Ensemble of One Model: Creating Model Variations for Transformer with Layer Permutation

Andrew Liaw[1], Jia-Hao Hsu[2] and Chung-Hsien Wu[3]
Department of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, TAIWAN
E-mail: [1]yhandrew.liaw@hotmail.com, [2]jiahaoxuu@gmail.com and [2]chunghsienwu@gmail.com

Abstract— **Ensemble involves combining the outputs of multiple models to increase performance. This technique has enjoyed great success across many fields in machine learning. This study focuses on a novel approach to increase performance of a model without any increase in number of parameters. The proposed approach involves training a model that can have different variations that perform well and different enough for ensemble. The variations are created by changing the order of the layers of a machine learning model. Moreover, this method can be combined with existing ensemble technique to further improve the performance.**

The task chosen for evaluating the performance is machine translation with Transformer, as Transformer is the current state-of-the-art model for this task as well as many natural language processing tasks. The IWSLT 2014 German to English and French to English datasets see an increase of at least 0.7 BLEU score over single model baseline with the same model size. When combined with multiple model ensemble, minimum increase of 0.3 BLEU is observed with no increase in parameters.

I. INTRODUCTION

The past decade has seen neural networks achieving the state of the art across many machine learning tasks. Popular tasks such as image classification in the field of computer vision [1], and translation in natural language processing [2] have been dominated by neural network-based methods. A common technique to boost the performance of neural network is ensemble. Ensemble has seen great success in high profile competitions, such as Kaggle [3]. Ensemble works by combining the output of multiple models. The most common methods involve averaging the output of different models or deciding on the output with a vote. This generally offers better performance over the uncombined outputs, as long as there is suffice difference between the outputs of each model. During training, neural network will converge to one of the many local minima. These different local minima result in different outputs, which makes ensemble of independently trained models an effective method to increase performance [4].

Since the common ensemble technique relies on having multiple independently trained models, this results in multiple times the number of parameters, which in turn multiples the training time. These training cost can be quite costly for large models, as state of the art models requires multiple expensive graphic processing unit (GPU) that consumes hundreds of watts per hour running in parallel [5]. Increase in number of parameters also results in increase in memory usage, and may be of concern for low-powered devices.

Some previous work to address the cost associated with training multiple models for ensemble focused on extracting multiple models from a single training run. These past works often involved the use of cyclic learning rate schedule. Cyclic learning rate schedule, such as snapshot ensemble [3], involved repeating cycles of resetting to initial learning rate and quickly dropping to a low value. One checkpoint was saved at the end of each cycle, which resulted in several models equaling the number of cycles. These models were used for ensemble.

The core concept for snapshot ensemble and similar methods is to take multiple snapshots during one training run. These snapshots are sub-optimal models compared to model trained with standard learning rate schedule, but the ensemble of these sub-optimal models can perform better than the standard approach. Later work improves on this method by using different cyclic learning rate schedules. Fast Geometric Ensembling (FGE) was proposed after discovery of connections between local minimum [6]. The ensemble method was further improved by Stochastic Weight Averaging (SWA) by using weight averaging as the ensemble method [7].

These cyclic learning rate schedule methods differ from our proposed method in that they require obtaining multiple models prior to ensemble, which multiples the number of parameters. While, our method need only one model, which is used to create variations. In addition, our method can be applied along with other ensemble techniques, which further improves the performance under the same number of parameters. This will be detailed in the experiment section.

This study proposed a method for Transformer [8] to ensemble without increasing parameter. The experiment was carried out on machine translation, a task in which Transformer is the state-of-the-art method [9, 10], and other tasks [11, 12]. The main contributions of this study are as follows:

1) Improve the model performance with the same number of parameters for machine translation with Transformer.

2) Able to create exponential amount of variations of one model.

3) The output of each variation is different enough from each other to increase performance with ensemble.

4) Can be combined with other ensemble method to further increase performance.

5) The proposed method is simple to implement for both training and evaluation.

II. METHODS

The goal is to create a form of ensemble without requiring any additional parameters. To achieve this, a method of creating variations of a model is needed. The proposed method has taken inspiration from the recent success of applying stochastic depth training to the Transformer.

A. Inspiration

Stochastic depth training was proposed to assist in training Residual Network (ResNet) [13]. As deep ResNet can suffer from vanishing gradient and slow training time, by randomly dropping out layers during training, stochastic depth training acts as regularization for very deep ResNet.

Recently, deep Transformer model for speech recognition utilize this method as regularization as well [14, 15]. Other research has applied stochastic depth training or similar methods for different uses, such as research into extracting small high-quality model for a large model trained with stochastic depth training [15]. This is achieved by training a Transformer with dropout applied to the layers. After training, a smaller model can be extracted from the full model by keeping a subset of layers. Depth-adaptive Transformer is a Transformer which can learn to use different number of computational steps for different inputs [16]. This is done by having the Transformer select a subset of layers to run depending on the input. The training method for depth-adaptive Transformer can be seen as a heavily modified form of stochastic depth training.

The above instance of applying stochastic depth training to Transformer shows the Transformer can be resilient to layer manipulation during training and evaluation. As such the manipulation of layers is used to create variations for ensemble.

B. Proposed Methods

The core idea is to create a model that can manipulate its layers to create variations with different outputs. These variations of a model can be ensembled to produce better result. Since each layer processes the input information and output processed information, switch the order of layers is essentially changing the order of how the information is processed. The idea of changing order of information processing to create variations is partially inspired by the decoding method in statistical phrase-based machine translation. The decoding in phrase-based translation forms different hypotheses by translating different parts of a sentence in different orders, then selects the best hypothesis by beam search [17]. Using an example of translating "How is the weather today?", one hypothesis may start by translating "how is", then "the weather", and finally "toady". Another hypothesis may start with "today", then "how is", and "weather". Both hypotheses may result in good quality translations, despite that they do not process the information in the same order.

The take away is by processing the information in a different order, the output can be different, and ensemble of different outputs can lead to better performance. To change the order of information processing, the order of layer of a model is modified. As such, different permutations of layer will serve as variations. Using alphabets to denote each layer, a 4-layered model could have order of (A, B, C, D) for one variation and (B, C, D, A) for another variation.

The method to generate variations is to permute the layers.

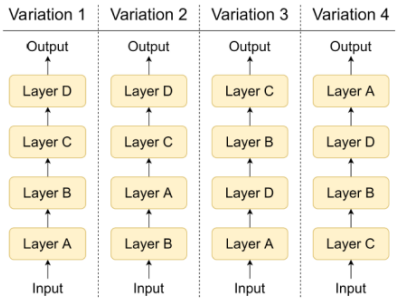

Fig. 1 Examples of different variations of a 4-layer model.

To provide structured permutation, the layers of a model is divided into pools. Permutation is done to the pools of layers. Using a 4-layered model as an example. The 4 layers could be divided into 2 pools with size 2 each. In which case, the first two layers can be shuffled and last two layers can be shuffled, but not between the two pools. Figure 2 lists out some valid and invalid permutations of layers for this example. For this work, the pools are always contiguous, therefore, the pools are described by their size. For example, a pool size of (2, 2) describes two pools with size two each. While, pool size of (1, 3, 1) refers to a model with a pool of size 1, followed by a pool of size 3, then a pool of size 1.

For the model to change the layer order and produce a valid

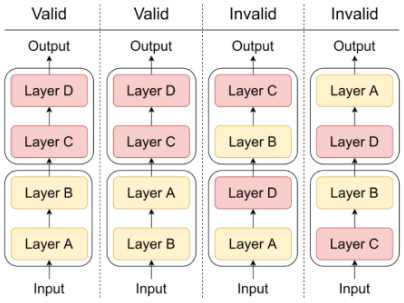

Fig. 2 Examples valid and invalid layer permutation for a 4-layer model with pool size (2, 2).

result, the training process needs to be modified to incorporate switching the layer order. Two methods of incorporating layer permutation into training are devised: set order and random order. Both methods involve switching the layer order at each update step during training.

For set order, a set of predetermined layer orders are decided on. During training, a layer order is randomly selected from the set for each step. For example, a four-layered model might be using layer order (A, B, C, D) and (A, C, B, D). At each step of the training process, one of these two-layer order is randomly selected to run the model. Algorithm 1 shows how the layer stack of a model is modified for set order training. This means for each update step, the same set of parameters are trained in a different configuration. This is similar to train multiple models with weight tying between different layers. Since all

the parameters are shared between the variations, no increase in parameter is needed.

Algorithm 1: One run of the layers for set order

Input: input of the layers: x, set of layer order: O
Output: output of the layers
o ← randomly select one order from O
for $i \in$ **o do**
  $x \leftarrow \mathbf{Layer}_i(x)$
end
return x

For random order, a random layer order is selected from all valid permutations of layers for each step of training. With the pool size dictating what constitute valid permutations. Algorithm 2 shows how the layer stack of a model is modified for random order training. An implication of using all valid permutations is an exponential amount of variations which could be trained together. For a six-layered model with pool size (6), there could be 720 permutations/variations. With the encoder-decoder architecture used in machine translation, if both encoder and decoder have 6 layers with pool size (6), a total of 518,400 variations are possible. This setup is used in the experiment, producing model that has 518,400 well performing variations, that can be ensembled for better performance.

Algorithm 2: One run of the layers for random order

Input: input of the layers: *x*, list of layer pool size: *S*
Output: output of the layers
$i \leftarrow 0$
L ← list of all layers index
for $s \in S$ **do**
  RandPerm(L(*i*...*i*+*s*-1))
  $i \leftarrow i + s$
end
for $j \in$ L **do**
  $x \leftarrow \text{Layer}_j(x)$
end
return x

Function RandPerm(list):
  return in-place random permutation of list

C. Evaluation

After training, the model is evaluated with random set of valid variations, then the outputs are ensembled. For set order training, valid variations are the set of layer orders used for training. For random order training, valid variations are any valid layer permutation as dictated by the pool size. The outputs of the models can be ensembled by any ensemble technique. The method of ensemble for the experiment is average ensemble.

III. EXPERIMENTS

The task used for evaluating the model was sentence-level translation. The baseline of single model Transformer was compared with a single model with ensemble of 5 variations. In addition, ensemble of 3 and 5 baselines was compared with ensemble of 3 and 5 models of the proposed method. The datasets used were the IWSLT 2014 German to English and French to English [18]. BLEU score [19] was used for evaluating the quality of translation. All experiments were conducted on a single Nvidia RTX 2080 Ti. The baseline was provided by Fairseq framework [20], and the model was implemented with the framework as well.

A. Experimental Setup

The datasets were cleaned and tokenized with Moses [21]. Byte pair encoding (BPE) [22] was used to split the vocabulary into subword units of the datasets. For both datasets, a total of 10k subword units was used. Label smoothing [23] of 0.1 was used. All models were trained with Adam [24] as optimizer with β_1=0.9, β_2=0.98, and ϵ= 〖10〗 ^(-9). The learning rate schedule was the same as the original Transformer [8] with 4000 step of warmup, and learning rate of 5× 〖10〗 ^(-4). Dropout [25] of 0.3 was applied. All models in the experiments were the Transformer [8] with 6 layers for encoder, 6 layers for decoder, model dimension of 512, 4 heads for multi-head attention, and 1024 dimension for inner dimension of position-wise feed-forward network. The data were batched with a maximum token count of 4096. This is the same setting as recommended by Fairseq [20] for the dataset. For generation of translated sentence, beam search of beam size 5 with length penalty of 0.6 was used. All ensembles were done by averaging the model output probability distribution during generation.

B. Experimental Results

This section compared the baseline of standard Transformer with the proposed method. The proposed method used a pool size of (6) for encoder and decoder, and random order training. After training, the proposed method randomly selected 5 variations for ensemble. All reported result in this section was repeated with 5 trials. The mean and standard deviation were reported.

Table 1 *Results of single model baseline and proposed method.*

IWSLT 2014 German to English		
Model	**BLEU score**	**Parameters**
Baseline	34.63 (σ: 0.109)	39M
Proposed Method	35.42 (σ: 0.168)	39M
IWSLT 2014 French to English		
Model	**BLEU score**	**Parameters**
Baseline	39.75 (σ: 0.083)	40M
Proposed Method	40.46 (σ: 0.124)	40M

For both datasets, the proposed method outperformed the baseline. From Table 1, the IWSLT 2014 German to English saw an improvement of 0.79 BLEU score, and IWSLT 2014 French to English saw an improvement of 0.71 BLEU score with the same number of parameters used by the baseline and the proposed method.

To show the performance of each variation, the average of the 25 randomly selected variations across 5 trials were reported in Table 2. This shows the training method did not negatively impact the performance of the model.

Table 2 *Results of variations from the proposed method.*

IWSLT 2014 German to English		
Model	**BLEU score**	**Parameters**
Baseline	34.63 (σ: 0.109)	39M
Single Variation	**34.69** (σ: 0.134)	39M
IWSLT 2014 French to English		
Model	**BLEU score**	**Parameters**
Baseline	39.75 (σ: 0.083)	40M
Single Variation	**39.77** (σ: 0.124)	40M

To demonstrate the proposed method can be combined with other ensemble techniques, the experiments combined average ensemble of multiple models with the proposed method. The baseline was the average ensemble of multiple models, while the proposed method was the ensemble of multiple variations across multiple models. Results of ensemble of 3 model and ensemble of 5 models were reported. For the proposed method, 5 variations were randomly selected from each model. In the ensemble of 3 models, this resulted in a total of 15 variations, and 25 variations for ensemble of 5 models.

Table 3: *Results of multiple model ensemble of baseline andproposed method.*

IWSLT 2014 German to English		
Model	**BLEU score**	**Parameters**
3 Model Baseline	36.43	39M × 3
3 Model Proposed Method	**36.76**	39M × 3
5 Model Baseline	36.82	39M × 5
5 Model Proposed Method	**37.12**	39M × 5
IWSLT 2014 French to English		
Model	**BLEU score**	**Parameters**
3 Model Baseline	40.91	40M × 3
3 Model Proposed Method	**41.24**	40M × 3
5 Model Baseline	41.36	40M × 5
5 Model Proposed Method	**41.70**	40M × 5

From Table 3, the proposed method showed improvements of at least 0.3 BLEU score across 3 model, and 5 model ensemble on both datasets, with no increase in parameter over baseline. As reference, a 66% increase of parameters between ensemble of 3 baseline model and 5 baseline models resulted in an improvement of 0.39 BLEU score for German to English, and 0.45 BLEU score for French to English.

Table 4: *Results of hyperparameter selection.*

IWSLT 2014 German to English			
Training	**Pool Size**	**1 Model**	**5 Model**
set order	(1, 4, 1)	35:07	36.86
random order	(1, 4, 1)	35.15	**37.12**
set order	(6)	35.27	36.94
random order	(6)	**35.42**	36.52
IWSLT 2014 French to English			
Training	**Pool Size**	**1 Model**	**5 Model**
random order	(1, 4, 1)	40.11	**41.70**
random order	(6)	**40.46**	41.42

The proposed method introduced two types of hyperparameters: training method and pool size. Both training methods and different pool sizes were tested on both datasets in Table 4. The result showed random order training and pool size of (6) performed the best for single model, while random order training with pool size of (1, 4, 1) performed the best for multiple model ensemble. In both cases, the random order training performed the best. For single model scenario, a large pool size was better. For multiple model ensemble, having restriction in permutation can help.

There were several neural machine translation approaches, and some of them which also used IWSLT 2014 database were selected for comparison, as shown in Table 5. These works differed in many aspects, such as ensemble [26], training step design [27], data augmentation [28], and model architecture design [29]. Among them, data augmentation has been the mainstream improvement method in recent years. It can be found from the table that the performance of our proposed method is similar to that of the existing methods, but for the method of ensemble, our method can improve the performance a lot. And it can be easily and universally applied to the existing models and other ensemble design.

Table 5: *Results of the existing methods on IWSLT'14 (De-En)*

Methods (Ensemble)	**BLEU score**
ConvS2S Ensemble[26]	34.61
Ours	37.12
Methods (Single model)	**BLEU score**
Tied-Transformer[27]	35.52
Soft contextual DA[28]	35.78
BERT-fused[29]	36.11
Ours	35.42

IV. CONCLUSIONS

Ensemble is a powerful machine learning tool. Our proposed method focuses on creating variations from a single model for ensemble. Two training method and one way to describe how to permute the layers are proposed. The proposed method results in better performance with the same number of parameters. In addition, our method can be combined with existing ensemble method to further improve performance. For single model ensemble, a large pool size is beneficial, while restriction for permutation is better for multiple model ensemble. Random order training performs better in both cases.

Experiments shows improvements of at least 0.7 BLEU score over the baseline with no increase in parameters. When combined with multiple model ensemble, the proposed method beats the baseline by at least 0.3 BLEU score with the same number of parameters. As reference, it requires 66% increase in parameters to increase BLEU score by 0.39 on IWSLT 2014 German to English between 3 model ensemble and 5 model ensemble. And the proposed method can perform well comparing with existing studies. As state-of-the-art models are on the scale that requires hundreds of GPUs to train, which is outside the reach of a single GPU. Future works on applying proposed method to training large models efficiently is needed.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, pp. 1097-1105, 2012.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104-3112.

[3] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger, "Snapshot ensembles: Train 1, get m for free," arXiv preprint arXiv:1704.00109, 2017.

[4] Lars Kai Hansen and Peter Salamon, "Neural network ensembles," IEEE transactions on pattern analysis and machine intelligence, vol. 12, no. 10, pp. 993-1001, 1990.

[5] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli, "Scaling neural machine translation," arXiv preprint arXiv:1806.00187, 2018.

[6] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 8803-8812.

[7] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson, "Averaging weights leads to wider optima and better generalization," arXiv preprint arXiv:1803.05407, 2018.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998-6008.

[9] Marcin Junczys-Dowmunt, "Microsoft translator at WMT 2019: Towards large-scale document-level neural machine translation," arXiv preprint arXiv:1907.06170, 2019.

[10] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov, "Facebook FAIR's WMT19 News Translation Task Submission," arXiv preprint arXiv:1907.06616, 2019.

[11] Ming-Hsiang Su, Chung-Hsien Wu, and Hao-Tse Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2061-2072, 2020.

[12] Yi-Hsuan Wang, Jia-Hao Hsu, Chung-Hsien Wu, and Tsung-Hsien Yang, "Transformer-based Empathetic Response Generation Using Dialogue Situation and Advanced-Level Definition of Empathy," in 2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP), 2021: IEEE, pp. 1-5.

[13] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, "Deep networks with stochastic depth," in European conference on computer vision, 2016: Springer, pp. 646-661.

[14] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, Sebastian Stüker, and Alexander Waibel, "Very deep self-attention networks for end-to-end speech recognition," arXiv preprint arXiv:1904.13377, 2019.

[15] Angela Fan, Edouard Grave, and Armand Joulin, "Reducing transformer depth on demand with structured dropout," arXiv preprint arXiv:1909.11556, 2019.

[16] Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli, "Depth-adaptive transformer," arXiv preprint arXiv:1910.10073, 2019.

[17] Philipp Koehn, Franz J Och, and Daniel Marcu, "Statistical phrase-based translation," UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2003.

[18] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico, "Report on the 11th iwslt evaluation campaign, iwslt 2014," in Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam, 2014, vol. 57.

[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," in Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311-318.

[20] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli, "fairseq: A fast, extensible toolkit for sequence modeling," arXiv preprint arXiv:1904.01038, 2019.

[21] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, and Richard Zens, "Moses: Open source toolkit for statistical machine translation," in Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions, 2007, pp. 177-180.

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," arXiv preprint arXiv:1508.07909, 2015.

[23] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton, "Regularizing neural networks by penalizing confident output distributions," arXiv preprint arXiv:1701.06548, 2017.

[24] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[25] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012.

[26] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin, "Convolutional sequence to sequence learning," in International Conference on Machine Learning, 2017: PMLR, pp. 1243-1252.

[27] Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin, "Tied transformers: Neural machine translation with shared encoder and decoder," in Proceedings of the AAAI Conference on Artificial Intelligence, 2019, vol. 33, no. 01, pp. 5466-5473.

[28] Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu, "Soft contextual data augmentation for neural machine translation," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5539-5544.

[29] Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu, "Incorporating bert into neural machine translation," arXiv preprint arXiv:2002.06823, 2020.