains a comparable performance. Specifically, our W2U method causes a slight loss with 0.15 WER on average compared with small models.

Diverse vocabulary clustering reveals a similar trend, which is that from random to phone-based method, then to unit-based method. This proves that clustering is more acceptable to models.

5 Analysis

We compare the classification accuracy of several clustering methods. We use the model with different clustering methods to predict the vocab id of each token. All the models are trained on clean 100h and use the best checkpoint to evaluate. The results are shown in Table 3. The results of "Random" and "Frequency" show that the models fail to classify the vocab accurately if there is no vocal feature as cluster guidance. For the number of vocabs N, we find that it is crucial for W2U. This is because the cluster center is randomly produced and N can easily affect the performance of cluster. However, the W2P is not sensitive and this proves that applying pronunciation as the cluster center is reasonable for ASR tasks.

We draw the word distribution of W2P and W2U methods in Figure 3. Besides the overlapped part, the unit part not only counts on prefix pronouncing but middle pronouncing as well. Thus the unit can also represent phonetic characteristics. This phenomenon explains why unit-based clustering behaves better and reveals a deeper comprehension of speech units from the perspective of a phone.

6 Related Work

Joulin et al., 2017 have proposed similar methods for splitting decoder output, yet they only considered the words frequency factor towards texts without inherent characteristics. Utilizing phonetic features to energize ASR is also mentioned by Qiu et al., 2023. Ji et al., 2022 find that all speech pre-trained models, which are trained by self-supervised learning, capture more articulatory features than conventional speech representation MFCC. Xu et al., 2023 validate the possibility of deploying Conformer on edge-computing devices.

7 Conclusion

In this work, we investigate the effect of mainstream accelerating methods on E2E ASR tasks and find that a large vocabulary still occupies a lot of computing time. We then adopt an intuitive method called ADD, which is to cluster vocabs depending on pronunciation to decrease the multiplication consumption. The result shows a speed boost with a little degradation in performance.

Limitations

There are some limitations we have to face. We only conduct our experiments on LibreSpeech dataset which is an English ASR dataset. Multiple language datasets may optimize our results. We did not apply more speedup methods in our experiments either. Furthermore, we spend much time discussing the improvement of the decoder part. However, regarding the long sequence in the encoder part, we do not pay much attention to that. This imbalance in length may cause an impact on inference speed.

References

Kunal Banerjee, Rishi Raj Gupta, Karthik Vyas, Biswajit Mishra, et al. 2020. Exploring alternatives to softmax function. *arXiv preprint arXiv:2011.11538*.

Maxime Burchi and Valentin Vielzeuf. 2021. Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Linhao Dong, Shuang Xu, and Bo Xu. 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)*, pages 5884–5888.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *ArXiv*, abs/2103.13630.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proc. Interspeech 2020*, pages 5036–5040.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. 2015. A neural transducer. *arXiv preprint arXiv:1511.04868*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Hang Ji, Tanvina Patel, and Odette Scharenborg. 2022. Predicting within and across language phoneme recognition performance of self-supervised learning speech pre-trained models. *arXiv preprint arXiv:2206.12489*.

Armand Joulin, Moustapha Cissé, David Grangier, Hervé Jégou, et al. 2017. Efficient softmax approximation for gpus. In *International conference on machine learning*, pages 1302–1310. PMLR.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Jinyu Li et al. 2022. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1).

Matthias R Mehl. 2006. Quantitative text analysis.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*.

Jin Qiu, Lu Huang, Boyu Li, Jun Zhang, Lu Lu, and Zejun Ma. 2023. Improving large-scale deep biasing with phoneme features and text-only data in streaming transducer. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE.

Yuan Shangguan, Rohit Prabhavalkar, Hang Su, Jay Mahadeokar, Yangyang Shi, Jiatong Zhou, Chunyang Wu, Duc Le, Ozlem Kalinli, Christian Fuegen, et al. 2021. Dissecting user-perceived latency of on-device e2e speech recognition. *arXiv preprint arXiv:2104.02207*.

Jacob R Stevens, Rangharajan Venkatesan, Steve Dai, Brucek Khailany, and Anand Raghunathan. 2021. Softermax: Hardware/software co-design of an efficient softmax for transformers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 469–474. IEEE.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 33–39, Suzhou, China. Association for Computational Linguistics.

Mingbin Xu, Alex Jin, Sicheng Wang, Mu Su, Tim Ng, Henry Mason, Shiyi Han, Yaqiao Deng, Zhen Huang, and Mahesh Krishnamoorthy. 2023. Conformer-based speech recognition on extreme edge-computing devices. *arXiv preprint arXiv:2312.10359*.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798.

Binbin Zhang, Di Wu, Zhendong Peng, Xingchen Song, Zhuoyuan Yao, Hang Lv, Lei Xie, Chao Yang, Fuping Pan, and Jianwei Niu. 2022. Wenet 2.0: More productive end-to-end speech recognition toolkit. *arXiv preprint arXiv:2203.15455*.

Yuhao Zhang, Chenghao Gao, Kaiqi Kou, Chen Xu, Tong Xiao, and Jingbo Zhu. 2023. Information magnitude based dynamic sub-sampling for speech-to-text. In *Proc. of Interspeech*.

Appendix

A Data Details

We conducted our experiments on LibriSpeech. LibriSpeech is an English ASR dataset as the training data. The training data is divided into three parts: 100 hours, 360 hours, and 500 hours. The dev and test data are split into 'clean' and 'other' categories based on speech clarity and complexity. We use all the training data to train models. The CMU Pronouncing Dictionary[1] is maintained by Carnegie Mellon University, to transfer words. This dictionary covers over 134,000 words and their pronunciations about North American English. In Table 2, we list models using beam size 1. Models using beam size 5 are listed in Table 4. Zhang et al. (2022) use beam size 10 in Wenet $w/$ LM.

Also, we list the decoder FLOPs calculating formula in Table 5 according to the work of Kaplan et al. (2020). For detailed parameters, we set $n_t = 25$, which denotes text length, and $d = 144$ in small models while $d = 256$ in base models, which denotes dimension of models, $d_{ff} = d * 8$, which denotes feedforward dimension, $n_a = 178$, which denotes phonetic features length, $n_{vocab} = 10000$, which denotes vocabulary size, $n_q = 10$, which denotes cluster numbers.

We trained the models for 100 epochs on eight NVIDIA 3090 GPUs. For the speed test, we restrict 10 cores of CPU to make a fair comparison.

About KD models, to make fair comparison, we choose the same base model as the teacher to help small models learn. For frequency clustering, we first count the most frequent phonetic prefix appearing in the corpus and array them up to down. Inside the same phonetic prefix, we sort them by frequency once again. For W2P clustering, we merge the same phonetic prefix together. To make clusters balance, we merge close clusters. What have to mention is we set a single cluster to contain the words without pronunciation. It is individually a special cluster.

B Approach Transfer

Transducer(Jaitly et al., 2015), which allows incremental output predictions as input data is received, is a popular method in ASR. Burchi and Vielzeuf, 2021 have proposed an efficient conformer, which contains a transducer decoder and achieves a SOTA result. We transfer our approach to this model. To make a fair comparison, we used a small version and trained it for 100 epochs with our methods. We choose a 1000-size vocab and a 10000-size to segment into 10 clusters separately since the original vocab of their result is 1000. We list the results in Table 6. All parameters and settings are provided in their link[2].

C Phonetic Distribution

For quick validation, we make it a probe task in Table 3. We just use 100 hours of training data of LibriSpeech to compare which cutoff method is relatively better. Besides, we randomly sample and count the centers of clusters and corresponding quantities in Table 7. With our entropy computing method, we can figure that more smoothing distribution of quantity means a higher entropy.

D HuBert Usage

We utilize HuBert(Hsu et al., 2021) to transfer words without pronunciation. As we mentioned in the main body, we train a HuBert model to predict the vocab id of each token. We transfer every token in the 100-hour text corpus of LibriSpeech to the id of the clusters it belongs to. All the tools[3] can be found. and checkpoints[4] can be found.

[1]The source can be obtained at http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[2]The code can be obtained at https://github.com/burchim/EfficientConformer.

[3]The tools can be obtained at https://github.com/facebookresearch/fairseq/tree/main/examples/hubert

[4]The checkpoints can be obtained at https://dl.fbaipublicfiles.com/hubert/hubert_base_ls960.pt

Method	Param. (M)	Test (WER) Clean	Other	Avg.	Speed (tokens/s)	FLOPs (M)	Rate of Acceleration
Wenet $w/$ LM (Zhang et al., 2022)	34.76	3.09	7.40	5.25	–	394.97	–
Base	45.42	3.11	7.34	5.23	60.00	426.97	100.00%
Base+Random	47.98	3.27(↓ 0.16)	8.23(↓ 0.89)	5.75	65.00	369.37	108.33%
Base+W2P		3.52(↓ 0.41)	7.76(↓ 0.42)	5.64			
Base+W2U		2.96(↑ 0.15)	7.76(↓ 0.42)	5.36			
Small*	17.52	3.59	8.57	6.08	103.67	91.82	172.77%
Small*+Random	18.92	3.66(↓ 0.07)	9.08(↓ 0.51)	6.37	118.33	25.67	197.22%
Small*+W2P		3.62(↓ 0.03)	8.86(↓ 0.29)	6.24			
Small*+W2U		3.70(↓ 0.11)	8.74(↓ 0.17)	6.22			

Table 4: The clustering effect on real performance. *small** denotes the small model with KD and AAN methods.

Operation	Base	Small	Small(w.attn)	Small(w.attn.our)
Self-Attention: Q K V	$3 * n_t * d * d$	$3 * n_t * d * d$	$n_t * d * d$	$n_t * d * d$
Self-Attention: Weight	$n_t * n_t * d$	$n_t * n_t * d$		
Self-Attention: Attn-Mul	$n_t * n_t * d$	$n_t * n_t * d$	$n_t * d$	$n_t * d$
Self-Attention: Project	$n_t * d * d$	$n_t * d * d$	$n_t * d * d$	$n_t * d * d$
Cross-Attention: QKV	$n_t * d * d + 2 * n_a * d * d$	$n_t * d * d + 2 * n_a * d * d$	$n_t * d * d + 2 * n_a * d * d$	$n_t * d * d + 2 * n_a * d * d$
Cross-Attention: Weight	$n_t * n_a * d$	$n_t * n_a * d$	$n_t * n_a * d$	$n_t * n_a * d$
Cross-Attention: Attn-Mul	$n_t * n_a * d$	$n_t * n_a * d$	$n_t * n_a * d$	$n_t * n_a * d$
Cross-Attention: Project	$n_t * d * d$	$n_t * d * d$	$n_t * d * d$	$n_t * d * d$
Feedforward	$2 * n_t * d * d_{ff}$	$n * 2 * d * d_{ff}$	$n * 2 * d * d_{ff}$	$n * 2 * d * d_{ff}$
Output	$n_t * d * n_{vocab}$	$n_t * d * n_{vocab}$	$n_t * d * n_{vocab}$	$n_t * d * n_{vocab}/n_q$
All	426973696	95460048	91820448	256739040
Rate of detracted Flops	–	0.0%	3.9%	41.7%

Table 5: Acceleration and performance comparison

Model	Beam	Dev (WER) Clean	Other	Test (WER) Clean	Other	FLOPs (M)
Transducer Small (Burchi and Vielzeuf, 2021)	1	4.11	10.76	4.35	10.63	101.53
	5	3.98	10.37	4.21	10.05	
Transducer Small+ADD	1	4.32	11.16	4.58	10.91	75.90
	5	4.11	10.69	4.34	10.42	

Table 6: The performance of Conformer + Transducer using our method.

Center	N	Quantity	Entropy
built, scientific, affection, resolution, philammon	5	6145, 313, 893, 480, 2169	-10.86
plate, prescribe, johnson, adverse, occasion	5	1708, 2513, 2278, 1513, 1988	-8.11
sprawl, r, terrestrial, though, clavering ue, subscription, adverse, penrod, marchioness	10	2188, 1411, 659, 243, 1167, 0, 340, 396, 2703, 893	-29.16
phil, vulgar, od, dense, tempest, changes, mum, anger, signor, dealing	10	1045, 1242, 1163, 924, 924, 416, 898, 901, 1891, 596	-23.63

Table 7: The distribution and cluster center of W2U