Revisiting Convolution-free Transformer for Speech Recognition

Zejiang Hou, Goeric Huybrechts, Anshu Bhatia, Daniel Garcia-Romero, Kyu J. Han, Katrin Kirchhoff

AWS AI Labs, USA

{zejiangh,huybrech,anshubha,dgromero,kyujhan,katrinki}@amazon.com

Abstract

Convolution augmented Transformer architectures have dominated the field of automatic speech recognition by showing better WER results when the models are trained on relatively smaller training data. In this work, we revisit the necessity of convolution modules in the ASR encoder architecture, given that the inductive bias brought by the convolution modules may only boost performance in a low training data regime. We show that with architectural improvements to the Transformer block, a convolution-free Transformer architecture (namely, Transformer++) can catch up with the best Conformer WER results as we scale up the training data. Moreover, we demonstrate that with large scale unsupervised pre-training, the proposed Transformer++ can achieve even better WER than the best Conformer results. Importantly, Transformer++ achieves state-of-the-art performance with top efficiency, where we show 40% CPU inference real-time factor (RTF) improvement and 25% GPU training speedup compared to Conformer.

Index Terms: speech recognition, speech encoder architecture, transformer, self-supervised learning.

1. Introduction

Deep neural networks have demonstrated remarkable improvement on end-to-end speech recognition (ASR). Inspired by the scaling success of Transformer [1] on natural language processing tasks, recent ASR efforts have been put on designing speech Transformers [2, 3, 4, 5, 6]. In particular, the hybrid attention-convolution architectures [7, 8, 9, 10] have received great attention, because of their ability to capture global and local features synchronously from the audio signals. The convolution augmented Transformer architecture (Conformer) [8] has become a dominant choice for both ASR and other speech processing tasks. Moreover, when combining with the recent advancement of self-supervised learning methods [11, 12, 13], Conformer has become the current state-of-the-art.

Albeit the empirical success of Conformer, the necessity of having convolution operations, especially when we train ASR models on larger scale datasets, lacks deeper analysis. The inductive biases inherent to the convolutions, such as translation equivariance and locality, help generalization when we train on relatively smaller amount of data. This explains the WER gains from Conformer when the model is trained on the 970 hours of the LibriSpeech dataset [8]. However, the benefit of having convolution modules diminishes as we move to larger datasets where the inductive biases can be trumped by large scale training, which is already seen from the vision and text domains [14]. Moreover, the use of large kernels and depth-wise convolutions raise concerns about inference efficiency. Our PyTorch profiler analysis shows that convolution operations (including both the convolutional down-sampling layers at the front and the convolutions in the encoder) are the major bottleneck in Conformer when we run inference on CPU.

In this paper, we first present systematic comparisons between Transformer and Conformer under different training data scales including 2.5k, 15k, and 96k hours of supervised data to understand the necessity of convolutions in the ASR encoder. Then, we introduce a simpler convolution-free Transformer architecture (Transformer++) that achieves the state-of-the-art ASR performance while offering significantly better training and inference efficiency. Transformer++ does not have any convolutions in the encoder blocks, and we use a simple frame stacking method for down-sampling at the front. The contributions of this paper are highlighted as follows:

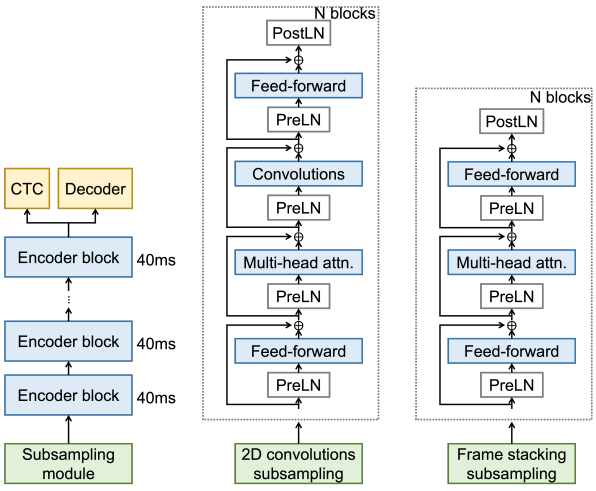
- We show that the gains from using convolutions in ASR diminishes as the training data and model size scale up.
- We propose a new speech model Transformer++, which outperforms vanilla Transformer by 12% relative in WER, and shows marginal difference with the best Conformer results.
- Combined with self-supervised pre-training, Transformer++ closes the WER gap compared to Conformer even when fine-tuning on the small-scale supervised data, and outperforms Conformer when fine-tuning on larger supervised data.
- Transformer++ demonstrates 40% CPU inference RTF improvement, and 25% training speedup in GPU hours.

2. Methodology

As shown in Figure 1(b), the Conformer incorporates convolution down-sampling at the front, followed by a number of blocks comprised of four modules in each. Conformer introduces the usage of convolution modules and the Macaron-style [15] feed-forward modules. In this work, we carefully revisit the design choices in Conformer from both macro and micro perspectives, and propose a convolution-free Transformer architecture (Transformer++) as shown in Figure 1(c). Compared to vanilla Transformer, Transformer++ has frame stacking based down-sampling, Macaron structure in the convolution-free encoder, FlashAttention, rotary positional encoding, and SwiGLU FFN.

2.1. Macro design

While the down-sampling module is usually overlooked in ASR architectural design, it counts for a significant portion of model computation. We find that replacing the 2D convolutional down-sampling in Conformer by a simple frame stacking module can improve the RTF by 30% with minimal WER impact. More specifically, to down-sample the frame rate at the front with a down-sampling factor of r and input feature dimension of d, the frame stacking module stacks consecutive r frames and concatenate them into $r \cdot d$ dimension. Then, a simple lin-



(a) General ASR encoder architecture (b) Conformer model (c) Proposed Conv. free Transformer

Figure 1: *Comparison between the Conformer architecture (b) and the proposed Transformer++ architecture (c).*

ear layer is used to project the concatenated features into the feature dimension of the encoder.

Inspired by Conformer using the Macaron-Net [15], we also adopt the Macaron structure in the encoder blocks. The Macaron style FFN layers sandwiching the self-attention module brings in improvements compared to the vanilla Transformer having single FFN in each block and an equivalent total parameter count. We discard the convolution modules but only retain the two feed-forward modules and the self-attention module. In summary, for input x_i to the i-th block, the output of the block in Transformer++ is produced by:

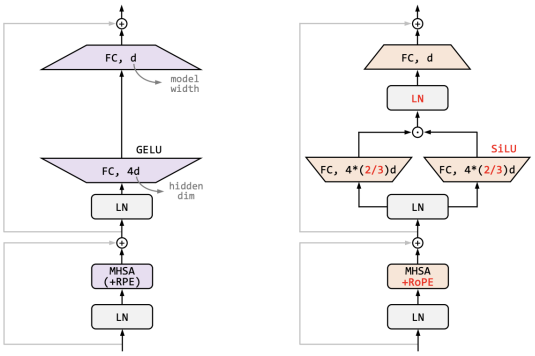
$$\begin{aligned} \tilde{x}_i &= x_i + \frac{1}{2}\text{FFN}(x_i), \\ \hat{x}_i &= \tilde{x}_i + \text{MHSA}(\tilde{x}_i), \\ y_i &= \text{LayerNorm}\big(\hat{x}_1 + \frac{1}{2}\text{FFN}(\hat{x}_i)\big). \end{aligned} \tag{1}$$

2.2. Micro design

Self-attention modules have been the mainstay for the success of Transformer, but the quadratic complexity with regard to sequence length makes it slow, especially on long sequences. Recent proposal of FlashAttention [16] is an IO-aware exact attention algorithm that uses tiling to reduce the access amount to the GPU high bandwidth memory. This makes the self-attention modules run faster, where we observe 20% training speedup from using FlashAttention without WER regression. Due to this training time saving, we apply FlashAttention to all our experiments including both Conformer and Transformer++.

Differently from Conformer which uses the relative sinusoidal positional encoding, we apply the rotary position embedding (RoPE) [17]. RoPE encodes the absolute position with a rotation matrix and incorporates the explicit relative position dependency in self-attention formulation. More specifically, the relative position information is incorporated into the token embedding vector by rotating the affine-transformed embedding vector, where the rotation angle is a pre-defined constant multiplied by the token's position index.

We adopt the pre-norm residual units [18] with dropout [19] for the feed-forward and the self-attention modules. We apply



(a) *Plain FFN.* (b) *SwiGLU+sub-LN FFN.*

Figure 2: *An illustration of the plain FFN and SwiGLU+sub-LN in Transformer++.*

Model	Vanilla Tran	Conformer		Transformer++	
Parameters (M)	112	136	356	112	344
Encoder layers	32	20	24	20	28
Encoder dim.	512	512	768	512	768
Attention heads	8	8	8	8	8
Conv. kernel	-	31	31	-	-
Decoder layers	1	1	1	1	1
Decoder dim.	512	512	768	512	768

Table 1: *Model hyper-parameters for Vanilla Transformer, Conformer, and our proposed Transformer++ models.*

the layer normalization [20] within the residual unit and on the input side. The feed-forward modules are composed of two linear layers and a Swish activation [21] in between. We use the default expansion ratio of $4\times$. We also apply weight scaling to the second linear layer of the feed-forward module. The scaling factor is derived by $1/\sqrt{2L}$, where L is the total number of blocks in the model.

2.3. Other architectural improvements

Although the main focus of this paper is to design a convolution-free ASR model, we notice that recent architectural advances in large language models have not yet been explored in the context of speech representation learning. Therefore, we also consider enhancing the proposed Transformer++ model by integrating gated linear unit with Swish activation (SwiGLU) [22] as the feed-forward module and sub-LN [23] as the normalization layer.

More specifically, SwiGLU is defined as the element-wise product between two linear transformations of the same input, one of which is Swish activated:

$$\text{SwiGLU}(x) = \text{Swish}(W^1 x + b^1) \odot (W^2 x + b^2). \tag{2}$$

The sub-LN method introduces another LayerNorm layer inside each module (including self-attention and feed-forward module) apart from the pre-norm layer. Figure 2 compares an improved FFN module with SwiGLU and sub-LN versus the plain FFN. To keep the number of parameters and FLOPs consistent, the hidden feature dimension of SwiGLU based FFN is $2/3\times$ of the plain FFN counterpart.

3. Experiments

3.1. Data

For supervised training, we prepare a 96k hours English dataset, comprised of various public domain and internal datasets with

Datasets	Train from scratch									Pre-train + fine-tune			
	2.5khrs			15khrs			96khrs			2.5khrs		96khrs	
	Tran	Tran++	Con	Tran	Tran++	Con	Tran	Tran++	Con	Tran++	Con	Tran++	Con
Public													
artie	15.7	12.4	11.5	8.7	6.7	6.3	7.4	5.8	4.9	9.2	8.5	4.1	3.8
commonvoice	21.6	17.7	16.7	13.7	11.7	11.2	11.8	10.1	9.1	13.7	12.9	8.0	7.9
voxpopuli	12.4	10.2	9.9	8.4	7.3	7.2	7.8	6.7	6.6	8.7	8.7	6.3	6.5
eval2000-swbd	9.1	7.4	7.0	6.4	5.5	5.4	5.5	5.6	5.2	6.0	6.2	5.0	5.4
eval2000-callhm	14.7	12.3	12.2	10.5	9.4	9.4	9.5	8.8	8.4	9.9	10.2	8.2	8.5
libri-speech-test-clean	8.2	6.0	5.7	3.8	3.1	2.9	2.9	2.3	2.2	4.5	4.3	1.8	2.0
librispeech-test-other	15.2	11.8	10.8	8.5	6.7	6.2	6.4	5.1	4.5	8.5	7.9	3.9	4.0
MSLT	10.8	9.2	9.0	8.4	7.8	7.8	8.1	7.5	7.4	8.3	8.1	7.2	7.1
CHiME5	33.1	29.4	28.7	25.2	22.9	22.5	26.8	20.5	19.9	22.5	21.8	17.5	19.4
People's speech	25.7	23.2	22.4	20.8	20.0	19.3	20.1	19.0	18.8	20.7	20.2	18.6	19.1
NCHLT Sadilar	21.5	15.7	15.2	13.1	10.7	10.0	11.3	10.0	10.0	13.1	12.8	8.5	8.6
EDACC	24.5	21.2	20.1	18.5	16.6	16.8	16.9	15.8	15.7	19.1	19.0	15.2	14.8
Voices (8k)	39.2	34.7	32.4	29.5	26.4	24.6	21.9	18.5	18.4	27.9	25.6	16.3	16.9
Voices (16k)	32.2	27.5	25.7	22.5	20.0	19.1	17.4	14.2	13.6	20.8	20.2	12.0	12.5
CORAAL	23.1	20.2	20.1	19.0	16.7	17.1	15.4	14.7	15.2	17.9	19.3	14.3	15.4
GMU 16k	14.7	11.0	9.7	8.0	6.1	5.4	6.4	4.9	4.7	7.4	6.7	4.0	4.1
Internal	18.9	15.8	15.3	13.8	12.1	11.8	11.6	10.2	9.9	12.9	12.7	9.4	9.5
Avg. WER	19.4	16.3	15.6	13.9	12.2	11.8	11.9	10.4	10.1	13.2	13.0	9.4	9.6

Table 2: *WER (%) comparisons on public and internal benchmarks between Transformer++ (Tran++) and other models including vanilla Transformer (Tran) and the state-of-the-art Conformer (Con). The competitors's results are based on our own reproduction to their best performance as possible. The internal benchmark results are aggregated from multiple internal datasets. We use 100M models for train-from-scratch, and 300M models for pre-train+fine-tune. In the train-from-scratch setup, Transformer++ shows marginal WER difference compared to Conformer as the training data scales up. Pre-training improves the WER noticeably and closes the WER gap.*

offline augmentations. To fully understand the gap between Conformer and Transformer under different training data scales, we further sub-sample two smaller training datasets: 2.5k hours and 15k hours datasets, both of which do not have any offline augmentations. Apart from supervised training, we also consider self-supervised pre-training on an internal dataset containing 300k hours of unlabeled English audios before fine-tuning the models on the supervised datasets. We conduct exhaustive comparisons between Transformer++ against vanilla Transformer and Conformer models on a variety of public and internal benchmarks under different acoustic conditions.

3.2. Training setup

We consider two model sizes, 100M and 300M parameters. Detailed configurations of Conformer, vanilla Transformer, and Transformer++ are presented in Table 1. We train the models with a joint CTC/Attention objective [24]. We use a single-layer Transformer decoder during training and discard it in inference. For regularization, we apply intermediate CTC [25] losses every six layers, a label smoothing factor of 0.1 on the attention decoder loss, a weight decay of 0.1, dropout of 0.1, layer dropping probability of 0.05. We train the models with the Apex AdamW [26] optimizer with $\beta_1 = 0.9, \beta_2 = 0.98$, and $\epsilon = 1 \times e^{-6}$. An exponential decay learning rate is adapted, with about 2 or 3 epochs of warmup and the peak learning rate of 0.005. We use sentencepiece model with a vocabulary size of 2048 for English training, and 4096 for the multilingual results.

3.3. Results

The WER results for all the datasets (no dataset specific parameters) are obtained using a CTC decoder with beamsearch (beam size 50) and shallow fusion with 4-gram LM trained on the transcripts of the training set. The LM weight is fixed at 0.6 for all experiments. Table 2 compares the 112M parameters Transformer++ model against the 112M parameters vanilla Transformer and the 136M parameters Conformer under three training data scales in the train-from-scratch setup. Our model outperforms the vanilla Transformer by 12 to 16% relative WER. Compared with Conformer, we observe that the gap between two models diminish as we increase the amount of training data. The absolute difference of the averaged WER shrinks to a marginal value of 0.3 when training on the largest 96k hours data. This suggests that the inductive bias and locality property of convolutions can be trumped by larger scale training. Next, We show how to close the gap between Transformer++ and Conformer by self-supervised pre-training.

3.4. Self-supervised pre-training

Recent advances of self-supervised pre-training further push the limits of ASR models. We study the impact of large scale unlabeled pre-training on Transformer++ performance. We adopt the BEST-RQ [12] method, which pre-trains the model to predict the masked speech signals with labels generated from a random-projection quantizer. The quantizer projects the speech inputs with a randomly initialized matrix, and performs a nearest-neighbor lookup in a randomly-initialized codebook. Neither the projection matrix nor the codebook is updated during pre-training. We pre-train the 344M Transformer++ and the 356M Conformer on 300k hours English audios for 500K steps.

After pre-training, we fine-tune the model on the 2.5k/96k hours data. Results are shown in Table 2. We observe that pre-training improves the train-from-scratch results significantly when we only have 2.5k hours supervised data, with more than 15% relative WER gains on both Transformer++ and Conformer. Moreover, we observe the gap between the two models gets further reduced thanks to the pre-training stage. When we fine-tune on the 96k hours dataset, Transformer++ even achieves a better WER of 9.4 compared to Conformer's 9.6.

3.5. Multilingual training

Apart from the monolingual results, we also evaluate the multilingual capability of Transformer++. For training, we prepare

Model	Params.	Languages					Avg.	RTF(↓)
		de	es	fr	it	pt		
Conformer	300M	10.6	6.6	9.6	6.1	9.8	8.5	0.271
Transformer++	300M	10.2	6.8	9.5	6.2	10.3	8.6	0.187

Table 3: *WER (%) comparisons on the multilingual FLEURS datasets between Transformer++ and Conformer models.*

Model	∼100M		∼300M	
	Inf. RTF	Train time	Inf. RTF	Train time
Conformer	0.119	102hrs	0.269	107hrs
Transformer++	0.068 ↓ 42%	76hrs ↓ 25%	0.183 ↓ 32%	77hrs ↓ 28%

Table 4: *Comparison of CPU inference RTF and GPU training time between Transformer++ and Conformer models under different model sizes.*

Model	Params.	RTF(↓)	Avg. WER
Conformer	136M	0.119	10.1
Transformer++			
Plain FFN	112M	0.068	10.4
SwiGLU+sub-LN FFN	112M	0.069	10.0

Table 5: *Ablation study of using SwiGLU and sub-LN based FFN modules in Transformer++.*

Model	RTF(↓)	Avg. WER
Conformer	0.119	10.1
−Conv in encoder blocks	0.102	10.2
−Conv downsampling	0.068	10.4

Table 6: *Ablation study on the removal of different convolution layers in terms of WER and RTF.*

a 35k hours supervised dataset consisting of five European languages: Germen (de), Spanish (es), French (fr), Italian (it), and Portuguese (pt). For evaluation, we use the testing splits of the FLEURS datasets [27]. To embed language information into the model, we add an embedding layer at the front to receive the language ID, and produces a language encoding vector having the same feature dimension as the encoder. This language encoding vector is appended to the sequence as a prefix frame after the frame stacking module in Transformer++. In decoding, we train a multilingual subword 4-gram LM on the training transcripts of the 35k hours data. The LM weight is fixed at 0.6. We use the 300M parameters Conformer and Transformer++ models in the multilingual training. The WER results are compared in Table 3. We observe that both the per-language and averaged WER of Transformer++ are on-par with Conformer, while Transformer++ obtains 30% inference RTF reduction.

3.6. Efficiency analysis

We compare the CPU inference RTF and the training wall-clock time on GPUs between Conformer and Transformer++. RTF is measured as the neural network forward call on c5d.24xlarge CPU instances. For training time, we report the GPU hours taken by training the 100M and 300M models on 96k hours dataset using 32 and 64 A100 GPUs, respectively. All measurements are done on the same hardware. As shown in Table 4, Transformer++ demonstrates 30 to 40% RTF improvement for different model sizes compared to Conformer. Moreover, we observe at least 25% training speedup from Transformer++, e.g., training 300M Transformer++ takes 77 hours for 400K steps, compared to 103 hours taken by the 300M Conformer.

3.7. Ablation study

Impact of SwiGLU and sub-LN. As introduced in Section 2.3, SwiGLU and sub-LN are recent architectural advances in Transformer. To understand its impact in speech representation learning, we compare Transformer++ with plain FFN and SwiGLU+sub-LN FFN on the 96k hours English train-from-scratch setup, and report the averaged WER results across all our internal/public benchmarks in Table 5. As observed, using SwiGLU+sub-LN maintains the same model parameter counts and inference RTF as plain FFN, while reducing the WER by 3.8% relatively, and helps closing the gap against Conformer.

Impact of removing different convolutions. To achieve the final convolution-free architecture, we remove both the convolutional down-sampling layers at the front and the convolution modules in the encoder blocks. We study the impact of removing convolutions at different positions in terms of WER and inference RTF. We use the 100M parameter models and train them from scratch on the 96k hours data. The WER and RTF results are aggregated across all our evaluation benchmarks. As shown in Table 6, removing only the convolutions from the encoder blocks leads to about 14% RTF reduction with negligible WER impact. Further replacing the convolutional down-sampling layers by the frame stacking module produces another 30% RTF reduction. Although this replacement incurs about 0.2 absolute regression, we have shown that by leveraging unsupervised pre-training and other architectural improvements like SwiGLU, the WER gap is closed out between Transformer++ and Conformer.

4. Related Works

The end-to-end ASR models are typically composed of an encoder to process the input sequence of speech frames, and a decoder to convert the extracted acoustic features to text output. The encoder architectures have evolved from the early CNN based [28, 29, 30, 31, 32] to Transformers [33, 2, 34, 6] for better ability to capture long-range dependencies between speech frame, and recently converge to hybrid attention-convolution based [8, 35] that can model global and local dependencies efficiently. Along the direction of improving ASR model efficiency, recent methods focus on designing more aggressive down-sampling schemes such as Squeezeformer [10], Fast Conformer [36], Efficient Conformer [37], and replacing the self-attention modules by more efficient global context learning methods with linear complexity [38, 39]. Nevertheless, these architectures still retain the convolution operations. In stark contrast, the Transformer++ is a convolution-free architecture.

5. Conclusion

In this paper, we revisit the use of convolutions in current state-of-the-art ASR models. Our experiments demonstrate diminishing gains from using convolutions as we gradually scale up the training data. We then propose a simpler and more efficient Transformer++ architecture without any convolution operations. With the recent advancement of self-supervised pre-training, Transformer++ achieves state-of-the-art WER results. Importantly, Transformer++ demonstrates 40% RTF improvements and 25% training speedup in GPU hours compared to Conformer. We hope our data points can facilitate more efficient and simplified ASR model explorations in future.

6. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[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 *ASRU*, 2019.

[3] T. Likhomanenko, Q. Xu, V. Pratap, P. Tomasello, J. Kahn, G. Avidov, R. Collobert, and G. Synnaeve, "Rethinking evaluation in asr: Are our models robust enough?" *arXiv preprint arXiv:2010.11745*, 2020.

[4] C. Liu, F. Zhang, D. Le, S. Kim, Y. Saraf, and G. Zweig, "Improving rnn transducer based asr with auxiliary tasks," in *SLT*, 2021.

[5] F. Zhang, Y. Wang, X. Zhang, C. Liu, Y. Saraf, and G. Zweig, "Faster, simpler and more accurate hybrid asr systems using wordpieces," *arXiv preprint arXiv:2005.09150*, 2020.

[6] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP*, 2020.

[7] Z. Wu, Z. Liu, J. Lin, Y. Lin, and S. Han, "Lite transformer with long-short range attention," *arXiv preprint arXiv:2004.11886*, 2020.

[8] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Interspeech*, 2020.

[9] 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 *ICML*, 2022.

[10] S. Kim, A. Gholami, A. Shaw, N. Lee, K. Mangalam, J. Malik, M. W. Mahoney, and K. Keutzer, "Squeezeformer: An efficient transformer for automatic speech recognition," in *NeurIPS*, 2022.

[11] Y. Zhang, J. Qin, D. S. Park, W. Han, C.-C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*, 2020.

[12] C.-C. Chiu, J. Qin, Y. Zhang, J. Yu, and Y. Wu, "Self-supervised learning with random-projection quantizer for speech recognition," in *ICML*, 2022.

[13] Y. Zhang, W. Han, J. Qin, Y. Wang, A. Bapna, Z. Chen, N. Chen, B. Li, V. Axelrod, G. Wang *et al.*, "Google usm: Scaling automatic speech recognition beyond 100 languages," *arXiv preprint arXiv:2303.01037*, 2023.

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

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

[16] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *NeurIPS*, 2022.

[17] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *arXiv preprint arXiv:2104.09864*, 2021.

[18] T. Q. Nguyen and J. Salazar, "Transformers without tears: Improving the normalization of self-attention," *arXiv preprint arXiv:1910.05895*, 2019.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," in *JMLR*, 2014.

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

[21] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[22] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202*, 2020.

[23] H. Wang, S. Ma, S. Huang, L. Dong, W. Wang, Z. Peng, Y. Wu, P. Bajaj, S. Singhal, A. Benhaim *et al.*, "Foundation transformers," *arXiv preprint arXiv:2210.06423*, 2022.

[24] T. Hori, S. Watanabe, and J. Hershey, "Joint CTC/attention decoding for end-to-end speech recognition," in *ACL*, 2017.

[25] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *ICASSP*, 2021.

[26] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[27] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *SLT*, 2023.

[28] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv preprint arXiv:1904.03288*, 2019.

[29] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. L. Y. Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," *arXiv preprint arXiv:1701.02720*, 2017.

[30] S. Kriman, S. Beliaev, B. Ginsburg, J. Huang, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, and Y. Zhang, "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *ICASSP*, 2020.

[31] S. Majumdar, J. Balam, O. Hrinchuk, V. Lavrukhin, V. Noroozi, and B. Ginsburg, "Citrinet: Closing the gap between non-autoregressive and autoregressive end-to-end models for automatic speech recognition," *arXiv preprint arXiv:2104.01721*, 2021.

[32] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," *arXiv preprint arXiv:2005.03191*, 2020.

[33] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," in *ICASSP*, 2020.

[34] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end asr: from supervised to semi-supervised learning with modern architectures," *arXiv preprint arXiv:1911.08460*, 2019.

[35] 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 *SLT*, 2023.

[36] D. Rekesh, N. R. Koluguri, S. Kriman, S. Majumdar, V. Noroozi, H. Huang, O. Hrinchuk, K. Puvvada, A. Kumar, J. Balam *et al.*, "Fast conformer with linearly scalable attention for efficient speech recognition," in *ASRU*, 2023.

[37] M. Burchi and V. Vielzeuf, "Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition," in *ASRU*, 2021.

[38] T. Parcollet, R. van Dalen, S. Zhang, and S. Bhattacharya, "Summarymixing: A linear-complexity alternative to self-attention for speech recognition and understanding," *arXiv preprint arXiv:2307.07421*, 2023.

[39] F. Mai, J. Zuluaga-Gomez, T. Parcollet, and P. Motlicek, "Hyperconformer: Multi-head hypermixer for efficient speech recognition," *arXiv preprint arXiv:2305.18281*, 2023.