HyperConformer: Multi-head HyperMixer for Efficient Speech Recognition

Florian Mai [⋆,†,‡], *Juan Zuluaga-Gomez* [⋆,†,‡], *Titouan Parcollet* [¶], *Petr Motlicek* [†,§]

[†]Idiap Research Institute, Martigny, Switzerland
[‡]LIDIAP, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland
[¶]University of Cambridge, United Kingdom
[§]Faculty of Information Technology, Brno University of Technology, Czech Republic
florian.mai@idiap.ch

Abstract

State-of-the-art ASR systems have achieved promising results by modeling local and global interactions separately. While the former can be computed efficiently, global interactions are usually modeled via attention mechanisms, which are expensive for long input sequences. Here, we address this by extending HyperMixer, an efficient alternative to attention exhibiting linear complexity, to the Conformer architecture for speech recognition, leading to HyperConformer. In particular, multi-head HyperConformer achieves comparable or higher recognition performance while being more efficient than Conformer in terms of inference speed, memory, parameter count, and available training data. HyperConformer achieves a word error rate of 2.9% on LibriSpeech test-clean with less than 8M neural parameters and a peak memory during training of 5.7GB, hence trainable with accessible hardware. Encoder speed is between 38% on mid-length speech and 56% on long speech faster than an equivalent Conformer.[1]

Index Terms: Hypernetworks, HyperMixer, Efficient Automatic Speech Recognition, LibriSpeech, SpeechBrain

1. Introduction

Automatic Speech Recognition (ASR) technologies have greatly benefited from deep learning, reaching unprecedented levels of accuracy and pushing successful products to real-life use cases. Various architectures of ASR systems co-exist and deliver superlative performance depending on the task or domain of interest [1]. A prevalent family of ASR systems uses self-attention and Transformer neural networks to consume the input speech sequence and build powerful representations both at the acoustic and linguistic levels [2]. Indeed, the ability of Multi-Head Self-Attention (MHSA) [3] to capture long-term dependencies via its sequence-long receptive field helped Transformer ASR architectures to outperform the previous state-of-the-art mostly composed with recurrent neural networks [2]. Nevertheless, ASR not only requires capturing global interactions describing the semantic and linguistic characteristics of the speech utterance but also modeling properly the local interactions that form the speech signal.

Conformer neural networks [4] have been introduced to specifically address this issue. They combine Transformer and Convolutional Neural Network (CNN) blocks to capture the global and local dependencies respectively, leading to improved Word Error Rate (WER). Most prominently, variations of the Conformer, named Branchformer [5] and E-Branchformer [6] reached the lowest WER on the widely-adopted LibriSpeech dataset [7] while being trained from scratch without external data. Following the local and global dependencies' assumption, Branchformer architecture physically create two branches per block (a dual path) in the architecture to capture independently and with adapted mechanisms (i.e., MHSA and CNN) both levels of dependencies. The latter branches are then merged and passed to the next architecture block. Such approaches are agnostic to the type of ASR decoding or processing, e.g., Transducers [8], CTC only [9], or CTC and attention [10]. However, they suffer from a major and well-documented efficiency issue as MHSA exhibits a quadratic complexity and memory time-dependency [11]. For instance, the MHSA block is among the most computationally demanding elements of any Transformer model. This is especially true for speech processing as input sequences are often long by nature e.g., longer than 30 seconds for a few LibriSpeech utterances [12, 13]. In addition, large-scale and Transformer-based Self-Supervised Learning (SSL) models for speech recognition are commonly trained with sentences voluntarily cropped at 20 to 25 seconds. The latter transformation is necessary to enable training with top-tier GPU e.g., Tesla V100 or A100 [14], also making it potentially intractable to train on more accessible compute infrastructures. This article focuses on retaining MHSA's global interactions capabili-

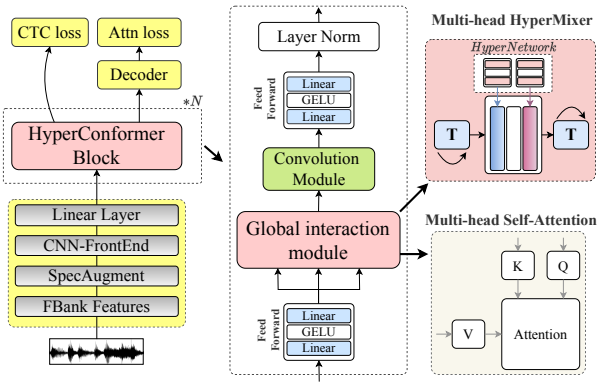


Figure 1: *Layout of the general Conformer architecture. Global interactions can be modeled either with attention leading to a* Conformer *or with HyperMixer to obtain* HyperConformer. ***T*** *represents the transpose operation. Skip connections are omitted for simplicity. The global interaction module is combined sequentially with a convolution module to capture local dependencies, critical for speech-related tasks.*

⋆Equal contribution. Order is determined by a coin flip.

Research supported by the Swiss National Science Foundation (*LAOS*, grant 200021-178862) and EU-H2020 (*CRiTERIA*, grant 101021866).

[1]The HyperConformer recipe is publicly available in: https://github.com/speechbrain/speechbrain/tree/develop/recipes/LibriSpeech/ASR/transformer/

ties beneficial to ASR while lowering significantly its computational and memory cost.

How to efficiently compute interactions between tokens in Transformer-like architectures is an active area of research [15]. Most works try to decrease the cost of attention directly, e.g., through a low-rank approximation [16], linearization [17], or the introduction of sparse attention patterns [18]. However, token mixing can also be achieved from outside the framework of attention, opening up considerably novel opportunities for improvement. MLPMixer [19] was the first to learn a fixed-size MLP for modeling global interactions, with many to follow in the vision domain [20, 21, 22]. However, the fixed size hinders their adoption for domains with variable length signals. Existing approaches for speech have strong locality biases [23, 24] and still rely on small attention modules for the best performance [24]. Recently, [11] proposed HyperMixer for text processing, which achieved competitive performance to attention at a substantially lower cost in terms of computation and data. Intuitively, HyperMixer constructs the token-mixing MLP of MLPMixer *dynamically as a function of the data*, hence being amenable to variable length inputs.

This article introduces HyperConformer, a novel and simple-to-implement alternative to MHSA. It benefits from the linear time and memory complexity of HyperMixer while capturing both global and local dependencies from the speech signal necessary for ASR. The contributions are threefold. First, we formally describe HyperConformer and its main components (§2). Then, we introduce *multi-head token mixing* to HyperMixer and HyperConformer to further improve the efficiency of both models. Finally, we open-source a training and inference ASR recipe within the widely adopted SpeechBrain toolkit [25]. Experiments are conducted in a relatively resource-constrained scenario with limited VRAM and neural parameters budgets to highlight the efficiency aspect of each evaluated model. Throughout the conducted ASR experiments on the LibriSpeech dataset (§3), HyperConformer consistently reaches the state-of-the-art Conformer baseline in terms of WER. In addition, HyperConformer shows between 37% and 56% reduction in processing time on mid-length and long speech, respectively. During training, it uses up to 30% less memory, hence being trainable on GPUs from the Ti 70 family. Overall, HyperConformer offers a more accessible alternative to any ASR system previously based on Conformer models.

2. HyperConformer

Figure 1 illustrates the different blocks of the introduced HyperConformer. It consists of four parts: Two feature mixing layers (feed-forward networks) at the bottom and top of the layer, a module for modeling local interactions, specifically the Convolution module introduced in [4], and a global interaction module. In the following, we discuss the global interaction modules bringing token mixing to the model. Other components of HyperConformer are identical to the Conformer [4].

2.1. Capturing Global Interactions

Let $X \in \mathbb{R}^{N \times d}$ represent N d-dimensional token vectors, also equivalent to a latent representation of speech coming from the previous layer on length N. The global interaction module $\mathrm{GI} : \mathbb{R}^{N \times d} \to \mathbb{R}^{N \times d}; X \mapsto X'$ is responsible for combining information from different tokens in such a way that every $X'_{:,j}$ contains information from every $X_{:,i}$. Such a behavior captures global interactions as it interconnects the different time steps of the given speech or latent sequence. This may be achieved, for instance, via multi-head attention or via HyperMixer.

2.2. Multi-Head Self-Attention

At the core, Multi-Head Self-Attention (MHSA) [3] relies on scaled dot-product attention:

$$\text{Attention}(X) = \text{Softmax}(\frac{XX^T}{\sqrt{d_k}})X,$$

which involves computing the dot product between every pair of input tokens, invoking memory and runtime complexity of $\mathcal{O}(N^2 \cdot d)$. The latter is responsible for the quadratic increase in memory and time consumption of standard Transformer architectures [11]. Further modeling capabilities are commonly obtained with the introduction of k parallel heads, allowing the model to attend to information from different representation subspaces, i.e., different views of the data:

$$\text{MHSA}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_k)W^O,$$
$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V),$$

with W^O, W_i^Q, W_i^K, W_i^V learnable weight parameters.

2.3. HyperMixer

From a high-level perspective, HyperMixer achieves token mixing over variable length sequences by dynamically constructing a *token mixing MLP* through the use of hypernetworks [26]. The latter models specialize in generating neural network parameters, e.g., weights and biases. A token-mixing MLP is a multilayer perceptron $\text{TM-MLP} : \mathbb{R}^{d \times N} \to \mathbb{R}^{d \times N}$ that combines information from different tokens *for each feature independently*, e.g., processing the Fbank coefficients of each time step of a sequence:

$$\text{TM-MLP}(X)_{i,:} = \text{LayerNorm}(W_1(\sigma(W_2^T X_{i,:}^T))), \quad (1)$$

where $W_1, W_2 \in \mathbb{R}^{N \times d'}$ are weight matrices with the hidden layer size d'. σ represents some non-linear activation function; we fix it to GELU [27] following [11]. Furthermore, we add layer normalization [28] for improved stability. Intuitively, the input layer W_1 decides to what degree each token's information should be sent to the hidden layer of TM-MLP, and the output layer W_2 decides for each token what information to extract from the hidden layer.

Importantly, W_1, W_2 themselves are not learnable parameters, which would require the input to be of the same fixed size at all times. Instead, $\text{HyperMixer}(X; d, d')$, parameterized through the embedding dimension d and the hidden layer size d', first dynamically generates W_1, W_2 from the inputs themselves with the two hypernetworks $\text{MLP}^1, \text{MLP}^2$:

$$W_k(X) = \begin{pmatrix} \text{MLP}^{\text{k}}(X_{:,1} + p_{:,1}) \\ \vdots \\ \text{MLP}^{\text{k}}(X_{:,N} + p_{:,N}) \end{pmatrix} \in \mathbb{R}^{N \times d'}, k \in 1, 2.$$

$\text{MLP}^1, \text{MLP}^2 : \mathbb{R}^d \to \mathbb{R}^{d'}$ contain the learnable parameters of HyperMixer, and $p_{:,j}$ are absolute position embeddings from standards Transformers [3]. After generating the weights, Equation 1 is applied. This determines the complexity of this model: $\mathcal{O}(N \cdot d \cdot d')$, which is the same asymptotic runtime as the feature mixing layers. Hence, HyperMixer turns the quadratic memory and inference time complexities to a linear regime.

2.4. Multi-Head HyperMixer

Analogously to MHSA, we propose an extension of HyperMixer to multi-head HyperMixer (MHHM) and HyperConformer, by introducing multiple token mixing heads. To this end, we create k parallel $\text{HyperMixer}^l(\cdot; d/k, d'/k), l \in 0..k-1$, which each operates on (d/k)-dimensional feature subsets of X, whose outputs are again concatenated:

$$\text{head}_l = \text{HyperMixer}^l(X_{:,(l\cdot(d/k)):(l+1\cdot(d/k))})$$
$$\text{MHHM}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_k)$$

As a result, and conversely to MHSA, the runtime complexity even further reduces to $\mathcal{O}(k\cdot(N\cdot(d/k)\cdot(d'/k))) = \mathcal{O}(\frac{N\cdot d\cdot d'}{k})$.

3. Experiments

This section details the experimental setup (3.1) used to evaluate HyperConformer against three baselines including the state-of-the-art Conformer. Models are compared both in terms of ASR performance (Section 3.2.1) and efficiency metrics (Section 3.2.2).

3.1. Experimental Setup

Our experiments aim at assessing the effectiveness and efficiency of HyperConformer in comparison to Conformer. Hence, we compare vanilla Transformer [3] and Conformer [4] models to HyperMixer and HyperConformer. In practice, we swap the global interaction module, i.e., attention, from `regularMHA` (which uses absolute position embeddings [3]) of Transformer and `RelPosMHAXL` (which uses relative position embeddings [29]) of Conformer to our multi-head `HyperMixer` implementation.

Datasets and Decoding. We validate HyperConformer, on the LibriSpeech dataset [7]. It is composed of ∼960h of transcribed speech in English. We perform ablations either training on the 100h set or the full, 960h set, and report results on the dev/test sets and clean/other partitions. Additionally, we use the text-only corpus for external language modeling (LM).[2] The LM is a Transformer based [3] only-encoder model composed of 12 encoder layers, $d_{ffn} = 3072$ and $d_{model} = 768$, which accounts for 93.3M parameters. Word error rates are reported using beam search with and without LM shallow fusion.

Neural Architectures. To gain a comprehensive understanding of performance and primary trade-offs, we ablate four different architectures in an encoder-decoder style: i) vanilla Transformer, ii) Conformer, iii) HyperMixer, and iv) HyperConformer. For the efficiency analysis only we also experiment with replacing `RelPosMHAXL` with `regularMHA` (Conformer-regular). All models use a 5K BPE sub-word unit [30] vocabulary. This remains consistent across all experiments and models. At the bottom of the encoder, we incorporated a front-end module consisting of a 2-layer CNN that receives 80-dim log Mel filterbank features. We use SpecAugment [31] during training with the default configuration in SpeechBrain. To correspond to accessible hardware as well as to emphasize low-compute resources performance, all models are conceived within a 25M parameter budget and trained with an 11GB memory constraint, corresponding to accessible GPU such as the Ti 80 family (or Ti 70 for the small version of HyperConformer).

[2]Pre-trained LM from SpeechBrain available in: `huggingface.co/speechbrain/asr-conformersmall-transformerlm-librispeech`.

Table 1: *Word error rates [%] on the official LibriSpeech dev and test sets for models trained on 960h LibriSpeech set. The results include the four proposed encoder models, including our novel architecture, HyperConformer. We ablate two different model sizes for each architecture and list results with and without LM. The last column list the peak memory consumption [GB] of each architecture under the same training conditions.*

Model	Par.	WER w/o LM dev		WER w/o LM test		WER w/ LM test		Peak Mem.
	[M]	clean	other	clean	other	clean	other	*[GB]*
Small sized models ($d_{model} = 144$)								
Transformer	6.1	7.7	15.6	7.8	15.8	3.9	8.2	6.45
HyperMixer	5.6	12.9	23.1	13.1	23.4	5.8	12.6	4.04
Conformer	8.7	4.7	11.4	5.0	11.3	3.1	6.8	8.18
HyperConformer	7.9	5.0	12.1	5.3	12.3	2.9	7.0	5.67
Medium sized models ($d_{model} = 256$)								
Transformer	16.2	4.6	10.7	4.7	10.9	2.7	6.1	7.6
HyperMixer	14.4	7.2	15.2	7.5	15.2	3.9	8.3	5.6
Conformer	24.1	3.6	8.8	3.8	8.7	2.6	5.9	10.7
HyperConformer	21.7	3.4	9.0	3.6	9.0	2.3	5.7	8.6

Hence, we select two model sizes for each architecture, i.e., 8 different scenarios. We use the same configuration, 10 encoder layers, and 8 attention or HyperMixing heads. However, we set $d_{model} = \{144, 256\}$ for `{base,medium}` models, respectively. The feed-forward network dimensions is set to $d_{ffn} = 4 \cdot d_{model}$ for all cases. For simplicity, we set the hidden layer size d' of TM-MLP to $d' = d_{ffn}$. We leave an exploration of this hyperparameter to future work.

Training Hyperparameters. Training is performed by combining the per-frame transformer decoder output probabilities and CTC [2]. The CTC loss [9] is weighted by $\alpha = 0.3$ during training. All the models use the same decoder, i.e., 4 Transformer layers. We follow the default training configuration of the LibriSpeech recipe from SpeechBrain.[3] It uses Adam [32] optimizer, learning rate (lr=$1e^{-3}$) scheduler with warmup [3] (25k steps warmup). We train for 110 epochs, i.e., ∼660k steps when full LibriSpeech and ∼70k when LibriSpeech 100h set. The recipe also uses dynamic batching, which reduces the overall training time. At decoding time, we use a beam size of 66 with a CTC weight of $ctc_w = 0.4$. All of our experiments can be run on accessible GPUs starting from the Ti 70 family.

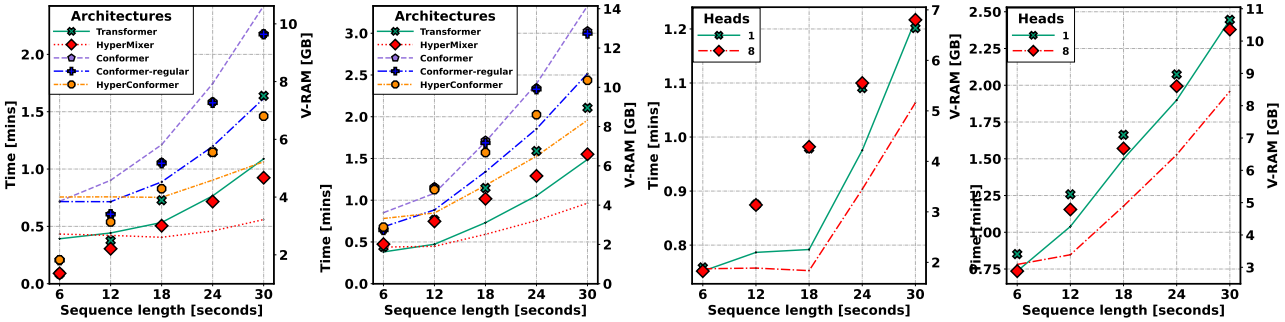
3.2. Results and Discussion

Our experiments are designed to answer two questions: 1) Does HyperConformer perform competitive to Conformer in terms of word error rates? 2) Is HyperConformer more efficient than Conformer?

3.2.1. Speech Recognition Results

We compare WERs of different state-of-the-art architectures for ASR, listing the results on Table 1. We find that HyperMixer alone achieves acceptable performance, especially in combination with a language model, but trails behind Transformer and Conformer, in all cases. We hypothesize that this is because the crucial local information in speech signals is difficult to pass through the hidden layer bottleneck of TM-MLP, which attention does not have. In contrast, HyperConformer performs comparable and often even better than Conformer in the medium-sized configuration. For instance, HyperConformer

[3]Please refer to the SpeechBrain recipe located in `recipes/LibriSpeech/ASR/transformer`.



(a) *Forward pass of small (left) and medium sized (right) models.*

(b) *Forward pass of 1 and 8 heads for HyperConformer.*

Figure 2: *Overall time (minutes) and GPU consumption (GB) required by different architectures for sequences of different lengths. The left plot of (a) and (b) shows the small model, the right plot shows the mid-size model. Each sequence length in the x-axis represents 1000 samples from the LibriSpeech dataset. For all plots: Lines denotes time (left y-axis) and markers GPU consumption (right y-axis). Batch size is 16 for all configurations.*

beats Conformer by 0.17% absolute WER on test-other with LM for the medium-sized model. We explain this as follows: In HyperConformer, i) the convolution module helps to model the local interactions between tokens, and ii) global interactions can be modeled in and passed through the multi-head HyperMixer's bottleneck effectively. Finally, we note that HyperConformer is amenable to scale, since moving from 7.9M $\rightarrow$ 21.7M, we obtain a 17.9% relative reduction in WER on test-other with LM, similar to Conformer.

3.2.2. Efficiency Analysis

In [11] is shown that HyperMixer has efficiency benefits regarding processing speed and training data size. Here, we investigate if these properties also transfer to the speech domain, particularly, HyperConformer on the ASR task.

Peak memory consumption The right hand side of Table 1 shows the peak memory consumption when training models of the same size on the same hardware. We observe that HyperConformer requires substantially less memory than Conformer (-30.6% with small size and -19.7% with medium size). The effect is stronger on small models than on large ones. Since larger models are wider (i.e., larger d and d'), the feature mixing components as well as TM-MLP require considerably more compute in comparison to attention, whose complexity depends primarily on the sequence length, which remains the same between training scenarios.

Resource consumption depending on sequence length The main advantage of HyperMixer is its linear complexity compared to attention's quadratic complexity. To investigate this property, we measure the peak memory and processing time of the encoder as a function of the length of the speech sample. To this end, we synthesize 1,000 sentences of 6, 12, 18, 24, and 30 seconds each by concatenating multiple signals from the LibriSpeech dataset. Figure 2a shows the resource consumption of all models. While HyperConformer and Conformer require similar processing time at short sequences, HyperConformer is considerably faster at mid-length (18s, small: 37.9%, mid: 15.2%) and long sequences (30s, small 56.1%, mid: 34.2%), demonstrating its better asymptotic complexity compared to Conformer. Note that Conformer with `regularMHA` is more efficient than `RelPosMHAXL`. However, this would lead to a performance loss [4], and HyperConformer is still substantially more efficient.

Number of heads An important technical novelties is the introduction of multi-head HyperMixer, which allows for multiple parallel views on the data analogous to multi-head attention, while at the same time reducing the model's complexity. In preliminary experiments, we found that HyperConformer with $k = 8$ heads performs as well as with $k = 1$ head. At the same time, moving from a single head to 8 heads reduces the number of parameters in the model by 7.1% in the small model and 20.8% in the mid-size model. Moreover, as Figure 2b shows, the processing time is reduced substantially by up to 12.6% (small) and 19.9% (mid-size) on the longest sequences.

Low-resource scenario HyperMixer is reported to work better than MHSA in the low-resource scenario [11]. Here, we conduct an initial experiment to test whether HyperConformer inhibits the same characteristic. To this end, we compare HyperConformer to Conformer on the 100h LibriSpeech subset, which is 10 times smaller than the full dataset. All other training parameters remain the same. Table 2 shows the results. In this scenario, HyperConformer performs around 20% better than Conformer, suggesting better data efficiency.

Table 2: *Performance of Conformer and HyperConformer when trained on 100h LibriSpeech (10$\times$ less data). Percentage in brackets shows relative WER reduction on test-other with LM.*

Model	Small size	Medium size
Conformer	8.29	7.57
HyperConformer	6.76 (-18.5%)	5.80 (-23.4%)

4. Conclusion

HyperConformer is a new architecture for efficient ASR introduced in this work. It integrates the benefits of the Convolution module from Conformer, which models local interactions, and the hypernetwork-based architecture, HyperMixer, which models global interactions. We were able to attain comparable or lower WERs (2.28/5.42 in test clean/other) HyperConformer when compared to Conformer. In addition, this novel architecture is substantially faster on long sequences, while also requiring less GPU memory during training. We believe HyperConformer is a green alternative to previous established Transformer and Conformer based models for ASR.

5. References

[1] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE access*, vol. 7, pp. 19 143–19 165, 2019.

[2] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2019, pp. 449–456.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 5036–5040. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-3015

[5] Y. Peng, S. Dalmia, I. Lane, and S. Watanabe, "Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 17 627–17 643. [Online]. Available: https://proceedings.mlr.press/v162/peng22a.html

[6] K. Kim, F. Wu, Y. Peng, J. Pan, P. Sridhar, K. J. Han, and S. Watanabe, "E-branchformer: Branchformer with enhanced merging for speech recognition," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 84–91.

[7] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[8] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[9] A. Graves and A. Graves, "Connectionist temporal classification," *Supervised sequence labelling with recurrent neural networks*, pp. 61–93, 2012.

[10] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[11] F. Mai, A. Pannatier, F. Fehr, H. Chen, F. Marelli, F. Fleuret, and J. Henderson, "Hypermixer: An mlp-based low cost alternative to transformers," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

[12] Y. Gao, J. Fernandez-Marques, T. Parcollet, A. Mehrotra, and N. Lane, "Federated Self-supervised Speech Representations: Are We There Yet?" in *Proc. Interspeech 2022*, 2022, pp. 3809–3813.

[13] T. Parcollet and M. Ravanelli, "The Energy and Carbon Footprint of Training End-to-End Speech Recognizers," in *Proc. Interspeech 2021*, 2021, pp. 4583–4587.

[14] A. Babu, C. Wang, A. Tjandra, K. Lakhotia, Q. Xu, N. Goyal, K. Singh, P. von Platen, Y. Saraf, J. Pino *et al.*, "Xls-r: Self-supervised cross-lingual speech representation learning at scale," *arXiv preprint arXiv:2111.09296*, 2021.

[15] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, 2022.

[16] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[17] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[18] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[19] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021.

[20] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay attention to mlps," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9204–9215, 2021.

[21] S. Chen, E. Xie, C. Ge, D. Liang, and P. Luo, "Cyclemlp: A mlp-like architecture for dense prediction," *arXiv preprint arXiv:2107.10224*, 2021.

[22] Z. Wang, W. Jiang, Y. M. Zhu, L. Yuan, Y. Song, and W. Liu, "Dynamixer: a vision mlp architecture with dynamic mixing," in *International Conference on Machine Learning*. PMLR, 2022, pp. 22 691–22 701.

[23] C. Xing, D. Wang, L. Dai, Q. Liu, and A. Avila, "Speech-MLP: a simple MLP architecture for speech processing," 2022. [Online]. Available: https://openreview.net/forum?id=-u8EliRNW8k

[24] J. Sakuma, T. Komatsu, and R. Scheibler, "Mlp-asr: Sequence-length agnostic all-mlp architectures for speech recognition," *arXiv preprint arXiv:2202.08456*, 2022.

[25] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[26] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rkpACe1lx

[27] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[28] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[29] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://aclanthology.org/P19-1285

[30] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 1715–1725.

[31] 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 *Proc. Interspeech 2019*, 2019, pp. 2613–2617. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2019-2680

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.