



An Effective Training Framework for Light-Weight Automatic Speech Recognition Models

Abdul Hannan¹, Alessio Brutti², Shah Nawaz³, Mubashir Noman⁴

¹University of Trento, Italy

²Fondazione Bruno Kessler, Italy

³Johannes Kepler University Linz, Austria

⁴Mohamed bin Zayed University of Artificial Intelligence, U.A.E.

Abstract

Recent advancement in deep learning encouraged developing large automatic speech recognition (ASR) models that achieve promising results while ignoring computational and memory constraints. However, deploying such models on low resource devices is impractical despite of their favorable performance. Existing approaches (pruning, distillation, layer skip etc.) transform the large models into smaller ones at the cost of significant performance degradation or require prolonged training of smaller models for better performance. To address these issues, we introduce an efficacious two-step representation learning based approach capable of producing several small sized models from a single large model ensuring considerably better performance in limited number of epochs. Comprehensive experimentation on ASR benchmarks reveals the efficacy of our approach¹, achieving three-fold training speed-up and up to 12.54% word error rate improvement.

Index Terms: Automatic Speech Recognition, Efficient Training, Feature Representation Learning, Light-weight Model

1. Introduction

Accelerated progress in Internet of Things (IoT) [1] and edge-cloud continuum [2] demands significant reduction in the neural architecture sizes for smooth operation with minimal computational and memory burden. Several efforts have been directed to create user and device-personalized models with adaptable model sizes while minimizing the latency using different techniques. Contemporary works aims at sparsification of deep models using pruning [3, 4] that primarily locate the intermediate representation (or input) that if removed, would minimally affect the model's output. Others either employ low-precision data and models [5] or opt for low-rank approximation methods [6, 7], or distill the knowledge from a large teacher model to a student model using a distribution's divergence-based loss such as Kullback–Leibler divergence, Jensen-shannon divergence, Wasserstein distance, etc [8, 9, 10, 11, 12, 13, 14, 15]. However, applying pruning solely leads to substantial performance degradation, directing the researchers to use pruning in conjunction with other approaches [4, 16, 17] to reduce the affect on performance up to some extent. Alternatively, few works utilized conditional computing approaches, such as Early Exit [18], Layer dropping [19, 20] etc., to reduce the computational load at inference time either by exiting at early stage or skipping some parts of the network. This dynamic behavior comes at the cost of computational flow alteration and addition of new modules, raising the computational and memory requirements.

In addition, dynamic dropping of layers leads to considerable reduction in ASR performance.

Curating resource-friendly architectures with the ideal performance is the underlying objective of machine learning, and is featuring new paradigms like tiny machine learning (tiny ML), green ML etc. Developing large models with billions of parameters to gain a point over counterparts might not be the right way to improve task-specific performance. Not only these architecture results in tons of carbon dioxide emission and consume a staggering amount of energy until convergence, the level of resource consumption of such large models does not align with these paradigms as well as their deployment on end-user devices is practically unviable. Similarly, compressing large models being undoubtedly effective, suffers from the drawback of inefficient resource utilization in addition to noticeable performance degradation. Moreover, training small and device-conditioned models every time for substantial number of epochs is also unfeasible. The reason is that small models demands prolong training time and large number of epochs to converge with subpar performance level. Therefore, it is desired to build smaller architectures capable of achieving favorable performance as well as requiring reduced computational and memory load.

In this regard, we propose an effective approach that exploits the knowledge of large reference model to obtain tiny models capable of performing favorably while requiring low computational resources. To summarize, our contributions are as below.

- We propose a two-step based approach to train small ASR models that achieve promising results on ASR task.
- The introduced approach enables us to train several small and tiny models requiring less compute resources while providing promising performance compared to equivalent small models.
- We perform extensive experiments on two challenging ASR benchmarks demonstrating the effectiveness of proposed approach.

2. Background

Representation Learning using Mask Auto-Encoders:

Mask auto-encoders [21, 22] are self-supervised frameworks that are utilized to learn rich semantic features from large amount of unlabeled data. Mask auto-encoder masks the large portion of its input $X \in \mathbb{R}^{H \times W}$, and feeds the unmasked patches to the encoder to obtain encoded features X_e . In decoder, learnable features are appended to the encoded features X_e which are forwarded to the decoder layers to obtain the reconstructed feature representations X_r . Mask auto-encoder utilizes the mean squared error loss (MSE) to measure the recon-

¹Our code is available at: <https://github.com/hannabdul/etf4asr>

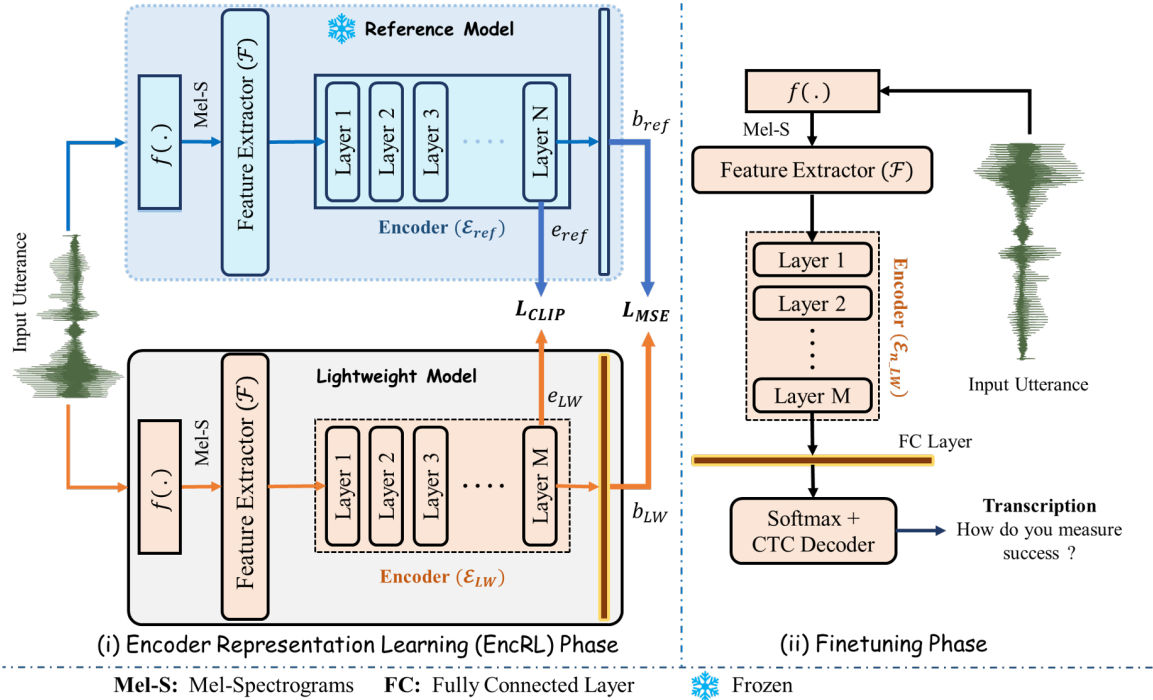


Figure 1: Here we illustrate the overall framework of the proposed method. (i) In the feature learning phase, the input utterance is fed to a large ASR model (reference model) to extract the features e_{ref} and b_{ref} . Afterwards, the same input is passed through the smaller model to obtain features e_{LW} and b_{LW} . To learn the knowledge of reference model, MSE loss is used between the outputs (b_{ref} and b_{LW}) of the classifiers of the reference and the small model. Symmetric cross-entropy loss is used on the features e_{ref} and e_{LW} to align the feature spaces. (ii) After transferring the knowledge to the encoder of light-weight model, CTC decoder is integrated and the model is finetuned to transcribe the input speech.

struction performance and enable the model to learn meaningful feature representations in self-supervised manner.

Knowledge Distillation: Knowledge distillation is a technique utilized to transfer the knowledge of a large or teacher model to a student model. Typically, the teacher model is trained on large set of available data and the knowledge is transferred to the student model by means of distillation loss. Existing works [23, 24] have utilized the cross-entropy loss having soft target distributions to distill the knowledge given as:

$$c_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \quad (1)$$

where c_i is the class probability of i th class, z_i is the logit and T is the temperature to control the distillation. This teacher-student architecture is trained in an end-to-end fashion. Given that we have several student networks, it is desired to train each student model using the knowledge distillation framework enabling the student model to perform well on the specific task.

In addition to the temperature controlled cross-entropy loss, MSE loss is also utilized to distill knowledge for classification task [25]. However, MSE is typically used in mask auto-encoder frameworks to learn meaningful representations from huge amount of unlabelled data. Inspired from these, we propose a two-step approach that first learns the semantically rich features from the encoder of large reference model followed by finetuning tiny models for limited number of epochs.

3. Proposed Framework

To mitigate the performance limitations and facilitate fast training, we propose a two-step feature learning based framework in Figure 1. The proposed framework requires: (i) *One time feature representation learning* to mimic the enriched representation of the large Reference model, (ii) *Finetuning* for limited number of epochs, outperforming the similar sized models trained from scratch for greater number of epochs.

Given a dataset $\mathcal{D} = \{(a_d^i, t_d^i)\}_{i=1}^K$ where a_d^i and t_d^i are the corresponding audio-transcription pair of the i^{th} instance with K total instances, we train a reference model \mathcal{M}_{ref} consisting of N encoder blocks for Z number of epochs. This trained model provides the *baseline results* as well as serves as the reference model for representation learning phase.

3.1. Encoder's Representation Learning (EncRL)

In this phase, our objective is to enable the light-weight model \mathcal{M}_{LW} to learn the information from the deep reference model \mathcal{M}_{ref} that \mathcal{M}_{ref} has learned by training on a large amount of data for a greater number of epochs Z . To this end, reference model \mathcal{M}_{ref} and light-weight model \mathcal{M}_{LW} are created having N and M number of layers, respectively, where $M \leq \frac{N}{2}$. Subsequently, we take the output features e_{LW} and e_{ref} of the last layer of the corresponding encoders and utilize the symmetric cross-entropy \mathcal{L}_{CLIP} loss to align the feature representations as shown in Figure 1. The \mathcal{L}_{CLIP} loss minimizes the distance between the feature representations (e_{LW} and e_{ref}) of the same samples while maximizing for the different samples. Similarly,

Table 1: WER evaluated on test-clean and test-other splits of LibriSpeech dataset. First row indicates the large ASR model trained for n number of epochs. * represents that the smaller model is trained for n number of epochs similar to large model. † represents the results of a Conformer model from [18] with twice number of attention heads and doubled feed-forward dimension in conformer submodules. × represents not reported.

Model	Epochs	# of Encoder Layers	Params (M)	WER (%)	
				clean	other
Conformer (ref)	150	12	18.60	5.60	14.38
Conformer†	×		31.601	6.5	17.7
Conformer*	150	6	9.462	8.61	20.97
Conformer†	×		15.963	7.6	20.0
Conformer (ours)	50		9.462	6.27	17.47
Conformer*	150	4	6.416	8.91	22.18
Conformer†	×		10.750	9.8	24.3
Conformer (ours)	50		6.416	7.95	21.82
Conformer*	150	2	3.370	15.57	32.57
Conformer†	×		5.537	17.6	36.1
Conformer (ours)	50		3.370	14.68	33.70

to enforce the light-weight model \mathcal{M}_{LW} to produce the similar output as the reference model \mathcal{M}_{ref} generates, we utilize mean squared error (\mathcal{L}_{MSE}) loss on the feature embeddings b_{LW} and b_{ref} of the two models. This loss combination will encourage the light-weight model \mathcal{M}_{LW} to learn rich semantic information from the model \mathcal{M}_{ref} . The total loss is given as:

$$\mathcal{L}_{EncRL} = \mathcal{L}_{CLIP} + \mathcal{L}_{MSE} \quad (2)$$

$$\mathcal{L}_{CLIP} = \max_{i \neq j} \left(\min_{i=j} \left(\sum_B e_{ref}^i \odot e_{LW}^j \right) \right) \quad (3)$$

$$\mathcal{L}_{MSE} = \frac{1}{B} \sum_i (b_{ref}^i - b_{LW}^i)^2 \quad (4)$$

where \odot represents element-wise multiplication.

During this phase, the reference model \mathcal{M}_{ref} is kept frozen and the transcription decoder is not utilized. We perform one-time feature representation learning of the light-weight model \mathcal{M}_{LW} by training for $\frac{2 \times Z}{3}$ epochs, enabling it to learn the enriched representation of \mathcal{M}_{ref} . Afterwards, we finetune the model \mathcal{M}_{LW} for a few epochs as explained in the next section.

3.2. Finetuning phase

After the feature representation learning phase, we integrate the Connectionist Temporal Classification (CTC) decoder in the light-weight model \mathcal{M}_{LW} . Then, we initialize the model \mathcal{M}_{LW} with the weights of the representation learning phase. We feed the input audio utterances to the model \mathcal{M}_{LW} and generate the output transcriptions. We finetune the model for $Z/3$ number of epochs. During finetuning, we utilize the connectionist temporal classification loss to measure the performance of the model. Similar to the model \mathcal{M}_{LW} , we further finetune the tiny models $\mathcal{M}_{n,LW}$ having number of layers less than M . Specifically, we finetune the model $\mathcal{M}_{n,LW}$ having encoder layers n where $n \in [2, 4, \dots, M]$. This enables the smaller models to perform favorably without requiring training for a large

Table 2: WER evaluated on TEDLIUM-v3 dataset (test split)

Model	# of Encoder Layers	Params (M)	WER (%)
Conformer (ref)	12	18.60	15.28
Conformer†		31.601	16.4
Conformer*	6	9.462	21.82
Conformer†		15.963	25.5
Conformer (ours)		9.462	19.72
Conformer*	4	6.416	49.30
Conformer†		10.750	35.4
Conformer (ours)		6.416	22.86

number of epochs.

For instance, given a mini-batch B in the dataset \mathcal{D} , the model processes the audio inputs a^i producing the corresponding embedding b^i , followed by a *Softmax* layer that converts the raw logits into the respective token probabilities p^i as illustrated in Figure 1. The tokens are forwarded to a built-in CTC decoder transforming the incoming data into the final transcription t_{pred}^i . Finally, the model’s transcription is optimized by quantification of CTC loss by comparing against the ground-truth transcription.

$$t_{pred}^i = CTC_Decoder (Softmax(b^i)) \quad (5)$$

$$\mathcal{L}_{ctc} = f_{CTC}(t^i - t_{pred}^i)$$

4. Experiments and Results

Implementation Details: The model is trained on NVIDIA A40 GPU with a batch size of 64 using AdamW optimizer with $L2$ normalization of $1e^{-6}$ and 10000 warmup iterations followed by exponentially decreasing the learning rate till the end of training. For each audio input, mel-spectrograms (using 80 mel-filterbanks) are computed for 20ms window length with a hop-length of 10ms along with SentencePiece byte-pair-encoding (BPE-256) tokenizer [26]. To suppress the over-fitting issue, we applied *SpecAug* augmentation [27] with frequency and time masking of 27 and 80 samples.

Architecture: We adapted open-sourced Conformer (small) architecture with feed-forward dimension of 256 and 12 encoder layers for the reference model. Consequently, the number of encoder layers M for light-weight model are 6. Furthermore, we replaced the 3-layered LSTM based decoder with a linear projection layer, whose output is forwarded to the CTC decoder for transcription.

Datasets: We evaluated our method on two challenging and publicly available ASR datasets: **LibriSpeech** [28] comprising of ≈ 1000 hours of read-aloud audiobooks and **TED-LIUM** (release-3) [29] consisting of ≈ 452 hours of transcribed English TED talks. We employed Word Error Rate (WER) metric to quantify the transcription errors.

4.1. Results

Comparison with small models: We compare the proposed framework with existing light-weight models, as well as smaller models that are trained for Z number of epochs. Table 1 reveals that our approach achieves superior performance as compared to the other smaller models for various model sizes (different number of encoder layers). For instance, our method ob-

Table 3: WER (%) comparison with conditional computation strategies on LibriSpeech test-clean split.

# of Encoder Layers	WER (%) _{Params(M)}					
	Ours		Early Exit [18]		Random Dropping [19]	
6	6.27	9.46	6.8	15.96	7.98	15.96
4	7.95	6.41	11.6	10.75	12.54	10.75
2	14.68	3.37	23.9	5.53	43.62	5.53

tains WER of 6.27% and 14.68% for the model having 6 and 2 encoder layers on LibriSpeech test-clean split, whereas the smaller models that are trained for greater epochs achieve WER of 8.61% and 15.67% respectively. Similarly, our approach achieves reasonable improvement in WER score of 3.5% and 0.36% for model having 6 and 4 encoder layers on LibriSpeech test-other split. Likewise, on Tedlium (v3) test split, our approach obtains absolute improvement of 2.1% and 26.44% in WER metric as compared to model trained from scratch.

In addition, to train W models of various sizes, our method requires $(W \times Z)/3$ training epochs compared to the other smaller models that require $W \times Z$ epochs, achieving $3\times$ faster training and better performance. In short, our approach offers flexibility to create several light-weight and small-sized models requiring considerably less training time and improved performance.

Comparison with Dynamic models: To further validate our framework’s efficiency, we compare the light-weight models finetuned with our approach against conditional computing approaches such as Early Exit (EE) [18] and Random Layer Dropping (RD) [19]. Table 3 depicts the performance superiority of our framework, surpassing EE and RD approaches for various model sizes. It is worth mentioning that EE and RD enables a single model to dynamically adjust the model’s size, hence, suitable for storage-constrained environments while compromising on the final accuracy. In contrast, our framework produces multiple light-weight models suitable for different resource settings with considerably better performance as compared to the counterparts.

Why EncRL is required? We performed the finetuning of smaller model having 6 encoder layers while initializing the weights from the final states of \mathcal{M}_{ref} . We observe that despite of intializing the weights from \mathcal{M}_{ref} model, the light-weight model struggles to provide better performance and achieves a WER of 7.41%. Whereas our approach provides better performance by achieving WER of 6.27%.

Vitality of Finetuning: To investigate the requirement of finetuning, we integrated the CTC decoder to the light-weight model obtained after representation learning. We then evaluated the model on LibriSpeech test-clean split. It was observed that the model failed to recognize the audio signal resulting in extremely high WER of ($\approx 95\%$) highlighting the need of finetuning phase.

4.2. Ablations

4.2.1. Quantifying the efficiency of Loss functions

We employed various loss functions during the representation learning phase including CLIP, MAE, MSE, and their combinations. Table 4 enlists the obtained WER for each loss function, highlighting that combination of CLIP and MSE loss pro-

Table 4: Evaluating the effect of different Loss functions used during Encoder’s Representation Learning phase. # of encoder layers used in these experiments is 6.

Loss Function	WER (%)
CLIP only	9.06
MAE only	6.40
MSE only	6.36
CLIP + MAE	6.42
CLIP + MSE	6.27

vide the rich feature representations which enable the model to obtain best performance after finetuning phase achieving WER score of 6.27%.

4.2.2. Do we need separate EncRL for various sizes of model?

To answer this question, we performed *EncRL* for a model having 4 encoder layers followed by finetuning for $Z/3$ epochs, and compared it with the finetuned model that is initialized with the last 4 layers of encoder’s state dictionary of *EncRL* with 6 layers. We observed slight improvement of 0.53% in WER when we separately perform encoder representation learning for 4 encoder layers demonstrating that the repeating encoder representation learning step does not provide significant improvement in performance but consumes twice as much computational cost.

4.2.3. Pruning

We applied pruning technique on the reference model to compare its performance against the proposed framework on LibriSpeech test-clean split. To prune the model, we employed two different schemes: (i) pruning conformer module completely by a given amount, (ii) pruning other conformer sub-modules except the convolutional sub-module. For the *former* case, we observed that pruning only 5% of weights significantly degrades the performance, scoring 13.64% WER. The performance reduces further if we increase the pruning amount by 2.5%, and totally collapses at pruning amount of 10% giving a WER of 49.44%. For the *latter* case, the model shows WER of 10.51% for 7.5% pruned weights which is 13.16% less than we achieved in the former pruning method. Similarly, the WER substantially increases by 11.69% and 23% for pruning amount of 15% and 22.5% highlighting the limitation of pruning to maintain good performance. However, our framework significantly outperforms the pruning schemes, providing a mere WER of 6.27%, 7.95% and 14.68% for equivalent pruning amount of 50%, 66.7%, and 83.33%, highlighting the efficacy of the framework.

5. Conclusion

In this work, we proposed a two-step framework to create ultra light-weight models in a resource efficient manner without significantly compromising on the final performance. We demonstrated that a reference model can be employed to train a general light-weight encoder-only model that serves as a starting point for multiple light-weight networks. This is accomplished by transferring the knowledge of the reference model to the light-weight model through feature alignment and adaptation. Later, the model with desired encoder depth is instantiated and initialized with the final state of the representation learning, and finetuned for a handful of epochs on the target dataset. Extensive experimentation on LibriSpeech and Tedlium datasets reveals the efficiency of the proposed framework.

6. References

- [1] B. Farahani, S. Tabibian, and H. Ebrahimi, "Towards a personalized clustered federated learning: A speech recognition case study," *IEEE Internet of Things Journal*, 2023.
- [2] K. Liang *et al.*, " $WH^2D^2N^2$: Distributed ai-enabled ok-asn service for web of things," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 5, pp. 1–16, 2023.
- [3] S. Kim *et al.*, "Learned token pruning for transformers," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2022, p. 784–794.
- [4] Z. Wei, L. Hao, and X. Zhang, "Model compression by iterative pruning with knowledge distillation and its application to speech enhancement." in *INTERSPEECH*, 2022, pp. 941–945.
- [5] H. D. Nguyen, A. Alexandridis, and A. Mouchtaris, "Quantization aware training with absolute-cosine regularization for automatic speech recognition," in *Interspeech 2020*, 2020, pp. 3366–3370.
- [6] G. I. Winata, S. Cahyawijaya, Z. Lin, Z. Liu *et al.*, "Lightweight and efficient end-to-end speech recognition using low-rank transformer," in *ICASSP*, 2020, pp. 6144–6148.
- [7] P. Wang and H. Van hamme, "Bottleneck low-rank transformers for low-resource spoken language understanding," 2022.
- [8] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *CoRR*, vol. abs/1503.02531, 2015. [Online]. Available: <http://arxiv.org/abs/1503.02531>
- [9] S. Watanabe, T. Hori, J. Le Roux, and J. R. Hershey, "Student-teacher network learning with enhanced features," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5275–5279.
- [10] T. Fukuda, M. Suzuki, G. Kurata, S. Thomas, J. Cui, and B. Ramabhadran, "Efficient knowledge distillation from an ensemble of teachers." in *Interspeech*, 2017, pp. 3697–3701.
- [11] R. Takashima, L. Sheng, and H. Kawai, "Investigation of sequence-level knowledge distillation methods for ctc acoustic models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6156–6160.
- [12] G. Kurata and G. Saon, "Knowledge distillation from offline to streaming rnn transducer for end-to-end speech recognition." in *Interspeech*, 2020, pp. 2117–2121.
- [13] J. W. Yoon, H. Lee, H. Y. Kim, W. I. Cho, and N. S. Kim, "Tutonet: Towards flexible knowledge distillation for end-to-end speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1626–1638, 2021.
- [14] K. Zhao, H. D. Nguyen, A. Jain, N. Susanj, A. Mouchtaris, L. Gupta, and M. Zhao, "Knowledge distillation via module replacing for automatic speech recognition with recurrent neural network transducer," in *23rd Interspeech Conference*, 2022.
- [15] J. W. Yoon, H. Lee, J. Y. Kang, and N. S. Kim, "Cons-kd: Dropout-robust knowledge distillation for ctc-based automatic speech recognition," *IEEE Access*, 2024.
- [16] K. Zhen, H. D. Nguyen, F.-J. Chang, A. Mouchtaris, and A. Rastrow, "Sparsification via compressed sensing for automatic speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6009–6013.
- [17] Q. Xiao, P. Ma, A. Fernandez-Lopez, B. Wu, L. Yin, S. Petridis, M. Pechenizkiy, M. Pantic, D. C. Mocanu, and S. Liu, "Dynamic data pruning for automatic speech recognition," in *Interspeech 2024*, 2024, pp. 4488–4492.
- [18] G. A. Wright, U. Cappellazzo, S. Zaiem, D. Raj, L. O. Yang, D. Falavigna, and A. Brutti, "Training dynamic models using early exits for automatic speech recognition on resource-constrained devices," *CoRR*, vol. abs/2309.09546, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2309.09546>
- [19] A. Hannan, A. Brutti, and D. Falavigna, "LDASR: An experimental study on layer drop using conformer-based architecture," in *Proc. of EUSIPCO*, 2024.
- [20] S. Zaiem *et al.*, "Fine-tuning strategies for faster inference using speech self-supervised models: a comparative study," in *ICAS-SPW*. IEEE, 2023, pp. 1–5.
- [21] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 16 000–16 009.
- [22] M. Noman, M. Naseer, H. Cholakkal, R. M. Anwer, S. Khan, and F. S. Khan, "Rethinking transformers pre-training for multi-spectral satellite imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 27 811–27 819.
- [23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: <https://arxiv.org/abs/1503.02531>
- [24] S. H. Krishnan Parthasarathi and N. Strom, "Lessons from building acoustic models with a million hours of speech," in *ICASSP*, 2019, pp. 6670–6674.
- [25] T. Kim, J. Oh, N. Y. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 2628–2635, main Track. [Online]. Available: <https://doi.org/10.24963/ijcai.2021/362>
- [26] T. Kudo and J. Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *EMNLP*, Brussels, Belgium, Nov. 2018, pp. 66–71.
- [27] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019*, 2019, pp. 2613–2617.
- [28] Panayotov *et al.*, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP*, 2015, pp. 5206–5210.
- [29] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko *et al.*, "TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *Speech and Computer: SPECOM 2018, Leipzig, Germany*, 2018, pp. 198–208.