UCONV-CONFORMER: HIGH REDUCTION OF INPUT SEQUENCE LENGTH FOR END-TO-END SPEECH RECOGNITION

Andrei Andrusenko[1], Rauf Nasretdinov [2], Aleksei Romanenko [1,2]

[1]ITMO University, St. Petersburg, Russia
[2]STC-innovations Ltd, St. Petersburg, Russia

andrusenkoau@itmo.ru, {nasretdinov, romanenko}@speechpro.com

ABSTRACT

Optimization of modern ASR architectures is among the highest priority tasks since it saves many computational resources for model training and inference. The work proposes a new Uconv-Conformer architecture [1] based on the standard Conformer model. It consistently reduces the input sequence length by 16 times, which results in speeding up the work of the intermediate layers. To solve the convergence issue connected with such a significant reduction of the time dimension, we use upsampling blocks like in the U-Net architecture to ensure the correct CTC loss calculation and stabilize network training. The Uconv-Conformer architecture appears to be not only faster in terms of training and inference speed but also shows better WER compared to the baseline Conformer. Our best Uconv-Conformer model shows 47.8% and 23.5% inference acceleration on the CPU and GPU, respectively. Relative WER reduction is 7.3% and 9.2% on LibriSpeech test_clean and test_other respectively.

Index Terms— End-to-end speech recognition, Conformer, U-Net, intermediate CTC loss, LibriSpeech

1. INTRODUCTION

The active development of the end-to-end approaches in ASR has led to many robust neural network architectures based on the convolutional neural networks [1, 2] and the attention mechanism [3]. A Conformer model [4] developed by combining these approaches is currently a state-of-the-art solution for many ASR systems based on both end-to-end [5] and hybrid [6] approaches. However, this model has several drawbacks related to the complexity of training and low inference speed. It mainly occurs due to the quadratic complexity of self-attention with respect to the length of the processed sequence [7]. As a result, a successful training of the Conformer model and its inference in production take much time and require a lot of computational resources.

Recently, various approaches have been proposed to make the self-attention mechanism more efficient [8, 9]. The work [10] uses the Fastformer [11] model with linear complexity, which is based on additive attention. In [12] authors proposed an approach based on kernelization.

Another way to speed up the Conformer is to reduce the input sequence length. The vanilla implementation of the Conformer model has an initial convolution block that reduces the input sequence length by 4 times; we further denote the overall reduction of an input sequence length by N times as a *reduction level xN*. In Conv-Transformer [13] and Efficient Conformer [14] the authors decrease the input sequence length by 8 times placing intermediate downsampling blocks after several Conformer layers. This approach significantly reduces the number of calculations for the rest of the model without considerable loss in recognition accuracy. However, this option works only for subword output units since the reduction level x8 is too high for graphemes. It may lead to a CTC loss [15] calculation error because the number of output embeddings may become less than the number of targets in the training transcription. The same problem arises for subwords when the input sequence length is reduced by 16 times. In this case, the effect of losing some latent acoustic information is added due to the rejection of even more intermediate signal representations than it is at reduction level x8. All these factors cause the problem of model convergence.

Inspired by the U-Net architecture [16], we propose the Uconv-Conformer model – an architecture for sequentially reducing the input sequence length by 16 times, followed by restoring the reduction level to x8 for the final layers of the Conformer model. To restore a latent acoustic information discarded during progressive downsampling, we use skip connections between blocks with the same reduction level, similar to the U-Net approach. This method allows us to reduce the number of processed embeddings for intermediate Conformer layers while maintaining the final reduction level acceptable for the CTC loss calculation using subwords.

In parallel with the active testing of the final prototype of our Uconv-Conformer model, a Squeezeformer [17] paper has been released. The key idea of this work is to use a

---

[1]https://github.com/andrusenkoau/espnet/blob/master_uconv/espnet2/asr/encoder/uconv_conformer_encoder.py

x4 **x8** **x16** **x8** **x4**

| Block 1 | Block 2 | Block 3 | Block 4 | Block 5 |

Input feats → Downsampling x4 / Conformer layers ... Downsampling x2 / Conformer layers ... Downsampling x2 / Conformer layers ... Upsampling x2 / Conformer layers ... Upsampling x2 / Conformer layers
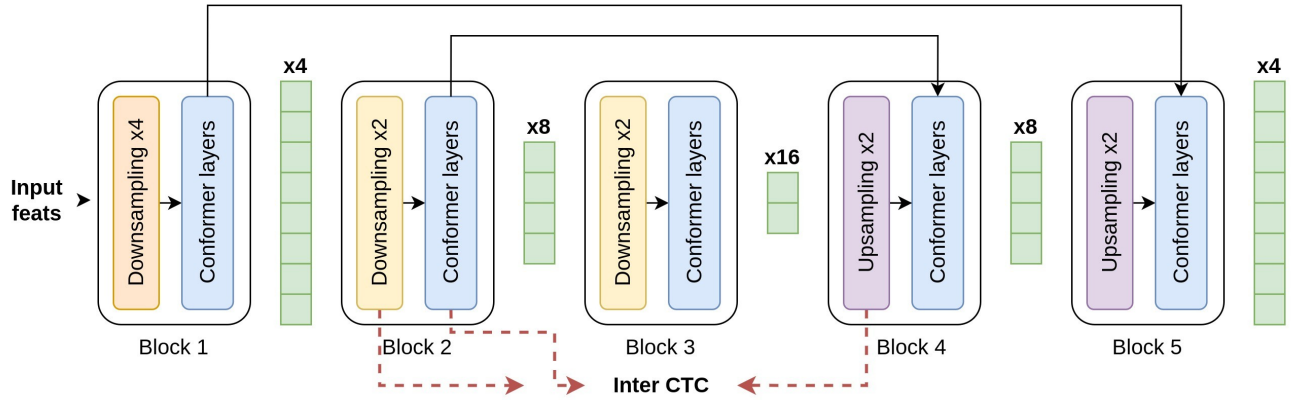
Inter CTC

Fig. 1. *The Uconv-Conformer architecture.*

temporal U-net structure similar to our approach. The main difference is that the Squeezeformer uses the intermediate reduction level x8 and the reduction level x4 for the final layers. Our approach is initially designed for higher reduction levels.

For additional comparison we implemented the Conv-Conformer architecture similar to the Conv-Transformer [13] with some simplifications. We also investigated the use of the intermediate CTC loss [18] for individual blocks of the Uconv-Conformer architecture. According to our results, it makes the training process more stable and improves the recognition accuracy for the low-resource data scenario.

As a measure of the model speed, we provide an extended comparison of the studied architectures in terms of inference speed on CPU and GPU.

2. METHODS

2.1. Uconv-Conformer architecture

The primary motivation behind the Uconv-Conformer model (Figure 1) is to reduce the length of embedding sequence processed by the intermediate part of the Conformer model. In the works [13] and [14], it is done by the sequential downsampling of the embedding sequence length by 8 times. According to our experience, reduction level x8 is more effective for subwords in English. This reduction level is too high for graphemes (the average pronouncing duration of a grapheme is less than a subword) and my lead to errors in CTC loss calculation and degradation of recognition accuracy.

It is possible to further increase the reduction level inside the network up to x16 (the largest reduction level in the model is further called as a *reduction depth*, and the reduction level for the final layer is called as a *final reduction*), but this leads to the same problems as the reduction level x8 for graphemes. To avoid this, we suggest using an upsample back to reduction level x8 for the final layers of the model in the case of subwords, that is similar to the U-Net approach. An additional upsampling block can be added for graphemes targets

to obtain the final reduction level x4. This can be useful for languages that work better with phonemes or graphemes [19].

However, adding only upsampling blocks is not enough, since some latent acoustic information is lost on the high reduction levels. To recover this loss we apply skip connections between model blocks with the same reduction level (the long dark arrows in Figure 1). Without using skip connections the recognition accuracy degrades noticeably.

As intermediate downsampling blocks (Downsampling x2), we use a sequence of three conv1d layers like in [13]. The first conv1d layer with filter size 3 and stride 1 converts the input dimension to an intermediate one (512 in our work). The second conv1d layer with filter size 3 and stride 2 reduces the input sequence length by 2 times. The final conv1d layer acts as the fully-connected layer and converts the result back to the attention dimension of the Conformer block. For the upsampling, we use the standard upsample module from PyTorch with nearest neighbor algorithm and scale factor 2.

2.2. Intermediate CTC loss

The proposed Uconv-Conformer architecture consists of several blocks operating at different reduction levels. To improve the training convergence of such a model, we analyzed the influence of the intermediate CTC loss [18] calculated at the output of such separate blocks (the red dashed arrows in Figure 1). It is necessary to bring all intermediate outputs to the same reduction level (x8 in our case) to calculate CTC loss under the same conditions. Otherwise, the network training stops converging. To calculate the final loss function, we use an expression like in [20] with lambda 0.5.

3. EXPERIMENTAL SETUP

3.1. Baseline model

As a starting point, we use the standard 12-layer Conformer encoder from the ESPnet librispeech recipe, which has 83.4M

parameters and initial downsampling block x4[2]. We refer to this model as Conformer-L. According to [6] and our experience with Conformers, such a large model size may be excessive for training on a small amount of data. Therefore, we reduced the number of model parameters to 21.8M. Now the number of conv2d convolution filters in the initial Downsampling x4 block is 64. The attention layer size is 280 with eight heads. The feed forward layer size is 1024. The conformer convolution filter size is 5. Thus we obtained a significant acceleration of the training and testing process without a critical loss in recognition accuracy. This model is referred to as the Conformer-S and used as the main baseline.

3.2. Conv-Conformer model

For additional comparison with our approach, we implemented the Conv-Conformer model similar to the Conv-Transformer [13]. The main difference is merging two time reduction steps into a single initial Downsampling x4. This model can be described as the first two blocks in Figure 1. We explored two options for the distribution of 12 layers between reduction level x4 and reduction level x8: 2-10 and 4-8. The more layers there are at reduction level x8, the faster the model operates. The size and number of Conformer layers are the same as in the baseline Conformer-S. The only difference is the presence of an additional Downsampling x2 block, consisting of three consecutive conv1d layers with the same size as in the Uconv-Conformer.

3.3. Uconv-Conformer model

We explored different reduction depths and final reduction levels for the proposed Uconv-Conformer model. The recently published Squeezeformer model can be compared to our Uconv_D8-F4 model in terms of its time reduction policy. The notation D8-F4 denotes that the reduction depth (D) is x8, and the final reduction (F) is x4. We also considered a deeper downsampling version – Uconv_D16-F4 with the same final reduction level x4. Models Uconv_D16-F8_v1 and Uconv_D16-F8_v2 have reduction depth x16 and final reduction x8. However, they have a different number of layers at each of the reduction levels. By varying the number of layers between different reduction levels it is possible to increase the model speed at the expense of its recognition accuracy and vice versa. The size and number of Conformer layers are the same as in Conformer-S. The number of parameters differs due to two additional Downsampling x2 blocks, each having about 1.4M parameters. For additional comparison with large Conformer-L we use Uconv-L_D16-F8_v1, which has the same size of the Conformer layers except for the additional Downsampling x2 blocks. Table 1 describes the pa-

rameters, time reduction policy, and layers distribution at different reduction levels.

3.4. Training details

To evaluate different scenarios with respect to the amount of available training data, we use the standard LibriSpeech [21] division into 100, 460, and 960 hours of training data. We use 80-dimensional Mel-scale log filter banks as acoustic features, calculated with 25ms window and 10ms shift. SpecAugment [22] is applied during the training process to prevent model overfitting. As acoustic targets, we use 256 BPE [23] unints obtained with SentencePiece tokenizer [24].

All models were trained for 100 epochs with Adam optimizer and weight decay 1e-6. Warmup with a peak LR value of 2e-3 at the 15th training epoch was used as a scheduler. We used the numel batch type with fixed batch bins 2M (sorting audio samples by length to reduce calculations on padded parts) and gradient accumulation is equal to 6.

CTC loss was used as the objective function. Decoding was performed using the beam search algorithm with beam 20 without using any external language model. All experiments were carried out in the ESPnet toolkit [25].

3.5. Measuring model speed performance

We provide the inference speed of the resulting model on the GPU and the CPU since many scenarios for using ASR systems still involve using the CPU. To do this, we measure the processing time of a 30-second test sample launched in one thread on the Intel Core i7-8700 and the single RTX 2080 Ti.

4. RESULTS

Table 1 provides the results for the 100h training data scenario. It can be seen that Conformer-S is quite a bit inferior to Conformer-L in recognition accuracy but significantly outperforms it in inference speed. The Conv-Conformer models show comparable WER results to Conformer-S, but have an advantage over it in inference time.

Next, we explored the different reduction depths and final reduction levels for Uconv-Conformer. The results show that all the models are faster than Conformer-S baseline, but returning to the final reduction level x4 deteriorates recognition accuracy. At the same time, the Uconv-Conformer models with final reduction level x8 show substantial performance boost and provide the best WER. Arranging more layers at reduction levels x8 and x16 allows to accelerate the model in terms of CPU inference speed. However, a slight accuracy degradation occurs due to the loss of some acoustic information at a high reduction level. Noteworthy, Uconv_D16-F8_v1 consumes about 28% less GPU memory during training than Conformer-S, which allows to increase the batch size and obtain additional training speed acceleration.

Table 1. *Results (WER, %) for training on LibriSpeech 100h. Layers distribution shows the number of Conformer layers at each reduction level. Inference speed on CPU/GPU presented in comparison with Conformer-S model. A negative value indicates an improvement, and a positive value – a degradation by X percent.*

Model	Params M	Reduction policy	Layers distrib.	Inference speed % CPU	GPU	dev clean	dev other	test clean	test other
Conformer-S (**baseline**)	21.8	x4	12	-	-	10.8	28.8	11.3	28.9
Conv-Conformer_v1	23.2	x4-x8	2-10	-48.4	-27.3	11.4	28.5	11.6	29.3
Conv-Conformer_v2	23.2	x4-x8	4-8	-38.5	-25.0	11.2	28.0	11.5	28.4
Uconv_D8-F4	23.2	x4-x8-x4	2-8-2	-38.3	-17.8	11.1	28.2	11.2	29.2
Uconv_D16-F4	24.6	x4-x8-x16-x8-x4	2-2-4-2-2	-44.2	-20.7	12.4	30.2	12.6	30.8
Uconv_D16-F8_v1	24.6	x4-x8-x16-x8	3-3-3-3	-47.8	-23.5	**10.4**	**27.1**	**10.7**	**27.1**
Uconv_D16-F8_v2	24.6	x4-x8-x16-x8	2-4-5-1	-56.3	-25.2	10.9	27.4	11.0	27.9
Uconv_D32-F8	26.0	x4-x8-x16-x32-x16-x8	2-2-2-2-2-2	-58.1	-27.7	11.5	28.6	11.6	29.0
Conformer-L	83.0	x4	12	+221.8	+113.3	10.7	27.8	10.9	28.4
Uconv-L_D16-F8_v1	87.3	x4-x8-x16-x8	3-3-3-3	+124.7	+71.2	**10.4**	**26.7**	**10.7**	**27.2**

Table 2 presents the experiments with all major architectures for 460 and 960 hours of available training data. Uconv_D16-F8_v1 outperforms Conformer-S and Conv-Conformer_v2 in all the cases. Our model even shows comparable results with Conformer-L despite much fewer model size and higher work speed. Furthermore, large Uconv-L_D16-F8_v1 demonstrates that Uconv-Conformer architecture shows a consistent superiority in recognition accuracy over standard Conformer in case of comparable model size.

Table 3. *Relative WER change, % of Uconv_D16-F8_v1 model after using intermediate CTC loss for different sizes of training data set.*

Training data	dev clean	dev other	test clean	test other
LibriSpeech 100h	**-6.7**	**-2.2**	**-6.5**	**-1.7**
LibriSpeech 460h	+2.1	+4.5	+4.0	+4.6
LibriSpeech 960h	+8.1	+5.9	+10.5	+9.1

in case of 460h and 960h training data. We can conclude that the intermediate CTC loss is only helpful for the low-resource scenario as an additional regularization.

Table 2. *Results (WER, %) for small and large models trained on LibriSpeech 460h and 960h.*

Model	dev clean	dev other	test clean	test other
LibriSpeech 460h				
Conformer-S (**baseline**)	5.1	16.9	5.5	16.8
Conv-Conformer_v2	5.2	16.4	5.4	16.2
Uconv_D16-F8_v1	**4.8**	**15.7**	**5.0**	**15.2**
Uconv_D16-F8_v2	4.9	15.7	5.1	15.6
Conformer-L	4.8	15.6	5.0	15.2
Uconv-L_D16-F8_v1	**4.5**	**15.0**	**4.7**	**14.5**
LibriSpeech 960h				
Conformer-S (**baseline**)	4.1	11.0	4.1	10.9
Conv-Conformer_v2	4.0	10.6	4.1	10.6
Uconv_D16-F8_v1	**3.7**	**10.1**	3.8	**9.9**
Uconv_D16-F8_v2	3.8	10.2	**3.7**	10.0
Conformer-L	3.6	9.9	3.7	9.7
Uconv-L_D16-F8_v1	**3.4**	**9.1**	**3.5**	**9.0**

5. CONCLUSION

This paper proposes the Uconv-Conformer architecture, which reduces the input sequence length by 16 times for intermediate Conformer layers, followed by upsampling to final reduction level x8 aimed at stabilization of convergence and CTC loss calculation. We also investigated the effect of the intermediate CTC loss, which improves the recognition accuracy for the low-resource scenario. The results obtained for small models trained on LibriSpeech 960 hours outperform the baseline Conformer with a similar number of parameters on test_clean and test_other by 7.3% and 9.2% relative WER. This method also accelerates inference on CPU and GPU by 47.8% and 23.5%, respectively.

Using an intermediate CTC loss for the Uconv-Conformer model (Table 3) improves the recognition accuracy only for 100h training data scenario. It leads to quality degradation

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, et al., "Jasper: An end-to-end convolutional neural acoustic model," in *Proc. Interspeech 2019*, 2019.

[2] Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, et al., "Quartznet: Deep automatic speech recognition with 1d time-channel separable convolutions," in *Proc. ICASSP*, 2020.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al., "Attention is all you need," Red Hook, NY, USA, 2017, NIPS'17, Curran Associates Inc.

[4] Anmol Gulati, James Qin, Chung-Cheng Chiu, et al., "Conformer: Convolution-augmented transformer for speech recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[5] Pengcheng Guo, Florian Boyer, Xuankai Chang, et al., "Recent developments on espnet toolkit boosted by conformer," in *ICASSP 2021*.

[6] Mohammad Zeineldeen, Jingjing Xu, Christoph Lüscher, et al., "Improving the training recipe for a robust conformer-based hybrid model," in *Proc. Interspeech*, 2022.

[7] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.*, 2022.

[8] Iz Beltagy, Matthew E. Peters, and Arman Cohan, "Longformer: The long-document transformer," *arXiv:2004.05150*, 2020.

[9] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, et al., "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, 2020.

[10] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe, "Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding," in *Proc. PMLR*, 2022.

[11] Chuhan Wu, Fangzhao Wu, Tao Qi, et al., "Fastformer: Additive attention can be all you need," *arXiv preprint arXiv:2108.09084*, 2021.

[12] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *Proc. ICML*, 2020.

[13] Wenyong Huang, Wenchao Hu, Yu Ting Yeung, et al., "Conv-Transformer Transducer: Low Latency, Low Frame Rate, Streamable End-to-End Speech Recognition," *Proc. Interspeech 2020*, pp. 5001–5005, 2020.

[14] Maxime Burchi and Valentin Vielzeuf, "Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition," *2021 IEEE ASRU*, pp. 8–15, 2021.

[15] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[17] Sehoon Kim, Amir Gholami, Albert Shaw, et al., "Squeezeformer: An efficient transformer for automatic speech recognition," *NeurIPS*, 2022.

[18] Jaesong Lee and Shinji Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *ICASSP 2021*, 2021.

[19] Andrei Andrusenko, Aleksandr Laptev, and Ivan Medennikov, "Exploration of end-to-end asr for openstt – russian open speech-to-text dataset," SPECOM, 2020.

[20] Jumon Nozaki and Tatsuya Komatsu, "Relaxing the Conditional Independence Assumption of CTC-Based ASR by Conditioning on Intermediate Predictions," in *Proc. Interspeech 2021*, 2021.

[21] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," *ICASSP*, 2015.

[22] Daniel S. Park, William Chan, Yu Zhang, et al., "Specaugment: A simple data augmentation method for automatic speech recognition," *Interspeech*, 2019.

[23] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[24] Taku Kudo and John Richardson, "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing," in *in Proc. EMNLP*, Brussels, Belgium, 2018.

[25] Shinji Watanabe, Takaaki Hori, Shigeki Karita, et al., "ESPnet: End-to-end speech processing toolkit," in *Proc. of Interspeech*, 2018.