Interspeech 2025
17-21 August 2025, Rotterdam, The Netherlands

Dynamic Acoustic Model Architecture Optimization in Training for ASR

Jingjing Xu[1,2], *Zijian Yang*[1], *Albert Zeyer*[1,2], *Eugen Beck*[2], *Ralf Schlüter*[1,2], *Hermann Ney*[1,2]

[1]Machine Learning and Human Language Technology Group, RWTH Aachen University, Germany
[2]AppTek GmbH, Germany

{jxu, zeyer, zyang, schlueter}@ml.rwth-aachen.de, ebeck@apptek.com

Abstract

Architecture design is inherently complex. Existing approaches rely on either handcrafted rules, which demand extensive empirical expertise, or automated methods like neural architecture search, which are computationally intensive. In this paper, we introduce DMAO, an architecture optimization framework that employs a grow-and-drop strategy to automatically reallocate parameters during training. This reallocation shifts resources from less-utilized areas to those parts of the model where they are most beneficial. Notably, DMAO only introduces negligible training overhead at a given model complexity. We evaluate DMAO through experiments with CTC on *LibriSpeech*, *TED-LIUM-v2* and *Switchboard* datasets. The results show that, using the same amount of training resources, our proposed DMAO consistently improves WER by up to $\sim 6\%$ relatively across various architectures, model sizes, and datasets. Furthermore, we analyze the pattern of parameter redistribution and uncover insightful findings.

Index Terms: speech recognition, dynamic architecture optimization, CTC

1. Introduction & Related Work

In the last decade, self-attention-based architectures such as Transformer [1], Conformer [2], and E-branchformer [3] have revolutionized the automatic speech recognition (ASR) field, significantly boosting the performance of acoustic models. For convenience and efficient scalability, identical neural blocks are stacked together to construct the acoustic encoder. However, this may result in inefficient use of parameters, as research [4, 5, 6] has shown that the model may perform different roles at varying depths, and such behavior is data-dependent. The basic components of the model architecture, such as feed-forward, convolutional, and attention layers, are specialized for different tasks. The question of how to design a better data-specific model architecture by optimally distributing the parameters of different types at various depths based on their usefulness remains under-explored.

The recently proposed Zipformer [7] divides the layers into stacks, with each stack having a different embedding dimension. However, the authors do not provide an explanation for the choice of these specific dimensions. In the works [8, 9], a block-wise scaling approach based on hand-crafted rules is employed to adjust the depth and width of blocks. However, the hyper-parameters governing the scaling process still require tuning to achieve optimal performance, thereby introducing additional training efforts. Neural architecture search (NAS) can be used to discover efficient architectures that outperform hand-designed ones and has been applied in [10, 11, 12] for the ASR task. However, NAS approaches often demand significantly more time and computational resources due to the extensive exploration of the architecture space and repeated evaluations of candidate models. These previous solutions are non-trivial. They either demand extensive expertise from researchers or require additional training efforts. Therefore, in this work, we aim to find a solution that dynamically optimizes the model architecture within a fixed resource budget during training, without incurring additional overhead.

The grow-and-drop paradigm has been utilized for training sparse networks from scratch [13, 14, 15, 16]. In those works, training begins with a randomly initialized sparse network with predefined sparsity, but the optimal architecture remains unknown. During the grow phase, neurons or connections are either added randomly or based on certain strategies, allowing the model to explore a variety of architectural configurations. After the growth phase, the network deactivates parameters that are considered unimportant.

In this work, we extend the grow-and-drop paradigm to dense network training. We propose a simple Dynamic Model Architecture Optimization (DMAO) framework that utilizes the grow-and-drop strategy to efficiently redistribute parameters during the training process. To enable flexible adaptation, we partition the dense model into finer groups of parameters. We define and compare various metrics to assess the importance of these parameter groups based on their contribution to the ASR model's performance. The parameter groups are ranked according to these metrics, after which we grow the top-ranked groups and remove the lowest-ranked ones. In this way, we enhance the model's capacity by reallocating resources from underutilized areas to those where they are most needed. We evaluate our approach by conducting experiments with the connectionist temporal classification (CTC) [17] model on *LibriSpeech*, *TED-LIUM-v2* and *Switchboard* datasets. The results demonstrate that DMAO consistently improves word-error-rate (WER) by up to $\sim$6% relative across different architectures (e.g. Conformer, E-Branchformer), model sizes, and datasets. We also investigate the optimal schedule for applying DMAO during training. Furthermore, we analyze how the parameter distribution changes with DMAO and provide possible explanations.

2. Dynamic Model Architecture Optimization

In this section, we present our approach for dynamically optimizing the acoustic model architecture during training. The core idea is to redistribute the parameters from parts of the model where they are least useful to those where they are most critical, thereby enhancing the model's capacity while maintaining a fixed budget for specific resource constraints. To enable more flexible global adjustments, we begin with partitioning the model into smaller parameter groups. We then rank these groups based on their importance, as computed using the metrics described in Sec. 2.2. The parameters are reallocated using the grow-and-drop paradigm, i.e., by removing the least important ones and duplicating the most important ones. The remaining parameters are left unchanged, and training proceeds with the updated model.

2.1. Model Partition

Conformer [18] and E-branchformer [3], which are among the most prevalent encoder architectures for ASR tasks, are used in this work. Conformer consists of feed-forward network (FFN), multi-head self-attention module (MHSA) and convolutional module (Conv). FFN has two weight matrices $W_{\text{ff}_1}, W_{\text{ff}_2}^T \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$, d_{model} is the model dimension and d_{ff} denotes the inner dimension of FFN. We partition the FFN into C groups,with sub-matrices $W_{\text{ff}_1}^c, {W_{\text{ff}_2}^c}^T \in \mathbb{R}^{d_{\text{model}} \times \frac{d_{\text{ff}}}{C}}$. MHSA has H heads and consists of four projection matrices, $W_k, W_q, W_v, W_o^T \in \mathbb{R}^{d_{\text{model}} \times (d_h \cdot H)}$, d_h is the dimension per head. We partition the MHSA into H groups, each with sub-matrices $W_k^h, W_q^h, W_v^h, {W_o^h}^T \in \mathbb{R}^{d_{\text{model}} \times d_h}$. Conv consists of weights $W_{\text{in}} \in \mathbb{R}^{d_{\text{model}} \times 4d_{\text{model}}}$, $W_{\text{conv}} \in \mathbb{R}^{k \times k \times 2d_{\text{model}} \times 1}$, and $W_{\text{out}} \in \mathbb{R}^{2d_{\text{model}} \times d_{\text{model}}}$. Likewise, we partition one Conv into M groups, with sub-matrices $W_{\text{in}}^m \in \mathbb{R}^{d_{\text{model}} \times \frac{4d_{\text{model}}}{M}}$, $W_{\text{conv}}^m \in \mathbb{R}^{k \times k \times \frac{2d_{\text{model}}}{M} \times 1}$ and $W_{\text{out}}^m \in \mathbb{R}^{\frac{2d_{\text{model}}}{M} \times d_{\text{model}}}$. This partition allows for easy adjustment of the hidden dimensions in each module. The E-branchformer can be partitioned into a similar manner. Suppose the encoder has L layers, we partition the model into $(2C + H + M) \times L$ groups.

2.2. Ranking Based on Importance Scores

We define several metrics to assess the importance of parameter groups. These scores are used to rank parameter groups according to their contribution to ASR performance. Additionally, they should be efficient to calculate and easily accessible, ensuring minimal computational overhead. Suppose the parameter group has N weight elements $w_i \in \mathbb{R}, i = \{1, ..., N\}$.

Magnitude refers to the absolute value of a weight and are widely used as metric for pruning ASR models [19, 20]. Weights with larger values after applying a norm to their magnitudes are considered more important, as they have a greater influence on the activation and final output [21, 22]. To calculate the importance score at training step t, we average the magnitudes of all weights in the parameter group. Additionally, to make the score more stable, we use exponential smoothing, as in [23], where α is a constant smoothing factor.

$$s_t = (1 - \alpha)s_{t-1} + \alpha \left(\frac{1}{N} \sum_{i=1}^{N} |w_i| \right) \tag{1}$$

Gradient represent the rate of change of the loss function with respect to the model's weights [24, 25]. Weights with small absolute gradients have a smaller impact on reducing the model's loss and are therefore less influential in the training process. To assess the importance of each parameter group, we compute the averaged L_2 norm of the gradients across all weights in that group and apply exponential smoothing as in Eq. (1) as well.

$$s_t = (1 - \alpha)s_{t-1} + \alpha \frac{1}{N} \sqrt{\sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}_{\text{ASR}}}{\partial w_i} \right)^2} \tag{2}$$

First-order Taylor Approximation provides a linear approximation of a function around a given point. We compute the importance score of an individual weight w_i using the first-order Taylor approximation to estimate the loss difference when w_i is removed or equivalently set to zero, as in

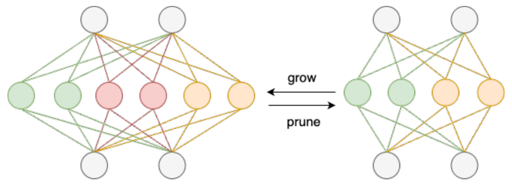

Figure 1: *Illustration of grow/prune parameter groups in an FFN module. Grey nodes indicate input/output. The parameter groups (green, red, yellow) include hidden nodes and their connections. Selecting the green group for doubling adds $\frac{d_{ff}}{C}$ new hidden nodes and their connections (in red), updating FFN weight matrices to $W_{ff_1}, W_{ff_2}^T \in \mathbb{R}^{d_{model} \times \frac{C+1}{C} d_{ff}}$. Pruning the red group removes its nodes and corresponding connections.*

[26, 23]. Compared to using gradients alone, this approach also takes the weight value into account, ensuring that both the size and effect of the weights are accounted for the decision-making process. For parameter groups, we compute the importance score using the same averaged L_2-norm as in Eq. (2).

$$s_t = (1 - \alpha)s_{t-1} + \alpha \frac{1}{N} \sqrt{\sum_{i=1}^{N} \left(\frac{\partial \mathcal{L}_{\text{ASR}}}{\partial w_i} w_i \right)^2} \tag{3}$$

Learnable Score Inspired by movement pruning [27, 28], we assign a learnable score s to each parameter group to indicate its importance. We directly scale all weight elements in that group as $w_i' = w_i \times s$. After each adaptation, the scales of all parameter groups (i.e., s) are reset to 1 to ensure a fair comparison in the next round. Empirically, we observe that the model may diverge if the scaling effect is removed. Therefore, we randomly sample training steps with a probability of 0.5, i.e., for half of the training steps, unscaled weights are used, and for the other half, scaled weights are applied. In this way, the model remains robust to the scaling effect.

2.3. Architecture Optimization

The adaptation is performed iteratively over I iterations during training. Let T_{end} denote the training step at which adaptation stops, and ΔT the number of training steps per iteration. At the end of each i^{th} iteration, i.e., at the $i\Delta T$ training step, we use the updated importance scores to rank the parameter groups and optimize the model architecture. Let $\delta \in [0, 0.5]$ denote the adaptation ratio, then we optimize the architecture by removing the bottom $\frac{\delta}{I}$ fraction of the parameters and doubling the top $\frac{\delta}{I}$ fraction. As a result, a total of 2δ parameters are modified. The grow-and-prune paradigm of FFN is illustrated in Figure 1 and can be easily analogized to all Conformer modules. With the partitioning design described in Sec. 2.1, only the hidden dimension of each module is modified, while the input and output dimensions remain unchanged. This design enables each parameter group to be independently doubled or pruned without altering the shapes of other groups. The initialization of the newly introduced weights is investigated in Sec. 3.2.3. Alternatively, we can use different resource constraints, such as the number of FLOPs, as used in [29].

3. Experiments

3.1. Experimental Setup

We conduct the experiments on the 960h *LibriSpeech* (*LBS*) corpus [30], the 200h *TED-LIUM-v2* (*TED-v2*) corpus [31] and the 300h *Switchboard* (*SWB*) corpus [32]. We use a phoneme-based CTC model following the setup in [33] as baseline, with log Mel-filterbank features as the input. It has a VGG front-end

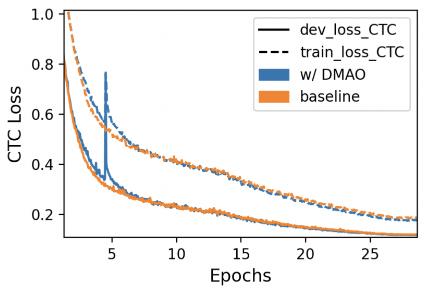

Figure 2: *Comparison of training loss between the baseline and the model w/ DMAO on LibriSpeech dataset.*

and 12 Conformer/E-branchformer blocks. The output labels are 79 end-of-word augmented phonemes. SpecAugment [34] is applied for data augmentation. For Conformer, the number of attention heads H is $\frac{d_{\text{model}}}{64}$ and $d_{\text{ff}} = 4 \times d_{\text{model}}$. For E-branchformer, the inner dimension of local extractor d_{inter} is set to $6 \times d_{\text{model}}$. In terms of model partition, we set both C and M to 4. We train all models for the same number of epochs: 50 epochs for *SWB* and *TED-v2*, and 30 epochs for *LBS*. The one-cycle learning rate scheduler [35] is used, with the learning rate increasing from 4×10^{-6} to 4×10^{-4} for 45% of the training, decreasing for the next 45%, and decaying to 10^{-7} in the final 10%. We use RETURNN [36] to train the acoustic models and RASR [37] for recognition. In inference, we apply Viterbi decoding with a 4-gram word-level language model. To further reduce computation, the smoothed scores in Eq. (1-3) are computed and updated every 1000 steps. All our config files and code to reproduce the results can be found online[1].

3.2. Experimental Results

3.2.1. *Overall Results with Conformer Encoder*

We present the overall results with Conformer across different model sizes in Table 1. The optimal DMAO setting including the selection of importance metrics is used, ablation study results are presented in the following sections. The results show that DMAO improves WER performance across all model sizes and datasets, achieving up to a 6% relative improvement. Furthermore, within each size category, the 'retrain' model with the optimized architecture outperforms the baseline model. To ensure a fair comparison, all training factors, except for the model architecture, remain consistent. This suggests that the updated architecture provides greater model capacity. Moreover, when compared to retraining with optimized architecture, the model with DMAO yields comparable or even slightly better results, indicating that directly applying adaptation during training does not disrupt model convergence. Figure 2 plots the training loss for both the baseline model and the model with DMAO. After the architecture update, we observe a sharp increase in training loss, as the model's weights and parameter distribution are altered. However, the loss decreases quickly, indicating that the model recovers rapidly. Following this, the training loss of the DMAO model remains slightly lower than the baseline, verifying our assumption that the updated model has greater capacity. To assess the reliability of the improvement, we compute bootstrap-estimated probabilities of WER reduction on each test set using the implementation from [38]. Absence of notation $^{+,++,+++}$ signifies a confidence level below 90%. The results indicate strong confidence that DMAO improves WER compared to the baseline on both the *LBS* and *SWB* datasets.

[1] https://github.com/rwth-i6/returnn-experiments/tree/master/2025-dynamic-model-architecture-optimization

Table 1: *Overall WERs [%] results using Conformer as the encoder architecture across different model sizes on three datasets. The baseline model does not use DMAO. The 'w/ DMAO' model applies model architecture optimization once at 20% (15% for LBS) of the total training steps, with the adaptation ratio δ set to 0.15, using the first-order Taylor approximation as the metric. The 'retrain' model uses the final architecture from the 'w/ DMAO' model but is trained from scratch.*

Params. [M]	Model	TED-v2		SWB		LBS	
		dev	test	Hub 5'00	Hub 5'01	dev-other	test-other
19.0	baseline	7.8	8.2	14.8	13.4	7.6	7.9
	w/ DMAO	**7.6**	**7.7**	**14.2**	**12.8**$^{+++}$	**7.3**	**7.8**$^{+++}$
	+retrain	7.7	7.9^{+}	14.2	13.1^{+++}	7.5	7.9^{+++}
42.1	baseline†	7.7	**7.9**	14.0	12.8	7.0	7.5
	w/ DMAO‡	**7.3**	**7.9**	**13.5**	**12.3**$^{+++}$	6.8	**7.2**$^{+++}$
	+retrain	7.4	**7.9**	13.9	12.5^{++}	**6.7**	**7.2**$^{+++}$
72.8	baseline	7.6	7.7	13.9	12.6	6.8	7.1
	w/ DMAO	7.4	7.7	13.8	12.2^{++}	**6.6**	**6.9**$^{++}$
	+retrain	**7.3**	**7.5**	**13.2**	**11.8**$^{+++}$	**6.6**	7.0

† used as the baseline for Table 2, Table 3 and Table 4. ‡ used to plot Figure 3.
$^{+,++,+++}$ denotes probability of improvement $> 90\%$, $> 95\%$, and $> 99\%$.

Table 2: *Comparison of WERs [%] using different metrics and various values of the smoothing factor α to compute importance scores for parameter groups. The baseline model does not use dynamic architecture adaptation, whereas the other models apply it once at 20% (15% for LBS) of the total training steps with adaptation ratio δ 0.15. The baseline is † from Table 1.*

Metric	Smooth factor α	SWB		LBS	
		Hub5'00	Hub5'01	dev-o	test-o
baseline	-	14.0	12.8	7.0	7.5
magnitude	0.9	14.0	12.8	7.7	7.9
gradient	0.9	13.6	12.3	7.0	**7.2**
first order Taylor	0.9	**13.5**	12.3	**6.8**	**7.2**
	1	13.7	12.3	7.1	7.5
	0.7	13.8	12.5	6.9	7.3
	0.5	13.6	12.3	7.0	7.3
learnable score	-	13.6	**12.2**	**6.9**	7.3

3.2.2. *Importance Score Metrics Comparison*

In Table 2, we compare the ASR results using different scoring metrics for ranking parameter groups in dynamic model architecture adaptation. Except for the magnitude-based scoring metric, all other metrics lead to WER improvements over the baseline, highlighting the importance of choosing a good metric. Among the metrics, the first-order Taylor approximation achieves the best performance, showing a $\sim 4\%$ improvement over the baseline on both the *SWB* Hub5'01 and *LBS* test-other sets. Additionally, the results suggest that exponential smoothing improves WER, making the scores more reliable.

3.2.3. *Initialization Comparison for Newly Introduced Weights*

In Table 3, we investigate different initialization strategies for the newly introduced weights after each model architecture update. The results show that directly copying the weights from the top δ parameters yields the best performance.

3.2.4. *Ablation Study of DMAO schedule*

Table 4 presents the results of an ablation study on the DMAO schedule. We observe that, applying DMAO too late in the training process can degrade performance, possibly because the model struggles to recover from the updates. On the other hand, applying it too early is suboptimal, as the scores computed in the initial training steps may be suboptimal. A good balance

Table 3: *WERs [%] comparison of different initialization for the newly introduced weights after each architecture optimization. The baseline is † and 'copy' is ‡ from Table 1. 'copy' refers to using the exact same weights from the top δ parameters, while 'copy + noise' adds extra Gaussian noise with a mean of 0 and a standard deviation of 0.01.*

Initialization	SWB		LBS	
	Hub5'00	Hub5'01	dev-other	test-other
baseline w/o DMAO	14.0	12.8	7.0	7.5
copy	**13.5**	**12.3**	**6.8**	**7.2**
copy + noise	13.7	**12.3**	6.9	**7.2**
random	13.6	**12.3**	7.0	7.4

Table 4: *WERs [%] of applying DMAO at different training stages, with varying adaptation ratios and number of iterations (first-order Taylor approximation as metric). T_{total} denotes the total number of training steps. The baseline is † from Table 1.*

$\frac{T_{\text{end}}}{T_{\text{total}}}$	Update ratio δ	Num. iters. I	TED-v2		SWB	
			dev	test	Hub5'00	Hub5'01
baseline w/o DMAO			7.7	7.9	14.0	12.8
20%	0.1	1	7.6	**7.7**	13.4	12.3
	0.15		**7.3**	7.9	13.5	12.3
		4	7.5	8.0	13.5	12.3
		8	7.5	7.9	13.9	12.3
	0.2	1	7.4	**7.7**	13.5	12.5
	0.25		7.5	**7.7**	**13.3**	**12.2**
10%	0.15		7.7	8.0	13.5	12.5
30%			7.7	8.1	14.0	12.5
50%			8.0	7.9	14.0	13.0

is observed when applying DMAO at around 20% of the total training steps, during a phase where the model has not yet plateaued but the loss is no longer changing rapidly. Applying DMAO once tends to outperform multiple iterations, as multiple architecture updates may introduce excessive disruption. Lastly, an adaptation ratio δ between 0.15 and 0.25 appears to be effective.

3.2.5. Applying DMAO to E-branchformer

To validate that DMAO can enhance model capacity regardless of the architecture, we also apply DMAO to the E-branchformer model and present the results in Table 5. The results align with those obtained using the Conformer, showing that DMAO generally enhances the performance of the E-branchformer encoder across various model sizes and datasets.

3.2.6. Updated Architecture Analysis

Figure 3 illustrates the distribution of parameters in the model after DMAO. We observe that in the lower layers, more MHSA is utilized, while fewer Convs are employed. In contrast, in the top layers, MHSA heads are removed, and more Convs and FFNs are introduced. To explain this, we illustrate Figure 4 to show that as the depth increases, the attention maps display highly diagonal patterns. Similar observations have also been made in works [39, 40, 2, 41] for the ASR task. As the range of learned context grows with increasing depth, the global view of the entire sequence appears to become less useful for the upper layers. Moreover, [4] reveals that attention roles can be divided into phonetic and linguistic localization. The lower layers primarily perform phonetic localization, where the model focuses on content-wise similar frames, which may be farther apart, to extract phonologically meaningful features. In contrast, higher layers focus on neighboring frames and aggregate the information for text transcription. This helps explain our observation, as Convs and FFNs are effective for local processing, while MHSA is specialised for global interaction. Additionally, we observe that more MHSA heads are used for *LBS* than for *TED-v2*. A possible explanation is that *LBS* has longer sequences than *TED-v2* (average 12.3s vs 8.2s), so more MHSA heads are needed to capture dependencies over greater distances.

Table 5: *WERs [%] of applying DMAO on E-branchformer across different model sizes. The same DMAO schedule from Table 1 is used, with DMAO applied once at 20% (15% for LBS) of total training steps, an adaptation ratio of δ=0.15, and the first-order Taylor approximation as the metric.*

Params. [M]	DMAO	SWB		LBS	
		Hub5'00	Hub5'01	dev-other	test-other
25.7	no	13.9	12.8	7.3	7.7
	yes	**13.8**	**12.5**	**6.9**	**7.4**
56.9	no	13.6	12.1	6.6	6.9
	yes	**13.0**	**11.7**	**6.5**	**6.8**
100.2	no	**12.9**	**11.9**	6.3	6.7
	yes	13.0	**11.9**	**6.2**	**6.6**

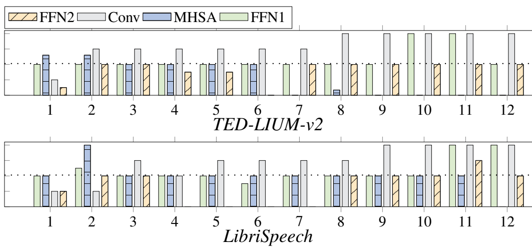

Figure 3: *Distribution of model parameters with DMAO‡ (shown in Table 1) across all depths. The y-axis represents the ratio of parameters after vs. before DMAO for each module. The dotted line indicates the baseline† (shown in Table 1).*

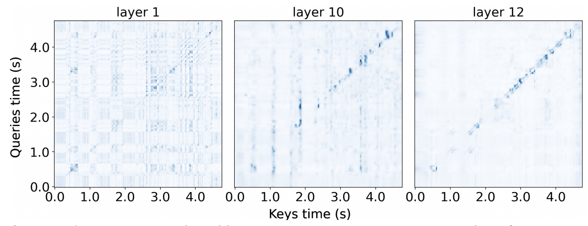

Figure 4: *Averaged self-attention scores across the 6 attention heads in the 1st, 10th, and 12th layers of a randomly selected sequence from LBS for the baseline† (from Table 1).*

4. Conclusion

We propose the DMAO training framework to dynamically optimize the acoustic encoder architecture during training. The optimization is achieved through the grow-and-drop paradigm, where the model's parameter allocation is balanced by redistributing parameters from underutilized regions to areas where they are most needed, thereby enhancing the model's capacity. We demonstrate the effectiveness of DMAO by applying it with a CTC encoder on various datasets, model sizes, and architectures (Conformer, E-Branchformer). The results show a consistent improvement of up to $\sim 6\%$ relative to the baseline when applying DMAO, with only negligible training overhead.

5. Acknowledgement

This work was partially supported by NeuroSys, which as part of the initiative "Clusters4Future" is funded by the Federal Ministry of Education and Research BMBF (funding ID 03ZU2106DD), and by the project RESCALE within the program *AI Lighthouse Projects for the Environment, Climate, Nature and Resources* funded by the Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection (BMUV), funding IDs: 6KI32006A and 6KI32006B.

6. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All you Need," in *NeuIPS*, USA, Dec. 2017, pp. 5998–6008.

[2] M. Burchi and V. Vielzeuf, "Efficient Conformer: Progressive Downsampling and Grouped Attention for Automatic Speech Recognition," in *ASRU*, Colombia, Dec. 2021, pp. 8–15.

[3] K. Kim, F. Wu, Y. Peng, J. Pan, P. Sridhar, and et. al., "E-Branchformer: Branchformer with Enhanced Merging for Speech Recognition," in *SLT*, Doha, Qatar, Jan. 2022, pp. 84–91.

[4] K. Shim, J. Choi, and W. Sung, "Understanding the Role of Self Attention for Efficient Speech Recognition," in *ICLR*, Virtual, Apr. 2022.

[5] A. Pasad, J. Chou, and K. Livescu, "Layer-Wise Analysis of a Self-Supervised Speech Representation Model," in *ASRU*, Cartagena, Colombia, Dec. 2021, pp. 914–921.

[6] K. Shim and W. Sung, "Similarity and Content-based Phonetic Self Attention for Speech Recognition," in *Interspeech*, H. Ko and J. H. L. Hansen, Eds., Incheon, Korea, Sep. 2022, pp. 4118–4122.

[7] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, and D. Povey, "Zipformer: A faster and better encoder for automatic speech recognition," in *ICLR*, Austria, May 2024.

[8] S. Mehta, M. Ghazvininejad, S. Iyer, L. Zettlemoyer, and H. Hajishirzi, "DeLighT: Deep and Light-weight Transformer," in *ICLR*, Virtual, May 2021.

[9] S. Mehta, M. H. Sekhavat, Q. Cao, M. Horton, Y. Jin, C. Sun, S. Mirzadeh, M. Najibi, D. Belenko, P. Zatloukal, and M. Rastegari, "OpenELM: An Efficient Language Model Family with Open Training and Inference Framework," 2024.

[10] Y. Liu, T. Li, P. Zhang, and Y. Yan, "NAS-SCAE: Searching Compact Attention-based Encoders For End-to-end Automatic Speech Recognition," in *Interspeech*, Incheon, Korea, Sep. 2022, pp. 1011–1015.

[11] A. Mehrotra, A. G. C. P. Ramos, S. Bhattacharya, L. Dudziak, and et. al., "NAS-Bench-ASR: Reproducible Neural Architecture Search for Speech Recognition," in *ICLR*, Virtual, May 2021.

[12] Y. Liu, T. Li, P. Zhang, and Y. Yan, "Improved Conformer-based End-to-End Speech Recognition Using Neural Architecture Search," 2021.

[13] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature communications*, vol. 9, no. 1, p. 2383, 2018.

[14] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the Lottery: Making All Tickets Winners," in *ICML*, vol. 119, Virtual, Jul. 2020, pp. 2943–2952.

[15] U. Evci, Y. Ioannou, C. Keskin, and Y. N. Dauphin, "Gradient Flow in Sparse Neural Networks and How Lottery Tickets Win," in *AAAI*, Virtual, Feb. 2022, pp. 6577–6586.

[16] H. Mostafa and X. Wang, "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization," in *ICML*, vol. 97, California, Jun. 2019, pp. 4646–4655.

[17] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, vol. 148, Pittsburgh, Pennsylvania, USA, Jun. 2006, pp. 369–376.

[18] A. Gulati, J. Qin, C. Chiu, N. Parmar, and et. al., "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Interspeech*, Shanghai, China, Oct. 2020, pp. 5036–5040.

[19] C. J. Lai, Y. Zhang, A. H. Liu, S. Chang, Y. Liao, Y. Chuang, K. Qian, S. Khurana, D. D. Cox, and J. Glass, "PARP: Prune, Adjust and Re-Prune for Self-Supervised Speech Recognition," in *NeurIPS*, virtual, Dec. 2021, pp. 21 256–21 272.

[20] J. Kim, S. Chang, and N. Kwak, "PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation," in *Interspeech*, Brno, Czechia, Aug. 2021, pp. 4568–4572.

[21] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Network," in *NeurIPS*, Montreal, Quebec, Canada, Dec. 2015, pp. 1135–1143.

[22] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," in *ICLR*, New Orleans, USA, May 2019.

[23] Z. Yang, Y. Cui, X. Yao, and S. Wang, "Gradient-based Intra-attention Pruning on Pre-trained Language Models," in *ACL*, Toronto, Canada, Jul. 2023, pp. 2775–2790.

[24] X. Ding, G. Ding, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global Sparse Momentum SGD for Pruning Very Deep Neural Networks," in *NeurIPS*, Vancouver, BC, Canada, Dec. 2019, pp. 6379–6391.

[25] D. Bekal, K. Gopalakrishnan, K. Mundnich, S. Ronanki, S. Bodapati, and K. Kirchhoff, "A Metric-Driven Approach to Conformer Layer Pruning for Efficient ASR Inference," in *Interspeech 2023*, Dublin, Ireland, Aug. 2023, pp. 4079–4083.

[26] P. Michel, O. Levy, and G. Neubig, "Are Sixteen Heads Really Better than One?" in *NeurIPS*, Vancouver, BC, Canada, Dec. 2019, pp. 14 014–14 024.

[27] V. Sanh, T. Wolf, and A. M. Rush, "Movement Pruning: Adaptive Sparsity by Fine-Tuning," in *NeurIPS*, virtual, Dec. 2020.

[28] F. Lagunas, E. Charlaix, V. Sanh, and A. M. Rush, "Block Pruning For Faster Transformers," in *EMNLP*, Punta Cana, Dominican Republic, Nov. 2021, pp. 10 619–10 629.

[29] J. Xu, E. Beck, Z. Yang, and R. Schlüter, "Efficient Supernet Training with Orthogonal Softmax for Scalable ASR Model Compression," in *ICASSP*, Apr. 2025, arXiv:2501.18895 To Appear.

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

[31] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks," in *LREC*, Reykjavik, Iceland, May 2014, pp. 3935–3939.

[32] J. Godfrey, E. Holliman, and J. McDaniel, "Switchboard: Telephone Speech Corpus for Research and Development," in *ICASSP*, vol. 1, San Francisco,USA, Mar. 1992, pp. 517–520.

[33] J. Xu, W. Zhou, Z. Yang, E. Beck, and R. Schlüter, "Dynamic Encoder Size Based on Data-Driven Layer-wise Pruning for Speech Recognition," in *Interspeech*, Greece, Sep. 2024, pp. 4563–4567.

[34] W. Zhou, W. Michel, K. Irie, M. Kitza, R. Schlüter, and H. Ney, "The Rwth ASR System for Ted-Lium Release 2: Improving Hybrid HMM With Specaugment," in *ICASSP*, Barcelona, Spain, May 2020, pp. 7839–7843.

[35] W. Zhou, W. Michel, R. Schlüter, and H. Ney, "Efficient Training of Neural Transducer for Speech Recognition," in *Interspeech*, Incheon, Korea, Sep. 2022, pp. 2058–2062.

[36] A. Zeyer, T. Alkhouli, and H. Ney, "RETURNN as a Generic Flexible Neural Toolkit with Application to Translation and Speech Recognition," in *ACL*, Melbourne, Australia, Jul. 2018, pp. 128–133.

[37] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/NN: the RWTH neural network toolkit for speech recognition," in *ICASSP*, Florence, Italy, May 2014, pp. 3281–3285.

[38] M. Bisani and H. Ney, "Bootstrap estimates for confidence intervals in ASR performance evaluation," in *ICASSP*, Montreal, Quebec, Canada, May 2004, pp. 409–412.

[39] S. Zhang, E. Loweimi, P. Bell, and S. Renals, "On The Usefulness of Self-Attention for Automatic Speech Recognition with Transformers," in *SLT*, Shenzhen,China, Jan. 2021, pp. 89–96.

[40] ——, "Stochastic Attention Head Removal: A Simple and Effective Method for Improving Transformer Based ASR Models," in *Interspeech*, Brno, Czechia, Aug. 2021, pp. 2541–2545.

[41] S. Kim, A. Gholami, A. E. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, "Squeezeformer: An Efficient Transformer for Automatic Speech Recognition," in *NeurIPS*, New Orleans, LA, USA, Dec. 2022.