Article

A 2D Convolutional Gating Mechanism for Mandarin Streaming Speech Recognition

Xintong Wang [1] and Chuangang Zhao [2,*]

[1] College of Science, Beijing Forestry University, Beijing 100083, China; xintongwang@bjfu.edu.cn
[2] School of Information Science & Technology, Beijing Forestry University, Beijing 100083, China
* Correspondence: zhaochuangang@bjfu.edu.cn

Abstract: Recent research shows recurrent neural network-Transducer (RNN-T) architecture has become a mainstream approach for streaming speech recognition. In this work, we investigate the VGG2 network as the input layer to the RNN-T in streaming speech recognition. Specifically, before the input feature is passed to the RNN-T, we introduce a gated-VGG2 block, which uses the first two layers of the VGG16 to extract contextual information in the time domain, and then use a SEnet-style gating mechanism to control what information in the channel domain is to be propagated to RNN-T. The results show that the RNN-T model with the proposed gated-VGG2 block brings significant performance improvement when compared to the existing RNN-T model, and it has a lower latency and character error rate than the Transformer-based model.

Keywords: streaming speech recognition; RNN-Transducer; gating mechanism; VGG

Citation: Wang, X.; Zhao, C. A 2D Convolutional Gating Mechanism for Mandarin Streaming Speech Recognition. *Information* **2021**, *12*, 165. https://doi.org/10.3390/info12040165

Academic Editor: Ricardo Ribeiro

Received: 7 March 2021
Accepted: 8 April 2021
Published: 12 April 2021

1. Introduction

With the development of artificial intelligence, automatic speech recognition (ASR) has contributed to improving the efficiency of human productive activities (e.g., recording meetings, automatically captioning videos, interacting with modern smart devices by sending voice commands directly, etc.). The ASR process generally converts the speech signals into symbol sequences; for example, the speech signal can be an utterance from a speaker, and the symbol sequence is its corresponding text. Whereas, offline ASR systems take the entire utterance as input to produce output symbols, streaming ASR systems process the speech signals as a streaming input, which means that hypotheses are generated as soon as possible once the first frame has arrived. We are interested in such low latency systems not only for ASR systems, but also some downstream tasks, such as spoken dialogue systems [1] and real-time translation systems [2].

Over the past few years, some end-to-end models for offline applications [3–7] have gained performance comparable to that of humans. However, these models cannot be directly applied to real-time scenarios because of their high latency. In contrast, recurrent neural networks (RNNs) are a natural architecture for building such a streaming model, which produces output that relies only on the current input and previous state history. Several models employing RNNs with LSTM [8] cells for streaming purposes have been proposed previously, including the recurrent neural aligner (RNA) [9], neural transducer [10], and RNN-Transducer (RNN-T) [11–13]. RNN-T is very well suited for on-device applications because it has the ability to perform streaming, high-accuracy, and low-latency [14].

However, the sequential nature of RNNs also restricts RNN-T to its input at the current time step, missing future information. Therefore, several models based on attention mechanisms have been proposed to make it possible for Transducer models to exploit contextual information. Transformer-Transducer (T-T) [15,16] has been proposed on speech recognition, with Transformer [17] becoming the state-of-the-art approach in the language modeling and machine translations fields [18–20]. They replaced LSTM with the encoder part of Transformer, which mainly includes multi-head attention mechanisms, feedforward networks, and layer normalization, have been proposed on speech recognition. Experiments that are based on T-T show that the accuracy of the streaming model considering contextual information is comparable to that of the offline models. Truncated self-attention adopted in [15] and masked self-attention adopted in [16] both reduce the error rate of the streaming model.

In general, the T-T model requires a deep Transformer. If each layer of the Transformer calculates the attention scores of the input sample points with the same context range, e.g., in [16], each layer masks the same number of future speech frames, the deep transformer will superimpose a high-latency. One solution is to make the context extracting mechanism independent of the deep network, acting only as an input layer. Thus, the depth-wise convolutional network is well suited as the so-called input layer.

The output of the depth-wise convolutional network incorporates the spatial and channel-domain features through its perceptual field. The multi-channel feature is a unique property of the convolutional network, also known as the width of the convolutional network, which is determined by the number of convolutional kernels. That is, the more convolutional kernels there are, the more channel-wise features can be extracted. Several studies and experiments [21,22] have shown that the shallow wide convolutional networks outperform the deep narrow convolutional networks for these tasks.

In this paper, we propose an Encoder architecture for Transducer that incorporates the properties of convolutional networks to extract contextual information and the ability of LSTM to learn historical information. Our model is trained on the AISHELL-1 [23] dataset, containing over 170 h of speech being recorded by 400 speakers in a quiet office and obtains a character error rate (CER) of 12.9%, outperforming the previously proposed LSTM-based Transducer models [11–13]. When compared with the Transformer-based Transducer model with the same convolutional networks, our model only has the latency of the model employed unidirectional Transformer, but it achieves a comparable CER of the model, which looks ahead for three frames.

The main contributions of this paper are as follows:

1. We combine convolutional networks with LSTM as the Encoder of Transducer to build a low-latency streaming speech recognition model. These convolutional networks are built in the form of VGG2 [24] networks, which are the first two layers of VGG16, a deep convolutional network architecture. Additionally, the maximum pooling layer is retained to reduce the frame rate, which improves the training efficiency.
2. We introduce a two-dimensional (2D) convolutional gating mechanism inside VGG2, called gated-VGG2, which controls what information will flow into LSTM. The gating mechanism employs half of the channel features that are generated by the convolutional network to form gate states acting on the other half of the channel features, so that twice the channel information can be learned, which improves the performance of the model.
3. There are no temporal dependencies in the gating mechanism, so that our model is easy to train in parallel.

This paper is organized, as follows, Section 2 presents the work related to this paper. Section 3 discribes the structure of the proposed model. Section 4 presents the experimental results on the AISHELL-1 dataset, and Section 6 provides concluding remarks.

2. Related Works

Developing a streaming speech recognition systems has been a hot issue in speech recognition in recent years. As already introduced in the previous section, RNN-T and T-T are two commonly used streaming models. Several pieces of research are devoted to fusing convolutional networks in the T-T model. Among the methods that were proposed in these studies, some of the methods for convolutional network enhancement come from the fields of natural language processing and computer vision.

T-T has been combined with the VGG network when it was first proposed in [15]. The VGG network plays two roles in T-T as an input layer: (1) adding relative position information to Transformer; and, (2) downsampling the input features through the pooling layer. Another study [25] also confirms that the convolutional approach is more helpful to extract the position information of the input sequence.

Whereas the convolutional network is used as the input layer in VGG T-T, the convolutional network alternates with the deep Transformer layer in Conv. Transformer-Transducer (ConvT-T) [26]. The convolutional network and Transformer form three blocks in ConvT-T, with the convolutional network in the latter two blocks incorporating more implicit features. Moreover, there is only unidirectional Transformer layers in ConvT-T, which achieves a low latency model.

Conformer [6] is designed as an architecture with a multi-head attention module and a convolutional module, and a pair of feedforward network modules. The multi-head attention module and feedforward network module follow the form of the Transformer, and the convolutional module is the key part of the Conformer, which is mainly a depth-wise convolutional layer that is sandwiched between two point-wise convolutional layers. Specifically, the conformer chooses gated linear units (GLU) [27] as the activation function for the first point-wise convolutional layer. The output channels of the first point-wise convolutional layer are twice the number of input channels, and the GLU resizes the output features to exactly the size of the input features, while it contains two times the channel information. In fact, the number of input channels is counted as the feature dimension at the current time step, so GLU is a feature frequency-wise gating mechanism in Conformer.

Researches on convolutional channels have focused on making convolutional networks more expressive by enhancing or suppressing some specific channel features. Some of the approaches that are derived from the above studies have become essential modules for convolutional networks, e.g., NetInNet [28] and SEnet [29]. NetInNet proposed 1x1 convolutional networks, in which the weights are learned for channels on a specific task, In contrast, SEnet added a gating mechanism to channels, which learned gating states through the global average of the features of that channel as the initial value.

In our work, we apply the convolution method to RNN-T. Similar to Conformer, GLU is chosen as the channel gating mechanism, but our GLU really acts on the multi-channel features that are generated by a set of convolutional kernels, which is a channel-wise gating mechanism. Additionally, we also experimentally tested another gating mechanism: gated tanh units (GTU). The results are shown in Section 4, where GTU outperforms GLU in our proposed model.

3. Transducer

Consider a model that consists of an Encoder, a Prediction network, and a Joint network, as illustrated in Figure 1. Given an input speech feature sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$ of length T and the output symbol sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_U)$ of length U, the Encoder encodes the inputs $x_{1:t}$ to obtain the acoustic feature representation f_t and the Prediction network encodes the symbols $y_{0:u-1}$ to produce the symbol representation g_{u-1}. We denote that $\mathcal{Y}$ is the output symbol space that consists of K symbols, ϕ is the blank symbol, indicating that it outputs nothing at the current time-step, and the extended output space $\bar{\mathcal{Y}} = (\phi \bigcup \mathcal{Y})$. For streaming, a Joint network would produce a probability distribution $P(k|t, u-1)$ over $\bar{\mathcal{Y}}$ for each combination of f_t at input time-step t and g_{u-1} at output time-step $u-1$, which finally goes through the softmax layer to produce a symbol y_u that will be the next input of the Prediction network if it is a non-blank symbol. Otherwise, g_{u-1} fuses with the next frame f_{t+1} to keep on predicting y_u. The Encoder can be expressed as

Equations (1)–(5), where y_0 is the ϕ and superscript k is the k-th element of the vectors in Equation (3).

$$f_t = Encoder(x_{1:t}), \quad t = 1, 2, \ldots, T, \tag{1}$$
$$g_{u-1} = Pred(y_{0:u-1}), \quad u = 1, 2, \ldots, U, \tag{2}$$
$$h(k, t, u) = Joint(f_t, g_{u-1}), \tag{3}$$
$$P(k \in \bar{\mathcal{Y}} | t, u) = softmax(h(k, t, u)), \tag{4}$$
$$y_u = \arg\max_k P(k \in \bar{\mathcal{Y}} | t, u). \tag{5}$$

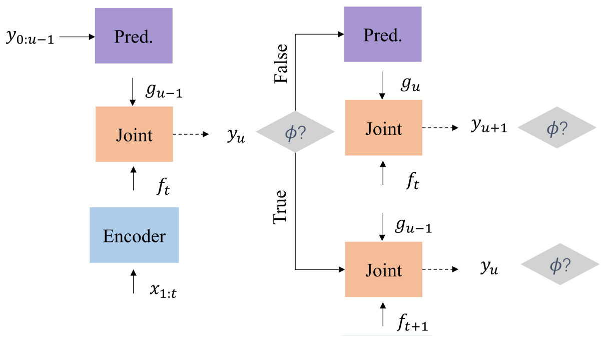

Figure 1. Tranducer decoding process.

3.1. RNN-Transducer

RNN-Transducer (RNN-T) employed LSTM for both Encoder and the Prediction network. The version of LSTM used in this paper is implemented according to the following composite function:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}), \tag{6}$$
$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right), \tag{7}$$
$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right), \tag{8}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}), \tag{9}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t, \tag{10}$$
$$h_t = o_t \odot \tanh(c_t), \tag{11}$$

where h_t and h_{t-1} are the hidden states at time-step t and $t-1$, c_t and x_t are the cell states and input at time-step t, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, repectively. The W_{ij} and b_{ij} refer to the learnable weights and bias between units with the index of gate name i and j. σ is the sigmid function, and $\odot$ is the Hadamard product. The previous cell states are not involved in the formation of the gates, and this is to allow the model to have fewer parameters, which is different from LSTM in [11,12].

The joint network is implemented in the form

$$Joint(f_t, g_{u-1}) = W_o(W_f f_t + W_g g_{u-1} + b_i) + b_o, \tag{12}$$

where W_f, W_g, W_o, b_i, and b_o are the learnable weights and biases, respectively.

3.2. Training

Transducer introduces ϕ in the hypothesis to align speech sequences and symbol sequences, called an alignment $\boldsymbol{a}$. By removing ϕ, the alignment $\boldsymbol{a}$ can be folded into a corresponding symbol sequence $\boldsymbol{y}$ (e.g., $\boldsymbol{a} = (\phi, y_1, \phi, y_2, y_3, \phi)$ is equivalent to $\boldsymbol{y} =$

(y_1, y_2, y_3)). Given an input speech sequence $\boldsymbol{x}$ and a target sequence $\boldsymbol{y}$, the sum over all conditional probabilities of the alignments is defined as the probability of generating $\boldsymbol{y}$

$$P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{a} \in A} P(\boldsymbol{a}|\boldsymbol{x}), \tag{13}$$

where A is a set consisting of all alignments equal to $\boldsymbol{y}$.

According to the forward-backward algorithm, the forward probability $\alpha(t,u)$ is defined as

$$\alpha(t,u) = P(y_1, y_2, \ldots, y_u | x_1, x_2, \ldots, x_t) \tag{14}$$
$$= \alpha(t-1,u)P(\phi|t-1,u) + \alpha(t,u-1)P(y_u|t,u-1). \tag{15}$$

Additionally, the backward probability $\beta(t,u)$ is defined as

$$\beta(t,u) = P(y_{u+1}, y_{u+2}, \ldots, y_U | x_t, x_{t+1}, \ldots, x_T) \tag{16}$$
$$= \beta(t+1,u)P(\phi|t+1,u) + \beta(t,u+1)P(y_{u+1}|t,u). \tag{17}$$

Thus, $P(\boldsymbol{y}|\boldsymbol{x})$ can be rewritten as the sum of the products of the forward and backward probabilities of all points that satisfy $\forall n = t+u$, where $1 \leq n \leq T+U$.

$$P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{t+u=n} \alpha(t,u)\beta(t,u). \tag{18}$$

The training model is to minimize the loss $-\ln P(\boldsymbol{y}|\boldsymbol{x})$ of the target sequence $\boldsymbol{y}$.

4. Extension to Gated-VGG2 RNN-T

So far, we have considered the RNN-T model in which each output conditioned its corresponding historical information. This model is too restrictive to consider contextual information of current frame. In this section, we extend RNN-T to include convolutional networks, pooling layers, and a gating mechanism as an input layer to fuse spatial and channel information of the input features, as illustrated in Figure 2. The above model could be called a gated-VGG2 RNN-T, since the encoder consists of a gated-VGG2 block and an N-layer LSTM, and the Prediction network is the multilayer LSTM.

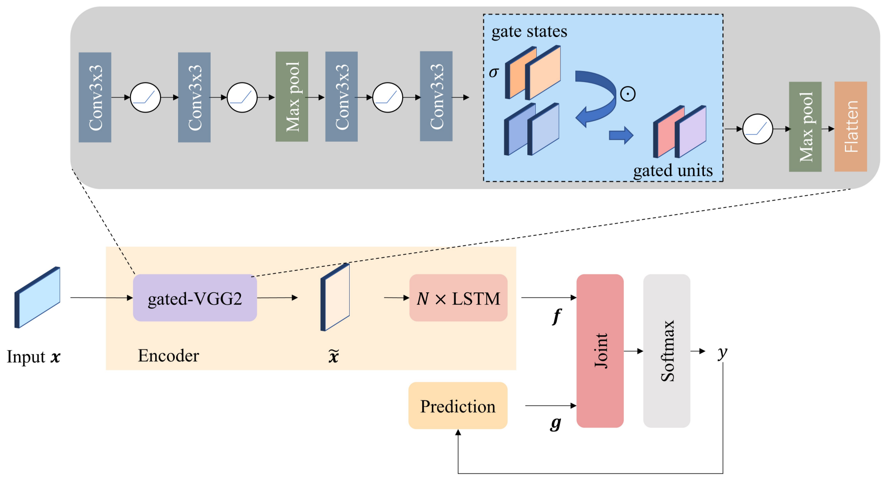

Figure 2. The architecture of proposed Transducer model.

The gated-VGG2 block is an architecture that is designed based on VGG2, the first two layers of VGG16, and a 2D convolutional gating mechanism. The VGG2 network is organized, as follows: there are four layers of convolutional networks, each of which applies the rectified linear unit (ReLU) [30] as the nonlinear activation function, and each two-layer convolutional network is followed by a maximum pooling layer. In our work, the gating mechanism is inserted between the last convolutional network and its activation function. When considering the input speech features as single-channel features, a set of convolutional networks will output multi-channel features. The gating mechanism executes element-wise multiplication of channel-specific spatial domain features, so we call it a 2D convolutional gating mechanism. After a series of convolutional networks, pooling layers, and the gating mechanism, we denote the output of the gated-VGG2 block as $\tilde{\boldsymbol{x}}$.

4.1. Convolutional Network

Each convolutional layer maps the input features $\boldsymbol{x}$ with C_{in} channels to output features $\boldsymbol{h}$ with C channels as Equation (19), where s is the s-th input channel, $\boldsymbol{k}_c$ is the parameters of the c-th kernel, and $\boldsymbol{h}_c$ is the c-th channel output. For simplicity of expression, the bias is neglected in the formula.

$$\boldsymbol{h}_c = \text{Conv}(\boldsymbol{x}) = \boldsymbol{k}_c * \boldsymbol{x} = \sum_{s=1}^{C_{in}} \boldsymbol{k}_c^s * \boldsymbol{x}^s. \tag{19}$$

4.2. Activation Function

We only consider the ReLU function as the nonlinear activation function for the convolutional network, because it has a constant gradient of 1 in regions that are larger than 0, which could avoid down-scaling the gradient, leading to gradient vanishing. The ReLU function is calculated, as follows, where $\boldsymbol{x}$ is the output of the previous convolutional layer.

$$\text{ReLU}(\boldsymbol{x}) = \max\{0, \boldsymbol{x}\}. \tag{20}$$

4.3. Max Pooling Layer

The max pooling layer outputs the maximum value within the receptive field. We denote that i is the time domain index, j is the frequency domain index , m, s are the kernel size and stride in the time domain, and n, d are the kernel size and stride in the frequency domain. Given input $\boldsymbol{x}$, the output of the max pooling layer is

$$\text{MaxPool}(\boldsymbol{x}) = \max\{x^{i,j}, x^{i+m-1,j}, x^{i,j+n-1}, x^{i+m-1,j+n-1}\}. \tag{21}$$

If the size of the input feature $\boldsymbol{x}$ is $T \times D$, where T and D represent the dimensions in the time domain and frequency domain, respectively, the dimension $T' \times D'$ of the output feature is

$$T' = \lceil \frac{T-m}{s} \rceil + 1, \tag{22}$$

$$D' = \lceil \frac{D-m}{d} \rceil + 1, \tag{23}$$

where $\lceil \cdot \rceil$ is the round up function.

4.4. 2D Convolutional Gating Mechanism

Employing the gating mechanism to control the information flow in a network has proved successful for RNNs. The input gate and forget gate of LSTM will have to learn when to release and write scaled information to the memory unit, and the output gate will have to learn what information needs to be trapped in the memory unit. Without these gates, LSTM would easily harm learnable short-term memories by storing long-term memories. In this paper, we introduce a two-dimensional (2D) convolutional gating mechanism to control what features can flow to the LSTMs. See Figure 3 (where $C_{in} = C = 4$ for simplicity), where input after convolutional operators is split in half along channel-dimension to form $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. Only half of the multi-channel features could be output, where the other half features will be transformed into gate states by a sigmoid function. Subsequently, these gate states act on the other half of the channel features to generate gated units, which are the output of the gating mechanism.

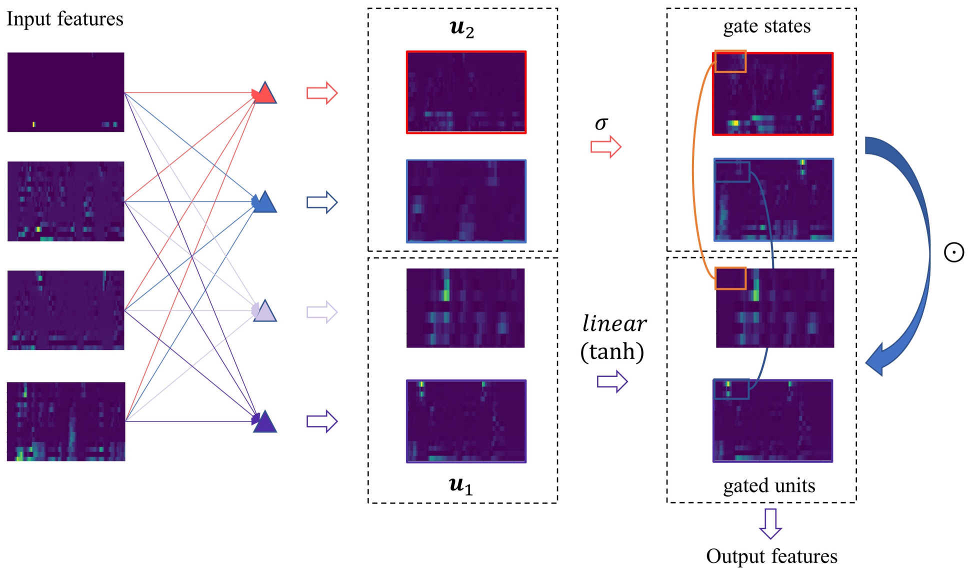

Figure 3. Two-dimensional (2D) convolutional gating mechanism.

Inspired by the work of [27], we consider both gated linear units (GLU) and gated tanh units (GTU) forms of gating mechanism to produce output $\boldsymbol{o}$ in Equations (24) and (25), where σ is the sigmoid activation function and $\odot$ is the Hadamard product between two matrices.

$$\mathrm{GLU} := \boldsymbol{u}_1 \odot \sigma(\boldsymbol{u}_2), \tag{24}$$
$$\mathrm{GTU} := \tanh(\boldsymbol{u}_1) \odot \sigma(\boldsymbol{u}_2). \tag{25}$$

The local gradients of GLU and GTU are calculated as

$$\nabla[\tanh(u_1) \odot \sigma(u_2)] = \tanh'(u_1)\nabla u_1 \odot \sigma(u_2) + \tanh(u_1) \odot \sigma'(u_2)\nabla u_2, \tag{26}$$
$$\nabla[u_1 \odot \sigma(u_2)] = \nabla u_1 \odot \sigma(u_2) + u_1 \odot \sigma'(u_2)\nabla u_2. \tag{27}$$

where both the values of tanh and tanh$'$ are between 0 and 1. Then we can get the corollary:

$$\nabla[\tanh(u_1) \odot \sigma(u_2)] < \nabla[u_1 \odot \sigma(u_2)]. \tag{28}$$

Essentially, the gradient in the backpropagation wil be downscaled, which may lead to a gradient vanishing when using the GTU, and the GLU does not have downscaled factors and, therefore, it can avoid this problem better.

However, the tanh function scales the input features to within the interval of $[-1,1]$ with mean 0, which can be seen as a data normalization operation that makes it easier for the model to converge to an optimal value in training, e.g., we found the better performance of GTU in our experiments.

Given the input speech features $\boldsymbol{x}$, the gated-VGG2 block output $\tilde{\boldsymbol{x}}$ according to the following combination of equations

$$\boldsymbol{h}^1 = \text{Conv}(\text{Conv}(\boldsymbol{x})), \tag{29}$$
$$\boldsymbol{u}^1 = \text{MaxPool}(\text{ReLU}(\boldsymbol{h}^1)), \tag{30}$$
$$\boldsymbol{h}^2 = \text{Conv}(\text{Conv}(\boldsymbol{u}^1)), \tag{31}$$
$$\boldsymbol{g} = \text{Gating}(\boldsymbol{h}^2), \tag{32}$$
$$\boldsymbol{u}^2 = \text{MaxPool}(\text{ReLU}(\boldsymbol{g})), \tag{33}$$
$$\tilde{\boldsymbol{x}} = \text{Flatten}(\boldsymbol{u}^2), \tag{34}$$

where $\tilde{x}$ is obtained by flattening the multi-channel features of u^2 into 1-dimensional features at every timestep.

5. Latency

In our proposed model, all of the latency comes from the convolutional layers and the pooling layers. We set the kernel size to 3×3 with stride 1 in the convolutional layer and 2×2 with stride 2 in the pooling layer. A padding is added to both ends of the input sequence before each convolutional operation. Generating $\tilde{x}_0$ actually requires waiting for x_6 to arrive, as shown in Figure 4. A latency of 60 ms will be introduced if the frame rate of the input features is 10 ms.

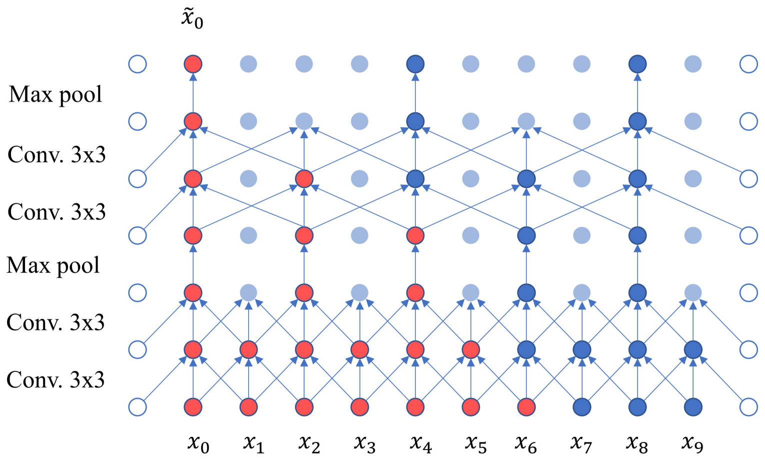

Figure 4. The latency of our proposed model.

6. Experiment

6.1. Corpus

We use the public AISHELL-1 Mandarin speech corpus for experiments. Table 1 shows the details of the corpus.

Table 1. AISHELL-1 Mandarin speech corpus.

AISHELL-1	Duration	Speaker	Utterance
train	150 h	340	120,098
validation	10 h	40	14,326
test	5 h	20	7176

6.2. Hyperparameter Setting

In all of the experiments, we extract the 80-dimensional log Mel-filter banks as features on 25 ms, with a 10 ms shift, and normalize to zero-mean and unit-variance.

Table 2 shows the Encoders used in all experiments. For (Bi)LSTM, hs represents the hidden layer size of the LSTM. For Transformer, d represents the input size, h represents the number of heads of the multi-head attention, and u represents the size of the feedforward network. The maximum pooling layers in our (gated-)VGG networks are both of size 2×2 with a stride of 2 to achieve a lower frame rate of 40 ms. The outputs in the first and second layers of the (Bi)LSTM are downsampled by a factor of 2, respectively, to bring the model to the same frame rate.

The corpus consists of 4233 Chinese characters (including “"blank” and “unk” tags), each of which is represented using a 320-dimensional embedding. We choose two-layer LSTM with 1024 hidden units as the Prediction network, and the size of the Joint network is 320. For the training step, we utilized adadelta [31] optimizer with an initial learning rate of 1.0. All of the models are trained 20 epochs on the training set and cross-validated using the validation set after each epoch. The training will stop early, if there is no performance improvement for three consecutive times. All the models in our experiments were built using the Espnet [32] toolkit.

Table 2. Encoder hyperparameters.

BiLSTM	**LSTM**	**VGG2**	**Gated-VGG2**	**Transformer**
4× BiLSTM hs = 640	5× LSTM hs = 1024	Conv3-64 Conv3-64	Conv3-64 Conv3-64	12× Transformer d = 512, h = 4, u = 2048
		maxpool	maxpool	
		Conv3-128 Conv3-128	Conv3-256 Conv3-256	
		maxpool	maxpool	

6.3. Performance

6.3.1. Gating Mechanism

We evaluated the effect of GLU and GTU, respectively. Table 3 shows the results; a lower CER is obtained using GTU than GLU on the model with our proposed gated-VGG2 block. We present the scaling of local gradients by the tanh function and the possible impact of the normalization of the tanh function on the performance in 3.1.5. The results show that the effect of local gradient scaling on the model is cancelled out.

Table 3. Comparison of two gating mechanisms: gated linear units and gated tanh units on CER.

Gating Mechanism	**CER(Test)**	**Sub.**	**Del.**	**Ins.**
GTU	12.9	11.9	0.7	0.3
GLU	13.1	12.1	0.7	0.3

6.3.2. Beam Search

In the decoding stage, we use the beam search algorithm in [11], where the size of the beam will affect the decoding speed and performance of the model. Figure 5 shows the decoding results of our proposed model (GTU) with an increasing beam width from 0 to 10, where a beam width of 0 represents greedy decoding. The results show that a beam width of 5 can achieve the balance of accuracy and cost.

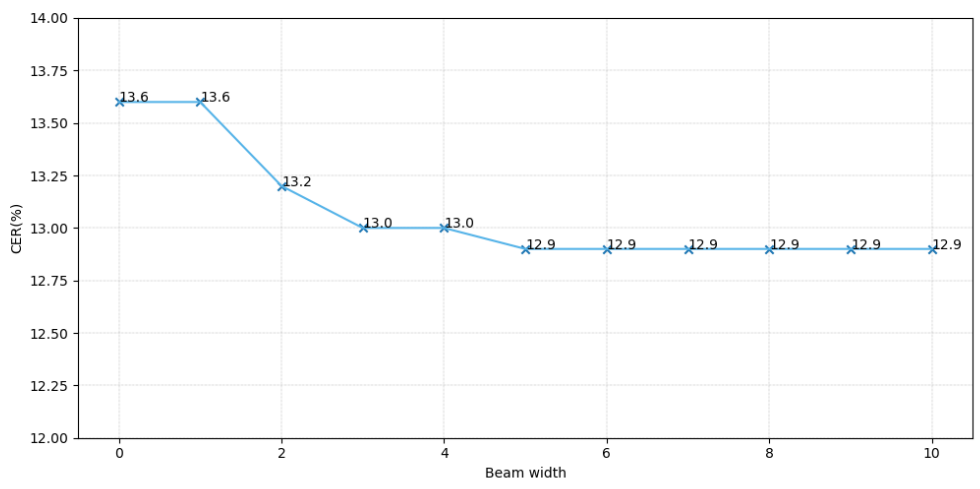

Figure 5. Decoding results of different beam width in beam search.

6.3.3. End-to-End Model

Table 4 shows the character error rate (CER) of our model (GTU) and other LSTM-based models. All of the models use the same structure and hyperparameters for the Prediction and Joint networks. In the decoding stage, we use a beam search of beam size 5 without language model. The streaming model that is based on our proposed model achieves the lowest CER and it outperforms the offline model with BiLSTM Encoder.

Table 4. Transducer models on the aishell-1 data set.

Encoder	CER(Test)	Sub.	Del.	Ins.
BiLSTM	13.2	12.3	0.6	0.3
LSTM	17.2	15.3	1.4	0.5
+ VGG2	13.4	12.4	0.7	0.3
+ Gated (ours)	12.9	11.9	0.7	0.3

Secondly, we compare the performance of our case with the Transformer-based model. In particular, to keep the same frame rate, we compared with the model with a VGG2-Transformer Encoder, which is a combination of VGG2 and the 12-layer Transformer in Table 2. As shown in Table 5, we test the model in full attention, unidirectional attention and with different sizes of lookahead, respectively. Increasing the lookahead sizes in the Transformer layers is effective in reducing CER, but it introduces a significant amount of latency. The CER of our proposed model is comparable to that of the lookahead three-frame model, with only unidirectional latency.

Table 5. Comparison of gated-VGG2 LSTM and streaming Transformer in terms of latency and CER.

Encoder	Latency	CER(Test)	Sub.	Del.	Ins.
gated-VGG2 LSTM	60 ms	12.9	11.9	0.7	0.3
VGG2-Transformer [15]					
Full attention	INF	11.4	10.0	1.1	0.3
unidirectional	60 ms	19.9	18.6	1.0	0.3
Lookahead 1	540 ms	16.9	15.8	0.8	0.3
Lookahead 2	1020 ms	14.6	13.5	0.8	0.3
Lookahead 3	1500 ms	13.0	12.0	0.7	0.3

7. Conclusions

In this paper, we proposed the gated-VGG2 block, a feature fusion module that is designed to capture the contextual information in streaming speech recognition through convolutional networks and enhance informative information through a gating mechanism. The experiments show that the streaming model based on the gated-VGG2 block achieves lower CER compared to other LSTM models, and has lower latency compared to a Transformer-based model with similar accuracy. In addition, the gated-VGG2 block illustrates the inability of previous VGG2 networks to adequately consider the importance of features. We hope that this insight can be useful for other tasks that rely on convolutional networks to represent features.

Author Contributions: Conceptualization, X.W.; Data curation, X.W.; Formal analysis, C.Z.; Funding acquisition, C.Z.; Software, X.W.; Supervision, C.Z.; Writing—original draft, X.W.; Writing—review & editing, C.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Selfridge, E.; Arizmendi, I.; Heeman, P.A.; Williams, J.D. Stability and accuracy in incremental speech recognition. In Proceedings of the SIGDIAL 2011 Conference, Portland, OR, USA, 17–18 June 2011; pp. 110–119.
2. Arivazhagan, N.; Cherry, C.; Te, I.; Macherey, W.; Baljekar, P.; Foster, G. Re-translation strategies for long form, simultaneous, spoken language translation. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7919–7923.
3. Chan, W.; Jaitly, N.; Le, Q.; Vinyals, O. Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 4960–4964.
4. Kim, S.; Hori, T.; Watanabe, S. Joint CTC-Attention Based End-to-End Speech Recognition Using Multi-Task Learning. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 4835–4839.
5. Karita, S.; Chen, N.; Hayashi, T.; Hori, T.; Inaguma, H.; Jiang, Z.; Someki, M.; Soplin, N.E.Y.; Yamamoto, R.; Zhang, W.; et al. A Comparative Study on Transformer vs RNN in Speech Applications. In Proceedings of the 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Sentosa, Singapore, 14–18 December 2019; pp. 449–456.
6. Gulati, A.; Qin, J.; Chiu, C.C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Pang, R.; et al. Conformer: Convolution-Augmented Transformer for Speech Recognition. *arXiv* **2020**, arXiv:2005.08100.
7. Emiru, E.D.; Xiong, S.; Li, Y.; Fesseha, A.; Diallo, M. Improving Amharic Speech Recognition System Using Connectionist Temporal Classification with Attention Model and Phoneme-Based Byte-Pair-Encodings. *Information* **2021**, *12*, 62. [CrossRef]
8. Sepp, H.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.
9. Sak, H.; Shannon, M.; Rao, K.; Beaufays, F. Recurrent Neural Aligner: An Encoder-Decoder Neural Network Model for Sequence to Sequence Mapping. In Proceedings of the Interspeech 2017: Conference of the International Speech Communication Association, Stockholm, Sweden, 20–24 August 2017; pp. 1298–1302.
10. Jaitly, N.; Sussillo, D.; Le, Q.V.; Vinyals, O.; Sutskever, I.; Bengio, S. A Neural Transducer. *arXiv* **2016**, arXiv:1511.04868.
11. Graves, A. Sequence Transduction with Recurrent Neural Networks. *arXiv* **2012**, arXiv:1211.3711.
12. Graves, A.; Mohamed, A.R.; Hinton, G. Speech Recognition with Deep Recurrent Neural Networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–30 May 2013; pp. 6645–6649.
13. Rao, K.; Sak, H.; Prabhavalkar, R. Exploring Architectures, Data and Units For Streaming End-to-End Speech Recognition with RNN-Transducer. In Proceedings of the 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Okinawa, Japan, 16–20 December 2017; pp. 193–199.
14. He, Y.; Sainath, T.N.; Prabhavalkar, R.; McGraw, I.; Alvarez, R.; Zhao, D.; Rybach, D.; Kannan, A.; Wu, Y.; Gruenstein, A.; et al. Streaming End-to-End Speech Recognition for Mobile Devices. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 6381–6385.
15. Yeh, C.F.; Mahadeokar, J.; Kalgaonkar, K.; Wang, Y.; Le, D.; Jain, M.; Schubert, K.; Fuegen, C.; Seltzer, M.L. Transformer transducer: End-to-end speech recognition with self-attention. *arXiv* **2019**, arXiv:1910.12977.
16. Zhang, Q.; Lu, H.; Sak, H.; Tripathi, A.; McDermott, E.; Koo, S.; Kumar, S. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020.

17. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
18. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
19. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the ICLR 2015: International Conference on Learning Representations 2015, San Diego, CA, USA, 7–9 May 2015.
20. Wang, Y.; Li, X.; Yang, Y.; Anwar, A.; Dong, R. Hybrid System Combination Framework for Uyghur–Chinese Machine Translation. *Information* **2021**, *12*, 98. [CrossRef]
21. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
22. Lu, Z.; Pu, H.; Wang, F.; Hu, Z.; Wang, L. The Expressive Power of Neural Networks: A View from the Width. *arXiv* **2017**, arXiv:1709.02540.
23. Bu, H.; Du, J.; Na, X.; Wu, B.; Zheng, H. AISHELL-1: An Open-Source Mandarin Speech Corpus and a Speech Recognition Baseline. In Proceedings of the 2017 20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA), Seoul, Korea, 1–3 November 2017; pp. 1–5.
24. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556
25. Mohamed, A.; Okhonko, D.; Zettlemoyer, L. Transformers with convolutional context for ASR. *arXiv* **2019**, arXiv:1904.11660.
26. Huang, W.; Hu, W.; Yeung, Y.; Chen, X. Conv-Transformer Transducer: Low Latency, Low Frame Rate, Streamable End-to-End Speech Recognition. In Proceedings of the Interspeech 2020, Shanghai, China, 25–29 October 2020; pp. 5001–5005.
27. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks. *arXiv* **2017**, arXiv:1612.08083.
28. Lin, M.; Chen, Q.; Yan, S. Network In Network. In Proceedings of the ICLR 2014: International Conference on Learning Representations (ICLR) 2014, Banff, AB, Canada, 14–16 April 2014.
29. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
30. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
31. Matthew, Z. ADADELTA: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701v1.
32. Watanabe, S.; Hori, T.; Karita, S.; Hayashi, T.; Nishitoba, J.; Unno, Y.; Soplin, N.E.Y.; Heymann, J.; Wiesner, M.; Chen, N.; et al. ESPnet: End-to-End Speech Processing Toolkit. *arXiv* **2018**, arXiv:1804.00015.