SCALABLE TRANSFORMERS FOR NEURAL MACHINE TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer has been widely adopted in Neural Machine Translation (NMT) because of its large capacity and parallel training of sequence generation. However, the deployment of Transformer is challenging because different scenarios require models of different complexities and scales. Naively training multiple Transformers is redundant in terms of both computation and memory. In this paper, we propose a novel scalable Transformers, which naturally contains sub-Transformers of different scales and have shared parameters. Each sub-Transformer can be easily obtained by cropping the parameters of the largest Transformer. A three-stage training scheme is proposed to tackle the difficulty of training the scalable Transformers, which introduces additional supervisions from word-level and sequence-level self-distillation. Extensive experiments were conducted on WMT EN-De and En-Fr to validate our proposed scalable Transformers. Pre-trained model has been included into supplement material.

1 INTRODUCTION

Transformers Vaswani et al. (2017) have demonstrated its superior performance on Machine Translation (NMT) Wu et al. (2016); Devlin et al. (2018). However, the scale (number of parameters and FLOPs) of the Transformers cannot be altered once trained. This is contradictory to the different scenarios of NMT which require models of different scales. For instance, NMT systems on a phone should have lower computational cost while those on clusters aim to achieve higher accuracy. A naive approach would be to separately train models of different scales. The Transformer of desired scale is then deployed to the target scenario. However, such a strategy requires training and maintaining multiple Transformers. A natural question is: can we build a single but scalable Transformers that can be flexibly scaled up or down to run with different FLOPs, without sacrificing the translation accuracy at corresponding FLOPs?

In this paper, we propose a scalable Transformers (ST) that can adjust feature dimensions of all encoder and decoder layers within a large range of widths (from 256 to 1024) without re-training. The largest model in our scalable Transformers has 1024 feature dimensions at all layers, and 6 encoder and 6 decoder layers. After properly training, its neural layers can be cropped to form sub-models. For example, activating the first 512 dimension in each layer, we would obtain a sub-Transformer with $1/4$ of its full parameters but can still perform translation accurately with limited performance drop. The sub-models of different scales share parameters with the largest model.

However, properly training the parameter-sharing sub-models of different scales is non-trivial. Jointly training them results in worse performance than independently-trained counterparts, due to the interference between the parameter-sharing models. To solve the issue, we propose to incorporate online word-level and offline sequence-level self-distillation Kim & Rush (2016) to effectively supervise the training of the scalable Transformers. During training, we randomly sample Transformers of different scales. The word-level predictions generated by the largest Transformer would serve as the additional supervisions for self-training smaller-scale sub-models with shared parameters. The training inference between sub-models can be mitigated by this strategy from two aspects. On the one hand, since the small sub-models share all their parameters with the largest Transformer, the predictions generated from the largest Transformer are easier to fit than the hard ground-truth supervisions. On the other hand, if the smaller sub-models are better trained, they would in turn enhance the performance of the largest model as their parameters are all included in the largest model.

We then generate offline sequence-level predictions from the largest Transformer as supervisions for further finetuning the smaller-scale sub-models. After three stage training, all Transformers reach their optimal performances. Our scalable Transformers has been tested on WMT'14 En-De and En-Fr benchmarks. All sub-models achieve better or comparable performances than their independently trained counterparts but with fewer overall parameters.

Our contributions can be summarized as threefold. (1) We present a novel scalable Transformers which can dynamically scale across a wide spectrum of parameters and FLOPs with shared parameter. (2) We propose a novel three-stage training strategy for the scalable Transformers, which include online word-level and offline sequence-level self-distillation for mitigating training interference between the sub-models. (3) We perform extensive experiments on WMT'14 En-De and En-Fr datasets, which show favourable performances of our scalable Transformers at different scales.

2 Related Work

NMT Architectures. Statistical Machine Translation (SMT) Forcada et al. (2011) dominated NMT in the early years. Seq2Seq Sutskever et al. (2014) then surpassed SMT and has become the mainstream for NMT. The main recent research turned to design architectures for seq2seq. NMT architectures evolved from LSTM Hochreiter & Schmidhuber (1997), CNN Gehring et al. (2017), DepthwiseCNN Kaiser et al. (2017) to Transformer Vaswani et al. (2017). The Evolved Transformer So et al. (2019) conducted architecture search using evolution algorithms and achieved good performance on NMT. Universal Transformer Dehghani et al. (2018) proposed to share all layers of the encoder and decoder. Scaling Neural Machine Translation (SNMT) Ott et al. (2018) performs extensive ablation study on training the Transformer. Previous NMT research focuses on architectures and training tricks while our scalable Transformers is proposed to build an architecture that can be flexibly adjusted to meet a large spectrum of resource constraints with a single model.

Distillation and self-distillation. Knowledge Distillation (KD) Hinton et al. (2015) was first proposed to transfer knowledge between teacher and student networks. DistillBERT Sanh et al. (2019) adopts KD to train a small BERT model given a large BERT model. Sequence-level distillation Kim & Rush (2016) generalize KD to sequence generation. KD generally transfers knowledge between teacher and independent students different capacities. There are also self-distillation methods Yang et al. (2019b); Xie et al. (2019); Yang et al. (2019a) that use its previous versions to provide additional supervisions for improving itself. In contrast, we generate both online soft and offline hard supervisions from the largest model to train its parameter-sharing sub-models.

Adaptive neural network. Tradition neural networks have fixed architectures and computation complexity in training and testing. Recently, neural networks with dynamic computation scales have been proposed in NLP and computer vision. Huang et al. (2016) proposed a stochastic depth training strategy for convolution neural networks, where residual layers are randomly dropped. Huang et al. (2017); Graves (2016); Figurnov et al. (2017); Graves (2016) proposed adaptive computational mechanism for RNN, CNN and Transformer. For adaptive computation, the calculation would stop if the confidence score at certain layer is higher than a threshold. Adaptive computation can therefore early terminate the calculation for saving cost on easy-to-predict samples. Slimmable Neural Network Yu et al. (2018); Yu & Huang (2019) was proposed to train width adjustable Convolution Neural Network (CNN). Although both slimmable NN and our scalable Transformers adjust layer width to achieve different model capacities. There are two key differences. (1) Slimmable NN can only support equal layer widths for all layers while our scalable Transformers allows flexibly setting different widths for different layers. (2) We handle training interference via a novel self-distillation training strategy while slimmable NN uses a small set of separate BN parameters, which are not feasible for our scalable Transformers with much freedom.

Parameter sharing between a network and its sub-networks Pham et al. (2018) has been investigated for Neural Architecture Search (NAS). Different from Pham et al. (2018), we aim to train all sub-network jointly with the largest network and achieve good performance with self-distillation.

3 SCALABLE TRANSFORMER

In this section, we first briefly revisit the Transformer for machine translation. We then explain how to modify conventional Transformer to implement our scalable Transformers (ST), which can flexibly choose its feature dimensions to match different target scales once trained (see Figure 1 for illustration). Since the training of the scalable Transformers is non-trivial, we propose a three-stage training strategy, which consists of independent training (stage 1), online word-level self-distillation (stage 2), offline sequence-level self-distillation (stage 3).

3.1 A REVISIT OF TRANSFORMER FOR MACHINE TRANSLATION

Embedding. Let $S \in \mathbb{R}^{N\times L}$ denote a source sentence of length L, where each word is represented by an one-hot vector of vocabulary size N. The input sentence S are encoded by word embedding W_e as $E = W_e S$, where $W_e \in \mathbb{R}^{N\times C}$.

Transformer layer. Each Transformer layer contains one multi-head attention sub-layer followed by FFN sub-layer. The input features (word embedding E for the 1st layer) are linearly projected into key, query, value, $K, Q, V \in \mathbb{R}^{L\times D}$ and propagated between positions as

$$K = W_k E + b_k, \quad Q = W_q E + b_q, \quad V = W_v E + b_v, \tag{1}$$
$$\text{Attention}(Q,K,V) = \text{softmax}\left(QK^T/\sqrt{d}\right)V,$$

where $W_k, W_q, W_v \in \mathbb{R}^{C\times D}$ and $b_k, b_q, b_v \in \mathbb{R}^D$. The multi-head attention further splits the Q, K, V features along the channel dimension to form groups of features. Another linear projection converts the attention output feature back to C-dimensional followed by a two-layer FFN sub-layer,

$$\text{FFN} = \text{ReLU}(XW_1 + b_1)W_2 + b_2, \tag{2}$$

where X is the input feature of the FNN, $W_1 \in \mathbb{R}^{C\times 4D}, b_1 \in \mathbb{R}^{4D}, W_2 \in \mathbb{R}^{4D\times C}, b_2 \in \mathbb{R}^C$ are the transformation parameters. Each Transformer layer contains the above two sub-layers. A residual connection is also added around each of the above sub-layers followed by layer normalization.

3.2 SCALABLE TRANSFORMERS

Architecture and word predictions. The Transformer adopts an encoder-decoder architecture, each of which consists of 6 Transformer layers. Given the last output feature $O \in \mathbb{R}^{N\times C}$, it is mapped back to the word embedding space to predict the output words as $\text{softmax}(OW_e^T)$, where W_e is the transposed word embedding matrix.

The goal of our scalable Transformers is that, once trained, the FLOPs and number of parameters of the Transformer can be flexibly adjusted according to different use scenarios and without network re-training. To achieve this goal, we make the *width* of our encoder and decoder layers to be flexibly modified within a pre-defined range. Our scalable Transformers shares the parameters with sub-models of different widths. Specifically, for each layer, the wider Transformer contains all the parameters and computations of the sub-Transformers. If the widest Transformer is properly trained, we can obtain smaller scale sub-Transformers by simply truncating Transformer layers' width and cropping the parameter matrices. Sub-Transformers cropped from the widest Transformer can still conduct accurate translation with limited performance drops. The widest Transformer layer has $C = M_{\max}$ and $D^i = M_{\max}$ as the input-output and attention feature dimension. The parameter matrices' sizes can be determined accordingly. For instance, we denote the attention matrix parameters at layer i of the widest Transformer as $W^i_{k,\max}, W^i_{q,\max}, W^i_{v,\max}$, which are all of size $M_{\max} \times M_{\max}$.

Narrower Sub-Transformer layers. Once the widest Transformer's parameters are defined, a narrower sub-Transformer's layer i can be obtained by cropping a subset of parameters from the widest layer i defined above. A narrower layer i would adopt a smaller input-output dimension $C < M_{\max}$ as well as a smaller attention feature dimension $D^i < M_{\max}$. Without loss of generality, here we only discuss how to obtain the attention-key parameters of the narrower layer i, $W^i_k \in \mathbb{R}^{C\times D^i}$ and $b \in \mathbb{R}^D$. The same operations can be generalized to obtain other parameters (W^i_q, W^i_v,W^i_o ,W^i_1, W^i_2, etc.) of the narrower Transformer's layer i. The attention-key parameters can be obtained as

$$W^i_k = W^i_{k,\max}[1{:}C, 1{:}D^i], \;\; b^i_k = b^i_{k,\max}[1{:}D^i],$$

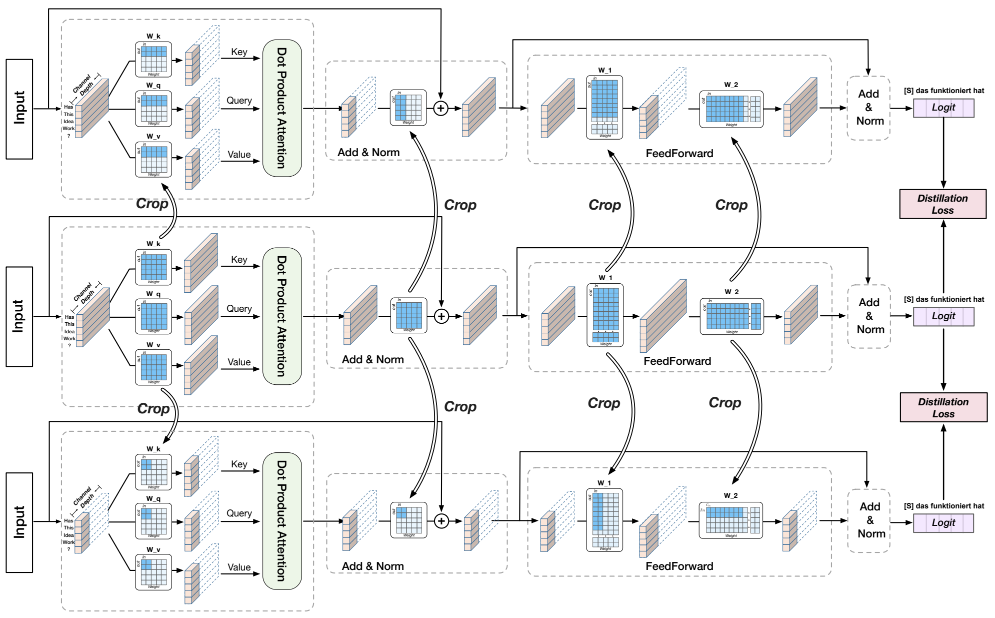

Figure 1: The proposed scalable Transformers (ST) can flexibly change different layers' feature width to meet different computational constraints. (Middle) The widest Transformer of a scalable Transformers. (Top) A narrower sub-Transformer with fixed input-output feature dimension and varying attention feature dimensions for different layers. The weights of sub-Transformers are directly cropped from the widest Transformer without re-training. (Bottom) A narrower sub-Transformer with the same input-output and attention feature dimension for all layers.

where $W^i_{k,\max}[1{:}C, 1{:}D^i]$ is the C-row and D^i-column top-left sub-matrix of the widest matrix W^i_k, and $b^i_k[1{:}D^i]$ denotes the first D^i dimensions of the widest bias vector b^i_k. It is obvious that the widest Transformer layers contains all the parameters and computations of the narrower ones, and thus avoid introducing redundant parameters. The narrower sub-Transformers can then be obtained by stacking such sub-Transformer layers with different D^i dimensions. Note that although the input-output dimension C can be flexibly adjusted, we make the entire narrower sub-Transformer share the same C dimensions across all layers because of the requirement of residual connections, while D^i can be different for different layers.

Input and output projections. Our scalable Transformers only modifies the width of the Transformer layers. The dimension of the word embeddings remains to be $M_{\max}$ for the sub-Transformers. To maintain the word embedding dimensions, for the widest Transformer, we use one additional linear projection to convert the word embeddings E to $E' = W'_{e,\max}E + b'_{e,\max}$, where $W'_{e,\max} \in \mathbb{R}^{M_{\max}\times M_{\max}}, b'_{e,\max} \in \mathbb{R}^{M_{\max}}$. Similarly, another linear projection is introduced to transform the FFN output features $O \in \mathbb{R}^{N\times M_{\max}}$ to $O' = W'_{o,\max}O + b'_{o,\max}$, which are then used for final word predictions. For narrower sub-Transformers with smaller intermediate feature width $C < M_{\max}$, we crop the parameters from the widest input and output projections for sub-Transformers so that the the word embedding can be shared across different scales,

$$\begin{aligned} W'_e &= W'_{e,\max}[1{:}M_{\max}, 1{:}C], \qquad & b'_e &= b'_{e,\max}[1{:}C], \\ W'_o &= W'_{e,\max}[1{:}C, 1{:}M_{\max}], \qquad & b'_o &= b'_{o,\max}. \end{aligned} \tag{3}$$

Scalable Transformer variants. In our experiments, we mainly experiment with two types of scalable Transformers. In the *type-1* scalable Transformers, each sub-Transformer's all layers adopt the same width from $\mathcal{M}$ for the input-output and attention feature dimensions, i.e., $C = D^1 = \cdots = D^{12}$. Therefore, there would be $|\mathcal{M}|$ sub-Transformers of different widths in total after the scalable Transformers is trained. In the *type-2* scalable Transformers, we fix the input-output dimension $C = M_{\max}$ but allows freely choosing the attention dimension D^i from any width in $\mathcal{M} = \{M_1, \cdots, M_{\max}\}$. Therefore, $|\mathcal{M}|^{12}$ different sub-Transformers exist after the ST is trained.

3.3 Training scalable Transformers with Self-distillation

Training the scalable Transformers is quite challenging as it requires all sub-Transformers with shared parameters to have superior performance. Simply training of all sub-Transformers independently cannot result in satisfactory performances because of the gradient interference between sub-models. To tackle this issue, we propose a novel three-stage training strategy. The key idea of training is to distill the knowledge from the widest Transformer and use them as the supervisions for narrower sub-Transformers. Compared with ground-truth annotations, the predictions by the widest Transformers are easier to fit for the sub-Transformers as all of them share parameters to certain degrees. On the other hand, if the sub-Transformers are properly trained, their parameters could also boost the performance of wider Transformers that contain them. We propose a three-stage training strategy, where stage-1 focuses on training the widest Transformer and its sub-Transformers independently, and stage-2 and stage-3 utilize word-level and sequence-level self-distillation.

Joint sub-Transformer pre-training. The stage-1 pre-training conducts jointly training on all the sub-Transformers. At each iteration j, we randomly sample a few sub-structures and always include the widest Transformer $T_{\max}$ for parameter updating. For each sub-Transformer, we randomly sample the input-output dimension C and the attention width D^i from $\{M_1, \cdots, M_{\max}\}$ for their Transformer layers following the layer width constraints of type-1 or type-2 models, i.e., we randomly sample from $|\mathcal{M}|$ and $|\mathcal{M}|^{12}$ sub-structures respectively for independent pre-training. All the sampled architectures are trained with the cross-entropy loss $\mathcal{L}_{ce}$ on ground truth words,

$$\mathcal{L}_1 = \mathcal{L}_{ce}(T_{\max}(S); G) + \sum_{T \in \mathbb{T}^{(j)}} \mathcal{L}_{ce}(T(S); G), \tag{4}$$

where G is the ground-truth sentence, $T_{\max}(S)$ and $T(S)$ denote the predicted words by the widest Transformer $T_{\max}$ and the sub-Transformer T given the input sequence S, and $\mathbb{T}^{(i)}$ denotes the set of sampled sub-Transformers at iteration j.

Although joint sub-model pre-training actually results in worse performance than training only the widest Transformer, it pretrains all the sub-structures to have gradients of similar scales and also makes compromises on the shared parameters. Our stage-1 pre-training is almost the same as slimmable NN Yu et al. (2018); Yu & Huang (2019) because Transformer uses Layer Normliazation to avoid Batch Normalization. If removing the separate BNs in slimmable NN, it is equivalent to jointly train the largest model and its sub-models.

Annealed word-level self-distillation. In stage-1 pre-training, the widest model is always updated in each iteration to ensure that it is more sufficiently trained than sub-Transformers. However, the interference between the models with shared parameters prevents them from effective learning. To mitigate the difficulty of training the sub-Transforms to fit the "hard" ground truth (one-hot vectors), we propose to distill the knowledge from the widest Transformer for training its sub-Transformers. Specifically, for each input sequence S, the soft predictions of the widest Transformer $T_{\max}(S)$ is used as additional training supervisions for its sub-Transformers T with the cross-entropy loss,

$$\mathcal{L}_2 = \mathcal{L}_{ce}(T_{\max}(S);\, G) + \lambda_2^{(j)} \sum_{T \in \mathbb{T}^{(j)}} \mathcal{L}_{ce}(T(S);\, G) + (1 - \lambda_2^{(j)}) \sum_{T \in \mathbb{T}^{(j)}} \mathcal{L}_{ce}(T(S);\, T_{\max}(S)),$$

where the first two terms are the same as stage-1 loss, $\mathcal{L}_{ce}(T(S)\,; T_{\max}(S))$ denotes using the widest model's predictions as learning targets for its sub-Transformers T, and $\lambda_2^{(j)}$ is a relative weight at iteration j balancing the soft supervisions. In early iterations, our pre-trained scalable Transformers from stage-1 still struggles because the predicted word probabilities $T_{\max}(S)$ might be noisy. Therefore, we make $\lambda_2^{(j)}$ closer to 1 when j is small to more rely on the hard ground-truth words. $\lambda^{(j)}$ then gradually decreases to involve the soft predictions as additional supervisions as

$$\lambda_2^{(j)} = \begin{cases} 1 - 0.5\frac{j}{t_j}, & \text{if } j < t_j, \\ 0.5, & \text{if } j \geq t_j. \end{cases} \tag{5}$$

t_j is an iteration threshold, after which $\lambda_2^{(j)}$ is fixed to be 0.5. In this way, the sub-Transformers can gradually fit the soft predictions. Because the sub-Transformers' are also parts of the widest Transformer. The widest Transformer can also be improved, which in turn further provides more accurate supervisions for training sub-Transformers.

Sequence-level self-distillation. After stage-2 training, both the ground-truth labels as well as the word-level soft predictions might still be noisy to hinder the further improvements of the models. For stage-3 training, we conduct offline beam search with the widest Transformer to generate refined sequence-level predictions for all the training sequences. For our widest Transformer, it is trained with only the sequence-level refined predictions; for its sub-Transformers, they are trained with both the offline beam-searched predictions and online word-level predictions with the below loss,

$$\mathcal{L}_3 = \mathcal{L}_{ce}(T_{\max}(S);\, T_{\max}^{\text{beam}}(S)) + \lambda_3 \sum_{T\in\mathbb{T}^{(j)}} \mathcal{L}_{ce}(T(S);\, T_{\max}^{\text{beam}}(S)) + (1-\lambda_3) \sum_{T\in\mathbb{T}^{(j)}} \mathcal{L}_{ce}(T(S);\, T_{\max}(S)), \tag{6}$$

where the 1st term is the loss for training the widest model, the 2nd and 3rd terms are for training the sub-Transformers with both offline sequence-level predictions $T_{\max}^{\text{beam}}(S)$ and online word-level predictions $T_{\max}(S)$, and λ_3 is for weighting the two loss terms and is empirically fixed to 0.1.

4 Experiments

4.1 Dataset and Experiment Setup

Datasets. We test the ST on WMT 2014 English-German (En-De) and English-French (En-Fr) datasets. We adopt the En-De pre-processed version by Vaswani et al. (2017) for fair comparison with previous approaches. Ablation study and hyperparameter tuning are conducted on WMT En-De validation (newstest2013) set and tested on the test (newstest2014). In En-Fr, we follow the same hyper-parameters of En-De. All ablation studies and experiments are conducted with type-1 scalable Transformers. BPE Sennrich et al. (2015) with shared dictionary between source and target was adopted for tokenization of both datasets. 32k and 40k joint dictionaries are created for En-De and En-Fr tasks. We measure the translation with case sensitive BLEU. All experiments use beam search 4 and length penalty 0.6. Following Vaswani et al. (2017), we apply compound splitting.

Implementation. Our scalable Transformers contains 6 encoder layers and 6 decoder layers. There are 13 different widths in total, i.e. $\mathcal{M} = \{256, 320, \cdots, 1024\}$. The attention head dimension is fixed to 64. We set different dropout rates within ST according to the features' dimensions. For En-De, we gradually increase the dropout rate $0 \to 0.3$ as the feature dimension $256 \to 1024$. For En-Fr, since it has larger-scale training data, we set dropout rates to 0 for widths in $[256, 576]$ and to 0.1 for widths in $[640, 1024]$. Our optimization follows Ott et al. (2018) and details are can be found in supplementary materials. Before stage-3 training, for the input sequence, we conduct prediction by the widest Transformer with beam search 4 and length penalty 0.6 to generate distillation targets. The predicted sequences are refined to remove the those whose source/target length ratio$>$ 20 or length$>$ 250 words. Our proposed and other compared methods' performances are reported based on the ensemble of the last 10 training epochs, following the experimental setting of Ott et al. (2018).

4.2 Ablation Study on WMT En-De

Independent vs. scalable Transformers. We first independently train separate Transformers, which have unified feature widths of $\{256, 512, 768, 1024\}$ for all layers but have fixed 1024-d word embedding and the additional input-output projections for fairly comparing with our type-1 scalable Transformers' sub-Transformers. Our scalable Transformers shows much better performance than the independently trained counterparts (see "Stage 1+2+3" vs "Indep." in Table 1).

Stage-1 vs. stage-2 vs. stage-3 training. We then show the necessity of the proposed three-stage training scheme. Simple joint training all sub-Transformers in stage-1 (equivalent to slimmable NN Yu et al. (2018) as we discussed in Sec. 3.3) results in even worse performance than independent training ("Stage 1" vs. "Indep." in Table 1), which illustrate the training interference between different sub-Transformers. Stage-2 training shows significant improvements over stage-1 results. Stage-3 training further improves the performance with offline sequence-level self-distillation ("Stage 1+2" and "Stage 1+2+3" in Table 1). We also test discarding stage-1 training ("Stage 2+3") and discarding stage-2 training ("Stage 1+3") from our three-stage training scheme. Both strategies lead to inferior performances compared with our proposed three-stage training scheme.

Feat. Dim.	**256**	**320**	**384**	**448**	**512**	**576**	**640**	**704**	**768**	**832**	**896**	**960**	**1024**
Dropout	0	0	0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
With Input-output Projections													
Indep.	23.32				25.54				26.14				26.74
Indep.+sep. teach	25.49				25.81				26.02				26.75
Scalable+sep. teach	25.53	25.49	25.71	25.78	25.99	26.02	26.05	26.01	26.19	26.21	26.19	26.33	26.59
Stage 1	23.76	23.85	24.27	24.57	24.86	25.25	25.56	25.94	26.23	26.21	26.50	26.58	26.61
Stage 1+2	24.76	24.93	24.93	25.35	25.61	25.58	25.84	26.00	26.34	26.37	26.45	26.53	26.65
Stage 2+3	24.91	24.59	25.11	25.12	25.32	25.36	25.41	25.53	25.57	26.01	25.99	26.19	26.21
Stage 1+3	25.41	25.29	25.85	25.89	26.01	25.99	26.11	26.23	26.31	26.45	26.44	26.41	26.59
Stage 1+2+3	25.59	25.59	26.06	26.05	26.15	26.11	26.06	26.23	26.30	26.49	26.61	26.65	26.71
Params (M)	45	51	59	68	79	91	104	119	135	152	171	191	209
FLOPs (G)	5.2	6.6	8.1	9.7	11.3	12.9	14.6	16.4	18.2	20.1	22.0	23.98	26.02
Without Input-output Projections													
Indep.	23.35				25.56				26.12				26.73
I/O Cropping	25.11	25.07	25.18	25.42	25.71	26.02	25.98	26.07	26.32	26.22	26.31	26.48	26.46
Params (M)	19	28	37	48	61	75	90	106	124	144	163	186	209
FLOPs (G)	5.2	6.6	8.7	9.7	11.3	12.9	14.6	16.4	18.2	20.1	22.0	23.98	26.02

Table 1: Ablation study on En-De **validation** (newtest2013) set with beam search 4. FLOPs are calculated with the assumption that source and target length are 20.

Independent models + separate widest teacher. The above experiments show that parameter-sharing sub-models and the full model can mutually boost their performance via the proposed training scheme. To show the improvement is not from simple knowledge distillation, we train an independent widest Transformer to generate the refined sequence-level predictions. After the training converges, we use the widest transformer as the teacher outputting both online soft and offline beam-searched hard targets to guide the training of 12 independent models with varying widths 256-1024. Results show that our scalable transformer can achieve better performance than independent distillation ("Indep.+sep. teach" vs. "indep.") due to the parameter sharing between models of different width, especially on narrower models. ("Indep.+sep. teach" vs. "Stage 1+2+3" in Table 1).

Scalable Transformer + separate widest teacher. An alternative approach to the above study is to keep the separate widest Transformer as the teacher to teach a proposed scalable Transformer. The performances by this strategy ("scalable+sep. teach" vs. "Stage 1+2+3") are only slightly lower than our proposed self-distilled scalable model. The reason might be that, although the scalable models share parameters and are jointly trained. The separate widest teacher's outputs might be less compatible with those of the narrower models than the scalable model's widest Transformer. In addition, this training strategy requires more training costs. For each forward process, only 46G FLOPs are required for our proposed training scheme, while 72G FLOPs are needed for this separate widest teacher strategy, as the separate teacher model needs to be separately forwarded to obtain the word-level soft targets. Training the separate widest model also requires additional resources.

Advantages of type-2 ST. Our type-2 ST can more flexibly adjust its FLOPs because it can freely choose different feature widths D^i for different layers. Surprisingly, after the type-2 model is trained, we found many of its sub-Transformers can even surpass the widest Transformer. Given the widest Transformers of our trained type-2 model from the last 10 epochs, for each layer width D^i, we randomly choose the feature dimension from $\{896, 960, 1024\}$ and conduct evaluation with $1,000$ random such sub-Transformers on the En-De validation set. We choose the best sub-Transformer on the validation (newtest2013) set and evaluate on the test (newtest2014) set. Such an optimal sub-model achieves 26.9 and 29.7 on the validation and test sets (see Table 6 in supplemental materials), which are even higher than the widest Transformer. We also calculate the mean and standard deviation of the BLEU scores of the top-10 sub-Transformers, which achieve 26.82 ± 0.036 and 29.60 ± 0.061 on validation and test sets. All top-10 sub-Transformers' performances are higher than those of the widest Transformer (26.7 for validation and 29.3 for test), demonstrating the effectiveness of our type-2 scalable Transformers. We will continue to investigate how to improve the efficiency on searching the optimal sub-model. To verify that the performance improvement of type-2 Transformer is not from the 1,000 searched sub-models, we store 1,000 checkpoints in the last 3 epochs when training an independent widest transformer. We average the best 10 models out of 1,000 checkpoints, which achieve 29.4 on En-De test set and is very similar to the baseline's 29.3. However, such a result is still worse than our searched optimal Type-2 sub-model's 29.7.

Training time and memory of scalable Transformer Our proposed scalable Transformer has 209M parameters and is trained for 212 hours on 8 V100 GPUs. If one separately trains 12 Trans-

Feat. Dim.	**256**	**320**	**384**	**448**	**512**	**576**	**640**	**704**	**768**	**832**	**896**	**960**	**1024**
Dropout	0	0	0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
Indep.	23.3				27.5				27.7				29.21
Stage 1	24.0	24.3	24.6	25.0	25.7	26.3	27.0	27.4	27.8	28.3	28.6	28.75	28.74
Stage 1+2	25.1	25.5	25.8	26.3	26.6	27.1	27.6	27.8	28.2	28.4	28.8	28.89	28.99
Stage 2+3	26.7	26.6	27.0	27.1	27.6	27.4	27.9	28.1	28.4	28.7	28.6	28.76	29.18
Stage 1+3	26.9	26.8	27.1	27.3	27.5	27.9	28.2	28.3	28.6	28.7	28.7	28.94	29.01
Stage 1+2+3	27.1	27.3	27.5	27.9	28.3	28.4	28.6	28.7	29.0	28.9	29.2	29.34	29.27

Table 2: Performance of type-1 scalable Transformers on En-De **test** (newtest2014) set.

Model	**EN-De**	**En-Fr**
Transformer Vaswani et al. (2017)	28.4	40.5
Weighted Transformer Ahmed et al. (2017)	28.9	41.0
Relative Transformer Shaw et al. (2018)	29.2	41.5
Scaling NMT Ott et al. (2018)	29.3	43.2
Evolved Transformer So et al. (2019) (NAS on 200 TPUs)	29.8	-
Baseline	29.2	43.1
Type-1 Stage 1 (widest)	28.7	42.8
Type-1 Stage 1+2+3 (widest)	29.3	43.1
Type-2 Stage 1+2+3 (widest)	29.5	43.1
Type-2 Stage 1+2+3 (sub-model)	29.7	43.3

Table 3: Comparison between scalable Transformers with state-of-the-art methods on the **test** sets.

formers with varying width, it would use 1290M parameters and 712 hours for training, both of which are much larger than those of our proposed method.

Other factors. We provide more ablation study in supplemental materials.

4.3 Final Results on WMT EN-De and En-Fr

We test the proposed scalable Transformers on WMT'14 EN-De test (newtest2014) set and En-Fr test set. Performances of type-1 ST on En-De test (newtest2014) are listed in Table 2, which show similar tendency as those on En-De validation set.

We further compare the widest model from our type-1 scalable Transformers and the randomly searched optimal sub-model from our type-2 scalable Transformers with the original Transformer Vaswani et al. (2017), weighted Transformer Ahmed et al. (2017), relative position Transformer Shaw et al. (2018), scaling neural machine translation Ott et al. (2018) and evolved Transformer So et al. (2019) in Table 3. The baseline Transformer is based on Ott et al. (2018). Our reproduction achieves 29.2 and 43.1 on En-De and En-Fr, comparable with the original results. The simple joint training (denoted as "Type-1 Stage-1 (widest)") results in slightly worse performance than the independently trained baseline.

The widest Transformer from type-1 ST achieves 29.3 and 43.1 on En-De and En-Fr, which are comparable with the baseline Transformer, but with 13 sub-models of different FLOPs and parameter sizes. The best randomly searched sub-Transformer of our type-2 ST (denoted as "Type-2 Stage 1+2+3 (sub-model)") on En-De can even reach 29.7, which has fewer parameters but outperforms state-of-the-art standard Transformers with 1024-dimensional features. Evolved Transformer performed architecture search with 200 TPUs to find the optimal translation architecture. It achieved 29.8 BLEU with 218M parameters. In contrast, our type-2 scalable Transformers is much easier to implement with existing libraries, only used 3 days on 8 V100 GPUs for training, and achieved comparable 29.7 BLEU with 221M parameters.

5 Conclusion

In this paper, we propose a novel scalable Transformers to achieve efficient and robust deployment of Transformers of different scales. We choose to make the scalable Transformers change its scale by modifying its layer width. Through carefully designed word-level and sequence-level self-distillation, the proposed scalable Transformers can be trained with an incremental increase of training time. After training, sub-Transformers of different scales can be easily obtained by cropping from the widest Transformer to achieve flexible deployment.

REFERENCES

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*, 2017.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1039–1048, 2017.

Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144, 2011.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

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*, pp. 5998–6008, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5628–5635, 2019a.

Chenglin Yang, Lingxi Xie, Chi Su, and Alan L Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2859–2868, 2019b.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1803–1811, 2019.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

A Optimization Details

ADAM Kingma & Ba (2014) optimizer is adopted with $\beta_1 = 0.9$, $\beta_2 = 0.98, \epsilon = 10^{-8}$. Other optimization details can be found in Table 4. The learning rate of each stage linearly increases for the first $4,000$ iterations from 5×10^{-4} to the maximum learning rate of each stage (denoted as "max lr" in Table 4). After reaching the maximum learning rate, the learning rate decays according to

$$\mathrm{lr} = \frac{\mathrm{lr}_{\max} \times \sqrt{4000}}{\sqrt{\#\mathrm{iteration}}} \tag{7}$$

where $\mathrm{lr}_{\max}$ stands for the maximum, 4000 is the number of iterations for warming up, $\#\mathrm{iteration}$ stands for the current iteration number. The learning rate scheme follows the original Transformer Vaswani et al. (2017). The total training epoch for each stage is denoted as "Epoches" in Table 4. Label smoothing of 0.1 is conducted following Ott et al. (2018). Weight decay is not used for all experiments. We train the proposed scalable Transformer with 8 V100 GPUs, each of which holds 3584 and 5120 tokens for En-De and En-Fr, respectively. We adopt gradient accumulation of 16 to further increase the training batch size. λ_2 or λ_3 weight the contributions of the cross-entropy loss and self-distillation loss and their settings are recorded in Table 4 (denoted as "λ_2/λ_3").

B Additional Ablation Studies and Experiments

Dropout scheme. Dropout has major impact on the final performance for training Transformers. Our scalable Transformer gradually increase the dropout rates from 0.1 to 0.3 as the feature dimensions increase from 256 to 1024. We test two alternative dropout strategies. The first one is to set dropout rates as 0.3 for all feature dimensions in $\{256, \cdots, 1024\}$. Another strategy is to set dropout rate to 0 for all feature dimensions except for dimension 1024, which has a dropout rate of 0.3. The results on En-De validation set is shown in Table 5, which show that our proposed dropout

Training stage	Epoches	max lr	λ_2/λ_3
EN-De stage-1	60	0.007	n/a
En-De stage-2	60	0.007	0.5
En-De stage-3	30	0.004	0.9
En-Fr stage-1	30	0.006	n/a
En-Fr stage-2	30	0.006	0.5
EN-Fr stage-3	15	0.004	0.9

Table 4: The hyperparameters for optimization on En-De and En-Fr datasets.

Feat. Dim.	**256**	**320**	**384**	**448**	**512**	**576**	**640**	**704**	**768**	**832**	**896**	**960**	**1024**
Proposed Dropout Stratety													
Dropout	0	0	0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
Stage 1+2+3	25.59	25.59	26.06	26.05	26.15	26.11	26.06	26.23	26.30	26.49	26.61	26.65	26.71
Uniform Maximal Dropout													
Dropout	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Stage 1+2+3	23.87	23.97	24.28	24.32	24.15	24.32	24.69	25.74	26.23	25.96	26.45	26.32	26.47
Widest Dropout Only													
Dropout	0	0	0	0	0	0	0	0	0	0	0	0	0.3
Stage 1+2+3	24.89	24.91	24.95	25.02	25.17	25.28	25.34	25.56	25.88	26.10	26.22	26.01	26.35

Table 5: Tuning dropout rates on type-1 scalable Transformer when training on En-De **validation** (newtest2013) set with beam search 4.

strategy leads to better performance than those simplified ones on all widths of the type-1 scalable Transformer.

Effects of input-output linear projections. Our scalable Transformers introduces two additional linear projections W_o' and W_e' to ensure that the same word embedding can be used across sub-Transformers. We show that the gain is not from the extra linear projections. We independently train another 4 Transformer to have the same word embedding and feature dimensions of $\{256, 512, 768, 1024\}$ respectively. Those 4 Transformers do not need the additional linear projections. The two "Indep." in Table 1 show that they have almost the same performance as independent Transformers with projections. Another possible way to replace the input-output projections is to crop the top corner of the embedding matrix to obtain narrower word embedding for the sub-Transformers. This strategy ("I/O Cropping" in Table 1) show inferior accuracy compared with the proposed additional input-output linear projections.

Beam search. We also test the impact of using beam search during inference. The results are listed in Tables 6 and 7. Beam search generally improves the translation accuracy, especially on stage-1 or stage-2 results. After stage-3 training, the improvements of "Stage 1+2+3" with beam search over "Stage 1+2+3" without beam search become marginal on the two datasets ($+0.17$ for En-De test set and $+0.04$ for En-Fr test set), which indicate that our three-stage training strategy can eliminate the need for beam search at inference to certain extent.

Width adaptive or depth adaptive? In our proposed scalable Transformer, we decide to tune layer width of the Transformer to achieve adjustable FLOPs and parameter sizes. To show tuning layer width is better than tuning network depth, we implement two independent Transformers with approximately the same number of parameters. One contains 6 encoder and 6 decoder layers with feature dimension 512. The other one contains 3 encoder and 3 decoder layers with feature dimension of 768. Their translation accuracy on En-De test (newtest2014) set is reported in Table 8, which show that decreasing the depth of the Transformer significantly impacts the translation accuracy $29.2 \rightarrow 26.3$, while decreasing the layer width has smaller performance drop $29.2 \rightarrow 27.4$. Therefore, it is better to tune the width of the Transformer layers than the overall depth of the Transformer to better maintain the translation accuracy.

Top-10 randomly searched type-2 sub-Transformers. We conduct random search on type-2 scalable Transformer's sub-models to find the optimal performance as stated in Sec. 4.2 of the main paper. Table 9 shows the top-10 sub-models' layer widths and their performances on En-De validation and test sets. The results clearly show that many sub-Transformers of our type-2 ST can even surpass the widest Transformer, which worth further studying how to effectively conduct sub-model search in the future.

Feat. Dim.	**256**	**320**	**384**	**448**	**512**	**576**	**640**	**704**	**768**	**832**	**896**	**960**	**1024**
Dropout	0	0	0	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
Beam Search 4													
Stage 1	24.0	24.3	24.6	25.0	25.7	26.3	27.0	27.4	27.8	28.3	28.6	28.75	28.74
Stage 1+2	25.1	25.5	25.8	26.3	26.6	27.1	27.6	27.8	28.2	28.4	28.8	28.89	28.99
Stage 1+2+3	27.1	27.3	27.5	27.9	28.3	28.4	28.6	28.7	29.0	28.9	29.2	29.34	29.27
Beam Search 1													
Stage 1	23.3	23.4	23.7	24.3	24.8	25.5	26.1	26.9	27.2	27.2	27.3	27.58	27.90
Stage 1+2	23.7	24.2	24.6	25.1	25.5	26.0	26.7	27.0	27.2	27.4	27.6	28.02	28.19
Stage 1+2+3	26.7	26.8	27.1	27.4	27.8	28.0	28.1	28.3	28.6	28.7	28.8	28.89	29.10
Params (M)	46	52	60	69	79	91	104	119	135	152	171	191	209
FLOPs (G)	5.2	6.6	8.1	9.7	11.3	12.9	14.6	16.4	18.2	20.1	22.0	23.98	26.02

Table 6: Performance of type-1 scalable Transformer on En-De **test** (newtest2014) set with beam search 4 and 1.

Feat. Dim.	**256**	**320**	**384**	**448**	**512**	**576**	**640**	**704**	**768**	**832**	**896**	**960**	**1024**
Dropout	0	0	0	0	0	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Beam Search 4													
Indep.													43.11
Stage 1	39.1	38.9	39.2	39.6	39.9	40.5	40.9	41.4	41.8	42.0	42.3	42.41	42.84
Stage 1+2	39.4	39.5	40.0	40.2	40.6	41.0	41.3	41.9	42.2	42.4	42.6	42.65	42.95
Stage 1+2+3	40.9	41.0	41.3	41.5	41.7	41.7	42.0	42.2	42.4	42.5	42.9	43.03	43.07
Beam Search 1													
Stage 1	38.0	38.0	38.5	38.9	39.3	39.7	40.5	40.8	41.3	41.5	41.7	41.92	42.11
Stage 1+2	38.6	38.6	39.1	39.4	39.8	40.2	40.7	41.1	41.4	41.7	41.9	42.10	42.40
Stage 1+2+3	40.4	40.6	40.8	41.0	41.4	41.5	41.9	42.1	42.3	42.4	42.7	42.84	43.03
Params (M)	57	63	71	80	91	103	116	130	146	164	182	202	224
FLOPs (G)	5.2	6.6	8.1	9.7	11.3	12.9	14.6	16.4	18.2	20.1	22.0	23.98	26.02

Table 7: Performance of type-1 scalable Transformer on En-Fr **test** set with beam search 4 and 1.

Encoder-decoder Layers	Layer Width	EN-De Test	Parameter (M)
6-6	512	27.5	61
3-3	768	26.3	73

Table 8: Performance of (top) deep but thin Transformer versus (bottom) shallow but wide Transformer with similar parameter sizes on En-De test set. The former network has smaller performance drop.

Model	**newtest 2013**	**newtest 2014**	FLOPs(G)
Widest: [1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024]	26.73	29.44	26.02
[1024, 1024, 1024, 1024, 896, 1024, 1024, 960, 1024, 1024, 1024, 896]	26.87	29.66	25.30
[1024, 1024, 1024, 1024, 960, 896, 1024, 896, 960, 1024, 960, 960]	26.85	29.64	24.86
[1024, 1024, 1024, 1024, 960, 896, 1024, 896, 960, 1024, 1024, 896]	26.85	29.62	24.86
[1024, 1024, 1024, 1024, 960, 1024, 1024, 896, 1024, 1024, 960, 896]	26.83	29.62	25.06
[1024, 1024, 1024, 1024, 960, 896, 1024, 896, 960, 1024, 960, 896]	26.83	29.60	24.69
[1024, 1024, 1024, 1024, 1024, 1024, 1024, 896, 1024, 1024, 1024, 960]	26.82	29.58	25.15
[1024, 1024, 1024, 1024, 1024, 1024, 1024, 960, 1024, 1024, 960, 960]	26.80	29.54	25.50
[1024, 1024, 1024, 1024, 960, 1024, 1024, 1024, 1024, 1024, 1024, 896]	26.77	29.69	25.57
[1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 960, 1024, 896]	26.77	29.47	25.50
[1024, 1024, 1024, 1024, 1024, 1024, 1024, 896, 1024, 1024, 1024, 960]	26.76	29.64	25.50

Table 9: BLEU scores of the top-10 randomly searched sub-Transformers from our type-2 scalable Transformer on En-De validation (newtest2013) and test (newtest2014) sets.