Branchformer: Parallel MLP-Attention Architectures to Capture Local and Global Context for Speech Recognition and Understanding

Yifan Peng [1] **Siddharth Dalmia** [2] **Ian Lane** [1] **Shinji Watanabe** [2]

Abstract

Conformer has proven to be effective in many speech processing tasks. It combines the benefits of extracting local dependencies using convolutions and global dependencies using self-attention. Inspired by this, we propose a more flexible, interpretable and customizable encoder alternative, *Branchformer*, with parallel branches for modeling various ranged dependencies in end-to-end speech processing. In each encoder layer, one branch employs self-attention or its variant to capture long-range dependencies, while the other branch utilizes an MLP module with convolutional gating (cgMLP) to extract local relationships. We conduct experiments on several speech recognition and spoken language understanding benchmarks. Results show that our model outperforms both Transformer and cgMLP. It also matches with or outperforms state-of-the-art results achieved by Conformer. Furthermore, we show various strategies to reduce computation thanks to the two-branch architecture, including the ability to have variable inference complexity in a single trained model. The weights learned for merging branches indicate how local and global dependencies are utilized in different layers, which benefits model designing. [1]

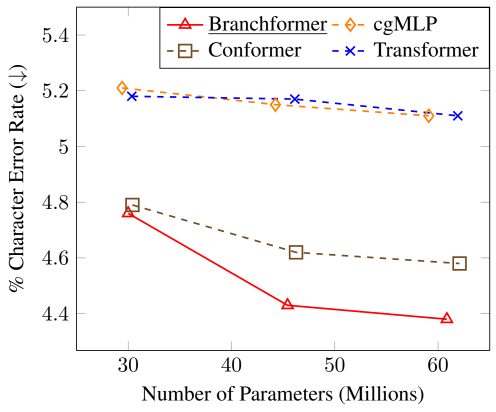


Figure 1. Character Error Rate (%) vs. Model Size. Our Branchformer outperforms previously proposed Conformer, cgMLP and Transformer at all scales on the benchmark Aishell ASR task.

1. Introduction

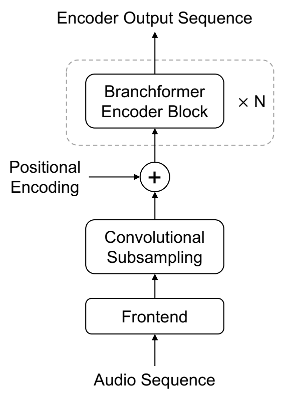
Conformer (Gulati et al., 2020), which is a variant of Transformer (Vaswani et al., 2017), has been successfully applied to various speech processing tasks (Guo et al., 2021; Arora et al., 2022; Chen et al., 2021). The key feature of Conformer is that it captures both local and global contextual information using convolution and self-attention, respectively. The effectiveness of combining local and global dependencies is also demonstrated by other studies in vision (Wu et al., 2021b) and language (Wu et al., 2020) tasks. Despite the superior performance of Conformer, there are some limitations. Conformer combines convolution and self-attention sequentially. This static single-branch architecture is hard to interpret and modify. It is unclear how local and global relationships are used in different encoder layers. Are they equally important in every layer? If not, which type of operation plays a major role in the initial layers? Understanding these questions can help researchers design better architectures.

To address these questions, we propose a flexible, interpretable and customizable encoder alternative, *Branchformer*, with two parallel branches to capture various ranged context in end-to-end automatic speech recognition (ASR) and spoken language understanding (SLU) tasks. In Branchformer, one branch employs self-attention or its variant to

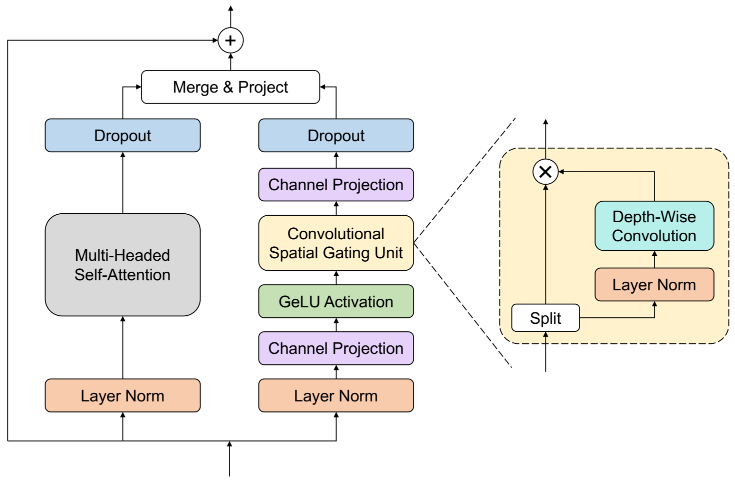
[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA [2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Correspondence to: Yifan Peng <yifanpen@andrew.cmu.edu>, Siddharth Dalmia <sdalmia@cs.cmu.edu>, Shinji Watanabe <swatanab@andrew.cmu.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

[1]Our code and models are released as part of the ESPnet toolkit: https://github.com/espnet/espnet.



(a) Overall architecture of the encoder. A stack of identical Branchformer blocks are used to capture local and global dependencies.



(b) Architecture of our Branchformer encoder block. It consists of two parallel branches. One branch employs attention to capture global context, while the other branch utilizes the MLP with convolutional gating to extract local context.

Figure 2. Architecture of the proposed Branchformer encoder and its components.

capture long-range dependencies, while the other branch utilizes an advanced multi-layer perceptron (MLP) to capture local dependencies. We adopt an MLP with gating called gMLP (Liu et al., 2021), which is originally proposed in vision and language tasks and has been successfully applied to CTC-based speech recognition (cgMLP) (Sakuma et al., 2022).

We conduct extensive experiments on three ASR and two SLU datasets. Results show that our model outperforms Transformer and cgMLP. It also matches with or outperforms the state-of-the-art Conformer. Furthermore, our Branchformer can be easily modified thanks to the disentangled two-branch design. The self-attention can be replaced by an efficient variant to reduce complexity with only minor performance degradation. If weighted average is used for branch merging, the learned weights represent the importance of each branch in different layers. This helps us understand how local and global relationships are utilized in a deep encoder model, which is beneficial for designing better architectures. Finally, by performing branch dropout during training, we can prune the model by removing the attention branch for faster inference speed without the need of re-training or fine-tuning of the Branchformer model. This allows practitioners to have two inference speeds within a single model and they can use either option depending on their need.

2. Related Work

2.1. RNNs, CNNs, Transformers and MLPs

Recurrent neural networks (RNNs) have been widely used in speech processing (Chan et al., 2016; Zeyer et al., 2018; Park et al., 2019). RNNs are capable of modeling temporal dependencies in an audio sequence, but they are difficult to parallelize and the long-range interactions are not well captured. Convolutional neural networks (CNNs) are another key building block in modern deep learning. CNNs focus on local dependencies and are shift-invariant, which are especially suitable for vision tasks. They can also be applied to speech recognition (Sainath et al., 2013; Li et al., 2019; Han et al., 2020).

In the past few years, Transformers with self-attention (Vaswani et al., 2017) have achieved superior performance in various speech applications (Karita et al., 2019). The self-attention mechanism is highly effective for modeling long-range global context, which is important for speech recognition and understanding. However, the time and memory complexity of self-attention is quadratic with respect to the sequence length. Researchers have proposed more efficient variants of self-attention (Tay et al., 2022) such as Longformer (Beltagy et al., 2020), Big Bird (Zaheer et al., 2020), Linformer (Wang et al., 2020) and Fastformer (Wu et al., 2021a). These modifications are orthogonal to our work, so any of them can be utilized in our framework. We will evaluate the Fastformer model as an example, because it shows strong performance with linear complexity.

More recently, multi-layer perceptrons (MLPs) are revisited in language and vision tasks (Tolstikhin et al., 2021; Liu et al., 2021). These MLP-based models have achieved comparable performance with Transformers, which is promising for further exploration. However, MLP-based models only accept fixed-length inputs, and thus cannot be directly applied to speech processing. To solve this problem, some MLP modules can be replaced by convolutions or other similar operations. For example, in (Sakuma et al., 2022), three variants of MLP-based models are proposed for CTC-based speech recognition. Among them, the MLP with convolutional gating (abbreviated as cgMLP in this paper) achieves the best performance. In this work, we employ cgMLP to extract local dependencies.

In addition to these standard modules, prior studies have also explored the combination of different types of modules (Sainath et al., 2015; Chen et al., 2018; Hao et al., 2019). Because different modules have complementary capacities, the combined model can potentially achieve better performance. Our Branchformer combines self-attention, convolution and MLP modules in a unified model.

2.2. Modeling Both Local and Global Context

Both local and global relationships play an important role in sequence modeling. Researchers have explored various methods to combine these two types of context in a unified model. Two well-known architectures are: Conformer (Gulati et al., 2020) and Lite Transformer (Wu et al., 2020).

Conformer is a variant of Transformer, which consists of two Macaron-like feed-forward layers, a multi-headed self-attention module and a convolution module. These modules are combined sequentially, resulting in a static single-branch architecture. Conformer-based methods have achieved state-of-the-art performance in many speech applications (Gulati et al., 2020; Guo et al., 2021). However, due to the single-branch design, it is difficult to analyze how local and global interactions are utilized in different layers. Conformer also enforces a fixed, interleaving pattern between self-attention and convolution, which may not always be optimal. As discovered in (Press et al., 2020), reordering the self-attention and feed-forward layers can improve Transformer's performance. Another limitation of Conformer is the quadratic complexity with respect to the sequence length due to self-attention.

Similar to this work, Lite Transformer (Wu et al., 2020) also adopts a two-branch architecture based on the standard self-attention and convolution to capture global and local dependencies. However, the motivation is quite different from ours.[2] Lite Transformer uses specialized parallel branches to reduce the overall computation and model size for mobile NLP applications. In speech, convolutions play an integral role in its modeling due to the local correlations in continuous speech data, which Conformer has exploited. Our Branchformer aims to re-design the architecture to make it more stable to train (Section 4.3), flexible to allow various attentions (Section 4.4), interpretable to present interesting design analysis (Section 4.6), and have different inference complexity in a single model (Section 4.7).

[2]The motivation plays a big role in the architecture choices. For example, Lite Transformer splits the input along the feature dimension before feeding them separately into the two branches. This reduces computation, but can harm the capacity of both branches. For better mobile efficiency, Lite Transformer also adopts a "flattened" feed-forward network (FFN), which we do not need. Actually, our Branchformer does not have an explicit FFN after the two branches. Instead, the two linear layers (i.e., channel projections) are integrated into the cgMLP branch. This design is novel compared to most of prior models.

3. Branchformer

3.1. Overall Architecture with Two Parallel Branches

The overall architecture of our Branchformer encoder is shown in Figure 2a. The raw audio sequence is first processed by a **frontend** module to extract log Mel features. Then, a **convolutional subsampling** module is applied to downsample the feature sequence in time. Positional encodings are added to the subsampled feature sequence, which is useful for self-attention modules in the following **Branchformer encoder blocks**. We employ a stack of N identical Branchformer blocks to capture both global and local relationships in the feature sequence.

The detailed architecture of the Branchformer encoder block is presented in Figure 2b. It consists of two parallel branches and a residual connection. The two branches share the same input, but focus on relationships of different ranges, which can be complementary to each other. In each branch, we first normalize the input using layer norm (Ba et al., 2016), then extract global or local dependencies using attention or cgMLP, respectively, which is followed by dropout (Srivastava et al., 2014). The outputs of two branches can be merged using concatenation or weighted average, with the original input added as a residual connection.

Compared with purely CNN or Transformer-based approaches, our Branchformer is able to capture both global and local context explicitly, which has been shown to be very important for various sequence processing tasks (Gulati et al., 2020; Wu et al., 2020), including ASR and SLU. Compared with the widely used Conformer which enforces a fixed interleaving pattern of self-attention and convolution operations, our two-branch architecture is more flexible, interpretable and customizable. As shown in our experiments (Section 4.6), the global and local contexts are not equally important in different encoder layers. Thus, a more flexible architecture may be more effective. We further evaluate strategies to reduce complexity, as discussed in Section 4.4 and Section 4.7.

3.2. Attention Branch for Global Context Modeling

3.2.1. Multi-Headed Self-Attention

In Figure 2b, the left branch in Branchformer aims to model global context in the input sequence. We employ the multi-headed self-attention mechanism (Vaswani et al., 2017) with relative positional encoding (Dai et al., 2019). In self-attention, the input is $\mathbf{X} \in \mathbb{R}^{T \times d}$, where T is the sequence length and d is the feature size. The input matrix can be further transformed into query, key and value matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{T \times d}$. The scaled dot product is calculated between every query and key, which is usually formulated as matrix multiplication $\mathbf{Q}\mathbf{K}^T/\sqrt{d}$. This computation has quadratic complexity with respect to T, which is not suitable for long sequences. Then, the raw scores are normalized using softmax and the output of self-attention is a weighted combination of the values, which can be written as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \tag{1}$$

In multi-headed self-attention, the queries, keys and values are projected h times with different learnable linear layers. After that, the self-attention operation is performed in parallel for each of these projected versions. Their outputs are concatenated and transformed to the original size, which generates the final output, as shown below:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1 \dots \text{head}_h)\mathbf{W}^O,$$

where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$, and the linear transforms are $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d \times d/h}$ and $\mathbf{W}^O \in \mathbb{R}^{d \times d}$.

3.2.2. More Efficient Attention

As discussed in Section 2.1, many efficient variants of self-attention can be used to replace the standard self-attention in our two-branch architecture. Here, we evaluate the Fastformer-based (Wu et al., 2021a) approach, which shows strong performance with linear complexity. The details of Fastformer are explained in Appendix A (Figure 8). The key component is the **attention-based pooling**, which summarizes a sequence into a single vector with global context. Specifically, let $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_T \in \mathbb{R}^d$ be the input sequence of the attention pooling module, the output $\mathbf{q} \in \mathbb{R}^d$ is a weighted sum: $\mathbf{q} = \sum_{i=1}^{T} \alpha_i \mathbf{q}_i$, where α_i is the attention weight. To obtain these weights, we first calculate the scaled dot product between a learnable parameter vector $\mathbf{w} \in \mathbb{R}^d$ and every input $\mathbf{q}_i$, and then perform softmax normalization:

$$\alpha_i = \frac{\exp(\mathbf{w}^T \mathbf{q}_i/\sqrt{d})}{\sum_{j=1}^{T} \exp(\mathbf{w}^T \mathbf{q}_j/\sqrt{d})}. \tag{2}$$

The complexity of attention-based pooling is linear with respect to the sequence length T, which is more efficient than self-attention and it is also capable of capturing global relationships.

3.3. MLP Branch for Local Context Modeling

In Figure 2b, the right branch of Branchformer focuses on more localized context in a sequence. This is realized by the MLP with convolutional gating (cgMLP) module (Sakuma et al., 2022), which employs depth-wise convolution and linear gating to learn strong representations from a sequence. cgMLP is more powerful than the standard convolution module in Conformer, which can potentially improve the capacity of our model. The architecture of cgMLP is depicted in Figure 2b. For brevity, the initial layer norm and the final dropout modules are omitted in the following formulation. The cgMLP module consists of a channel projection, an activation function such as Gaussian error Linear Unit (GeLU) (Hendrycks & Gimpel, 2016), a **convolutional spatial gating unit** (CSGU) and another channel projection. Specifically, given an input sequence $\mathbf{X} \in \mathbb{R}^{T \times d}$, the output is calculated as follows:

$$\mathbf{Z} = \text{GeLU}(\mathbf{X}\mathbf{U}) \in \mathbb{R}^{T \times d_{\text{hidden}}}, \tag{3}$$

$$\tilde{\mathbf{Z}} = \text{CSGU}(\mathbf{Z}) \in \mathbb{R}^{T \times d_{\text{hidden}}/2}, \tag{4}$$

$$\mathbf{Y} = \tilde{\mathbf{Z}}\mathbf{V} \in \mathbb{R}^{T \times d}, \tag{5}$$

where $\mathbf{U} \in \mathbb{R}^{d \times d_{\text{hidden}}}, \mathbf{V} \in \mathbb{R}^{d_{\text{hidden}}/2 \times d}$ denote the two channel projections. Usually, the hidden dimension d_{hidden} is larger than the original dimension d (e.g., $d = 256, d_{\text{hidden}} = 2048$), which is similar to the position-wise feed-forward layers in Transformer and Conformer.

The key component in cgMLP is the CSGU based on linear gating (Equation (4)), which exploits a depth-wise convolution to capture local dependencies. First, the input feature sequence $\mathbf{Z} \in \mathbb{R}^{T \times d_{\text{hidden}}}$ is equally split along the feature dimension, resulting in two new sequences $\mathbf{Z}_1, \mathbf{Z}_2 \in \mathbb{R}^{T \times d_{\text{hidden}}/2}$. Then, $\mathbf{Z}_2$ is normalized with layer norm and processed by a depth-wise convolution along the time dimension:

$$\mathbf{Z}_2' = \text{DWConv}(\text{LayerNorm}(\mathbf{Z}_2)). \tag{6}$$

The final output of CSGU is the element-wise product of $\mathbf{Z}_1$ and $\mathbf{Z}_2'$, i.e., $\tilde{\mathbf{Z}} = \mathbf{Z}_1 \otimes \mathbf{Z}_2'$, where $\otimes$ denotes element-wise multiplication. This is a type of linear gating, as we do not use other nonlinear activations before the multiplication.

The computational costs of the two channel projections are $O(Tdd_{\text{hidden}})$ and $O(Tdd_{\text{hidden}}/2)$, respectively. The depth-wise convolution has $O(TKd_{\text{hidden}}/2)$ complexity, where K is the kernel size. The overall complexity is linear with respect to T, although there is a constant factor K.

3.4. Merging Two Branches

We employ concatenation or weighted average to merge two branches. Concatenation is used as the default method since it is simple and effective, but weighted average is more interpretable. The branch weights indicate how global and local relationships are utilized in different layers.

3.4.1. Concatenation

Let $\mathbf{Y}_{\text{att}}, \mathbf{Y}_{\text{mlp}} \in \mathbb{R}^{T\times d}$ be the output sequences from the attention branch and cgMLP branch, respectively. We first concatenate them along the feature dimension and then project the result back to the original dimension:

$$\mathbf{Y}_{\text{merged}} = \text{concat}(\mathbf{Y}_{\text{att}}, \mathbf{Y}_{\text{mlp}})\mathbf{W}_{\text{merge}} \in \mathbb{R}^{T\times d}, \quad (7)$$

where $\mathbf{W}_{\text{merge}} \in \mathbb{R}^{2d\times d}$ is a learnable matrix.

3.4.2. Weighted Average

The concatenation-based merging is very effective, but it is difficult to interpret and modify. To mitigate this issue, weighted average is also utilized in our work, where the weights are dynamically generated by the model. More specifically, we first summarize the output sequence of each branch into a single vector using the attention-based pooling as described in Section 3.2.2:

$$\mathbf{y}_{\text{att}} = \text{AttPooling}(\mathbf{Y}_{\text{att}}), \quad (8)$$
$$\mathbf{y}_{\text{mlp}} = \text{AttPooling}(\mathbf{Y}_{\text{mlp}}), \quad (9)$$

where $\mathbf{y}_{\text{att}}, \mathbf{y}_{\text{mlp}} \in \mathbb{R}^{d}$. Then, the two vectors are projected to two scalars and normalized using softmax to get branch weights:

$$w_{\text{att}}, w_{\text{mlp}} = \text{softmax}(\mathbf{W}_{\text{att}}\mathbf{y}_{\text{att}}, \mathbf{W}_{\text{mlp}}\mathbf{y}_{\text{mlp}}), \quad (10)$$

where $\mathbf{W}_{\text{att}}, \mathbf{W}_{\text{mlp}} \in \mathbb{R}^{1\times d}$ represent linear transforms. Finally, the merged output is a weighted average: $\mathbf{Y}'_{\text{merged}} = w_{\text{att}}\mathbf{Y}_{\text{att}} + w_{\text{mlp}}\mathbf{Y}_{\text{mlp}}$, where $\mathbf{Y}'_{\text{merged}} \in \mathbb{R}^{T\times d}$ captures both global and local dependencies.

To make it possible to prune the two-branch model for faster inference, we can drop the entire attention branch (i.e., setting the weight of the attention branch to zero and directly using the other branch) with a certain probability during training. We call this technique **branch dropout**, which is evaluated in Section 4.7.

3.5. Complexity Analysis

In this section, we analyze the complexity of the components in our proposed Branchformer. Let T be the sequence length and d be the feature dimension.

For the attention-based branch, we consider self-attention and Fastformer. In the standard self-attention formulation, the scaled dot product is calculated between all pairs of feature vectors in the sequence, which leads to $O(T^2d)$ complexity. In Fastformer, the attention-based pooling has compleixty $O(Td)$, as discussed in Section 3.2.2. The linear projections in both self-attention and Fastformer have complexity $O(Td^2)$. Fastformer is more efficient for longer sequences because its complexity is linear instead of quadratic w.r.t. the sequence length.

Table 1. Results presenting the % Character Error Rate (CER) of our proposed Branchformer model on the Aishell Mandarin ASR task. Previously published papers and our reproduction of the baselines using cgMLP, Transformer, Lite Transformer and Conformer models are shown for comparison. No LM rescoring is used unless specified.

Method	Params (M)	dev (↓)	test (↓)
SpeechBrain (Ravanelli et al., 2021)			
Transformer	-	5.60	6.04
WeNet (Yao et al., 2021)			
Transformer	-	-	5.30
Conformer	-	-	**4.61**
ESPnet (Watanabe et al., 2018)			
Transformer (+ RNN LM)	30.4	5.9	6.4
Conformer	46.2	**4.5**	4.9
Our Baselines (reproduced based on ESPnet)			
cgMLP	44.3	4.61	5.15
Transformer	46.1	4.83	5.17
Lite Transformer	51.0	4.70	5.06
Conformer	46.3	4.24	4.62
Our Proposed Model			
Branchformer	45.4	**4.19**	**4.43**

For the MLP-based branch, the overall computational cost is linear w.r.t. the sequence length, although there is a constant factor related to the kernel size, as discussed in Section 3.3.

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets

We evaluate our models on ASR and SLU tasks.[3] For ASR, three widely-used public datasets are used: (1) Aishell (Bu et al., 2017), which consists of 170 hours of Mandarin speech data; (2) Switchboard (SWBD) 300h (Godfrey et al., 1992), which contains about 300 hours of English telephone conversations, and (3) LibriSpeech 960h (Panayotov et al., 2015), which consists of about 960 hours of English read audiobooks. For SLU, the recently released SLURP corpus (Bastianelli et al., 2020) is used for intent classification and entity prediction. SLURP is an English SLU dataset which is substantially larger and linguistically more diverse than previous SLU datasets. The Speech Commands dataset (Warden, 2018) is also employed. The vocabulary contains 35 words. Each utterance is around one second.

4.1.2. Implementation Details

Our models are implemented using PyTorch (Paszke et al., 2019) and the ASR and SLU experiments are conducted using the ESPnet toolkit (Watanabe et al., 2018; Arora et al., 2022; Guo et al., 2021). We follow the correspond-

[3]We also tested the efficacy of Branchformer on machine translation. Results are shown in Appendix G.

Table 2. Results presenting the % Word Error Rate (WER) of our proposed Branchformer model on the Switchboard ASR task. Previously published papers and our reproduction of the baselines using Transformer, cgMLP and Conformer models are shown for comparison. No LM rescoring is used. $^\diamond$Implemented with LSTM.

Method	Params (M)	swb (↓)	chm (↓)	eval2000 (↓)
Tüske et al. (2020) $^\diamond$	-	7.6	**14.6**	-
Park et al. (2019) $^\diamond$	-	**7.2**	**14.6**	-
ESPnet (Watanabe et al., 2018)				
Transformer	-	9.0	18.1	13.6
Conformer	-	**7.2**	**14.6**	**10.9**
Our Baselines (reproduced based on ESPnet)				
cgMLP	32.6	8.7	16.3	12.5
Transformer	44.4	9.0	16.0	12.5
Conformer	44.5	**7.8**	14.5	11.1
Our Proposed Model				
Branchformer	43.7	**7.8**	**14.1**	**10.9**

Table 3. Results presenting the % WER of our proposed Branchformer model on the LibriSpeech ASR task. Previously published papers and our reproduction of the Conformer baseline are shown for comparison. No LM rescoring is used unless specified.

Method	Params (M)	dev (↓)		test (↓)	
		clean	other	clean	other
LSTM (Park et al., 2019)	-	-	-	2.8	6.8
ContextNet (Han et al., 2020)	112.7	2.0	4.6	**2.1**	4.6
Conformer (Gulati et al., 2020)	118.8	**1.9**	**4.4**	**2.1**	**4.3**
SpeechBrain (Ravanelli et al., 2021)					
Transformer + Transformer LM	-	-	-	2.5	5.9
WeNet (Yao et al., 2021)					
Conformer	-	-	-	2.7	6.5
ESPnet (Watanabe et al., 2018)					
Transformer + RNN LM	99.4	2.3	5.9	2.5	6.2
Conformer	116.2	2.3	6.1	2.6	6.0
Our Baseline (reproduced based on ESPnet)					
Conformer	116.2	**2.2**	5.6	2.5	**5.5**
Our Proposed Model					
Branchformer	116.2	**2.2**	**5.5**	**2.4**	**5.5**

ing recipes in ESPnet for data preparation, model training and evaluation. The 80-dim log Mel filterbank features are extracted. The window and hop lengths vary for different datasets. SpecAugment (Park et al., 2019) and speed perturbation are performed for data augmentation. We apply two 3×3 convolutions with a stride of 2 to subsample the original speech feature sequence. For different tasks, we adjust the feature size d, hidden size d_{hidden}, number of layers and attention heads h in the Branchformer encoder to make it comparable with previous models, but we always employ a 6-layer Transformer decoder. Our experiments are conducted using 4 Tesla V100 GPUs with 32GB memory. The Adam optimizer (Kingma & Ba, 2015) with weight decay 1e-6 is utilized. We also apply dropout (Srivastava et al., 2014) with probability 0.1 and label smoothing (Müller et al., 2019) with weight 0.1 to mitigate overfitting. We adopt the Transformer learning rate scheduler (Vaswani et al., 2017). The CTC weight is set to 0.3 for joint training with the attention decoder (Kim et al., 2017). The joint CTC-attention decoding (Hori et al., 2017) is also performed. We average the best 10 checkpoints based on validation performance for inference. The exponential moving average technique is applied to the Switchboard corpus, but it is not used in other datasets. Table 8 in Appendix B summarizes the hyper-parameters in each dataset for reproducibility.

Table 4. Results presenting the accuracy and SLU-F1 of our proposed Branchformer model on the SLURP SLU benchmark. Previously published papers and our reproduction of baselines using cgMLP, Transformer and Conformer are shown for comparison.

Method	Intent Classification		Entity Prediction	
	Params (M)	Acc. (↑)	Params (M)	SLU-F1 (↑)
SLURP Paper (Bastianelli et al., 2020)	-	0.783	-	0.708
SpeechBrain (Ravanelli et al., 2021)				
LSTM + pretrained ASR encoder	-	0.751	-	0.633
ESPnet (Arora et al., 2022)				
Conformer	109.3	**0.863**	-	**0.719**
Our Baselines (reproduced based on ESPnet)				
cgMLP	77.0	0.859	77.0	0.734
Transformer	90.2	0.872	90.3	0.717
Conformer	109.3	0.877	109.4	0.769
Our Proposed Model				
Branchformer	110.1	**0.881**	95.6	**0.777**

4.2. Main Results

In this section, we compare the proposed Branchformer encoder with previous studies. Here, we adopt the standard self-attention (Section 3.2.1) and concatenation-based merging (Section 3.4.1). For fair comparisons, we reproduced cgMLP, Transformer and Conformer baselines using ESPnet based on our own environments.

Table 1 shows the results on Aishell without a language model (LM). Compared with cgMLP and Transformer, our Branchformer reduces the character error rate (CER) by 0.7% absolutely on the test set. It also outperforms Conformer by 0.2%, achieving the best performance among these methods.

Table 2 presents the word error rates (WERs) on Switchboard 300h without an LM. Again, our Branchformer outperforms cgMLP and Transformer baselines by a large margin. It matches with Conformer on the Switchboard (swb) subset, and outperforms it on the CallHome (chm) subset by 0.4%. The performance of Branchformer is also comparable with previous best results without LM fusion.

Table 3 compares WERs on LibriSpeech 960h. The full results are shown in Table 9 of Appendix C. Our Branchformer achieves 2.4/5.5 without an LM and 2.1/4.5 with an LM on the test clean/other sets. This is comparable with the Conformer baseline, and is better than results reported by other open-source toolkits. Note that our numbers are worse than the best reported Conformer-Transducer result (Gu-

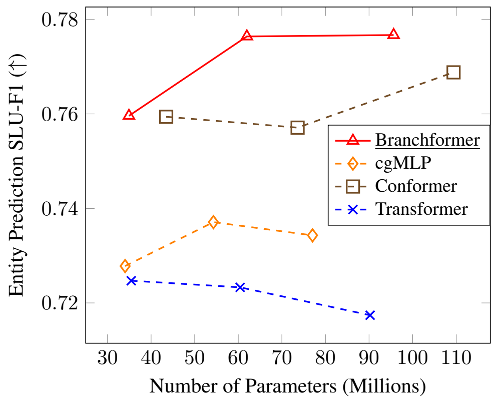


Figure 3. SLURP Entity Prediction SLU-F1 vs. Model Size. Our Branchformer outperforms the previously proposed Conformer, cgMLP and Transformer models at all scales for the SLURP benchmark Spoken Language Understanding task.

Table 5. Accuracy performance of Branchformer vs. other architectures on Google Speech Commands (35 commands). Training the vanilla Conformer model is unstable on this dataset, but Branchformer achieves similar performance as other models.

Method	Params (M)	Accuracy (↑)	
		dev	test
SpeechBrain (Ravanelli et al., 2021)			
TDNN (+ xvector)	-	-	0.974
ESPnet (Arora et al., 2022)			
Conformer (w/o BatchNorm)	-	**0.974**	**0.975**
Our Baselines (reproduced based on ESPnet)			
cgMLP	30.7	0.966	0.966
Transformer	42.9	**0.973**	**0.974**
Conformer (w/ BatchNorm)	43.0	diverged	
Our Proposed Model			
Branchformer	41.8	**0.973**	0.973

lati et al., 2020) achieved by Google, because our model is based on a Transformer decoder instead of RNN-Transducer and our code-base is different.

We evaluate our approach on two SLU tasks using SLURP, as shown in Table 4. Our Branchformer achieves the best accuracy for intent classification and the best SLU-F1 for entity prediction among the reported results, which demonstrates the effectiveness of our model in different tasks.

4.3. Model Scalability and Training Stability

We compare our Branchformer with baselines at different model scales. In Figure 1, the CER on Aishell decreases for all models as the model size increases. At all three scales, the proposed Branchformer achieves the lowest CER. Figure 3 shows the SLU-F1 scores on SLURP. Again, our model outperforms the baselines at all model scales. These results demonstrate the scalability and effectiveness of Branchformer.

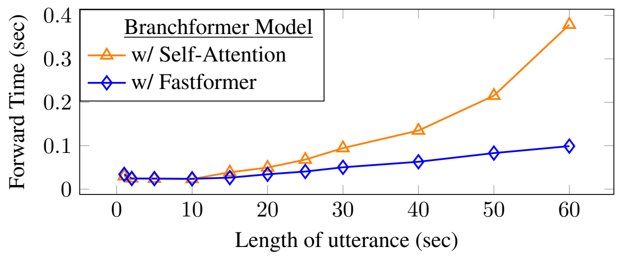


Figure 4. Encoder forward time vs. input audio length using different attention mechanisms for modeling global dependencies in Branchformer. Branchformer w/ Fastformer achieves linear scaling in forward time with different utterance lengths.

Table 6. Comparison of the Fastformer-based model with others on Aishell (% CER) and Switchboard 300h (% WER). Fastformer has linear complexity w.r.t. the sequence length T, while self-attention has quadratic complexity. K denotes the convolution kernel size.

Method	Complexity	Aishell		SWBD 300h	
		dev	test	swb	chm
cgMLP	$O(TK)$	4.61	5.15	8.7	16.3
Transformer	$O(T^2)$	4.83	5.17	9.0	16.0
Conformer	$O(T^2)$	4.24	4.62	**7.8**	14.5
Branchformer					
w/ self-attention	$O(T^2)$	**4.19**	**4.43**	**7.8**	**14.1**
w/ Fastformer	$O(TK)$	4.22	4.58	7.9	14.5

In our experiments, we have found that Branchformer is more stable to train than Conformer, especially on short utterances and limited data. This can be seen in our results on Google Speech Commands in Table 5. Our reproduction of the standard Conformer model diverged, which is consistent with the description reported in the ESPnet recipe (Arora et al., 2022). This is probably due to the batch norm used in Conformer or the difficulty of optimizing deeper layers in Conformer (i.e., sequential combination of convolution and self-attention layers). Branchformer on the other hand is more stable and achieves similar results with other methods.

4.4. Results of Efficient Attention

As discussed in Section 3.2.2, the standard self-attention can be replaced by more efficient attention variants (e.g., Fastformer) to reduce complexity. We evaluate this approach on Aishell and Switchboard 300h. Results are presented in Table 6. The encoder forward time for inputs of different lengths is shown Figure 4. We randomly generate inputs with batch size 12 and run the encoder 10 times on a Tesla V100 GPU. Then the average time is reported. Branchformer with self-attention performs the best, but its

Table 7. Comparison of two methods for merging the branches of Branchformer on Aishell. Branchformer w/ concatenation performs better. Branchformer w/ weighted average slightly degrades performance but exhibits other desirable properties.

Method	Params (M)	CER (%)	
		dev	test
cgMLP	44.26	4.61	5.15
Transformer	46.13	4.83	5.17
Conformer	46.25	4.24	4.62
Branchformer			
w/ concatenation	45.43	**4.19**	**4.43**
w/ weighted average	43.88	4.23	4.61

complexity is quadratic in T. The Fastformer-based variant has linear complexity. Although its performance degrades slightly, it is still comparable with Conformer and is much better than cgMLP and Transformer. This demonstrates the flexibility of our architecture, thanks to the disentangled two-branch design and the strong cgMLP branch.[4]

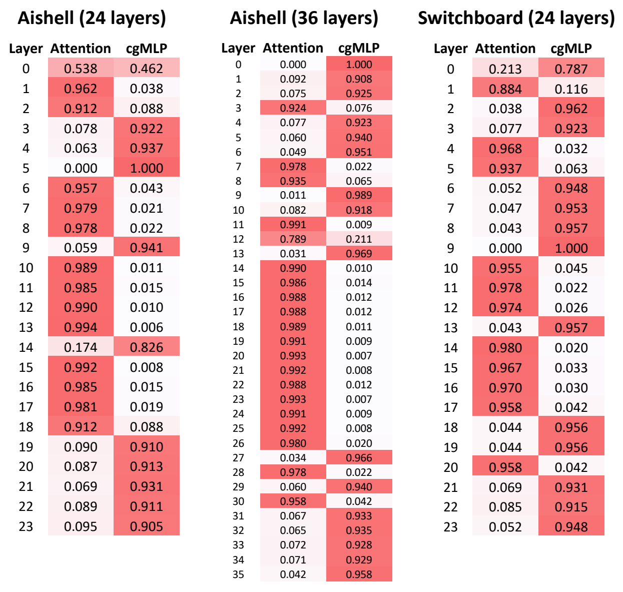
4.5. Comparison of Merging Operations

Two merging operations are discussed in Section 3.4. Table 7 compares their performance on Aishell. Weighted average is slightly worse than concatenation, but still matches with Conformer. In subsequent sections, we will show that weighted average is more intrepretable and customizable.

4.6. Layer-Wise Analysis of Local/Global Branches

The learned weights for merging two branches introduced in Section 3.4.2 can represent the importance of local and global context in different layers. We visualize the average weights on the validation set in Figure 5. The standard deviation is usually smaller than 0.01, which means the weights are very consistent for different samples. We can observe certain patterns in the weights. In almost all layers, one branch is dominant. In initial layers, the two types of branches are utilized in an interleaving fashion, which is similar to the Conformer design. This indicates the model is using both local and global relationships to learn strong hidden representations. In later layers, multiple consecutive attention blocks are observed, showing that global context is more important in the intermediate layers. Finally, multiple consecutive cgMLP blocks are leveraged to extract the local contextual information, which is then used by the Transformer decoder. Interestingly, similar patterns have also been discovered in prior studies. Sainath et al. have explored various ways to unify convolutional, recurrent and linear layers in a single model (2015) and found that it is

[4]We also replaced the self-attention in Conformer with Fastformer and trained this efficient Conformer on Aishell. The CERs on dev and test sets are 4.69% and 5.56%, respectively, which are much worse than the original Conformer results.


Aishell (24 layers)

Layer	Attention	cgMLP
0	0.538	0.462
1	0.962	0.038
2	0.912	0.088
3	0.078	0.922
4	0.063	0.937
5	0.000	1.000
6	0.957	0.043
7	0.979	0.021
8	0.978	0.022
9	0.059	0.941
10	0.989	0.011
11	0.985	0.015
12	0.990	0.010
13	0.994	0.006
14	0.174	0.826
15	0.992	0.008
16	0.985	0.015
17	0.981	0.019
18	0.912	0.088
19	0.090	0.910
20	0.087	0.913
21	0.069	0.931
22	0.089	0.911
23	0.095	0.905

Aishell (36 layers)

Layer	Attention	cgMLP
0	0.000	1.000
1	0.092	0.908
2	0.075	0.925
3	0.924	0.076
4	0.077	0.923
5	0.060	0.940
6	0.049	0.951
7	0.978	0.022
8	0.935	0.065
9	0.011	0.989
10	0.082	0.918
11	0.991	0.009
12	0.789	0.211
13	0.031	0.969
14	0.990	0.010
15	0.986	0.014
16	0.988	0.012
17	0.988	0.012
18	0.989	0.011
19	0.991	0.009
20	0.993	0.007
21	0.992	0.008
22	0.988	0.012
23	0.993	0.007
24	0.991	0.009
25	0.992	0.008
26	0.980	0.020
27	0.034	0.966
28	0.978	0.022
29	0.060	0.940
30	0.958	0.042
31	0.067	0.933
32	0.065	0.935
33	0.072	0.928
34	0.071	0.929
35	0.042	0.958

Switchboard (24 layers)

Layer	Attention	cgMLP
0	0.213	0.787
1	0.884	0.116
2	0.038	0.962
3	0.077	0.923
4	0.968	0.032
5	0.937	0.063
6	0.052	0.948
7	0.047	0.953
8	0.043	0.957
9	0.000	1.000
10	0.955	0.045
11	0.978	0.022
12	0.974	0.026
13	0.043	0.957
14	0.980	0.020
15	0.967	0.033
16	0.970	0.030
17	0.958	0.042
18	0.044	0.956
19	0.044	0.956
20	0.958	0.042
21	0.069	0.931
22	0.085	0.915
23	0.052	0.948

Figure 5. Visualization of branch weights in Branchformer with different sizes (24 vs. 36 layers) and on different datasets (Aishell vs. Switchboard). We see a consistent behavior of interleaving dominant branch in the beginning few layers followed by multiple consecutive layers of dominant global and local branches each.

better to employ RNNs for capturing global temporal dependencies in the middle stage and linear layers for capturing local interactions in the final stage. Press et al. has successfully improved Transformer by putting more self-attention blocks before feed-forward layers (2020).

In the above discussion, we assume that the self-attention branch captures more global context in a sequence, but self-attention also has the potential to focus on local context. To analyze the behavior of self-attention in different architectures, we calculate the diagonality metric for the attention weight matrices in every encoder layer, as proposed by Zhang et al. (2021). A higher diagonality means the attention weight matrix concentrates more on its diagonal, thus capturing more local context. We compute the average diagonality over all samples in the validation set and all attention heads in each layer. As shown in Figure 6, Branchformer has lower diagonality than Transformer, showing that the self-attention branches of Branchformer capture more global relationships due to the disentangled two-branch design. Similar patterns of the self-attention weights have been observed in Lite Transformer (Wu et al., 2020), because it also has specialized parallel branches. The definition of diagonality and some examples of the attention weights in Branchformer and Transformer can be found in Appendix D.

Based on the layer-wise analysis of two branches, the interleaving pattern of local and global blocks imposed by Conformer may not always be optimal. Our two-branch architec-

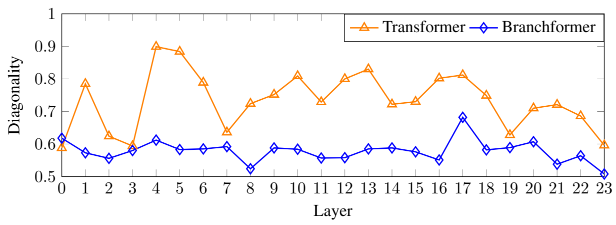


Figure 6. Diagonality of self-attention in each encoder layer. A higher diagonality means the attention weight matrix concentrates more on its diagonal, thus capturing more local context (Zhang et al., 2021).

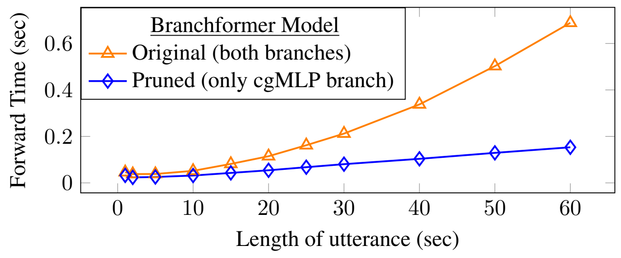


Figure 7. Encoder forward time vs. input audio length. The pruned model with a single branch has linear complexity, while the original two-branch model has quadratic complexity.

ture dynamically determines the importance of each branch, which can be more flexible and powerful. To verify such observation and hypothesis, we combined the Conformer blocks with Branchformer blocks sequentially, resulting in a two-stage encoder model. Detailed results are presented in Appendix E (Table 10). With Conformer blocks first and Branchformer blocks last, the performance is better than the vanilla Conformer and is similar to our Branchformer.

4.7. Model Pruning Using Branch Dropout

As described in Section 3.4.2, if we apply branch dropout to the attention branch during training, then our Branchformer can work in two modes for inference. In one mode, both branches are employed, which is more effective but slower. In the other mode, only the cgMLP branch is utilized, which reduces complexity from quadratic to linear and still keeps comparable performance with the same cgMLP trained from scratch. Note that this approach does not require fine-tuning or re-training. We simply train a single model with branch dropout, and it can work in two modes for inference. This demonstrates our two-branch model is more flexible and customizable. The encoder forward time for different input lengths is shown in Figure 7. We randomly generate an input with batch size 8 and run the encoder 10 times on a Tesla V100 GPU. The average time is reported. Detailed results with different dropout rates are shown in Table 11 of Appendix F.

5. Conclusion

In this work, we present a novel encoder architecture called Branchformer. It has two parallel branches, one for capturing global interactions using attention and the other for more localized context using cgMLP. Due to the disentangled architecture, Branchformer can be customized easily by replacing the self-attention component with a more efficient attention. With the weighted average-based merging and branch dropout, we show that Branchformer can become flexible to have two different inference time complexity in a single trained model. By studying the branch weights in Branchformer, we could understand how local and global dependencies get utilized in different layers, which is helpful for model designing. Besides its flexible, interpretable, and customizable design, Branchformer outperforms Transformer and cgMLP by a large margin in various ASR and SLU benchmarks. It also achieves comparable or superior performance than the state-of-the-art Conformer while being more stable towards training in extreme data regimes.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback on our paper. This work used the Extreme Science and Engineering Discovery Environment (XSEDE) (Towns et al., 2014), which is supported by National Science Foundation grant number ACI-1548562. Specifically, it used the Bridges system (Nystrom et al., 2015), which is supported by NSF award number ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

References

Arora, S., Dalmia, S., Denisov, P., Chang, X., Ueda, Y., Peng, Y., Zhang, Y., Kumar, S., Ganesan, K., Yan, B., Thang Vu, N., Black, A. W., and Watanabe, S. ESPnet-SLU: Advancing Spoken Language Understanding Through ESPnet. In *Proceedings of ICASSP*, 2022.

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

Bastianelli, E., Vanzo, A., Swietojanski, P., and Rieser, V. SLURP: A Spoken Language Understanding Resource Package. In *Proceedings of EMNLP*, 2020.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Bu, H., Du, J., Na, X., Wu, B., and Zheng, H. AISHELL-1: An open-source Mandarin speech corpus and a speech recognition baseline. In *Conference of the Oriental Chapter of the International Coordinating Committee* *on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*, 2017.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of ICASSP*, 2016.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *Proceedings of ACL*, 2018.

Chen, S., Wu, Y., Chen, Z., Wu, J., Li, J., Yoshioka, T., Wang, C., Liu, S., and Zhou, M. Continuous Speech Separation with Conformer. In *Proceedings of ICASSP*, 2021.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of ACL*, 2019.

Godfrey, J., Holliman, E., and McDaniel, J. SWITCHBOARD: telephone speech corpus for research and development. In *Proceedings of ICASSP*, 1992.

Gulati, A., Qin, J., Chiu, C.-C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proceedings of Interspeech*, 2020.

Guo, P., Boyer, F., Chang, X., Hayashi, T., Higuchi, Y., Inaguma, H., Kamo, N., Li, C., Garcia-Romero, D., Shi, J., et al. Recent developments on ESPnet toolkit boosted by conformer. In *Proceedings of ICASSP*, 2021.

Han, W., Zhang, Z., Zhang, Y., Yu, J., Chiu, C.-C., Qin, J., Gulati, A., Pang, R., and Wu, Y. ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context. In *Proceedings of Interspeech*, 2020.

Hao, J., Wang, X., Yang, B., Wang, L., Zhang, J., and Tu, Z. Modeling Recurrence for Transformer. In *Proceedings of NAACL*, 2019.

Hendrycks, D. and Gimpel, K. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Hori, T., Watanabe, S., and Hershey, J. R. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of ACL*, 2017.

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., and Zhang, W. A Comparative Study on Transformer vs RNN in Speech Applications. In *Proceedings of ASRU*, 2019.

Kim, S., Hori, T., and Watanabe, S. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *Proceedings of ICASSP*, 2017.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *Proceedings of ICLR*, 2015.

Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Nguyen, H., and Gadde, R. T. Jasper: An End-to-End Convolutional Neural Acoustic Model. In *Proceedings of Interspeech*, 2019.

Liu, H., Dai, Z., So, D., and Le, Q. V. Pay attention to MLPs. In *Proceedings of NeurIPS*, 2021.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *Proceedings of NeurIPS*, 2019.

Nystrom, N. A., Levine, M. J., Roskies, R. Z., and Scott, J. R. Bridges: a uniquely flexible HPC resource for new communities and data analytics. In *Proceedings of the 2015 XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, 2015.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In *Proceedings of ICASSP*, 2015.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., and Le, Q. V. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. In *Proceedings of Interspeech*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of NeurIPS*, 2019.

Press, O., Smith, N. A., and Levy, O. Improving Transformer Models by Reordering their Sublayers. In *Proceedings of ACL*, 2020.

Raunak, V., Dalmia, S., Gupta, V., and Metze, F. On Long-Tailed Phenomena in Neural Machine Translation. In *Findings of EMNLP*, 2020.

Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., Mori, R. D., and Bengio, Y.

SpeechBrain: A General-Purpose Speech Toolkit, 2021. arXiv:2106.04624.

Sainath, T. N., Kingsbury, B., Mohamed, A.-r., Dahl, G. E., Saon, G., Soltau, H., Beran, T., Aravkin, A. Y., and Ramabhadran, B. Improvements to Deep Convolutional Neural Networks for LVCSR. In *Proceedings of ASRU*, 2013.

Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks. In *Proceedings of ICASSP*, 2015.

Sakuma, J., Komatsu, T., and Scheibler, R. MLP-based architecture with variable length input for automatic speech recognition, 2022. URL https://openreview.net/forum?id=RA-zVvZLYIy.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient Transformers: A Survey. *ACM Comput. Surv.*, Apr 2022.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A. P., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. MLP-Mixer: An all-MLP Architecture for Vision. In *Proceedings of NeurIPS*, 2021.

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G. D., Roskies, R., Scott, J. R., and Wilkins-Diehr, N. XSEDE: Accelerating Scientific Discovery. *Computing in Science & Engineering*, 16(5):62–74, Sept.-Oct. 2014.

Tüske, Z., Saon, G., Audhkhasi, K., and Kingsbury, B. Single Headed Attention Based Sequence-to-Sequence Model for State-of-the-Art Results on Switchboard. In *Proceedings of Interspeech*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proceedings of NeurIPS*, 2017.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *CoRR*, abs/1804.03209, 2018.

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Enrique Yalta Soplin, N., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., and Ochiai, T. ESPnet: End-to-End Speech Processing Toolkit. In *Proceedings of Interspeech*, 2018.

Wu, C., Wu, F., Qi, T., Huang, Y., and Xie, X. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*, 2021a.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. CvT: Introducing Convolutions to Vision Transformers. In *Proceedings of ICCV*, 2021b.

Wu, Z., Liu, Z., Lin, J., Lin, Y., and Han, S. Lite Transformer with Long-Short Range Attention. In *Proceedings of ICLR*, 2020.

Yao, Z., Wu, D., Wang, X., Zhang, B., Yu, F., Yang, C., Peng, Z., Chen, X., Xie, L., and Lei, X. WeNet: Production oriented Streaming and Non-streaming End-to-End Speech Recognition Toolkit. In *Proceedings of Interspeech*, 2021.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big Bird: Transformers for Longer Sequences. In *Proceedings of NeurIPS*, 2020.

Zeyer, A., Irie, K., Schlüter, R., and Ney, H. Improved Training of End-to-end Attention Models for Speech Recognition. In *Proceedings of Interspeech*, 2018.

Zhang, S., Loweimi, E., Bell, P., and Renals, S. On The Usefulness of Self-Attention for Automatic Speech Recognition with Transformers. In *Proceedings of SLT*, 2021.

A. Fastformer Architecture

Figure 8 shows the overall architecture of Fastformer (Wu et al., 2021a), which has linear complexity w.r.t. the sequence length. The attention-based pooling is described in Section 3.2.2 and Equation (2).

In Fastformer, the input is first transformed into query, key and value sequences. Then, the attention pooling module extracts a single vector from the queries, which has global contextual information. The vector is multiplied with every key vector, resulting in a new sequence.

Next, we repeat the procedure described above for the new key sequence after multiplication. We perform another attention pooling to obtain a global representation for the new key sequence, and multiply it to every vector in the value sequence. This element-wise multiplication is followed by a final linear transform to generate the output of the entire Fastformer block. Note that we also add the query to the output, which is similar to a residual connection.

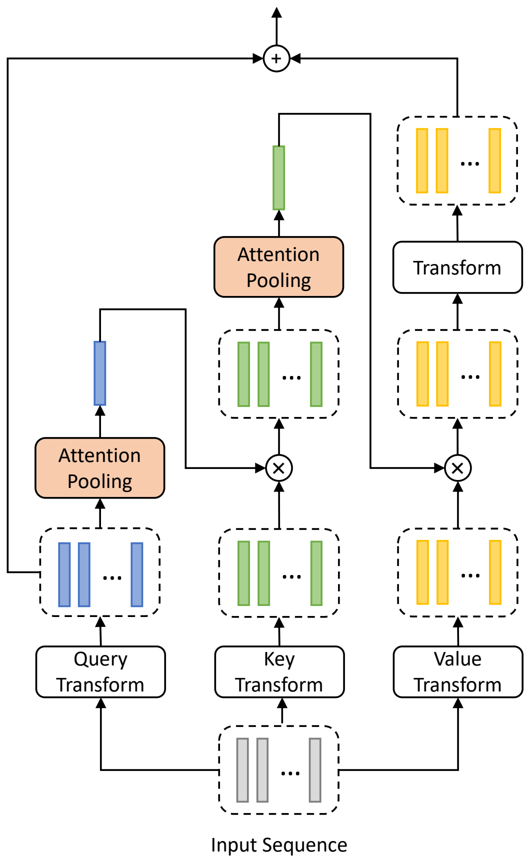


Figure 8. Architecture of the Fastformer block (Wu et al., 2021a), which is an efficient variant of Transformer with linear complexity. The attention-based pooling module extracts a global representation from a sequence based on attention mechanism, as shown in Section 3.2.2 and Equation (2). $\otimes$ denotes element-wise multiplication.

B. Implementation Details

The hyper-parameters used in our main experiments are presented in Table 8.

C. Full Results on LibriSpeech 960h

Table 9 shows the full results on LibriSpeech 960h, both with and without LMs. A brief version is in Table 3.

Table 8. Implementation details in different tasks and datasets. We show the Branchformer configurations used to generate main results in Section 4.2. T is the input sequence length.

	Aishell	Switchboard 300h	LibriSpeech 960h	SLURP		Speech Commands
				intent	entity	
Frontend						
window length	512	400	512	512	512	400
hop length	128	160	256	128	128	160
SpecAug						
time warp window	5	5	5	5	5	5
num of freq masks	2	2	2	2	2	2
freq mask width	(0, 27)	(0, 30)	(0, 27)	(0, 30)	(0, 30)	(0, 30)
num of time masks	10	2	10	2	2	2
time mask width	(0, 0.05T)	(0, 40)	(0, 0.05T)	(0, 40)	(0, 40)	(0, 40)
Architecture						
feature size d	256	256	512	512	512	256
hidden size d_{hidden}	2048	2048	2048	3072	2048	2048
attention heads h	4	4	8	8	8	4
num of encoder layers	24	24	22	18	18	24
depth-wise conv kernel	31	31	31	31	31	31
Training						
epochs	60	90	60	50	50	150
learning rate	1e-3	2e-3	2.5e-3	1e-3	1e-3	3e-4
warmup steps	35k	50k	40k	35k	35k	15k
weight decay	1e-6	1e-6	1e-6	1e-6	1e-6	1e-6
dropout rate	0.1	0.1	0.1	0.1	0.1	0.1
ctc weight	0.3	0.3	0.3	0.3	0.3	0
label smoothing weight	0.1	0.1	0.1	0.1	0.1	0.1
exponential moving average	NA	0.9999	NA	NA	NA	NA

D. Diagonality of Attention Weight Matrices

Section 4.6 analyzes the diagonality metric (Zhang et al., 2021) of the self-attention weight matrices in different encoder layers, which shows that the self-attention branches of Branchformer extract more global context compared with the original Transformer.

We follow the definition of diagonality in (Zhang et al., 2021). Let $\mathbf{A} \in \mathbb{R}^{T\times T}$ denote an attention weight matrix, where T is the sequence length. Each element a_{ij} is the attention weight between the ith and jth feature vectors in the sequence. Each row is a probability distribution, and the weights in each row sum to 1. Let's first define the centrality C_i of the ith row:

$$C_i = 1 - \frac{\sum_{j=1}^{T} a_{ij}|i-j|}{\max\limits_{1\le j\le T} |i-j|}. \tag{11}$$

Then, the diagonality D of the entire matrix is defined as the average centrality over all rows:

$$D = \frac{1}{T}\sum_{i=1}^{T} C_i. \tag{12}$$

Figure 9 plots some attention weights from Branchformer and Transformer. We can see that Transformer has more diagonal attention weights, which means these attention heads are capturing local context. Our Branchformer has fewer diagonal patterns, showing that it extracts more global relationships.

E. Results of Two-Stage Mixed Models

As mentioned in Section 4.6, we show the results using the two-stage mixed encoder. Specifically, we combined the Conformer blocks with Branchformer blocks sequentially, resulting in a two-stage encoder model. Results are presented in

Table 10.

If we use Conformer blocks first and Branchformer blocks later, the performance is better than the vanilla Conformer and is similar to our Branchformer. However, if the order is reversed, the performance degrades. These results verify our previous analysis. It is better to have interleaving local and global blocks in earlier layers and have more specialized blocks in later layers.

F. Effects of Branch Dropout

The branch dropout is introduced in Section 3.4.2. The results and discussions can be found in Section 4.7. We apply dropout to the attention branch during training so that we can prune the two-branch model for inference. This approach does not need fine-tuning or re-training. The trained model can operate in two different speeds, with one being linear and the other being quadratic w.r.t. the sequence length.

There is a trade-off between the performance of the two modes. As shown in Table 11, as we increase the dropout rate, the original model degrades in performance, but the pruned model improves. With a relatively large dropout rate, the pruned model is comparable with the same model trained from scratch.

The inference time is shown in Figure 7 in Section 4.7.

G. Preliminary Results on Machine Translation

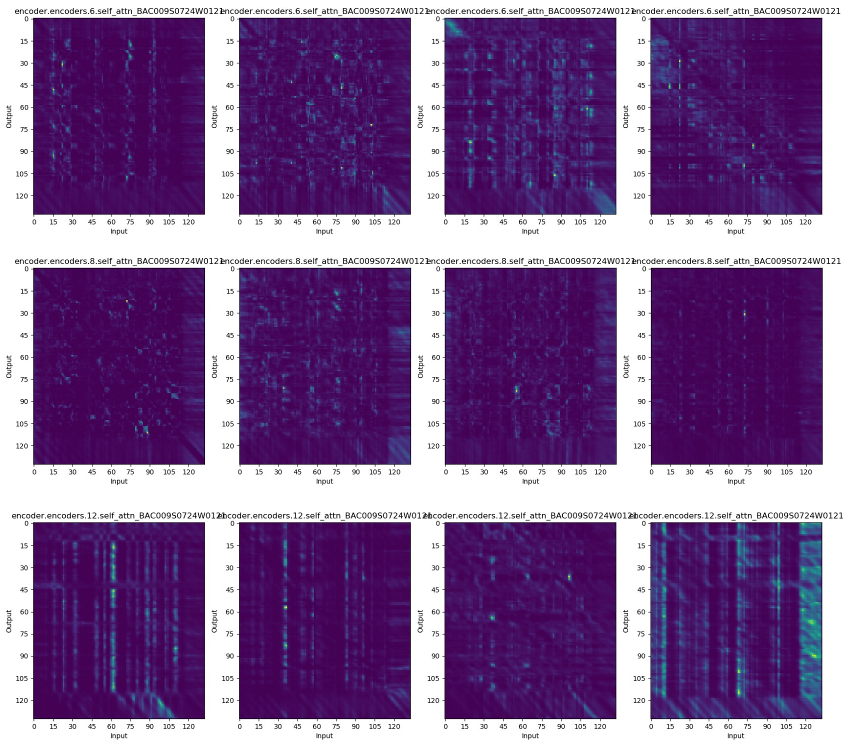
Our Branchformer is a general encoder model which might be useful for other sequence modeling tasks in addition to speech processing. We have conducted preliminary experiments on the neural machine translation (NMT) task using ESPnet. The dataset is IWSLT 14 De-En, which has been widely used for evaluating NMT systems. The results are presented in Table 12. Our proposed Branchformer achieves higher BLEU scores than the standard Transformer. We also observed that Branchformer converged faster during training. Note that the text input sequence is much shorter than a speech feature sequence, so we need to use a smaller convolution kernel size (e.g., around 5) in the cgMLP branch.

Table 9. WERs (%) on the LibriSpeech 960h dataset. Previous studies and our reproduced baselines are shown for comparison. The Transformer LM used in our experiments was downloaded from ESPnet.

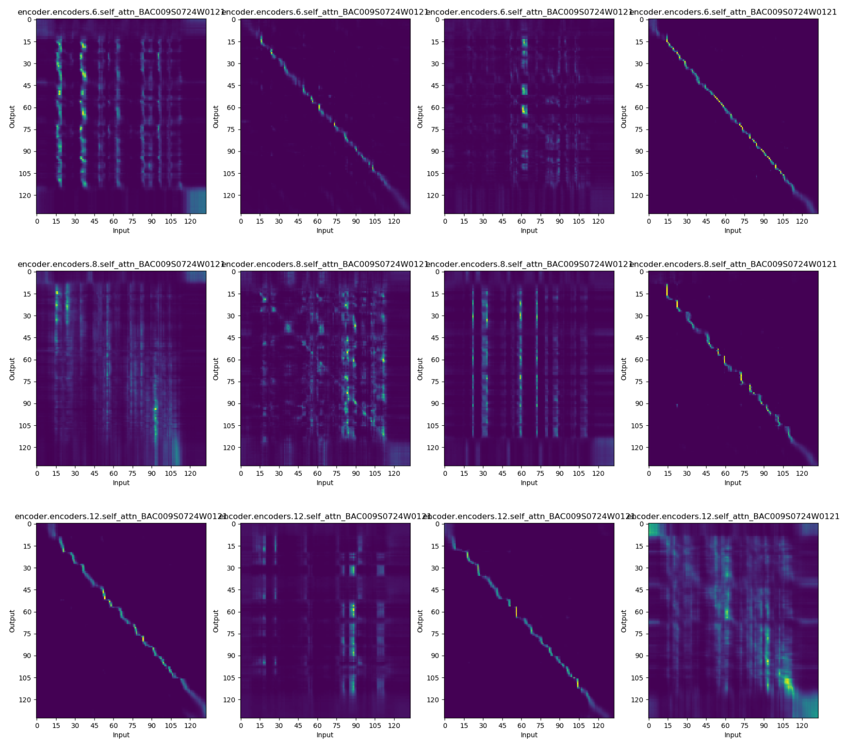
Method	Params (M)	LM	WER (%)			
			dev		test	
			clean	other	clean	other
Park et al. (2019)						
LSTM	-	-	-	-	2.8	6.8
LSTM	-	RNN	-	-	2.5	5.8
Han et al. (2020)						
ContextNet	112.7	-	2.0	4.6	2.1	4.6
ContextNet	112.7	LSTM	-	-	1.9	4.1
Gulati et al. (2020)						
Conformer	118.8	-	1.9	4.4	2.1	4.3
Conformer	118.8	LSTM	-	-	1.9	3.9
SpeechBrain (Ravanelli et al., 2021)						
Transformer	-	Transformer	-	-	2.5	5.9
WeNet (Yao et al., 2021)						
Conformer	-	-	-	-	2.7	6.5
Conformer	-	4-gram	-	-	2.7	6.0
ESPnet (Watanabe et al., 2018)						
Transformer	99.4	RNN	2.3	5.9	2.5	6.2
Conformer	116.2	-	2.3	6.1	2.6	6.0
Conformer	116.2	Transformer	1.9	4.6	2.1	4.7
Our Baselines (reproduced based on ESPnet)						
cgMLP	108.0	-	2.3	6.3	2.6	6.1
Transformer	109.6	-	2.5	6.5	2.8	6.5
Conformer	116.2	-	**2.2**	5.6	2.5	**5.5**
cgMLP	108.0	Transformer	1.9	4.5	**2.1**	4.6
Transformer	109.6	Transformer	2.1	4.7	2.3	5.0
Conformer	116.2	Transformer	**1.8**	4.3	**2.1**	**4.5**
Our Proposed Model						
Branchformer	116.2	-	**2.2**	**5.5**	**2.4**	**5.5**
Branchformer	116.2	Transformer	1.9	**4.2**	**2.1**	**4.5**

Table 10. Results of the two-stage encoder on Aishell.

Method	Layers	Params (M)	CER (%)	
			dev	test
Single-stage (using one type of encoder)				
Conformer	12	46.3	4.24	4.62
Branchformer	24	45.4	4.19	**4.43**
Two-stage (using two types of encoders sequentially)				
Conformer + Branchformer	6+12	45.8	**4.18**	4.47
Branchformer + Conformer	12+6	45.8	4.35	4.57



(a) Branchformer



(b) Transformer

Figure 9. Examples of the self-attention weights in Branchformer and Transformer. Transformer has more diagonal attention weights, which means the attention blocks are capturing more local dependencies in a sequence.

Table 11. CERs (%) on Aishell with different dropout rates for the self-attention branch. The two branches are merged using weighted average. We apply branch dropout during training. For inference, we can employ the original Branchformer model with both branches, or prune it by removing the attention branch.

Branch Dropout Rate	Original Model		Pruned Model	
	dev	test	dev	test
0.0	4.23	4.61	98.74	98.78
0.3	4.23	4.63	5.09	6.13
0.5	4.31	4.72	4.85	5.63
0.6	4.30	4.65	4.68	5.29
0.7	4.40	4.80	4.67	5.22
0.8	4.46	4.91	4.60	5.10
Conformer	4.24	4.62	-	-
cgMLP	4.61	5.15	-	-

Table 12. BLEU scores of the machine translation task on a widely used corpus, IWSLT 14 De-En. Our experiments are based on ESPnet.

Method	Params (M)	BLEU	
		valid	test
Raunak et al. (2020)			
Transformer	42	33.44	32.15
Our Results Using ESPnet			
Transformer	41.8	34.10	33.13
Branchformer	43.4	**34.96**	**33.72**