NSYSU-MITLab System for Open Automatic Speech Recognition 2020 Challenge

Hung-Pang Lin
Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
m083040013@g-mail.nsysu.edu.tw

Chia-Ping Chen
Department of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
cpchen@mail.cse.nsysu.edu.tw

***Abstract*—In this report, we describe the system Team NSYSU-MITLab has implemented for the Open Automatic Speech Recognition 2020 Challenge (OpenASR20). We proposed an end-to-end automatic speech recognition (ASR) system consists of Conformer architecture as an encoder and Transformer as a decoder. The feature encoded by Conformer also be used by connectionist temporal classification (CTC) for joint training and decoding. We also employ a speaker classifier for domain adversarial training, which can improve the robustness of the encoder. We participate in Cantonese and Vietnamese of OpenASR20 in Constrained training condition. The best performance we have achieved with the proposed methods is the word error rate of 61.4526% for Cantonese and 74.9092% for Vietnamese.**

***Index Terms*—speech recognition, Transformer, Conformer, connectionist temporal classification, domain adversarial training**

I. Introduction

Transformer [1] is a sequence-to-sequence architecture that can replace the recurrent neural networks (RNN) in previous automatic speech recognition (ASR) work [2]. [3] shows that RNN-based ASR architecture joint training and decoding with connectionist temporal classification (CTC) can improve robustness and achieve fast convergence. CTC can be used in Transformer-based ASR architecture as well [4]. Convolutions have also been successful for ASR to capture local information of input acoustic features. The encoder architecture named Conformer [5] combines convolution neural networks and Transformer to obtain both local and global context. Domain adversarial training [6] can reduce the mismatch between training data and testing data. [7] shows that adversarial training is also helpful in the ASR task. In this report, we used the Conformer encoder and Transformer decoder to build an end-to-end ASR architecture with CTC joint training and decoding. Furthermore, we applied adversarial training with speaker classifier to improve the robustness of encoder for speakers' variety. The model architecture is illustrated in Fig. 1.

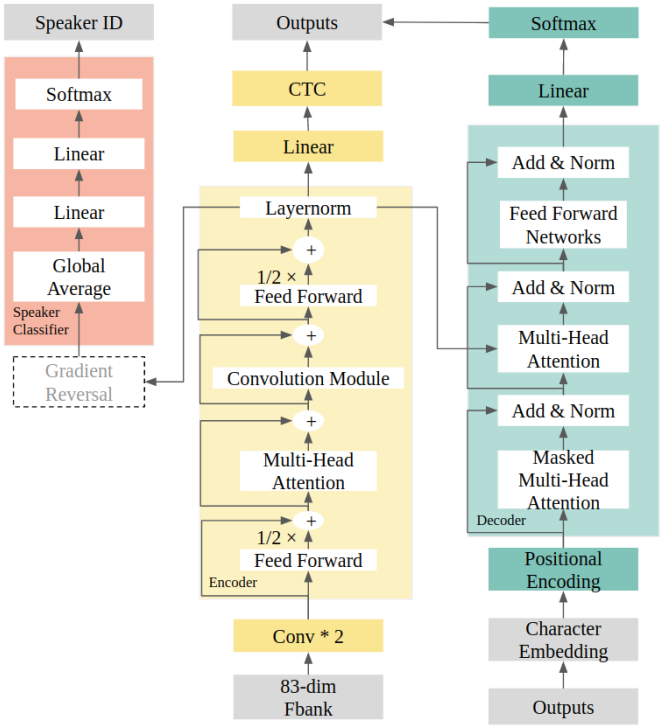

Fig. 1. Model architecture

II. Network Architecture

A. Multi-head Attention

Transformer [1] applies multi-head attention mechanism as follow:

$$\text{att}(Q,K,V) = \text{softmax}(\frac{QK^\top}{\sqrt{att}})V \tag{1}$$

$$H_h = \text{att}(QW_h^q, KW_h^k, VW_h^v) \tag{2}$$

$$\text{MHA}(Q,K,V) = [H_1, H_2, ..., H_{head}]W^{head} \tag{3}$$

where $K, V \in \mathbb{R}^{k \times att}$ and $Q \in \mathbb{R}^{q \times att}$ are input sequences for this attention layer, q is the length of Q and k is the length of K and V, att is the number of feature dimensions. $W_h^q, W_h^k, W_h^v \in \mathbb{R}^{att \times (att/head)}$ are learnable weight matrices and $H_h \in \mathbb{R}^{q \times (att/head)}$ is h-th head output ($h = 1, ..., head$),

where $head$ is the number of heads. All heads are concatenated and then multiplied by a learnable weight matrix $W^{head} \in \mathbb{R}^{att \times att}$ to get the multi-head attention output.

B. Subsample

The acoustic feature is a sequence of 80-dim FBank with 3-dim pitch $X^{fbank} \in \mathbb{R}^{T \times 83}$, where T is the length of the feature. We subsample X^{fbank} to $X^{sub} \in \mathbb{R}^{seq \times att}$ by two-layer CNN with ReLU activation, att channels, stride size 2 and kernel size 3, where seq is the length of the output sequence.

C. Encoder Architecture

Conformer [5] is a state-of-the-art encoder architecture for speech recognition, which combines convolution neural networks and Transformer modules [1] to capture both global and local information of input sequence. The architecture of the Conformer encoder is shown in the left half of Fig. 2. X^{sub} will go through several Conformer encoder layers and obtain encoded features $X_e \in \mathbb{R}^{seq \times att}$. Suppose that the input of i-th encoder block is X_i, the i-th block output is calculated as:

$$X_i^{'} = X_i + \frac{1}{2}\text{FF}(X_i) \tag{4}$$

$$X_i^{''} = X_i^{'} + \text{MHA-RPE}(X_i^{'}, X_i^{'}, X_i^{'}) \tag{5}$$

$$X_i^{'''} = X_i^{''} + \text{Conv}(X_i^{''}) \tag{6}$$

$$X_{i+1} = \text{Layernorm}(X_i^{'''} + \frac{1}{2}\text{FF}(X_i^{'''})) \tag{7}$$

where FF(), MHA-RPE(), Conv(), Layernorm() denote the feed-forward network, multi-head attention with relative sinusoidal positional encoding scheme from Transformer-XL [8], convolution module, and layer normalization [9], respectively. The output of feed-forward network is calculated as:

$$FF(X) = \text{Swish}(\text{Layernorm}(X_i)W_1^{ff} + b_1^{ff})W_2^{ff} + b_2^{ff} \tag{8}$$

where Swish() denotes the Swish activation [10], $W_1^{ff} \in \mathbb{R}^{att \times ff}$, $W_2^{ff} \in \mathbb{R}^{ff \times att}$ are learnable weight matrices, $b_1^{ff} \in \mathbb{R}^{att}$, $b_2^{ff} \in \mathbb{R}^{ff}$ are learnable bias vectors.

The convolution module starts with a pointwise convolution and a gated linear unit (GLU) [11], followed by a 1-D depthwise convolution layer. Batchnorm [12] and Swish activation are deployed after the depthwise convolution. The architecture of the convolution module is shown in the right half of Fig. 2.

D. Decoder Architecture

Transformer's decoder receives the output of Conformer encoder X_e and prefix sequence of token IDs $Y[1:u] = Y[1], ..., Y[u]$. The architecture of the decoder is shown in the right part of Fig. 1. The output of j-th decoder block is calculated as:

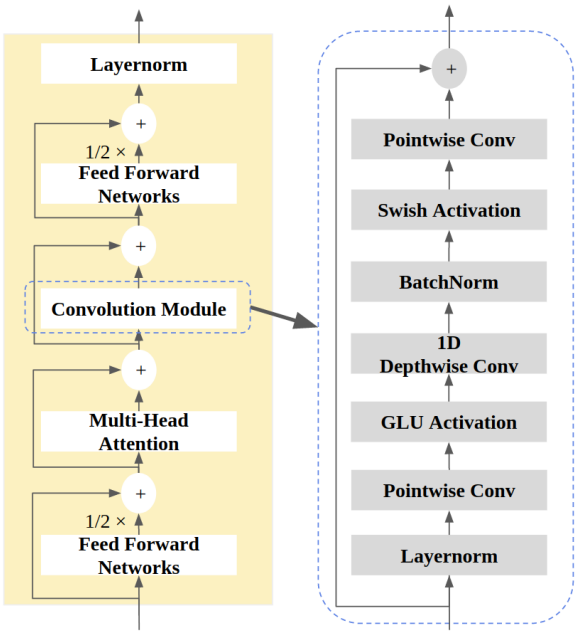

Fig. 2. Conformer encoder model architecture and convolution module

$$E = \text{Embed}(Y[1:u]) \tag{9}$$

$$Z_0 = E + \text{PE} \tag{10}$$

$$Z_j^{'} = \text{Layernorm}(Z_j + \text{MHA}(Z_j, Z_j, Z_j)) \tag{11}$$

$$Z_j^{''} = \text{Layernorm}(Z_j^{'} + \text{MHA}(Z_j^{'}, X_e, X_e)) \tag{12}$$

$$Z_{j+1} = \text{Layernorm}(Z_j^{''} + \text{FF}(Z_j^{''})) \tag{13}$$

where Embed() will transforms the sequence of the token IDs Y into a sequence of learnable embedding $E \in \mathbb{R}^{u \times att}$. PE is sinusoidal positional encoding [1], Z_j is the input of j-th decoder block. Given the token IDs $Y[1:u]$ and the output of encoder X_e, posterior probabilities of the next token ID is calculated as:

$$[p_{s2s}(Y[2]|Y[1], X_e), ..., p_{s2s}(Y[u+1]|Y[1:u], X_e)] = \text{softmax}(Z_d W^{att} + b^{att}) \tag{14}$$

$$p_{s2s}(Y|X_e) = \prod_u p_{s2s}(Y[u+1]|Y[1:u], X_e) \tag{15}$$

where Z_d is the output of the decoder, $W^{att} \in \mathbb{R}^{att \times token}$, $b^{att} \in \mathbb{R}^{token}$ are learnable parameters and $token$ is the number of token IDs.

E. Joint Training and Decoding

Joint training with CTC (connectionist temporal classification) [13] can result in faster convergence [4]. CTC layer

receives the output of the encoder X_e and then computes the probability $p_{ctc}(Y|X_e)$ as:

$$C = \text{softmax}(X_e W^{ctc} + b^{ctc}) \tag{16}$$

$$p(\pi|X_e) = \prod_{t=1}^{seq} C[t, \pi[t]] \tag{17}$$

$$p_{ctc}(Y|X_e) = \sum_{\pi \in \beta^{-1}(Y)} p(\pi|X_e) \tag{18}$$

where $W^{ctc} \in \mathbb{R}^{att \times token}, b^{ctc} \in \mathbb{R}^{token}$ are learnable parameters. π is an alignment between X_e and Y. $C[t, \pi[t]]$ is the probability of the alignment between the output token $\pi[t]$ and the t-th frame in X_e. The many-to-one mapping $\beta(\pi)$ removes all blank symbols $\varnothing$ and repeated labels from π, for example, $\beta(\text{a}\varnothing\text{aabb}) = \text{aab}$. $\beta^{-1}(Y) = \{\pi | Y = \beta(\pi)\}$ is a set of any possible π that can form Y after removing redundant symbols.

The joint multi-task loss function is calculated as:

$$L_{mtl} = -\alpha \log p_{s2s}(Y|X_e) - (1-\alpha) \log p_{ctc}(Y|X_e) \tag{19}$$

where α is a hyperparameter. During decoding, we compute the sum of log probabilities from the Transformer decoder, CTC, and language model:

$$\hat{Y} = \arg\max_{Y \in y^*} \{\lambda \log p_{s2s}(Y|X_e) + (1-\lambda) \log p_{ctc}(Y|X_e) + \gamma \log p_{lm}(Y)\} \tag{20}$$

where $p_{lm}(Y)$ denotes the language model probability of Y, λ and γ are hyperparameters, y^* is a set of output hypotheses.

F. Adversarial Training

Speakers' variety between the training set and the testing set may lead to poor recognition results. To address this problem, we employ adversarial training [6] to encourage the encoder to learn speaker-invariant representation. The output of the encoder $X_e \in \mathbb{R}^{seq \times att}$ is reshaped to $X_e^{'} \in \mathbb{R}^{att}$ by global average pooling and then received by the speaker classifier, which consists of two linear layers. A gradient reversal layer [6] is inserted before this speaker classifier. The architecture of the speaker classifier is shown in the left part of Fig. 1. The model parameters are updated as:

$$\theta_{dec} \leftarrow \theta_{dec} - \epsilon \frac{\partial L_{mtl}}{\partial \theta_{dec}} \tag{21}$$

$$\theta_{spk} \leftarrow \theta_{spk} - \epsilon \beta \frac{\partial L_{spk}}{\partial \theta_{spk}} \tag{22}$$

$$\theta_{enc} \leftarrow \theta_{enc} - \epsilon (\frac{\partial L_{mtl}}{\partial \theta_{enc}} - \beta \frac{\partial L_{spk}}{\partial \theta_{enc}}) \tag{23}$$

where θ_{enc}, θ_{dec}, θ_{spk} refer to the parameters of encoder, decoder, and speaker classifier, respectively. L_{spk} is a cross-entropy loss function for the speaker classifier, and β is a hyperparameter. Fig. 3 illustrates how adversarial training works during backpropagation.

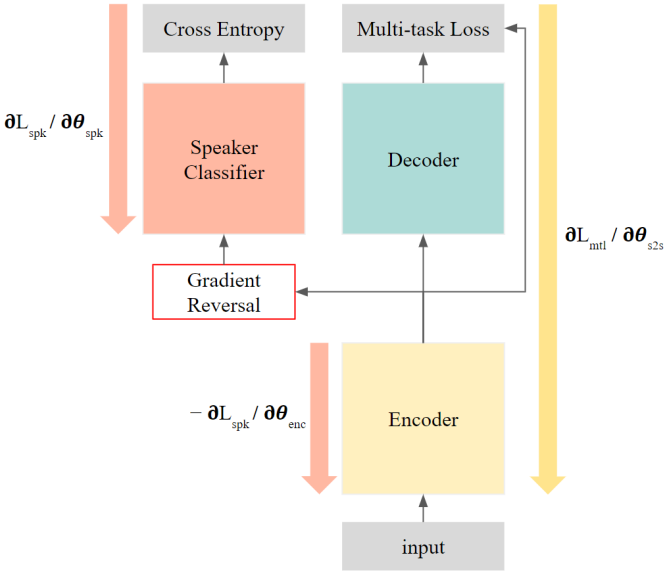

Fig. 3. Parameters updating with adversarial training

TABLE I
DETAILS OF TRAINING SET

Dataset	Language	utterances	hours
Build	Cantonese	120	20
Build-ref	Cantonese	10,229	17.89
Build-align	Cantonese	10,229	14.64
Build	Vietnamese	126	20.14
Build-ref	Vietnamese	10,176	10.98
Build-align	Vietnamese	10,176	8.82

III. EXPERIMENT

A. Dataset

The languages we participate in are Cantonese and Vietnamese. We tokenize the Cantonese dataset with character and tokenize the Vietnamese dataset with byte-pair-encoding (BPE) [14] using SentencePiece [15]. For the Build dataset in both languages, we cut the original audio file based on the time information given by the reference file. We refer to the Build dataset cut by reference file as "Build-ref". However, we noticed there are some non-speech segments at the beginning or end of the Build-ref audio files. To address this problem, we use the model trained on Build-ref to decode Build-ref and utilize the CTC blank symbol generated during decoding to estimate the time information of the non-speech segment in the audio file. We refer to the Build dataset cut by CTC alignment as "Build-align". Table. I shows the information of Build-ref and Build-align. For the Dev dataset, we cut the original audio file based on the time information given by the reference file. For the Eval dataset, we segment the original audio file every 7 seconds with 1 second overlap. We applied speed perturbation [16] and SpecAugment [17] for data augmentation.

TABLE II
MODEL HYPERPARAMETERS FOR DIFFERENT MODEL SIZES

Model	Small	Medium	Large
Num Params (M)	6.5	20.5	44.7
Encoder Layers	6	8	12
Decoder Layers	3	4	6
Attention Dimension	128	256	256
Feed-Forward Dimension	1024	1024	2048

TABLE III
THE PERFORMANCE COMPARISONS OF DIFFERENT MODEL SIZES ON THE DEV DATASET

Model	Language	hours	WER(%)
Large	Cantonese	8.47	**61.0**
Medium	Cantonese	5.73	63.6
Small	Cantonese	3.64	63.0
Large	Vietnamese	6.06	70.6
Medium	Vietnamese	4.14	71.5
Small	Vietnamese	4.21	**68.9**

TABLE IV
THE PERFORMANCE COMPARISONS OF DIFFERENT HYPERPARAMETER β ON THE DEV DATASET FOR CANTONESE

Model	Language	β	WER(%)
Large	Cantonese	0	62.9
Large	Cantonese	0.3	62.1
Large	Cantonese	0.5	61.0
Large	Cantonese	0.7	**60.7**
Large	Cantonese	1	61.8

TABLE V
RESULTS OF EVAL DATASET IN CANTONESE AND VIETNAMESE

Model	Language	Training Set	WER(%)
Large	Cantonese	Build-ref	**61.45**
Large	Cantonese	Build-align	65.15
Small	Vietnamese	Build-ref	**74.61**
Small	Vietnamese	Build-align	74.61

B. Experimental Setups

The kernel size in the depthwise convolution is 32. The inner dimension of the speaker classifier is 256. The multi-task loss weight was $\alpha = 0.3$ for the CTC joint training. We use the Adam optimizer [18] with square root learning rate scheduling [1] (25000 warmup steps, 32 minibatch size and 50 epochs). The decoding hyperparameters λ and γ are 0.5 and 0.7. We applied 10% dropout [19] on the output of each module in Conformer encoder before it was added to the module input. We also applied label smoothing [20] with a penalty of 0.1. We use a 2-layer LSTM language model with width 1024 trained on the Build dataset transcripts without any external dataset. We implement our system on the open ASR toolkit ESPnet [21]. All models are trained on 1 Nvidia GeForece GTX 1080 Ti GPU.

C. Results

First, we explore three different model sizes, small, medium, and large, by using different combinations of encoder layers, decoder block layers, attention feature dimension att, and feed-forward inner dimension ff. Table. II describes their architecture hyperparameters. The result of different model sizes tested on the Dev dataset is shown in Table. III, while other hyperparameters are kept unchanged during the comparison. We can see that the large model has the best performance in Cantonese and the small model has the best performance in Vietnamese. Table. III also shows the hours required for training.

In Table. IV, we compare the result of different hyperparameter β on the Dev dataset for Cantonese, where β controls the proportion of adversarial learning in the loss function. Adversarial training is not used when β is 0. We can see that adversarial training can improve system performance when β is not 0. The best result is obtained when β is 0.7.

Table. V shows the result of our system on the Eval dataset for Cantonese and Vietnamese. β is 0.7 in all of our submission. We only participate in Constrained training conditions. The best result in our submission is the word error rate (WER) of 61.4526% for Cantonese and 74.9092% for Vietnamese.

IV. CONCLUSION

We proposed an end-to-end ASR system consists of the Conformer encoder and Transformer decoder with CTC joint training and decoding. In addition, we employed a speaker classifier for adversarial training. We have achieved 61.4526% WER for Cantonese and 74.9092% WER for Vietnamese in the Constrained training condition with the proposed methods.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[2] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5884–5888, IEEE, 2018.

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

[4] T. Nakatani, "Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," 2019.

[5] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[7] Y. Shinohara, "Adversarial multi-task learning of deep neural networks for robust speech recognition.," in *Interspeech*, pp. 2369–2372, San Francisco, CA, USA, 2016.

[8] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

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

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

[11] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*, pp. 933–941, 2017.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[13] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[14] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[15] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[16] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[17] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

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

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[21] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proceedings of Interspeech*, pp. 2207–2211, 2018.