2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW) | 979-8-3503-7451-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSPW62465.2024.10627008

TRAINING EARLY-EXIT ARCHITECTURES FOR AUTOMATIC SPEECH RECOGNITION: FINE-TUNING PRE-TRAINED MODELS OR TRAINING FROM SCRATCH

George August Wright[1], *Umberto Cappellazzo*[1], *Salah Zaiem*[2], *Desh Raj*[3], *Lucas Ondel Yang*[4], *Daniele Falavigna*[5], *Mohamed Nabih Ali*[5], *Alessio Brutti*[5]

[1]University of Trento; [2]LTCI, Télécom Paris, Institut Polytechnique de Paris; [3]Johns Hopkins University; [4]Universite Paris-Saclay, LISN, CNRS; [5]Fondazione Bruno Kessler

ABSTRACT

The ability to dynamically adjust the computational load of neural models during inference is crucial for on-device processing scenarios characterised by limited and time-varying computational resources. A promising solution is presented by early-exit architectures, in which additional exit branches are appended to intermediate layers of the encoder. In self-attention models for automatic speech recognition (ASR), early-exit architectures enable the development of dynamic models capable of adapting their size and architecture to varying levels of computational resources and ASR performance demands. Previous research on early-exiting ASR models has relied on pre-trained self-supervised models, fine-tuned with an early-exit loss. In this paper, we undertake an experimental comparison between fine-tuning pre-trained backbones and training models from scratch with the early-exiting objective. Experiments conducted on public datasets reveal that early-exit models trained from scratch not only preserve performance when using fewer encoder layers but also exhibit enhanced task accuracy compared to single-exit or pre-trained models. Furthermore, we explore an exit selection strategy grounded in posterior probabilities as an alternative to the conventional frame-based entropy approach. Results provide insights into the training dynamics of early-exit architectures for ASR models, particularly the efficacy of training strategies and exit selection methods.

Index Terms— SSL, fine-tuning, early-exit, ASR

1. INTRODUCTION

The edge-cloud continuum is an emerging complex ecosystem that integrates compute-enabled edge devices, distributing the overall computation workload among them [1]. The availability of computational resources differs considerably across devices, and varies over time due to resource sharing between multiple services. Therefore, it is crucial to have neural models that can dynamically change their trade-off between computation and performance. To this end, we investigate the use of early-exit architectures applied to large-vocabulary automatic speech recognition (ASR).

Previous work for on-device neural processing focused on reducing model size through compression [2], knowledge distillation [3, 4], pruning [5], and quantization [6]. Although effective, these approaches provide static solutions which require handling multiple models with varying trade-offs. A preferable approach is to dynamically adapt a single model architecture to the memory and computational capabilities of each hosting device. A solution is represented by "early-exit" architectures that introduce intermediate exit branches [7, 8]. The input is not processed by all layers of the neural network but only a subset of them, returning the result at an intermediate layer and bypassing operations in the layers not traversed. Figure 1 displays examples of early-exiting networks, where layer-specific classifiers/decoders (the "exit layers") are appended to intermediate encoder layers. Early-exit architectures leverage the observation that, for simpler inputs, the lower layers of the model may have already learned sufficient parameters for accurate predictions. Early-exit architectures enable *resource-aware* processing (Fig. 1, left), where a single model can be used on heterogeneous devices, as well as *result-aware* processing (Fig. 1, right), where the model selects the earliest exit that achieves the same performance as the entire network.

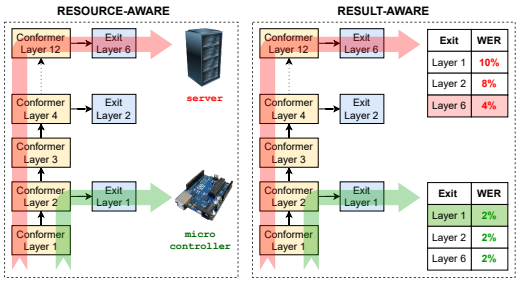

Fig. 1: Examples of *resource-aware* and *result-aware* uses of early-exits. Left: the micro-controller can afford only two layers, whereas the server can handle the entire model. Right: the first input requires processing with the entire network; in the second case, the best transcription is produced after only two encoder layers.

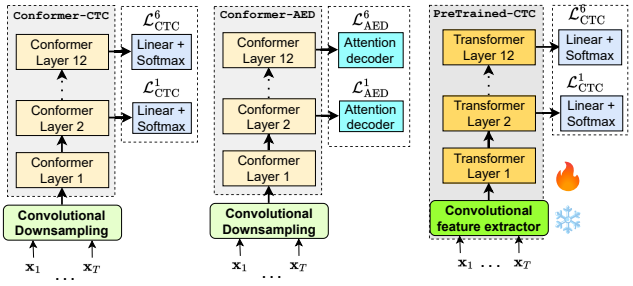

Fig. 2: Early-exit architectures (from left to right): Conformer-CTC, Conformer-AED, and pre-trained-CTC. Conformer-based models are trained from scratch. pre-trained-CTC are initialised from pre-trained models (Wav2Vec2.0 or WavLM) and fine-tuned with early-exit CTC losses, freezing the convolutional feature extractor.

In this work, we investigate the use of early-exit self-attention

Table 1: Hyperparameters for the early-exit model architectures shown in Fig. 2.

Feature	Conformer CTC	Conformer AED	Wav2Vec2 CTC	WavLM (Base+) CTC
# params (M)	31.0	13.3	94.0	94.7
Encoder	12-layer Conf.	12-layer Conf.	12-layer Transf.	12-layer Transf.
Attention dim.	256	144	768	768
Number heads	8	4	8	8
Feed-forward dim.	2048	1024	3072	3072
Decoder	Linear	4-layer Transf.	Linear	Linear
Inputs	80-d MFCC	80-d MFCC	Waveform *	Waveform *
Loss function	$\mathcal{L}_{\text{CTC}}$	$\mathcal{L}_{\text{CE}}$ & $\mathcal{L}_{\text{CTC}}$	$\mathcal{L}_{\text{CTC}}$	$\mathcal{L}_{\text{CTC}}$
Output units	BPE	BPE	Grapheme	Grapheme
LM rescoring	✗	✓	✗	✗
Data augmentation	✗	✓	✓	✓

architectures applied to ASR models. Following common practice in ASR, previous work mainly focuses on accelerating inference using pre-trained backbones fine-tuned with early-exit objectives [9, 10, 11]. In this paper we additionally investigate training models from scratch using different early-exit losses as depicted in Fig. 2. We demonstrate that early-exit models, optimised on the joint loss of all early-exit layers, outperform "single-exit" models optimised only on the final layer output. Moreover, our experiments suggests that training from scratch native early-exit architectures is to some extent more effective than fine-tuning pre-trained self-supervised models. Our overall contributions are:

1. We investigate early-exit training strategies, both training from scratch and fine-tuning pre-trained self-supervised models, considering 4 different architectures;
2. We evaluate a confidence-based exit selection based on N-best posteriors that offers a slightly better trade-off than entropy.

2. RELATED WORK

Early-exiting methods were first introduced for computer vision in BranchyNet [7] by adding two branches to AlexNet [12]. The authors optimised the joint loss of the exits and defined a confidence measure, based on the entropy of the output class distribution, to decide the exit level. More recently, Scardapane et al. [13] have provided a theoretical framework for multi-exit neural architectures. Early-exit classifiers have also been implemented in tiny (KB-sized) models [14]. Besides early-exiting, other methods for dynamically adapting the model architecture for efficient inference have been explored, such as HydraNet [15].

In speech recognition, early-exiting was first introduced in HuBERT-EE [9] to accelerate inference for a pre-trained HuBERT [16] model, using confidence measures based on CTC confidence or output entropies, with no significant performance degradation. Similarly, Zaiem et al. [11] have investigated early-exit fine-tuning strategies in the context of a large pre-trained WavLM [17] model, comparing them with approaches based on layer removal and input down-sampling.The issue of overthinking in ASR encoders has been analysed in [10], in which the authors report theoretical lower bounds of speed/quality trade-offs for early-exit strategies. Exit-selection strategies were proposed based on comparisons of output distribution and transcriptions between successive exits. Similar investigations using the entropy of the output distribution have also been conducted for recurrent neural networks [18].

All aforementioned studies [9, 10, 11] employ pre-trained models by fine-tuning the Transformer component, as is common for ASR. They are primarily focused on *improving inference efficiency* by selecting the best early exit according to some criteria. Analogously, in natural language processing, research on early-exit strategies has focused on accelerating the inference of large pre-trained language models such as BERT [19, 20, 21]. Conversely, in this work, our objective is to *understand the training dynamics* of early-exit models (both trained from scratch and initialised from pre-trained models) by conducting exhaustive experiments on multiple datasets. We demonstrate that training the model from scratch, jointly optimizing all exited layers of the model, provides significant performance improvements over conventional single-exit models and fine-tuned pre-trained models (particularly at the lowest exits).

3. EARLY-EXIT MODELS FOR ASR

Given an input sequence $\mathbf{X}$, such as a raw waveform $\{x_1, \ldots, x_N\}$ or acoustic features $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^d$, an ASR system estimates the output sequence $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{X}), \tag{1}$$

where $\mathbf{y} \in \mathcal{Y}^*$, for some vocabulary $\mathcal{Y}$, such as graphemes, phonemes, or byte pair encoding (BPE) units. The distribution $P(\mathbf{y}|\mathbf{X})$ is usually estimated using a parameterised model Θ (such as a neural network), i.e., $P(\mathbf{y}|\mathbf{X};\Theta)$, which is learned using input-output pairs ($\mathbf{X}$,$\mathbf{y}$).

For convenience, Θ is often factored into an encoder, which extracts high-dimensional representations $\mathbf{h}_1^T$ from $\mathbf{X}$, and a decoder, which maps $\mathbf{h}_1^T$ to the output sequence $\mathbf{y}_1^U$. Since $U \ll T$ in general, ASR decoders either use (i) an *alignment function* ($\mathcal{B} : \mathbf{a}_1^T \rightarrow \mathbf{y}_1^U$) for sequence training, or (ii) an *attention mechanism* with label-based cross-entropy training. We apply early-exiting to ASR by adding decoders at several intermediate layers of the encoder (as shown in Fig. 2). Assuming that M such intermediate exits are added (with hypothesis $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^M$), the overall model is trained by optimizing the joint objective

$$\mathcal{L}_{EE}(\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^M, \mathbf{y}) = \sum_{m=1}^{M} \mathcal{L}(\hat{\mathbf{y}}^m, \mathbf{y}), \tag{2}$$

where $\mathcal{L}(\hat{\mathbf{y}}^m, \mathbf{y}) = -\log P(\mathbf{y}|\mathbf{X};\Theta_m)$, and Θ_m denotes the subset of Θ used for exit m. In this work, we implement early-exiting in four diverse model architectures. Conformer-CTC and Conformer-AED share a Conformer encoder [22] architecture, with early exits appended to every other layer of a 12-layer encoder, but features different decoders. In addition we consider two pre-trained backbones based on Wav2Vec-2.0 [23] and WavLM [17] models. Their hyper-parameters are summarised in Tab. 1.

Conformer-CTC: The Conformer encoder is used to obtain $\mathbf{h}_1^T$, and the decoder is a linear layer with softmax. The intermediate loss function is connectionist temporal classification (CTC) [24]:

$$\mathcal{L}_{\text{CTC}}(\hat{\mathbf{y}}, \mathbf{y}) = -\log \sum_{\mathbf{a}_1^T \in \mathcal{B}^{-1}(\mathbf{y}_1^U)} \prod_{t=1}^{T} P(a_t|\mathbf{h}_1^T), \tag{3}$$

where $a_t \in \mathcal{Y} \cup \{\phi\}$, and $\mathcal{B}$ maps $\mathbf{a}_1^T$ to $\mathbf{y}_1^U$ by removing repeated tokens and ϕ.

Conformer-AED: To test the robustness of early-exits with complex decoders, we use an attention-based encoder-decoder (AED) model [25]. Retaining the architecture of the Conformer-CTC encoder, we replace its linear decoder with four Transformer layers with cross-attention on $\mathbf{h}_1^T$. This decoder contains two output heads, trained with a CTC loss and a sequence-to-sequence

cross-entropy loss respectively [26]. The overall loss function is:

$$\mathcal{L}_{\text{AED}}(\hat{\mathbf{y}}, \mathbf{y}) = \lambda_{\text{CTC}}\mathcal{L}_{\text{CTC}}(\hat{\mathbf{y}}, \mathbf{y}) + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}), \quad (4)$$

where $\mathcal{L}_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{u=1}^{U} \log P(y_u|\mathbf{h}_1^T, \mathbf{y}_1^{u-1})$, and λ's are hyperparameters. Following the SpeechBrain recipe [27], we set λ_{CTC} and λ_{CE} to 0.3 and 0.7, respectively. During inference, only the cross-entropy head is used, and a Transformer-based language model trained with the same tokenization is used to rescore the hypothesis.

Wav2Vec2-CTC and WavLM-CTC: Whereas the Conformer-based models are trained *from scratch* by optimizing Eq. 2, for Wav2Vec2-CTC and WavLM-CTC we instead apply early-exit fine-tuning on a *pre-trained* encoder (Wav2Vec-2.0 [23] and WavLM Base+) encoder using the joint CTC losses (Eq. 3). The models operate on raw waveforms processed using a convolutional feature extractor, which is frozen during fine-tuning. Note that the pretraining of the two backbones is also different, as WavLM Base+ is trained on $94k$ hours of English speech.

4. EARLY-EXIT SELECTION

For an early-exit model at inference time, an uncertainty measure can be used to decide at which exit to output. That is, an exit layer is selected when its uncertainty drops below a given threshold that is estimated to guarantee a desired performance level. Since the encoder layer outputs are converted to posterior probabilities through a softmax module, their uncertainties are suitably represented by their average frame entropies:

$$\Xi^m = -\frac{1}{T|\mathcal{Y}|}\sum_{t=1}^{t=T}\sum_{y\in\mathcal{Y}} P[y|\mathbf{h}_t^m]\log(P[y|\mathbf{h}_t^m]) \quad (5)$$

where, $P[y|\mathbf{h}_t^m]$ is the probability in the m^{th} encoder output at time t for each output token $y \in \mathcal{Y}$. While entropy is a common choice in literature, we also investigate a metric that estimates sentence confidence, computed by applying a softmax to the scores of the N-best hypotheses provided by each decoder:

$$\Psi^m = \frac{e^{s_1^m}}{\sum_1^K e^{s_k^m}} \quad (6)$$

where s_k^m is the log-probability of the k^{th} hypothesis at layer m, i.e. $s_k^m = \log(P[\hat{\mathbf{y}}_k^m|\mathbf{X}; \mathbf{\Theta}_m])$, and K is the number of N-best hypotheses. Preliminary experiments, aimed at finding the optimal performance/complexity trade-off, suggested the value $K = 300$.

5. EXPERIMENTS

We carry out experiments using LibriSpeech [28], TED-LIUM (release 3, [29]), and VoxPopuli [30]. LibriSpeech contains ≈1,000 hours of read-aloud audiobooks, partitioned into ≈960h for training and ≈20h for evaluation. TED-LIUM comprises of ≈452h of transcribed English speeches from TED video conferences for training and ≈6h for evaluation. VoxPopuli is a multi-lingual corpus formed of ≈400K hours of recordings from European Parliament events. For this work, we use the English subset, consisting of ≈543h for training ≈60h for evaluation.

5.1. Implementation details

We consider four models: Conformer-CTC (Eq. 3), Conformer-AED (Eq. 4), Wav2Vec2-CTC and WavLM-CTC. The Conformer models take as input 80 Mel Frequency Cepstral Coefficients (MFCCs). This MFCC sequence is passed through a series of 1D convolution sub-sampling layers. The output of this block is applied to a positional encoding module that feeds a stack of 12 Conformer blocks. Wav2Vec2-CTC also consists of a convolutional feature extractor followed by a 12-layer self-attention encoder, but instead takes raw waveforms as input. Both Conformer-CTC and Conformer-AED use a BPE-based tokenizer [31], with 256 and 5000 tokens respectively. The exit decoders of Wav2Vec2-CTC instead produce 32 grapheme-based tokens (28 characters + 1 blank token + 2 sentence boundary tokens + 1 unknown token) per its official recipe.

The code for both Conformer-CTC and Wav2Vec2-CTC models is available[1], while the Conformer-AED model and WavLM-CTC follow the related SpeechBrain recipe. Tab. 1 summarises the main hyperparameters for the four models.

6. RESULTS

All results reported in this section are expressed in terms of word error rates (WERs) computed on the standard test partitions of the three datasets. Tab. 2 reports the performance on LibriSpeech at different exits, training Conformer-CTC and Conformer-AED models from scratch and fine-tuning Wav2Vec2-CTC and WavLM-CTC. For each model, we also report the performance of the corresponding single-exit model for comparison.

The Conformer-CTC model with 12 layers achieves 6.5% on test-clean and 17.7% on test-other. As expected, WER is higher in the lower layers. The performance decreases significantly only in the lowest two exits (Layer 2, 4). In the middle layers (6, 8, 10), performance is comparable to that of the uppermost layer, while requiring significantly fewer parameters. Similar trends are achieved with Conformer-AED, but with significantly better overall performance (2.3% and 6.0% WER in the 12th layer for test-clean and test-other, respectively). This absolute improvement is attributed to the Transformer-based decoders and language model rescoring, allowing to reach state-of-the-art on LibriSpeech. Tab. 2 shows that both models based on pre-trained backbones (Wav2Vec2-CTC and WavLM-CTC) exhibits similar behaviors. However, since in pre-training they have been optimised solely on the loss of the highest layer, they displays higher WERs at lower exits with a noticeable performance gain beginning at the 8th layer. This low-layer degradation is much less evident in the both models trained from scratch. In summary, although smaller and trained on less data, the Conformer-CTC/AED models perform better than those based on pre-trained models in the lowest three layers. Apart from the lowest exits, the early-exit Conformer-CTC/AED models achieve better WERs than the corresponding single-exit counterparts (column "no-EE" in Tab. 2). This indicates the beneficial effects of the compound loss, acting as a regulariser and improving both robustness and generalization, as observed in previous studies incorporating losses at lower layers [13, 7, 32]. In other words, using a single model with multiple exits not only reduces the computational burden of training multiple single exit models but also delivers better performance. Note that the same behavior is not observed for both the pre-trained models, where the performance of the 12th layer of the EE model is worse than the corresponding 12-layer no-EE model. These results reflect that Wav2Vec2-CTC and WavLM-CTC have not been trained from scratch but only fine-tuned with early exits.

These findings suggest that for early-exit architectures,

[1] https://github.com/augustgw/early-exit-transformer and https://github.com/augustgw/wav2vec2-ee

Table 2: WERs of the 4 models on LibriSpeech at different layers. "EE": early-exit models; "no-EE": individual single-exit models trained independently. "Layer": exited layer in EE models or total number of layers in no-EE models. Due to time constraints, experiments with No-EE models are focused on low layers, which are the most relevant, and are not available for all cases ("-").

Layer	Conformer-CTC				Conformer-AED				Wav2Vec2-CTC				WavLM-CTC			
	test-clean		test-other		test-clean		test-other		test-clean		test-other		test-clean		test-other	
	no-EE	EE	no-EE	EE	no-EE	EE	no-EE	EE	no-EE	EE	no-EE	EE	no-EE	EE	no-EE	EE
2	17.6	23.9	36.1	43.8	18.9	20.1	38.0	40.1	35.7	33.7	56.7	56.0	28.2	28.0	47.6	48.5
4	9.8	11.6	24.3	25.7	12.8	12.5	25.8	25.2	17.4	17.4	35.5	36.7	12.9	13.9	27.2	27.3
6	7.6	6.8	20.0	18.1	8.4	7.7	20.1	17.1	10.7	9.6	24.8	23.7	–	8.7	–	18.4
8	–	5.9	–	16.3	–	4.4	–	11.5	–	5.8	–	15.9	–	4.8	–	12.4
10	–	5.2	–	15.8	–	2.8	–	6.9	–	4.5	–	12.6	–	4.0	–	9.5
12	6.5	5.1	17.7	15.1	2.5	2.3	6.1	6.0	3.4	4.3	8.6	12.2	3.0	3.6	6.5	8.8

Table 3: WERs on the **TED-LIUM** and **VoxPopuli** at different exits, training the model Conformer-CTC from scratch.

Layer	TED-LIUM		VoxPopuli	
	no-EE	EE	no-EE	EE
2	42.7	43.8	27.3	36.7
4	35.4	23.4	19.7	21.1
6	25.5	18.0	18.7	17.3
8	–	16.1	–	15.4
10	–	14.9	–	14.7
12	16.4	14.6	16.3	14.3

training a model from scratch might be more efficient than fine-tuning a large but conventionally pre-trained model. It is worth noting that the same trends are observed considering different decoders, different training losses, and independently of the use of a language model. We hypothesise that both Wav2Vec2.0 and WavLM models would display similar improvements in low-layer performance if trained from scratch with early-exits. Unfortunately, these experiments were not affordable.

Finally, experiments on TED-LIUM and VoxPopuli, shown in Tab. 3, confirm the observations drawn on LibriSpeech. In these experiments, we also observe superior performance in models trained with the compound early-exit loss as compared to those trained with single exits, for layers higher than 4. Moreover, experiments on LibriSpeech-100h (not reported here for the sake of space) have shown similar trends, suggesting the effectiveness of the approach also for low-resource ASR settings.

6.1. Exit selection during inference

In this section we analyse exit selection at inference time using either average frame entropy (Eq. 5) or sentence confidence (Eq. 6). We follow the common thresholding approach, selecting the first exit below a predefined entropy threshold or above a predefined posterior threshold. Previous studies [11, 10] have observed that although the overall performance of lower layers is typically inferior to that of the final layers, in many cases the performance is on par. Being able to identify those cases would considerably reduce the overall computational cost. Fig. 3 shows the average selected exit with the corresponding WER when varying the threshold for two models and the two metrics (the other models are not reported for the sake of readability). The closer the curve to the chart origin, the better. We observe that, as expected, better models deliver better performance in exit selection: the Conformer-AED lines are well below the others. Sentence confidence (dotted lines) on average selects lower exits than entropy at the same WER values, indicating that it provides a better trade-off between saving computation and maintaining performance. However, estimating the sentence confidence maybe not always computationally viable and other factors, such as the computational capabilities of the hosting device, may influence which exit selection method is most appropriate.

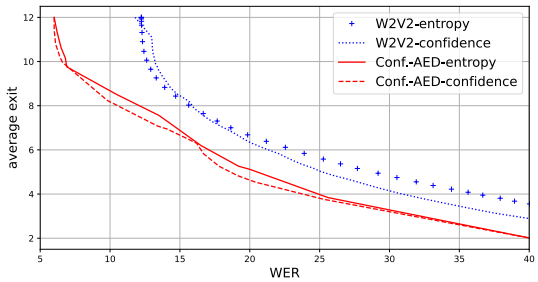

Fig. 3: Average exit selection (y-axis) and WER (x-axis) when varying the exit selection threshold for Conformer-AED and Wav2Vec2-CTC, using both entropies and sentence confidence as exit metrics.

7. CONCLUSION AND FUTURE WORKS

In this paper, we have investigated early-exit architectures for ASR by comparing the training and inference of models based on the Conformer and on pre-trained models (Wav2Vec2 and WavLM). In contrast to previous studies, we bring additional attention to the training dynamics of early-exit models. In particular, we demonstrated the benefits of training models from scratch using early exits, as compared to fine-tuning a pre-trained model, on three datasets. Future works will investigate weighting schemes for the compound loss in Eq. 2 or alternative training strategies [13], including distillation (similar to [32]) from upper layers. Additionally, methods to approximate confidence-based methods for exit selection could improve their practicality on low-resource devices.

Acknowledgments. This work was partially funded by the PNRR ICSC National Research Centre for High Performance Computing, Big Data and Quantum Computing (CN00000013), and by PNRR project FAIR - Future AI Research (PE00000013) under the NRRP MUR program funded by the NextGenerationEU. We also acknowledge support from the JSALT23 workshop, hosted at Le Mans University, France, and sponsored by Johns Hopkins University with unrestricted gifts from Amazon, Facebook, Google, and Microsoft.

8. REFERENCES

[1] K. Liang, R. Ma, Y. Hua, H. Wang, N. Hu, T. Song, H. Gao, and H. Guan, "WH2D2N2: Distributed AI-enabled OK-ASN service for Web of Things," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, pp. 1 – 16, 2023.

[2] G. Cerutti, R. Prasad, A. Brutti, and E. Farella, "Compact recurrent neural networks for acoustic event detection on low-energy low-complexity platforms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 654–664, 2020.

[3] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[4] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. of ICASSP*, Vancouver (Canada), May, 26-31 2013, pp. 7893–7897.

[5] H. Song, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding," in *Proc. of 4th International Conference on Learning Representations (ICLR)*, 2016.

[6] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceeding of CVPR*, 2016, pp. 4820–4828.

[7] T. Teerapittayanon, B. McDanel, , and H. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," *arXiv:1709.01686*, 2017.

[8] M. Phuong and Lampert, "Distillation-based training for multi-exit architectures," in *Proc. of ICCV*, 2019.

[9] "HuBERT-EE: Early exiting HuBERT for efficient speech recognition," .

[10] D. Berrebbi, B. Yan, and S. Watanabe, "Avoid overthinking in self-supervised models for speech recognition," in *IEEE ICASSP*, 2022.

[11] S. Zaiem, R. Algayres, T. Parcollet, S. Essid, and M. Ravanelli, "Fine-tuning strategies for faster inference using self-supervised models: A comparative study," *arXiv:2303.06740*, 2023.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[13] S. Scardapane, M. Scarpiniti, E. Maccarelli, and M. Uncini, "Why should we add early exits to neural networks?," *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, 2020.

[14] N. P. Ghanathe and S. Wilton, "T-RECX: Tiny-resource efficient convolutional neural networks with early-exit," in *Proceedings of ACM International Conference on Computing Frontiers*, 2023, pp. 123–133.

[15] R. T. Mullapudi, W. R. Mark, N. M. Shazeer, and K. Fatahalian, "HydraNets: Specialized dynamic architectures for efficient inference," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8080–8089, 2018.

[16] W. N. Hsu, B. Bolte, Y. 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.

[17] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao, J. Wu, L. Zhou, S. Ren, Y. Qian, Y. Qian, M. Zeng, and F. Wei, "WavLM: Large-scale self-supervised pre-training for full stack speech processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1505–1518, 2022.

[18] R. Tang, K. V. S. M. Kumar, J. Xin, P. Vyas, W. Li, G. Yang, Y. Mao, C. Murray, and J. Lin, "Temporal early exiting for streaming speech commands recognition," in *Proc. of ICASSP*, 2022.

[19] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "BERT loses patience: Fast and robust inference with early exit," in *NIPS*, 2020.

[20] W. Liu, P. Zhou, Z. Zhao, Z. Wang, H. Deng, and Q. Ju, "FastBERT: a self-distilling BERT with adaptive inference time," in *ACL*, 2020.

[21] J. Xin, R. Tang, J. Lee, Y. Yu, and J. Lin, "DeeBERT: Dynamic early exiting for accelerating bert inference," in *ACL*, 2020.

[22] Y. Lu, Z. Li, D. He, Z. Sun, B. Dong, T. Qin, L. Wang, and T. Liu, "Understanding and improving transformer from a multi-particle dynamic system point of view," *arXiv:1906.02762*, 2019.

[23] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "Wav2Vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," 2020.

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

[25] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015.

[26] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *IEEE ICASSP*, 2016.

[27] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," 2021, arXiv:2106.04624.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. of ICASSP*, 2015, pp. 5206–5210.

[29] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *Speech and Computer*, A. Karpov, O. Jokisch, and R. Potapova, Eds. 2018, pp. 198–208, Springer International Publishing.

[30] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," in *Proceedings of the ACL*. Aug. 2021, Association for Computational Linguistics.

[31] S. R., B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. of ACL*, 2016, pp. 1715—1725.

[32] S. Geng, P. Gao, Z. Fu, and Y. Zhang, "RomeBERT: Robust training of multi-exit BERT," *arXiv, preprint arXiv:2101.09755*, 2021.