DropDim: A Regularization Method for Transformer Networks

Hao Zhang, Dan Qu, Keji Shao, and Xukui Yang

***Abstract*—We introduce DropDim, a structured dropout method designed for regularizing the self-attention mechanism, which is a key component of the transformer. In contrast to the general dropout method, which randomly drops neurons, DropDim drops part of the embedding dimensions. In this way, the semantic information can be completely discarded. Thus, the excessive co-adapting between different embedding dimensions can be broken, and the self-attention is forced to encode meaningful features with a certain number of embedding dimensions erased. Experiments on a wide range of tasks executed on the MUST-C English-Germany dataset show that DropDim can effectively improve model performance, reduce over-fitting, and show complementary effects with other regularization methods. When combined with label smoothing, the WER can be reduced from 19.1% to 15.1% on the ASR task, and the BLEU value can be increased from 26.90 to 28.38 on the MT task. On the ST task, the model can reach a BLEU score of 22.99, an increase by 1.86 BLEU points compared to the strong baseline.**

***Index Terms*—End-to-end, transformer, regularization, dropout.**

I. Introduction

THE end-to-end (E2E) neural network has achieved great success in sequence-to-sequence problems such as Automatic Speech Recognition (ASR) [1], Machine Translation (MT) [2], and Speech Translation (ST) [3]. In this paradigm, a single neural network is used to directly map the input to the target, removing the independent part of the cascade method. Transformer [4] is currently the most popular E2E structure and has achieved state of the art performance in many sequences modeling tasks [5], [6]. This ability is partly because it makes few assumptions about structural information of data, which makes the transformer a universal and flexible architecture [7]. As a side effect, the lack of structural bias makes the transformer prone to overfitting for small-scale data.

Dropout [8] is a widely used regularization strategy, which multiplies each neuron with a sample of a Bernoulli random variable during training, with probability p of dropping/zeroing out the neuron. The final model can be understood as the average of multiple models. Several variations of dropout have also been proposed, such as dropblock [9], spatial dropout [10], and zoneout [11]. While most transformer-based models still use the regular dropout [12], [13].

DropAttention is recommended in [14]. By randomly setting attention weights to zero, the final model was encouraged to use the full context of the input sequences for prediction, rather than relying solely on a small piece of features. And considering that the multi-head attention mechanism is dominated by a small number of attention heads [15], Zhou *et al.* [16] recommends DropHead, which randomly discards part of the attention heads during the training.

The above methods directly process the attention head. In this paper, we recommend another simple structured dropout method from the perspective of self-attention layer input, called DropDim. The key motivation behind DropDim is that each embedding dimension contains certain semantic information in transformer. In other words, the basic unit is a vector instead of a single neuron [14]. Thus, the independence assumption in dropout needs to be relaxed. We achieve this by dropping the embedding dimension. In this way, semantic information can be completely removed. The excessive co-adapting between different embedding dimensions can be broken, and the self-attention is forced to encode meaningful features with a certain number of embedding dimensions erased.

There exists some structured dropout like the one suggested by us [17]–[21].The most similar to our method is cutoff [22], which is a set of simple yet efficient data augmentation strategies, including token cutoff, feature cutoff, and span cutoff. However, the motivation and application location are very different. Cutoff processes the input sentence to generate a restricted perspective, which helps enrich empirical observations and better cover the data space. While we process the input of the attention mechanism to ensure that self-attention layer can encode more meaningful features without utilizing the information from the removed embedding dimensions at all.

Experiments on a wide range of tasks executed on the MUST-C English-Germany dataset [23] show that DropDim can effectively improve model performance, reduce over-fitting, and show complementary effects with other regularization methods. When combined with label smoothing [24], the WER can be reduced from 19.1% to 15.1% on the ASR task, and the BLEU value can be increased from 26.90 to 28.38 on the MT task. On the ST task, the model can reach a BLEU score of 22.99, an increase by 1.86 BLEU points compared to the strong baseline.

II. Method

DropDim is a simple regularization method similar to dropout. Its main difference from dropout is that it drops embedding

Manuscript received September 21, 2021; revised December 20, 2021; accepted December 25, 2021. Date of publication January 5, 2022; date of current version February 1, 2022. This work was supported by the National Natural Science Foundation of China under Grants 61673395 and 62171470. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Odette Scharenborg. *(Corresponding authors: Dan Qu; Xukui Yang.)*

Hao Zhang, Dan Qu, and Xukui Yang are with the School of the Information Engineering University, Zhengzhou 450000, China (e-mail: haozhang0126@163.com; qudanqudan@sina.com; gzyangxk@163.com).

Keji Shao is with the Jiangnan Institute of Computing Technology, Wuxi 214000, China (e-mail: shaokjppl@163.com).

Digital Object Identifier 10.1109/LSP.2022.3140693

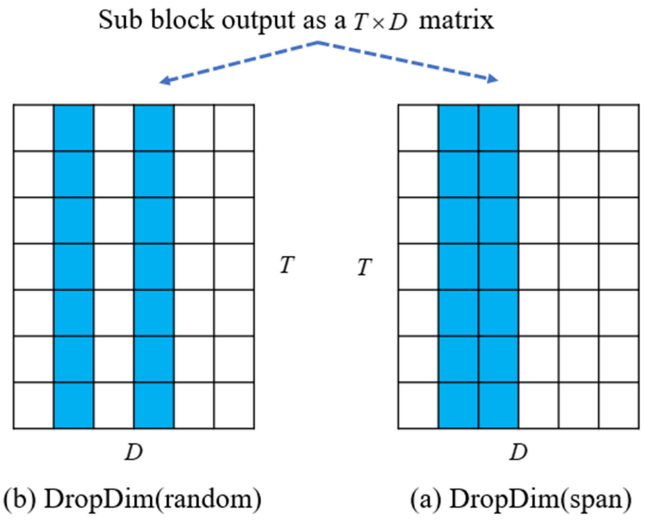

Fig. 1. Schematic illustration of the proposed DropDim, including DropDim(random) and DropDim(span). The blue area indicates that the corresponding element is dropped. Similar to dropout we do not apply DropDim during inference.

Algorithm 1: DropDim.

1: **Input:** Transformer sub block output $\boldsymbol{h}$ as a $\boldsymbol{T} \times \boldsymbol{D}$ matrix, drop rate p, $mode$
2: **if** $mode == Inference$ **then**
3: return $\boldsymbol{h}$
4: **end if**
5: Randomly sample ξ: $\xi_i \sim \text{Bernoulli}(p)$, $i \leqslant D$
6: For each element ξ_i, create a mask vector $M_i \in \mathbb{R}^T$ with all elements equal to ξ_i.
7: Apply the mask: $\boldsymbol{h} = \boldsymbol{h} \times \boldsymbol{M}$
8: Normalization: $\boldsymbol{h} = \boldsymbol{h} \times count(\boldsymbol{M})/count\_ones(\boldsymbol{M})$

dimensions with a certain probability instead of randomly dropping independent units.

$$\boldsymbol{h} = DropDim(\boldsymbol{x} + sub\_block(\boldsymbol{x})) \tag{1}$$

Where $\boldsymbol{x}$ denotes the input of transformer encoder or decoder sub-block. Unless otherwise specified, DropDim is applied to all sub-blocks.

In detail, we propose two structured dropout methods: DropDim(random) and DropDim(span), which are shown in Fig. 1.

1) DropDim(random) means dropping independent embedding dimension. Algorithm 1 describes its pseudo code.
2) DropDim(span) means dropping span of the embedding dimension. Let α be the pre-defined value that determined the max length of the consecutive drop. Then, the span length l is chosen from the uniform distribution on the interval [0, α] and the starting index s for the span is randomly sampled as: $s \in \{0, 1, \ldots, D - l\}$ with D is the size of embedding dimension. Afterward, the embeddings between the s-th and $(s + l - 1)$-th positions are dropped.

III. Experimental Setup

A. Datasets

To prove the effectiveness and generalization of DropDim, we conduct experiments on three types of sequence-to-sequence tasks: ASR, MT, and ST. MUST-C is a speech translation corpus, including triplets of speech, transcription, and translation, which can perform different tasks flexibly. It is built from English TED talks, covering 8 language pairs. We conduct experiments on the English-German language pair. There are 234 K utterances totaling 408 hours in this parallel corpus, which is divided into training set (400 hours, 229,703 pronunciations), development set (3 hours, 1423 pronunciations), and tst-COMMON[1] set (5 hours, 2641 pronunciations).

B. Preprocessing and Evaluation

All the acoustic features used in this paper are 80-dimensional MFCC extracted using Kaldi [25] with global cepstral mean and variance normalization. The text data in the source language is normalized with lowercase conversion, tokenization, and punctuation removal. We apply punctuation normalization instead of punctuation removal for the text data in the target language. The processing of the text is realized by Moses scripts.[2] We apply BPE [26] on the combination of source and target text to obtain shared subword units, and the vocabulary size is set to 8 K.

The case-insensitive BLEU [27] calculated by the multi-bleu.pl script[3] is used to evaluate translation tasks (MT, ST). Moreover, we use Word Error Rate (WER) to evaluate the ASR system.

C. Training Settings

The models involved in this paper all adopt the Transformer structure. We conduct experiments based on Espnet [28] and use PyTorch-lightning [29] to organize our code. For both encoder and decoder in the MT model, the number of layers is 6, the number of attention heads is 4, the embedding dimension is 256, and the filter size of the feed-forward neural network is 2048. The attention and residual dropout rates are 0.0 and 0.1, respectively. For the ST and ASR model, the hyperparameters of encoder and decoder are 12 and 6, respectively. Other parameters remain the same with MT model. We train our models with Adam optimizer [30] on 4 NVIDIA V100 GPUs.

IV. Experimental Results

A. Main Results

We first experimentally verify the effectiveness of DropDim for ASR, MT, and ST tasks, and compare it with other methods. The results are summarized in Table I.

DropDim vs Dropout: The original transformer model did not apply dropout to the output of the residual connection. For the convenience of discussion, we define a transformer baseline in which DropDim in 1 is replaced with dropout. In all three tasks, DropDim and dropout share a similar trend and DropDim has a

[1]MuST-C is a multilingual dataset and this testset is the commonly shared utterances between the languages.
[2][Online]. Available: https://www.statmt.org/moses/
[3][Online]. Available: https://github.com/moses-smt/mosesdecoder/scripts/generic/multi-bleu.perl

TABLE I
MODEL PERFORMANCE UNDER DIFFERENT METHODS

Models	ASR(WER%↓)	MT(BLEU↑)	ST(BLEU↑)
Transformer(base)	19.1	26.90	21.13
+dropout*	18.7	27.10	21.54
+DropAttention	18.3	27.38	21.34
+Drophead	18.1	27.51	21.57
+label smoothing(0.1)	16.8	27.88	21.31
+Dropdim(random)	17.8	27.22	21.85
+label smoothing(0.1)	15.3	28.32	22.59
+Dropdim(span)	17.7	27.42	21.97
+label smoothing(0.1)	15.1	28.38	22.99

*: The Application Location of Dropout Here is the Same as DropDim.

TABLE II
EXPERIMENTS ON DIFFERENT DATASETS AND LANGUAGE PAIRS IN ST TASK. LABEL SMOOTHING IS USED IN ALL METHODS BY DEFAULT

Models	Librispeech En-Fr	MUST-C En-Fr
base	16.08	33.15
+Dropdim(random)	17.12	34.46
+Dropdim(span)	17.34	24.57

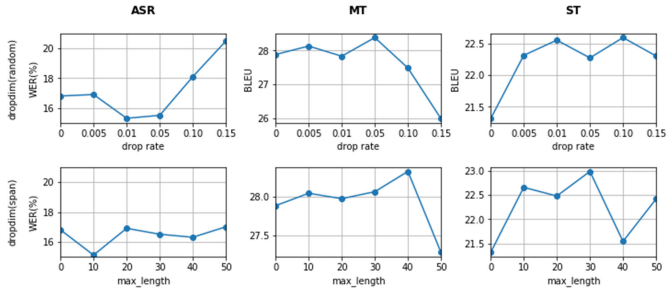

Fig. 2. The effect of different hyperparameters on model performance.

TABLE III
BLEU SCORE OF MODELS WHEN DROPDIM IS APPLIED ON DIFFERENT PART DURING TRAINING

task	Part		
	encoder	decoder	all
ASR(WER%↓)	16.9	16.2	15.1
MT(BLEU↑)	27.82	27.71	28.38
ST(BLEU↑)	22.66	22.13	22.99

large gain compared to the dropout results. This shows evidence that the DropDim is a more effective regularizer compared to dropout.

Comparison with DropAttention and Drophead: We compare with two popular structured dropout methods designed for self-attention mechanisms. All three methods can improve model performance. Specifically, on speech-related tasks (ASR, ST), DropDim achiceves better results. We believe this is because compared to DropAttention and Drophead, DropDim processes the input of the self-attention mechanism, which can completely drop semantic information and produce a stronger regularization effect. However, compared with speech, the text is coarse-grained, too strong regularization will hurt the MT model.

In the above comparison, DropAttention, Drophead, and dropout* do not use label smoothing. The same trend can be observed when label smoothing is added, and the results were not listed due to the limited number of pages.

Span vs Random: DropDim(span) delivers better results on all three tasks. This is due to the fact that in the case of high dimensions (256 in this article), it may not be that a single embedding dimension determines a semantics, but several adjacent embedded dimensions jointly express a semantics. And semantic information can be removed more effectively with continuous drop, which brings greater challenges to model training. We leave the specific details for future research.

Integration with other regularization techniques: Label smoothing [24] is a common regularization technique. The results in Table I show that both label smoothing and DropDim can improve model performance. Moreover, the performance can be further improved when combining DropDim and label smoothing. This shows that the overfitting problem is a direction that still needs to be studied, and there is no method that can completely solve it when used alone.

Results under different langugae pair and datasets: In order to further proves the effectiveness and generality of Dropdim, we performed additional experiments on the Librispeech En-Fr [31] and MUST-C En-Fr datasets in ST task. The results are listed in Table II. The experimental results showed the same trend as on MUST-C En-De datasets. The speech in Librispeech En-Fr was recorded on-site, which contains more noise, resulting the transcription and translation are not well aligned with the original audio. So, the model performance on this dataset is lower than on MUST-C En-Fr.

B. Ablation Studies

Effect of Hyperparameters: The hyperparameters of Dropdim determine the number of embedding dimensions that are dropped. The influence of hyperparameters on model performance is summarized in Fig. 2. It can be observed that determining a sweet point of the drop ratio is critical to the generalization ability of the resulting model. Specifically, when DropDim (random) is used, the best drop ratio for ASR, MT, and ST are 0.01, 0.05, and 0.10, respectively. While using DropDim (span), the best max_length (α) is 10, 40, and 30, respectively. Dropping too many embedding dimensions will bring a very small improvement to the model and even hurt the performance. This may be because the model unable to extract knowledge from the input for learning.

Effect on Different Part. In this section, we study the effect of DropDim on different parts of the model. Specifically, we apply DropDim separately on the encoder, decoder, and the whole model. The results are shown in Table III. We can see that the proposed approach is more effective when used on the whole model because of the strong regularization effect.

C. Analysis

DropDim drops more semantic information: We first train the model normally on ST task and then apply dropout or DropDim when testing. For a fair comparison, DropDim (random) is used here because with the same drop rate, the same number of neurons are dropped. The difference is that neurons are randomly dropped when dropout is used while using DropDim (random) will drops the entire embedding dimension. It can be seen that as the drop ratio increases, the yellow curve in Fig. 3 drops rapidly, indicating that DropDim can remove semantic information more effectively than dropout.

Results under different amounts of data: We test the effectiveness of DropDim under different amounts of data on the ST task. To simulate different data resource scenarios, we randomly

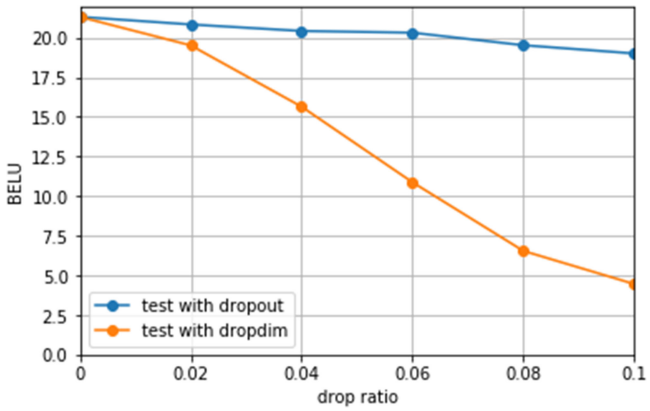

Fig. 3. The performance of the model using dropout or DropDim during testing.

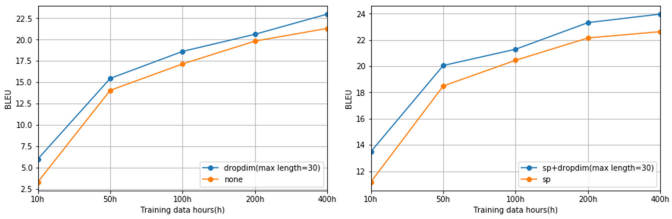

Fig. 4. Model performance with or without DropDim under different amounts of data. Left: No data augmentation is used. Right: Speed Perturbs of 0.9, 1.0, 1.1 are used for data augmentation.

select 10-hour, 50-hour, and 100-hour speech training data from the MUST-C English-German dataset, which originally contains 400-hour speech. Then the model is trained with or without DropDim. In addition, Speed Perturb [32] is a commonly used data augmentation method under low resource conditions. We also study the combination of DropDim and Speed Perturb. The results are shown in Fig. 4. The blue curve in the figure continues to be higher than the yellow curve, showing that DropDim can continuously improve the model performance under different amounts of data. Moreover, the combination with Speed Perturb can produce better results, indicating that the two methods are complementary.

D. Visualization

Attention Visualization: We further analyze how DropDim improve the model performance by visualizing the encoder-encoder and encoder-decoder attention maps. Specifically, the encoder-encoder attention is extracted from the last layer of the encoder, and the encoder-decoder attention is extracted from the last layer of the decoder. Fig. 5 shows an example. The attention in the ST model tends to be smooth between input frames. However, after applying DropDim, both types of attention become more concentrated, indicating that DropDim can not only help the model to remove interference but also make the model better align with the target.

Training Process Visualization: It can be seen from the network training curve in Fig. 6 that when DropDim is used, the network can underfit the training loss, which is in sharp contrast with the usual situation where networks tend to over-fit to the training data. This is consistent with specaugment [33], which

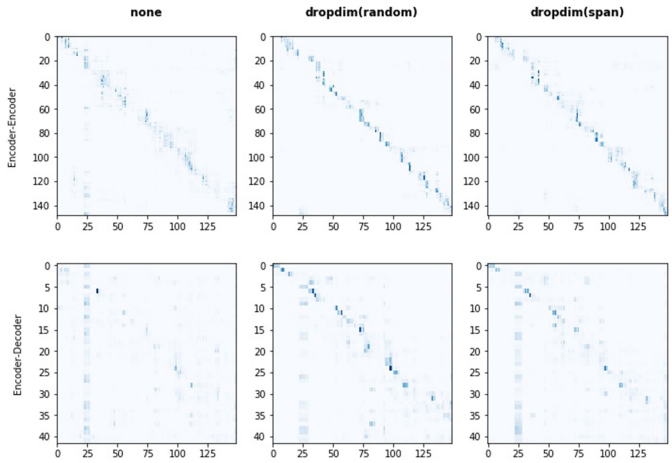

Fig. 5. The visualization of attention of different modules under different strategies. The horizontal coordinates represent the sequence of speech frames. In the attention map of Encoder-Encoder, the vertical coordinates represent the sequence of speech frames. In the attention map of Encoder-Decoder, the vertical coordinates represent the text token sequence.

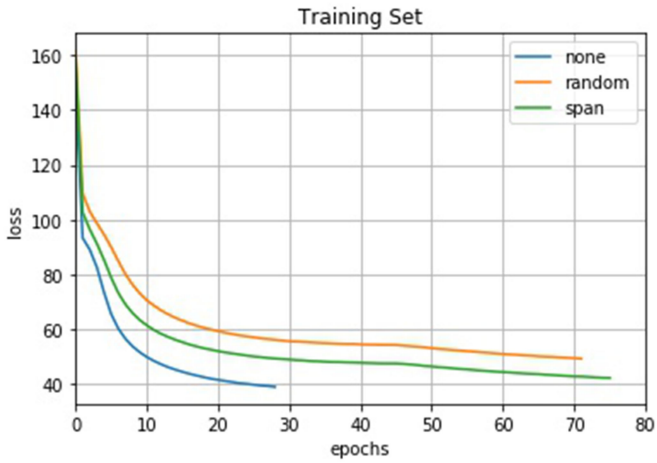

Fig. 6. Training process under different strategies.

converts over-fitting problems into under-fitting problems, and improves model performance through longer training time.

This also reveals to us that the combination of Dropdim and specaugment is not suitable. Because although this combination can improve performance, the training time will be further increased. Dropdim is more suitable to be combined with Speed Perturbs, because the latter improves the model performance by increasing the amount of data, which is different from Dropdim and specaugment.

V. Conclusion

In this paper, we introduce DropDim, a simple regularization method for training transformer models. We have verified the effectiveness of DropDim on a wide range of tasks, showing that it can continuously improve the model performance. Experimental analysis show that DropDim is also competitive under low-resource conditions and is complementary to other data augmentation methods.

REFERENCES

[1] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 4960–4964.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 3104–3112.

[3] R. J. Weiss, J. Chorowski, N. Jaitly, Y. Wu, and Z. Chen, "Sequence-to-sequence models can directly translate foreign speech," in *Proc. Interpseech*, 2017, pp. 2625–2629.

[4] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5998–6008.

[5] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 5884–5888.

[6] M. A. D. Gangi, M. Negri, and M. Turchi, "Adapting transformer to end-to-end spoken language translation," in *Proc. Interpseech*, 2019, pp. 1133–1137.

[7] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," 2021, *arXiv:2106.04554*.

[8] N.Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Proc. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 10727–10737 .

[10] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 648–656.

[11] D. Krueger *et al.*, "Zoneout: Regularizing RNNs by randomly preserving hidden activations," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 354–364.

[12] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Proc. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 12449–12460.

[13] Q. Zhang *et al.*, "Transformer transducer: A streamable speech recognition model with transformer encoders and RNN-T loss," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7829–7833.

[14] L. Zehui *et al.*, "Dropattention: A regularization method for fully-connected self-attention networks," in *Proc. Annual Meet. Assoc. Computat. Linguist.*, 2019, pp. 1872–1876.

[15] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one," in *Proc. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 14014–14024.

[16] W. Zhou, T. Ge, F. Wei, M. Zhou, and K. Xu, "Scheduled drophead: A regularization method for transformer models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2020, pp. 1971–1980.

[17] H. Wang, Y. Zou, and W. Wang, "Specaugment++: A hidden space data augmentation method for acoustic scene classification," in *Proc. Interpseech*, 2021, pp. 551–555.

[18] Z. Xie *et al.*, "Data noising as smoothing in neural network language models," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 621–627.

[19] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 1019–1027.

[20] S. Dalmia, Y. Liu, S. Ronanki, and K. Kirchhoff, "Transformer-transducers for code-switched speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2021, pp. 5859–5863.

[21] S. Dalmia, B. Yan, V. Raunak, F. Metze, and S. Watanabe, "Searchable hidden intermediates for end-to-end models of decomposable sequence tasks," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 1882–1896.

[22] D. Shen, M. Zheng, Y. Shen, Y. Qu, and W. Chen, "A simple but tough-to-beat data augmentation approach for natural language understanding and generation," 2020, *arXiv:2009.13818*.

[23] M. A. D. Gangi, R. Cattoni, L. Bentivogli, M. Negri, and M. Turchi, "Must-c: A multilingual speech translation corpus," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 2012–2017.

[24] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 2818–2826.

[25] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. Autom. Speech Recognit. Understanding Workshop*, 2011, pp. 512–518.

[26] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proc. Assoc. Comput. Linguistics*, 2015, vol. 1, pp. 1715–1725.

[27] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proc. Assoc. Comput. Linguistics*, 2002, pp. 311–318.

[28] H. Inaguma *et al.*, "ESPnet-ST: All-in-one speech translation toolkit," in *Proc. Assoc. Comput. Linguistics*, 2020, pp. 302–311.

[29] W. Falcon, "Pytorch lightning," vol. 3. Accessed: May 1, 2021. [Online]. Available: https://github.com/PyTorchLightning/pytorch-lightning

[30] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2014, pp. 748–759.

[31] A. C. Kocabiyikoglu, L. Besacier, and O. Kraif, "Augmenting librispeech with french translations: A multimodal corpus for direct speech translation evaluation," in *Proc. Int. Conf. Language Resour. Evaluat.*, 2018, pp. 621–627.

[32] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interpseech*, 2015, pp. 3586–3589.

[33] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interpseech*, 2019, pp. 2613–2617.