pISSN 2005-8063
eISSN 2586-5854
2021. 3. 31.
Vol.13 No.1
pp. 45-51

말소리와 음성과학
Phonetics and Speech Sciences

한국음성학회지
https://doi.org/10.13064/KSSS.2021.13.1.045

Hyperparameter experiments on end-to-end automatic speech recognition*

Hyungwon Yang · Hosung Nam**

Department of English Language and Literature, Korea University, Seoul, Korea

Abstract

End-to-end (E2E) automatic speech recognition (ASR) has achieved promising performance gains with the introduced self-attention network, Transformer. However, due to training time and the number of hyperparameters, finding the optimal hyperparameter set is computationally expensive. This paper investigates the impact of hyperparameters in the Transformer network to answer two questions: which hyperparameter plays a critical role in the task performance and training speed. The Transformer network for training has two encoder and decoder networks combined with Connectionist Temporal Classification (CTC). We have trained the model with Wall Street Journal (WSJ) SI-284 and tested on devl93 and eval92. Seventeen hyperparameters were selected from the ESPnet training configuration, and varying ranges of values were used for experiments. The result shows that “num blocks” and “linear units” hyperparameters in the encoder and decoder networks reduce Word Error Rate (WER) significantly. However, performance gain is more prominent when they are altered in the encoder network. Training duration also linearly increased as “num blocks” and “linear units” hyperparameters’ values grow. Based on the experimental results, we collected the optimal values from each hyperparameter and reduced the WER up to 2.9/1.9 from dev93 and eval93 respectively.

Keywords: automatic speech recognition, transformer, neural network, hyperparameters, optimization

1. Introduction

Automatic Speech Recognition (ASR) has been widely studied by many researchers, as it is successfully implemented to speech related applications. Since ASR transforms human’s speech signals to sentence as a text, a number of tasks processing human’s speech information adopt speech recognition result. A chatbot task (Wei et al., 2018), for instance, analyzes human’s intentions based on the sentences produced by ASR system to interact with people. Sentiment analysis which evaluates the emotional state of a person also utilizes speech recognition result in order to adapt the sentiment recognition models in a noisy environment (Lakomkin et al., 2019). In practice, sentiment recognition task is usually tested in a noisy background. Therefore, building a noise-robust ASR model is crucial to the task.

* This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2019 S1A5A8033772).
** hnam@korea.ac.kr, Corresponding author
Received 31 January 2021; Revised 16 March 2021; Accepted 16 March 2021

Speech recognition model is generally built up with 3 different models: Acoustic Model (AM), Pronunciation Model (PM), and Language Model (LM). Human language is comprised of various syllables and AM learns how each syllable is pronounced and sequentially arranged in speech production. PM contains the mapping information of words and their realized pronunciation affected by complex phonological rules embedded in the language. Lastly, the proper structures of word sequence and grammatical information that make a sentence natural is stored in LM.

Traditionally, Hidden Markov Model with Gaussian Mixture Model (HMM-GMM) is frequently used to make an AM to learn phone sequences from the speech signals. HMM-GMM learns triphone information, three consecutive phone sequence, by cumulating mixture information of diagonal covariance Gaussians. To build a PM, a dictionary that properly maps each word to phoneme sequences is required. Therefore, generating a dictionary is crucial for creating a PM. However, this task is tricky and laborious because it needs to be updated whenever new words are found in a corpus and cover multiple pronunciation if each word is pronounced in many ways. For example, tomato has two ways to be pronounced: [təméitou] or [təmɑ́ːtou]. Trigram modeling is one of the best options to build a LM. Good-turing (Gale & Sampson, 1995) or modified kneser-ney (James, 2000) smoothing algorithms are suggested and applied to LM to predict unseen words in efficiency. However, a few drawbacks such as a sparse representation of language and computation overhead remain.

After neural network merges in ASR field, a lot of researchers have achieved a state-of-the-art performance in the speech recognition tasks (Chang et al., 2020; Miao et al., 2020; Nakatani, 2019; Watanabe et al., 2017). Especially, end-to-end (E2E) ASR models have been developed by utilizing recurrent neural network or transformer network (Vaswani et al., 2017). The greatest benefit of the neural network based model is that it trains all the AM, PM, and LM with the single neural network. To do so, training a speech recognition model becomes simpler compared to the previous algorithms that need to train each model separately and combine them.

Recently, ASR toolkits providing numerous neural network architectures such as ESPnet (Watanabe et al., 2018; Watanabe et al., 2020), or KoSpeech (Kim et al., 2020) are also introduced and they ease the speech recognition model building process. Although training a model by composing neural networks is not trick, achieving a better performance in a speech recognition task is challenging. A few papers (Koutsoukas et al., 2017; Popel & Bojar, 2018) try to suggest the solutions to improve the performance but not mainly focus on speech recognition task.

In this paper, we examine the impact of hyperparameters provided in the neural network. We select transformer network constructed with attention mechanism, since it shows a significant improvement in speech recognition tasks and is frequently applied in many studies. Our study aims to two goals. 1. Finding out which hyperparameter plays a critical role in performance improvement. 2. The tendency of training speed variation with respect to the hyperparameter values.

2. Experiments

2.1. Experimental setup

The training network consists of two transformer architectures both in encoder and decoder parts and they are jointly trained with Connectionist Temporal Classification (CTC; Graves et al., 2006).

We implement the hyperparameter experiments on ESPnet framework compiled with Pytorch backend and the network training is carried out using 6 NVIDIA Tesla T4 GPUs.

2.2. Hyperparameter setup

ESPnet transformer network provides total 30 adjustable hyper-parameters in the training configuration. In this paper, only 17 hyperparameters are selected because in the perspective of training network, chosen hyperparameters are more closely related to the network learning process. The selected hyperparameters are trained with proportionally ranged values based on their provided default values in the configuration. Excluded 13 hyperparameter values are fixed with the given default set in the network training configuration.

When each hyperparameter is applied in the training process with the preset values sequentially, other hyperparameters except the selected value, is fixed to the given default in order to prevent dependencies among the hyperparameter values. It is critical to keep the independency in the hyperparameter experiments, since this paper aims to investigate each hyperparameter's impact on the network performance.

2.2.1. Encoder hyperparameters

Transformer architecture has 9 hyperparameter variables in the encoder network. 3 hyperparameter variables ("output size", "input layer", "positional dropout rate") are fixed to default values, and the other 6 hyperparameters ("normalized before", "attention heads", "linear units", "num blocks", "dropout rate", "attention dropout rate") are set to diversely ranged values. Table 1. below describes how the encoder hyperparameters are initialized and the bold text indicates the default value.

Table 1. The range of hyperparameters in the transformer encoder network

Hyperparameters	Values				
output size	**256**				
input layer	**2d conv**				
normalized before	**True**		false		
attention heads	1	**2**	4	8	
linear units	512	1,024	**2,048**	4,096	
num blocks	2	4	6	8	**12**
dropout rate	0.0	**0.1**	0.2	0.3	0.4
positional dropout rate	**0.1**				
attention dropout rate	**0.0**	0.1	0.2	0.3	0.4

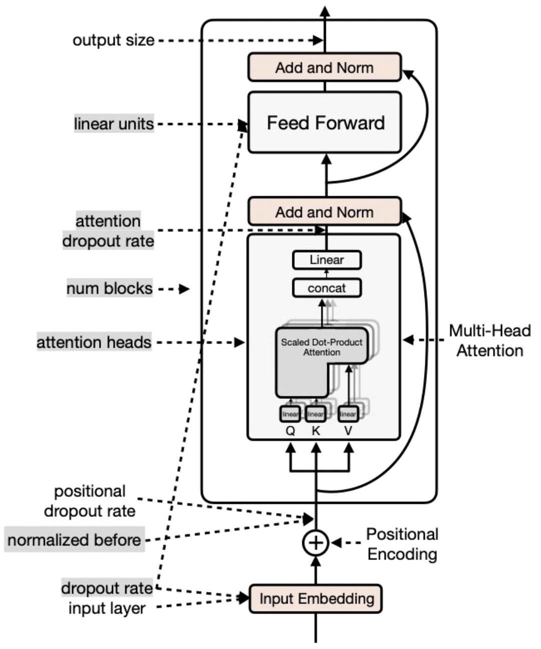

Figure 1. Encoder network structure and its related hyperparameters

The hyperparameters involved in the training encoder network is described in Figure 1 and the selected hyperparameters for the experiment are highlighted in gray.

An "output size" hyperparameter is fixed to 256 so that the number of "attention heads" is able to be adjusted. The number of "attention heads" split the number of Q, K, and V variables and then split variables are concatenated after they passed scaled dot-product attention. Therefore, the dimension of scaled dot-product attention should meet the "output size" in order to evade dimension mismatch.

Since input feature extraction is not the main focus in the study, convolutional network as a default setting for extracting input features is applied to input layer hyperparameter. "positional dropout rate" was not included in the experiment list because ESPnet toolkit ignores the value in the configuration but automatically initializes this hyperparameter with "dropout rate" hyperparameter.

2.2.2. Decoder hyperparameters

In the decoder network, 7 hyperparameters are ready to be adjusted, but only 5 hyperparameter variables ("attention heads", "linear units", "num blocks", "dropout rate", "self attention dropout rate") are extracted for experiment, and the rest 2 variables ("positional dropout rate", "src attention dropout rate") are remained as default. Table 2. shows the types of hyperparameters used in the transformer network and their values. The default values of the hyperparameter in the box are bolded.

Table 2. The range of transformer decoder network hyperparameters

Hyperparameters	Values				
attention heads	1	**2**	4	8	
linear units	512	1,024	**2,048**	4,096	
num blocks	2	4	**6**	8	12
dropout rate	0.0	**0.1**	0.2	0.3	0.4
positional dropout rate	**0.1**				
self attention dropout rate	**0.0**	0.1	0.2	0.3	0.4
src attention dropout rate	**0.0**				

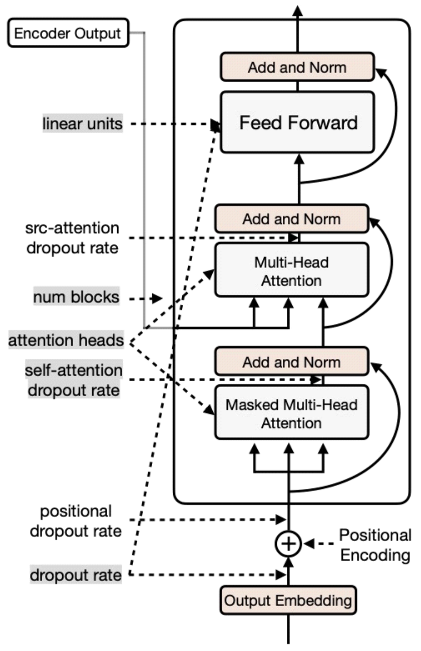

Figure 2. Decoder network structure and its related hyperparameters

As Figure 2 depicts, hyperparameters for the experiment are listed right next to the decoder network architecture and among them, the shaded variables in gray are trained with multiple values.

In the decoder section, we ignore two hyperparameters, "src attention dropout rate" and "positional dropout rate" because they automatically share "self attention dropout rate", and "dropout rate" values respectively during training network.

2.2.3. Model hyperparameters

A few hyperparameters which are not directly related to the network are included in the experiment because of their significant impact on the network performance. There are 14 hyperparameters in total, and 6 of them are chosen to the training process. Table 3. elaborates the detail information about the indirectly related hyperparameters. Hyperparameters set to default are bolded.

Table 3. The range of the model hyperparameters

Hyperparameters	Values				
batch type	**folded**				
batch size	**32**				
accum grad	**8**				
max epoch	**50**				
patience	**none**				
init	chainer	**xavier uniform**	xavier normal	kai-minguni-form	kaiming normal
optim	**adam**				
lr	**0.005**				
scheduler	**warmuplr**				
warmup steps	10,000	20,000	**30,000**	40,000	
keep nbest model	5	**10**	15	20	
ctc weight	0.0	0.1	0.2	**0.3**	0.4
lsm weight	0.0	**0.1**	0.2	0.3	0.4
length normalized loss	true			**false**	

Rather than directly participating in the training process to improve speech recognition performance, model hyperparameters mainly relate to managing GPU memory in efficiency or boosting up the speed of the training process.

"batch type", "batch size", and "accum grad" hyperparameters which contribute to memory management are initialized to default values. When it comes to "max epoch", instead of using default value 100, we minimize it to 50 so as to save the training duration to finish all the combinatory hyperparameter experiments. "patience" given default 'none' in the experiment, decides whether to stop training process when the model is not improved after the repeated epochs. "learning rate" and "scheduler" are fixed to default but in order to see the impact of "scheduler", we diversify a "warmup steps" value in the training process. An "optim" hyperparameter is another important factor in the neural network training, but it is already experimented a lot in the previous studies (Kingma & Ba, 2014; Okewu et al., 2019). Thus, "optim" is set to default value in this study.

2.3. Dataset

Since Wall Street Journal (WSJ) has been trained and tested frequently in ASR experiments, this article carried out the hyperparameter experiment with WSJ dataset. Before training the model, WSJ, which consists of csr_1 and csr_2_comp was extracted into SI-284 set for training and dev93 and eval92 set for evaluation. The training set provides total 37,416 English sentences including noise, space, and symbols (e.g., exclamation or question marks) and 333 sentences are prepared in an equivalent condition for evaluation set.

3. Results

After training speech recognition models with selected hyperparameters, each model decoded WSJ dev93 and eval92 set and we extracted Word Error Rate (WER) information from the result. The Table 4 shows all the decoding result in WER, with respect to the hyperparameter values. The decoding result is sorted by model, encoder and decoder parts and the best WER score (mean WER score from dev93 and eval92 sets) in each hyperparameter set is bolded in the table. This paper mainly focuses on how the WER score and training speed vary as the hyperparameter values change. Therefore, the speech recognition performance achieved from the experiment could be lower than the other state-of-the-art studies' because the model is not fully trained in optimal condition.

Table 4. WER from each hyperparameter model on WSJ dev93 and eval92

Hyperparameters		Values / WER				
MODEL	init	chainer	xavier uniform	xavier normal	kaiming uniform	kaiming normal
		42.0/35.1	**17.0/12.7**	17.3/14.0	17.7/13.1	17.6/13.4
	warmup steps	10,000	20,000	30,000	40,000	
		15.6/12.4	16.0/12.8	17.3/12.7	17.3/13.6	
	keep nbest model	5	10	15	20	
		17.0/12.8	17.3/12.7	16.9/13.0	16.9/13.4	
	ctc weight		0.1	0.2	0.3	0.4
			17.3/13.6	17.0/13.1	**17.3/12.7**	16.5/13.0
	lsm weight		0.1	0.2	0.3	0.4
			17.3/12.7	17.3/13.2	17.5/12.9	18.0/13.3
	length normalized loss	true	false			
		17.7/14.0	**17.3/12.7**			
ENCODER	attention heads	1	2	4	8	
		17.6/13.3	**16.7/13.0**	17.3/12.7	17.6/13.4	
	linear units	512	1,024	2,048	4,096	
		18.9/14.8	18.0/14.0	17.3/12.7	**16.3/12.3**	
	num blocks	2	4	6	8	12
		24.5/19.7	20.2/16.0	18.5/15.0	17.3/13.5	**17.3/12.7**
	dropout rate	0.0	0.1	0.2	0.3	0.4
		17.4/14.4	**17.3/12.7**	17.7/13.4	17.8/13.7	20.4/15.7
	attention dropout rate	0.0	0.1	0.2	0.3	0.4
		17.3/12.7	16.5/13.0	15.6/12.8	**15.8/12.6**	15.9/12.7
	normalized before	true	false			
		17.3/12.7	14.1			
DECODER	attention heads	1	2	4	8	
		17.3/13.0	17.1/12.9	**17.3/12.7**	17.6/12.8	
	linear units	512	1,024	2,048	4,096	
		17.7/13.5	17.5/13.4	17.2/12.7	**16.9/12.9**	
	num blocks	2	4	6	8	12
		19.7/16.0	17.2/13.6	17.3/12.7	**16.3/12.5**	16.3/12.5
	dropout rate	0.0	0.1	0.2	0.3	0.4
		16.5/13.3	**17.3/12.7**	16.9/13.8	16.3/13.6	17.5/13.5
	self attention dropout rate	0.0	0.1	0.2	0.3	0.4
		17.3/12.7	16.5/13.8	17.0/13.9	16.6/13.8	16.5/13.5

WER, word error rate.

3.1. Models accuracy

3.1.1. Model hyperparameters

An "init" hyperparameter controls weight initialization and xavier method especially using uniform distribution shows the best performance (17.0/12.7 WER) among the other methods. The "init" hyperparameter provides chainer option which is used in chainer toolkit and this option initializes weight with LeCun normalization (LeCun et al., 2012). WER decreases gradually when "warmup steps" declines. Since the "warmup steps" hyperparameter modifies the "learning rate" value in which "warmup steps" reaches its value, setting it to a lower value tweaks the learning rate much earlier and helps the model to converge fast. Similar to "warmup steps", a "keep nbest model" hyperparameter also shows tendency that WER in eval92 set declines as value decreases thought the WER in dev93 set shows the opposite result. The result implies that the more the number of hypothesis to choose, the more the task difficulty of finding a correct one increases. Thus, lowering the complexity by reducing "keep nbest model" might be crucial in the network training. However, too much low value could degrade or halt WER score because the score increases a bit (12.7 to 12.8 WER in eval92) when the hyperparameter is adjusted to 5 from 10. A "length normalized loss" hyperparameter derives better score from false setting (17.3/12.7 WER). "ctc weight" and "lsm weight" hyperparameters do not show any meaningful result because their WER scores fluctuate regardless of the values' shift.

3.1.2. Encoder hyperparameters

Two hyperparameters in the encoder network, "linear units" and "num blocks" improve the decoding performance as the hyperparameter values increase. Compared to "num blocks" (17.3/12.7 WER), a "linear units" (16.3/12.3 WER) hyperparameter makes a better progress in the decoding performance. A "dropout rate" hyperparameter, however, shows convex like tendency in its result. When it is set to the lowest value, 0.0, its WER is 17.4/14.4, but it is dropped (17.3/12.7 WER) as the value increases by 0.1 but as soon as the value rises, the performance is degraded. "attention heads" and "attention dropout rate" do not draw a meaningful result in this experiment.

3.1.3. Decoder hyperparameters

As seen in the encoder network hyperparameters, "linear units" and "num blocks" hyperparameters in the decoder network also show linearity in their improvement curve. However, the performance increases rather slowly as the value changes. WER scores from "num blocks" and "linear units" in the encoder network are dropped by 7.2/7.0 and 2.6/2.5 respectively, but 3.4/3.5, and 0.8/0.6 in the decoder network. A "dropout rate" hyperparameter in the decoder network does not act like the one in the encoder network, but it reaches the lowest WER at the value 0.1 and maintains high WER at the other values. Meaningful result is not found in "attention heads" and "self attention dropout rate".

3.1.4. Optimal hyperparameters

According to the experiment, the study shows which hyperparameters are relatively critical to speech recognition performance. Based on the result, we collect optimal values from each hyperparameter and train the model again to see whether the model develop the decoding performance.

However, the optimal values gained independently may not guarantee the best decoding performance since each value was tested without considering dependencies among the 17 hyperparameter values. In order to find the optimal hyperparameter set, several research (Wang et al., 2019; You et al., 2019) have considered dependencies of two or three hyperparameters in their experiment. In light of this fact, our study needs to proceed the experiment considering all the dependencies of the hyperparameters, but this is almost impossible due to the large number of experimental trials. If we take the dependencies into account, the number of trials will be 20,480,000,000 based on the equation (1). In (1), X, Y, Z is the number of value sets (2, 4, 5), and i, j, k is the number of hyperparameters (2, 8, 7) respectively.

$$Values^{hyperparamets} = X^{i}Y^{j}Z^{k} \tag{1}$$

This trial number will go down to 71 if the dependencies of the hyperparameter values are not considered. This paper trains the model with optimal values drawn from the independent condition to see that the model is still able to increase the speech recognition performance even though the optimal values were not considered the dependencies of the hyperparameters.

The trained model that ignores dependencies of the hyperparameters proves that the decoding performance has been improved by reducing WER score to 13.4/10.4 and this performance could be developed by running more epochs.

3.2. Training speed

While the network trains models with different hyperparameter values, we mark the training duration between the 1st and 2nd epochs to see how the training duration varies depending on the hyperparameter values. In terms of training speed, the result shows that the number of "linear units" and "num blocks" hyperparameters brings a meaningful change in the encoder and decoder networks.

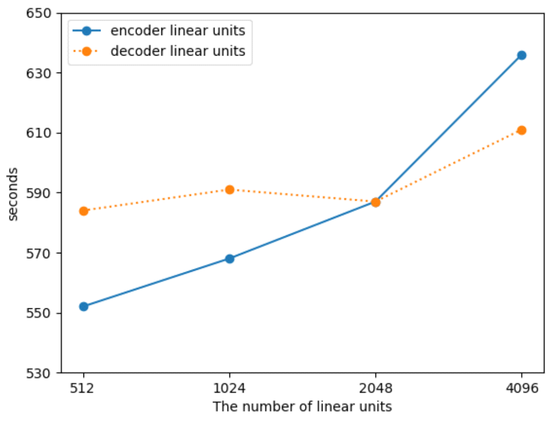

Figure 3. The training speed of linear units per a single epoch

Figure 3 shows two hyperparameters' training speed as the number of "linear units" varies. When the number of "linear units" in the encoder network increases, the training speed declines gradually. In contrast, the identical hyperparameter's behavior of training speed variation in the decoder network does not gradually increase but is stabilized its speed around 590 seconds except when the number of "linear units" hits highest value, 636 seconds, in this experiment. Although the number of "linear units" hyperparameter has a direct influence on the training speed, the degree of change in both the encoder and decoder network is not similar.

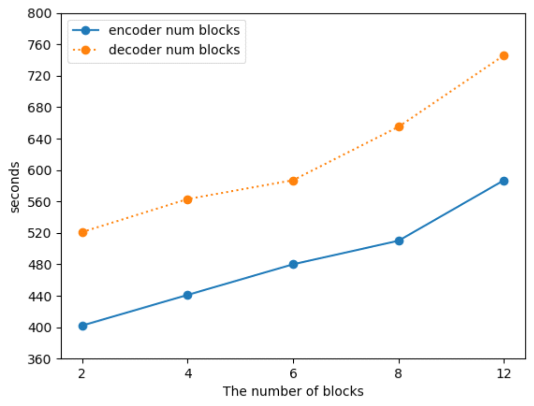

Figure 4. The training speed of num blocks per a single epoch

Figure 4 presents that “num blocks” hyperparameters in both the encoder and decoder networks show steady increase with respect to training duration. In terms of the pattern of speed increase in both hyperparameters, the amount of the speed duration increased is similar as the number of blocks pile up. However, the mean speed gap between the two hyperparameters is 130 seconds and it predicts that a computational cost of “num blocks” in the decoder network is more expensive, thought the number of “num blocks” hyperparameter remains identical in both networks.

4. Conclusions

This paper investigates the impact of hyperparameters in both ASR performance and training speed based on the E2E transformer network. Total 17 hyperparameters out of 30 are selected and trained with diversely ranged values.

In model hyperparameters, “init”, “warmup steps”, “keep nbest model” and “length normalized loss” show a significant impact on the speech recognition performance. This study also finds that the number of “num blocks” and “linear units” brings a gradual improvement as their values increase but this tendency is more strongly presented in the encoder network hyperparameters.

Training speed varies when the number of “num blocks” and “linear units” hyperparameters change. As the result indicates, WER score from the “linear units” hyperparameter in the encoder network is more significantly affected by the number of values, when the other in the decoder network does not show a strong correlation between the WER score and the number of values. When it comes to the training speed comparison of “num blocks” hyperparameters both in the encoder and decoder networks, the hyperparameter in the encoder network is more computationally expensive than the other.

Two hyperparameters, “linear units” and “num blocks”, that bring a significant impact on WER score may be differently evaluated if we consider the training speed result. Since training duration is critical in real-world application, A speech recognition model needs to be trained fast in order to save hardware resources and research time. In light of this aspect, the “linear units” hyperparameter in the decoder network may not worth experimenting with various values because its training time usually takes longer with little improvement. The same hyperparameter in the encoder network, however, achieves relatively fast performance in training duration with quick convergence though the WER score lags behind a bit.

When it comes to the “num blocks” hyperparameter, the gap between the two WER scores from encoder and decoder network is relatively large in lowest value 2. This gap almost perishes when the hyperparameter value is grown up to 12. While this WER score gap between the two networks gets lower as the hyperparameter value increases, the training duration gap between the two networks has been almost stable. Therefore, if the “num blocks” hyperparameter value needs to be set high, it may be efficient to tweak this value in the encoder network and vice versa.

Finally, after the experiment, we train the model with the optimal values found from the result, we obtained the improved model that shows 13.4/10.4 WER. Compared to the best result in the experiment 16.3/12.3, the optimal model dropped WER by 2.9/1.9.

References

Chang, X., Zhang, W., Qian, Y., Le Roux, J., & Watanabe, S. (2020, May). End-to-end multi-speaker speech recognition with transformer. *Proceedings of the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6134-6138). Barcelona, Spain.

Gale, W. A., & Sampson, G. (1995). Good turing frequency estimation without tears. *Journal of Quantitative Linguistics*, *2*(3), 217-237.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006, June). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 369- 376). Pittsburgh, PA.

James, F. (2000). *Modified kneser-ney smoothing of n-gram models* (RIACS Technical Report 00.07). Mountain View, CA: Research institute for advanced computer science. Retrieved from https://www.researchgate.net/profile/Frankie-James/publication/255479295_Modified_Kneser-Ney_Smoothing_of_n-gram_Models/links/54d156750cf28959aa7adc08/Modified-Kneser-Ney-Smoothing-of-n-gram-Models.pdf

Kim, S., Bae, S., & Won, C. (2020). KoSpeech: open-source toolkit for end-to-end Korean speech recognition. *arXiv*. Retrieved from https://arxiv.org/abs/2009.03092

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv*. Retrieved from https://arxiv.org/abs/1412.6980

Koutsoukas, A., Monaghan, K. J., Li, X., & Huan, J. (2017). Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics*, *9*(1), 1-13.

Lakomkin, E., Zamani, M. A., Weber, C., Magg, S., & Wermter, S. (2019, May). dorporating end-to-end speech recognition models for sentiment analysis. *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)* (pp. 7976-7982). Montreal, QC.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In G. Montavon, G. B. Orr & K. S. Müller (Eds.), *Neural networks: tricks of the trade* (2nd ed., Vol. 7700, pp. 9-48). Berlin, Germany: Springer.

Miao, H., Cheng, G., Gao, C., Zhang, P., & Yan, Y. (2020, May). Transformer-based online CTC/attention end-to-end speech recognition architecture. *Proceedings of the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6084-6088). Barcelona, Spain.

Nakatani, T. (2019, September). Improving transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration. *Proceedings of the Interspeech 2019*. Graz, Austria

Okewu, E., Adewole, P., & Sennaike, O. (2019, July). Experimental comparison of stochastic optimizers in deep learning. *Proceedings of the International Conference on Computational Science and its Applications* (pp. 704-715). Saint Petersburg, Russia.

Popel, M., & Bojar, O. (2018). Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, *110*(1), 43-70.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser L., & Polosukhin, I. (2017). Attention is all you need. *arXiv*. Retrieved from https://arxiv.org/abs/1706.03762

Wang, C., Wu, Y., Du, Y., Li, J., Liu, S., Lu, L., Ren S., ⋯ Zhou, M. (2019). Semantic mask for transformer based end-to-end speech recognition. *arXiv*. Retrieved from https://arxiv.org/abs/1912.03010

Watanabe, S., Boyer, F., Chang, X., Guo, P., Hayashi, T., Higuchi, Y., Hori, T., ⋯ Zhang, W. (2020). The 2020 ESPnet update: new features, broadened applications, performance improvements, and future plans. *arXiv*, *arXiv:2012.13006*

Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., ⋯ Ochiai, T. (2018). Espnet: end-to-end speech processing toolkit. *arXiv*. Retrieved from https://arxiv.org/abs/1804.00015

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., & Hayashi, T. (2017). Hybrid CTC/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, *11*(8), 1240-1253.

Wei, C., Yu, Z., & Fong, S. (2018, February). How to build a chatbot: chatbot framework and its capabilities. *Proceedings of the 2018 10th International Conference on Machine Learning and Computing* (pp. 369-373). Macau, China.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., & Hsieh, C. J. (2019). Large batch optimization for deep learning: training bert in 76 minutes. *arXiv*. Retrieved from https://arxiv.org/abs/1904.00962

• **Hyungwon Yang**
PhD Student, Dept. of English Language and Literature
Korea University
145, Anam-ro, Seongbuk-gu, Seoul, Korea
Tel: +82-2-3290-1980
Email: hyung8758@korea.ac.kr
Fields of interest: Phonetics, Language Engineering

• **Hosung Nam,** Corresponding author
Professor, Dept. of English Language and Literature
Korea University
145, Anam-ro, Seongbuk-gu, Seoul, Korea
Tel: +82-2-3290-1991
Email: hnam@korea.ac.kr
Fields of interest: Phonetics, Language Engineering