Multiscale Collaborative Deep Models for Neural Machine Translation

Xiangpeng Wei[1,2]*, **Heng Yu**[3]†, **Yue Hu**[1,2]†, **Yue Zhang**[4,5], **Rongxiang Weng**[3], **Weihua Luo**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Machine Intelligence Technology Lab, Alibaba Group, Hangzhou, China
[4]School of Engineering, Westlake University, Hangzhou, China
[5]Institute of Advanced Technology, Westlake Institute for Advanced Study, Hangzhou, China
{weixiangpeng,huyue}@iie.ac.cn yue.zhang@wias.org.cn
{yuheng.yh,wengrx,weihua.luowh}@alibaba-inc.com

Abstract

Recent evidence reveals that Neural Machine Translation (NMT) models with deeper neural networks can be more effective but are difficult to train. In this paper, we present a **M**ulti**S**cale **C**ollaborative (MSC) framework to ease the training of NMT models that are substantially deeper than those used previously. We explicitly boost the gradient backpropagation from top to bottom levels by introducing a *block-scale collaboration* mechanism into deep NMT models. Then, instead of forcing the whole encoder stack directly learns a desired representation, we let each encoder block learns a fine-grained representation and enhance it by encoding spatial dependencies using a *context-scale collaboration*. We provide empirical evidence showing that the MSC nets are easy to optimize and can obtain improvements of translation quality from considerably increased depth. On IWSLT translation tasks with three translation directions, our extremely deep models (with 72-layer encoders) surpass strong baselines by +2.2∼+3.1 BLEU points. In addition, our deep MSC achieves a BLEU score of 30.56 on WMT14 English→German task that significantly outperforms state-of-the-art deep NMT models.

1 Introduction

Neural machine translation (NMT) directly models the entire translation process using a large neural network and has gained rapid progress in recent years (Sutskever et al., 2014; Sennrich et al., 2016). The structure of NMT models has evolved quickly, such as RNN-based (Wu et al., 2016), CNN-based (Gehring et al., 2017) and attention-based (Vaswani et al., 2017) systems. All of these models follow the encoder-decoder framework with attention (Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) paradigm.

Deep neural networks have revolutionized the state-of-the-art in various communities, from computer vision to natural language processing. However, training deep neural networks has been always a challenging problem. To encourage gradient flow and error propagation, researchers in the field of computer vision have proposed various approaches, such as residual connections (He et al., 2016), densely connected networks (Huang et al., 2017) and deep layer aggregation (Yu et al., 2018). In natural language processing, constructing deep architectures has shown effectiveness in language modeling, question answering, text classification and natural language inference (Peters et al., 2018; Radford et al., 2018; Al-Rfou et al., 2019; Devlin et al., 2019). However, among existing NMT models, most of them are generally equipped with 4-8 encoder and decoder layers (Wu et al., 2016; Vaswani et al., 2017). Deep neural network has been explored relatively little in NMT.

Recent evidence (Bapna et al., 2018; Wang et al., 2019a) shows that model depth is indeed of importance to NMT, but a *degradation* problem has been exposed: by simply stacking more layers, the translation quality gets saturated and then degrades rapidly. To address this problem, Bapna et al. (2018) proposed a transparent attention mechanism to ease the optimization of the models with deeper encoders. Wang et al. (2019a) continued this line of research but construct a much deeper encoder for Transformer by adopting the *pre-norm* method that establishes a direct way to propagate error gradients from the top layer to bottom levels, and passing the combination of previous layers to the next. While notable gains have been reported over shallow models, the improvements of translation quality are limited when the model depth is beyond 20. In addition, degeneration of translation quality is still observed when the depth is beyond 30. As a result, two questions arise naturally: *How*

* Work done at Alibaba Group.
† Corresponding Author.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 414–426
July 5 - 10, 2020.

to break the limitation of depth in NMT models? and *How to fully utilize the deeper structure to further improve the translation quality?*

In this paper, we address the degradation problem by proposing a **M**ulti**S**cale **C**ollaborative (MSC) framework for constructing NMT models with very deep encoders.[1] In particular, the encoder and decoder of our model have the same number of *blocks*, each consisting of one or several stacked layers. Instead of relying on the whole encoder stack directly learns a desired representation, we let each encoder block learn a fine-grained representation and enhance it by encoding spatial dependences using a bottom-up network. For coordination, we attend each block of the decoder to both the corresponding representation of the encoder and the contextual representation with spatial dependences. This not only shortens the path of error propagation, but also helps to prevent the lower level information from being forgotten or diluted.

We conduct extensive experiments on WMT and IWSLT translation tasks, covering three translation directions with varying data conditions. On IWSLT translation tasks, we show that:

- While models with traditional stacking architecture exhibit worse performance on both training and validation data when depth increases, our framework is easy to optimize.
- The deep MSC nets (with 72-layer encoders) bring great improvements on translation quality from increased depth, producing results that substantially better than existing systems.

On the WMT14 English→German task, we obtain improved results by deep MSC networks with a depth of 48 layers, outperforming strong baselines by +2.5 BLEU points, and also defeat state-of-the-art deep NMT models (Wu et al., 2019; Zhang et al., 2019a) with identical or less parameters.[2]

2 Background

Given a bilingual sentence pair $(\mathbf{x}, \mathbf{y})$, an NMT model learns a set of parameters $\boldsymbol{\Theta}$ by maximizing the log-likelihood $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\Theta})$, which is typically decomposed into the product of the conditional probability of each target word: $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\Theta}) = \prod_{t=1}^{T_{\mathrm{y}}} P(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x}; \boldsymbol{\Theta})$, where T_{y} is the length of sentence $\mathbf{y}$, $\mathbf{y}_{<t}$ is the partial translation that contains the target tokens before position t. An encoder-decoder framework is commonly adopted to model the conditional probability $P(\mathbf{y}|\mathbf{x}; \boldsymbol{\Theta})$, in which the encoder and decoder can be implemented as RNN (Wu et al., 2016), CNN (Gehring et al., 2017), or Self-Attention network (Vaswani et al., 2017). Despite variant types of NMT architectures, multiple-layer encoder and decoder are generally employed to perform the translation task, and residual connections (He et al., 2016) are naturally introduced among layers, as $\mathrm{H}^l = \textsc{Layer}(\mathrm{H}^{l-1}; \boldsymbol{\Theta}^l) + \mathrm{H}^{l-1}$, where H^l is the output of the l-th layer, $\textsc{Layer}(\cdot)$ is the layer function and $\boldsymbol{\Theta}^l$ be the parameters.

We take the state-of-the-art Transformer as our baseline model. Specifically, the encoder consists of a stack of L identical layers, each of which comprises two subcomponents: a self-attention mechanism followed by a feed-forward network. Layer normalization (Ba et al., 2016) is applied to the input of each subcomponent (i.e., *pre-norm*) and a residual skip connection (He et al., 2016) adds each subcomponent's input to its output. Formally,

$$\begin{aligned} \mathrm{O}_e^l &= \textsc{Attn}(\mathrm{Q}_e^l, \mathrm{K}_e^l, \mathrm{V}_e^l; \boldsymbol{\Theta}_e^l) + \mathrm{H}_e^{l-1}, \\ \mathrm{H}_e^l &= \textsc{Fnn}(\textsc{Ln}(\mathrm{O}_e^l); \boldsymbol{\Theta}_e^l) + \mathrm{O}_e^l, \end{aligned} \tag{1}$$

where $\textsc{Ln}(\cdot)$, $\textsc{Attn}(\cdot)$ and $\textsc{Ffn}(\cdot)$ are layer normalization, attention mechanism, and feed-forward networks with ReLU activation in between, respectively. $\{\mathrm{Q}_e^l, \mathrm{K}_e^l, \mathrm{V}_e^l\}$ are query, key and value vectors that are transformed from the normalized $(l-1)$-th encoder layer $\textsc{Ln}(\mathrm{H}_e^{l-1})$.

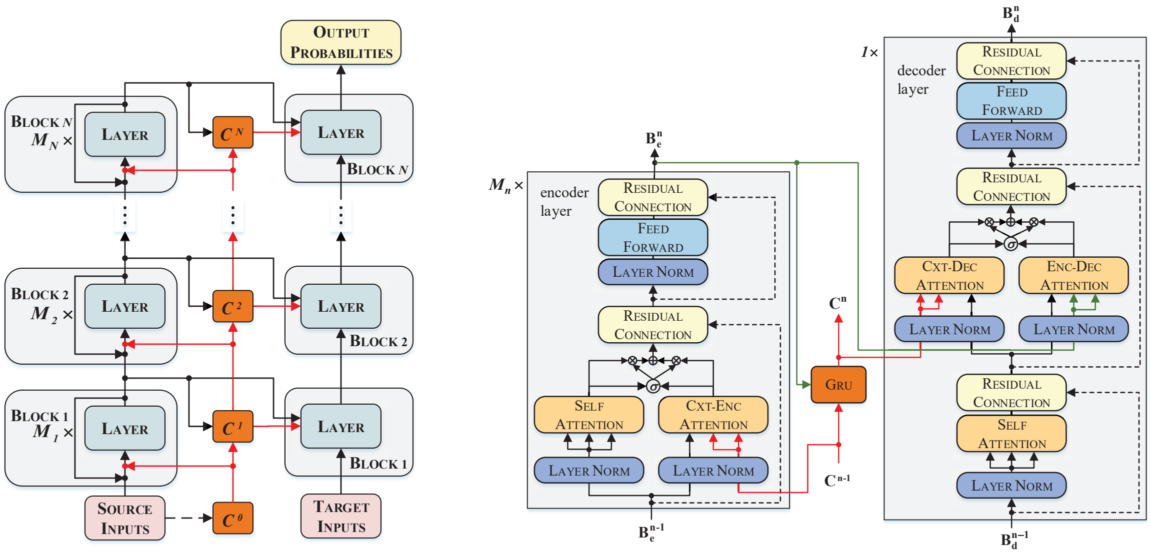
The decoder is similar in structure to the encoder except that it includes a standard attention mechanism after each self-attention network, which attends to the output of the encoder stack H_e^L:

$$\begin{aligned} \mathrm{O}_d^l &= \textsc{Attn}(\mathrm{Q}_d^l, \mathrm{K}_d^l, \mathrm{V}_d^l; \boldsymbol{\Theta}_d^l) + \mathrm{H}_d^{l-1}, \\ \mathrm{S}_d^l &= \textsc{Attn}(\textsc{Ln}(\mathrm{O}_d^l), \mathrm{K}_e^L, \mathrm{V}_e^L; \boldsymbol{\Theta}_d^l) + \mathrm{O}_d^l, \\ \mathrm{H}_d^l &= \textsc{Fnn}(\textsc{Ln}(\mathrm{S}_d^l); \boldsymbol{\Theta}_d^l) + \mathrm{C}_d^l, \end{aligned} \tag{2}$$

where $\{\mathrm{Q}_d^l, \mathrm{K}_d^l, \mathrm{V}_d^l\}$ are transformed from the normalized $(l-1)$-th decoder layer $\textsc{Ln}(\mathrm{H}_d^{l-1})$ and $\{\mathrm{K}_e^L, \mathrm{V}_e^L\}$ are transformed from the top layer of the encoder. The top layer of the decoder H_d^L is used to generate the final output sequence. In the

[1]In our scenario, we mainly study the depth of encoders. The reason is similar in (Wang et al., 2019a): 1) encoders have a greater impact on performance than decoders; 2) increasing the depth of the decoder will significantly increase the complexity of inference.

[2]MSC not only performs well on NMT but also is generalizable to other sequence-to-sequence generation tasks, such as abstractive summarization that is introduced in Appendix A.



(a) Overall framework. (b) Detailed illustration of the n-th block.

Figure 1: Illustration of Multiscale Collaborative Deep NMT Model. N is the number of encoder and decoder blocks. The n-th block of the encoder consists of M_n layers, while each decoder block only contains one layer.

following sections, we simplify the equations as

$$\begin{aligned} \mathrm{H}_e^l &= \mathcal{F}(\mathrm{H}_e^{l-1}; \mathbf{\Theta}_e^l) + \mathrm{H}_e^{l-1}, \\ \mathrm{H}_d^l &= \mathcal{G}(\mathrm{H}_d^{l-1}, \mathrm{H}_e^L; \mathbf{\Theta}_d^l) + \mathrm{H}_d^{l-1}, \end{aligned} \quad (3)$$

for the encoder and decoder, respectively.

As discussed by Wang et al. (2019a), applying layer normalization to the input of each subcomponent is the key to learning deep encoders, as it establishes a direct way to pass gradient from the top-most layer to bottom layers:

$$\frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^l} = \frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^L} \times \Big(1 + \sum_{j=l}^{L-1} \frac{\partial \mathcal{F}(\mathrm{H}_e^j; \mathbf{\Theta}_e^{j+1})}{\partial \mathrm{H}_e^l}\Big), \quad (4)$$

where $\mathcal{L}$ is the cross entropy loss. However, as pointed out by Wang et al. (2019a) that it can be difficult to deepen the encoder for better translation quality. We argue that as the right-most term in Eq. (4) approaches 0 for the lower levels of the encoder, the parameters of which cannot be sufficiently trained using the error gradient $\frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^L}$ only. To solve this problem, we propose a novel approach to shorten the path of error propagation from $\mathcal{L}$ to bottom layers of the encoder.

3 Multiscale Collaborative Deep Model

In this section, we introduce the details of the proposed approach, a **M**ulti**S**cale **C**ollaborative (MSC) framework for constructing extremely deep NMT models. The framework of our method consists of two main components shown in Figure 1(a). First, a *block-scale collaboration* mechanism establishes shortcut connections from the lower levels of the encoder to the decoder (as described in 3.1), which is the key to training very deep NMT models. We give explanation by seeing the gradient propagation process. Second, we further enhance source representations with spatial dependencies by *contextual collaboration*, which is discussed in Section 3.2.

3.1 Block-Scale Collaboration

An intuitive extension of naive stacking of layers is to group few stacked layers into a *block*. We suppose that the encoder and decoder of our model have the same number of blocks (i.e., N). Each block of the encoder has M_n ($n \in \{1, 2, ..., N\}$) identical layers, while each decoder block contains one layer. Thus, we can adjust the value of each M_n flexibly to increase the depth of the encoder. Formally, for the n-th block of the encoder:

$$\mathrm{B}_e^n = \mathrm{BLOCK}_e(\mathrm{B}_e^{n-1}), \quad (5)$$

where $\mathrm{BLOCK}_e(\cdot)$ is the block function, in which the layer function $\mathcal{F}(\cdot)$ is iterated M_n times, i.e.

$$\begin{aligned} \mathrm{B}_e^n &= \mathrm{H}_e^{n,M_n}, \\ \mathrm{H}_e^{n,l} &= \mathcal{F}(\mathrm{H}_e^{n,l-1}; \mathbf{\Theta}_e^{n,l}) + \mathrm{H}_e^{n,l-1}, \\ \mathrm{H}_e^{n,0} &= \mathrm{B}_e^{n-1}, \end{aligned} \quad (6)$$

where $l \in \{1, 2, ..., M_n\}$, $\mathrm{H}_e^{n,l}$ and $\mathbf{\Theta}_e^{n,l}$ are the representation and parameters of the l-th layer in the n-th block, respectively. The decoder works in a similar way but the layer function $\mathcal{G}(\cdot)$ is iterated only once in each block,

$$\begin{aligned}\mathrm{B}_d^n &= \textsc{Block}_d(\mathrm{B}_d^{n-1}, \mathrm{B}_e^n) \\ &= \mathcal{G}(\mathrm{B}_d^{n-1}, \mathrm{B}_e^n; \mathbf{\Theta}_d^n) + \mathrm{B}_d^{n-1}.\end{aligned} \tag{7}$$

Each block of the decoder attends to the corresponding encoder block. He et al. (2018) proposed a model that learns the hidden representations in two corresponding encoder and decoder layers as the same semantic level through layer-wise coordination and parameter sharing. Inspired by this, we focus on efficiently training extremely deep NMT models through directly attending decoder to the lower-level layers of the encoder, rather than only to the final representation of the encoder stack.

The proposed block-scale collaboration (BSC) mechanism can effectively boost gradient propagation from prediction loss to lower level encoder layers. For explaining this, see again Eq. (4), which explains the error back-propagation of pre-norm Transformer. Formally, we let $\mathcal{L}$ be the prediction loss. The differential of $\mathcal{L}$ with respect to the l-th layer in the n-th block $\mathrm{H}_e^{n,l}$ can be calculated as:[3]

$$\begin{aligned}&\frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^{n,l}} \\ &= \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^N} \times \frac{\partial \mathrm{B}_e^N}{\partial \mathrm{H}_e^{n,l}} + \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^n} \times \frac{\partial \mathrm{B}_e^n}{\partial \mathrm{H}_e^{n,l}} \\ &= \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^N} \times (1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathrm{H}_e^{n,k}}{\partial \mathrm{H}_e^{n,l}} + \sum_{i=n+1}^{N} \sum_{j=1}^{M_i} \frac{\partial \mathrm{H}_e^{i,j}}{\partial \mathrm{H}_e^{n,l}})}_{(a)} \\ &+ \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^n} \times (1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathrm{H}_e^{n,k}}{\partial \mathrm{H}_e^{n,l}})}_{(b)},\end{aligned} \tag{8}$$

where term (a) is equal to Eq. (4). In addition to the straightforward path $\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^N}$ for parameter update from the top-most layer to lower ones, Eq. (8) also provides a complementary way to directly pass error gradient $\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^n}$ from top to bottom in the current block. Another benefit is that BSC shortens the length of gradient pass chain (i.e., $M_n \ll L$).

3.2 Contextual Collaboration

To model long-term spatial dependencies and reuse global representations, we define a GRU (Cho et al., 2014) cell $\mathcal{Q}(\mathbf{c}, \bar{\mathbf{x}})$, which maps a hidden state $\mathbf{c}$ and an additional input $\bar{\mathbf{x}}$ into a new hidden state:

$$\begin{aligned}\mathrm{C}^n &= \mathcal{Q}(\mathrm{C}^{n-1}, \mathrm{B}_e^n), n \in [1, N] \\ \mathrm{C}^0 &= \mathcal{E}_e,\end{aligned} \tag{9}$$

where $\mathcal{E}_e$ is the embedding matrix of the source input $\mathbf{x}$. The new state C^n can be fused with each layer of the subsequent blocks in both the encoder and the decoder. Formally, B_e^n in Eq.(5) can be re-calculated in the following way:

$$\begin{aligned}\mathrm{B}_e^n &= \mathrm{H}_e^{n,M_n}, \\ \mathrm{H}_e^{n,l} &= \mathcal{F}(\mathrm{H}_e^{n,l-1}, \mathrm{C}^{n-1}; \mathbf{\Theta}_e^{n,l}) + \mathrm{H}_e^{n,l-1}, \\ \mathrm{H}_e^{n,0} &= \mathrm{B}_e^{n-1}.\end{aligned} \tag{10}$$

Similarly, for decoder, we have

$$\begin{aligned}\mathrm{B}_d^n &= \textsc{Block}_d(\mathrm{B}_d^{n-1}, \mathrm{B}_e^n) \\ &= \mathcal{G}(\mathrm{B}_d^{n-1}, \mathrm{B}_e^n, \mathrm{C}^n; \mathbf{\Theta}_d^n) + \mathrm{B}_d^{n-1}.\end{aligned} \tag{11}$$

The above design is inspired by multiscale RNNs (MRNN) (Schmidhuber, 1992; El Hihi and Bengio, 1996; Koutnik et al., 2014; Chung et al., 2016), which encode temporal dependencies with different timescales. Unlike MRNN, our MSC enables each decoder block to attend to multi-granular source information with different space-scales, which helps to prevent the lower level information from being forgotten or diluted.

Feature Fusion: We fuse the contextual representation with each layer of the encoder and decoder through attention. A detailed illustration of our algorithm is shown in Figure 1(b). In particular, the l-th layer of the n-th encoder block $\mathcal{F}(\cdot; \mathbf{\Theta}_e^{n,l})$, $l \in [1, M_n]$ and $n \in [1, N]$,

$$\begin{aligned}&\mathrm{O}_e^{n,l} \\ &= g_e \odot \textsc{Attn}_h(\mathrm{H}_e^{n,l-1}, \mathrm{H}_e^{n,l-1}, \mathrm{H}_e^{n,l-1}; \mathbf{\Theta}_e^{n,l}) \\ &+ (1 - g_e) \odot \textsc{Attn}_c(\mathrm{H}_e^{n,l-1}, \mathrm{C}^{n-1}, \mathrm{C}^{n-1}; \mathbf{\Theta}_e^{n,l}) \\ &+ \mathrm{H}_e^{n,l-1}, \\ &g_e = \sigma(W_1 \textsc{Attn}_h(\cdot) + W_2 \textsc{Attn}_c(\cdot) + b),\end{aligned} \tag{12}$$

where g_e is a gate unit, $\textsc{Attn}_h(\cdot)$ and $\textsc{Attn}_c(\cdot)$ are attention models (see Eq. (1)) with different parameters. $\mathrm{O}_e^{n,l}$ is further processed by $\textsc{Ffn}(\cdot)$ to output the representation $\mathrm{H}_e^{n,l}$. Symmetrically, in the decoder, S_d^n in Eq. (2) can be calculated as

$$\begin{aligned}\mathrm{S}_d^n &= g_d \odot \textsc{Attn}_h(\mathrm{O}_d^n, \mathrm{B}_e^n, \mathrm{B}_e^n; \mathbf{\Theta}_d^n) \\ &+ (1 - g_d) \odot \textsc{Attn}_c(\mathrm{O}_d^n, \mathrm{C}^n, \mathrm{C}^n; \mathbf{\Theta}_d^n) \\ &+ \mathrm{O}_d^l\end{aligned} \tag{13}$$

[3]For a detailed derivation, we refer the reader to Appendix B.

where O_d^n is the output of the self-attention sublayer defined in Eq. (2). g_d is another gate unit.

4 Experiments

We first evaluate the proposed method on IWSLT14 English↔German (En↔De) and IWSLT17 English→French (En→Fr) benchmarks. To make the results more convincing, we also experiment on a larger WMT14 English→German (En→De) dataset.

4.1 Settings

Dataset. The dataset for IWSLT14 En↔De are as in Ranzato et al. (2016), with $160k$ sentence pairs for training and 7584 sentence pairs for validation. The concatenated validation sets are used as the test set (dev2010, dev2012, tst2010, tst2011, tst2012). For En→Fr, there are $236k$ sentence pairs for training and 10263 for validation. The concatenated validation sets are used as the test set (dev2010, tst2010, tst2011, tst2012, tst2013, tst2014, tst2015). For all IWSLT translation tasks, we use a joint source and target vocabulary with $10k$ byte-pair-encoding (BPE) types (Sennrich et al., 2016). For the WMT14 En→De task, the training corpus is identical to previous work (Vaswani et al., 2017; Wang et al., 2019a), which consists of about 4.5 million sentence pairs. All the data are tokenized using the script `tokenizer.pl` of Moses (Koehn et al., 2007) and segmented into subword symbols using jointly BPE with $32k$ merge operations. The shared source-target vocabulary contains about $37k$ BPE tokens. We use newstest2013 as the development set and newstest2014 as the test set. Following previous work, we evaluate IWSLT tasks with tokenized case-insensitive BLEU and report tokenized case-sensitive BLEU (Papineni et al., 2002) for WMT14 En→De.

Model Settings. For IWSLT, the model configuration is `transformer_iwslt`, representing a `small` model with embedding size 256 and FFN layer dimension 512. We train all models using the Adam optimizer ($\beta_1/\beta_2 = 0.9/0.98$) with adaptive learning rate schedule (warm-up step with 4K for shallow models, 8K for deep models) as in (Vaswani et al., 2017) and label smoothing of 0.1. Sentence pairs containing 16K∼32K tokens are grouped into one batch. Unless otherwise stated, we train `small` models with 15K maximum steps,

Depth	36-layer	54-layer	72-layer
dec. (N)	6	6	6
enc. ($N \times M$)	6×6	6×9	6×12

Table 1: Deep architectures of MSC on IWSLT tasks. We simply set $M_1 = \cdots = M_N = M$.

and decode sentences using beam search with a beam size of 5 and length penalty of 1.0.

For WMT14 En→De, the model configuration is `transformer_base/big`, with a embedding size of $512/1024$ and a FFN layer dimension of $2048/4096$. Experiments on WMT are conducted on 8 P100 GPUs. Following Ott et al. (2018), we accumulate the gradient 8 iterations and then update to simulate a 64-GPU environment with a batch-size of 65K tokens per step. The Adam optimizer ($\beta_1/\beta_2 = 0.9/0.98$ for `base`, $\beta_1/\beta_2 = 0.9/0.998$) for `big`) and the warm-up strategy (8K steps for `base`, 16K steps for `big`) are also adopted. We use relatively larger batch size and dropout rate for deeper and bigger models for better convergence. The `transformer_base/big` is updated for 100K/300K steps. For evaluation, we average the last 5/20 checkpoints for `base/big`, each of which is saved at the end of an epoch. Beam search is adopted with a width of 4 and a length penalty of 0.6. We use `multi-bleu.perl` to evaluate both IWSLT and WMT tasks for fair comparison with previous work.

4.2 Results

We first evaluate 36-layer, 54-layer and 72-layer MSC nets on IWSLT tasks. Table 1 summarizes the architecture. As shown in Table 2, applying MSC to the vanilla Transformer with 6 layers slightly increases translation quality by +0.26∼+0.37 BLEU (①→②). When the depth is increasing to 36, we use relatively larger dropout rate of 0.3 and achieve substantially improvements (+1.4∼+1.8 BLEU) over its shallow counterparts (③ v.s. ②). After that, we continue deepening the encoders in order, however, our extremely deep models (72 layers, ⑤) suffer from *overfitting* issue on the small IWSLT corpora, which cannot be solved by simply enlarging the dropout rate. We seek to solve this issue by applying L2 regularization to the weights of encoders with greatly increased depth. Results show that this works for deeper encoders (⑥).

We also report the inference speed in Table 2 (the last column). As expected, the speed decreases

#	Model	Param.	En→De	De→En	En→Fr	ΔTrain/ΔDec
1	small, 6 layers, $dp_a = dp_r = 0.1$	10.5M	27.23	32.73	41.19	17/1800
2	w/ MSC	15.6M	27.49	33.10	41.53	17/1736
3	MSC, 36 layers, $dp_a = dp_r = 0.3$	43.3M	29.04	34.86	42.90	23/1498
4	w/ 54 layers	60.0M	29.32	35.16	43.62	27/1412
5	w/ 72 layers	76.6M	-	-	-	-
6	w/ 72 layers, $\lambda_{l_2} = 10^{-5}$	76.6M	**29.67**	**35.81**	**44.15**	30/1340

Table 2: BLEU scores [%] of IWSLT translation tasks. **ΔTrain/ΔDec**: training time (hours)/decoding time (tokens per second) with a batch size of 32 and a beam size of 5. Dropout is applied to the residual connection (dp_r) and attention weights (dp_a). We apply L2 regularization to the weights of deeper encoders with $\lambda_{l_2} = 10^{-5}$, which is only applied to the IWSLT tasks as the corpora are smaller and thus more regularization is required.

Model (small, 36 layers)	**BLEU**
Bapna et al. (2018)	28.09
Wang et al. (2019a)	28.63
MSC	29.04
Model (small, 72 layers)	
Bapna et al. (2018)	failed
Wang et al. (2019a)	28.34
MSC	29.67

Table 3: Comparison with existing methods on IWSLT14 En→De translation. For a fair comparison, we implemented all methods on the same Transformer backbone as well as model settings.

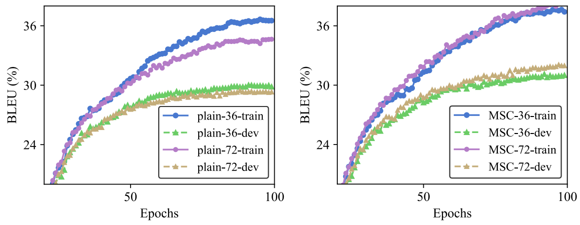
Model	**Param.**	**BLEU**
Vaswani et al. (2017)	213M	28.4
Bapna et al. (2018)	137M	28.0
Dou et al. (2018)	356M	29.2
He et al. (2018)	‡210M	29.0
Wang et al. (2019a)	137M	29.3
Zhang et al. (2019a)	560M	29.62
Wu et al. (2019)	‡268M	**29.9**
TRANSFORMER (base)	63M	27.44
MSC, 6 layers (base)	73M	27.68
MSC, 36 layers (base)	215M	29.71
MSC, 48 layers (base)	272M	**30.19**
TRANSFORMER (big)	211M	28.86
MSC, 6 layers (big)	286M	29.17
MSC, 18 layers (big)	512M	**30.56**

Table 4: BLEU scores [%] on WMT14 En→De translation. ‡ denotes an estimate value.

with the depth of MSC increasing, which is consistent with observation of Wang et al. (2019a). Compared to the baseline, MSC (72 layers) reduces decoding speed by 26%. We leave further investigation on this issue to future work.

For fair comparisons, we implement existing methods (Bapna et al., 2018; Wang et al., 2019a) on the same vanilla Transformer backbone. We separately list the results of 36-layer and 72-layer encoders on the IWSLT14 En→De task in Table 3. The method of Bapna et al. (2018) fail to train a very deep architecture while the method of Wang et al. (2019a) is exposed a degradation phenomenon (28.63→28.34). In contrast, MSC in both 36-layer and 72-layer cases outperform these methods. This suggests that our extremely deep models can easily bring improvements on translation quality from greatly increased depth, producing results substantially better than existing systems.

Table 4 lists the results on the WMT14 En→De translation task and the comparison with the current state-of-the-art systems. The architectures ($N \times M$) of the 18-layer, 36-layer and 48-layer encoders are set as 6×3, 6×6 and 6×8 respectively. We can see that incorporating our MSC into the shallow base/big contributes to +0.24/+0.31 BLEU (27.44→27.68/28.86→29.17) improvements under the same depth. When the depth grows, MSC demonstrates promising improvements of +1.39∼+2.51 BLEU points over its shallow counterparts. It is worth noting that deep MSC with the base setting significantly outperforms the shallow one with the big setting (29.17→30.19), though both of them have around the same number of parameters. Compared to existing models, our MSC outperforms the transparent model (Bapna et al., 2018) (+2.2 BLEU) and the DLCL model (+0.9 BLEU) (Wang et al., 2019a), two recent approaches for deep encoding. Compared to both the depth scaled model (Zhang et al., 2019a) and the current



(a) Plain Network (b) MSC Network

Figure 2: Illustration of the degradation problem on IWSLT14 En→De task. We randomly select 3K sentence pairs from our training data for evaluation. For a fair comparison, we implemented all models on the same Transformer backbone as well as model settings.

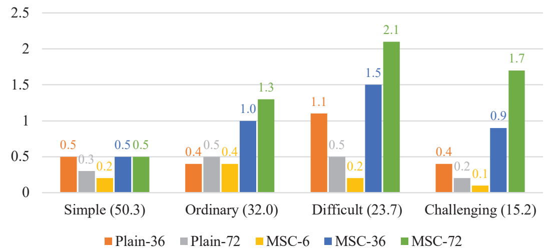


Figure 3: Comparison between plain nets and MSC nets on fine-grained test sets with increasing translation difficulty from "Simple" to "Challenging". Improvements (BLEU [%]) of translation quality over the 6-layer plain net. Higher is better. The results of this baseline are enclosed in the parentheses.

SOTA (Wu et al., 2019), our MSC achieves better performance with identical or less parameters.

4.3 Analysis

Analysis of Degradation. We examine 36-layer and 72-layer *plain* and MSC nets, respectively. For plain networks, we simply stack dozens of layers. As we can see from Figure 2(a), the plain nets suffer from the degradation problem, which is not caused by overfitting, as they exhibit lower training BLEU. In contrast, the 72-layer MSC exhibits higher training BLEU than the 36-layer counterpart and is generalizable to the validation data. This indicates that our MSC can be more easily optimized with greatly increased depth.

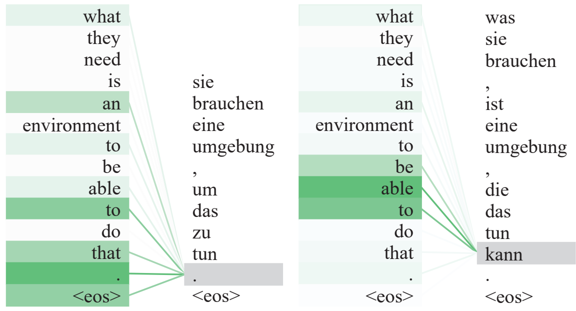
Analysis of Handling Complicated Semantics. Although our MSC can enjoy improvements of BLEU score from increased depth, what does the models benefit from which is still implicit. To better understand this, we show the performance of deep MSC nets in handling sentences with complicated semantics. We assume that complicated sentences are difficult to fit with high prediction losses. Then we propose to use the modified prediction losses to identify these sentences:

$$
\begin{aligned}
s(\mathbf{x}, \mathbf{y}) = & \mathbb{E}\big[-\log P(\mathbf{y}|\mathbf{x}; \mathbf{\Theta})\big] \\
& + \mathrm{Std}\big[-\log P(\mathbf{y}|\mathbf{x}; \mathbf{\Theta})\big],
\end{aligned} \tag{14}
$$

where $\mathbb{E}\big[-\log P(\mathbf{y}|\mathbf{x}; \mathbf{\Theta})\big]$ is approximated by:

$$
\begin{aligned}
& \mathbb{E}\big[-\log P(\mathbf{y}|\mathbf{x}; \mathbf{\Theta})\big] \\
& \qquad \approx \frac{1}{K}\sum_{k=1}^{K} -\log P(\mathbf{y}|\mathbf{x}; \mathbf{\Theta}^{(k)}),
\end{aligned} \tag{15}
$$

where $\{\mathbf{\Theta}^{(k)}\}_{k=1}^{K}$ indicates model parameters for the last K ($K = 20$) checkpoints. $\mathrm{Std}[\cdot]$ is the



(a) 6-layer (b) 72-layer

Figure 4: Visualization of the attention weights from the top-most layer of the decoder for both shallow and deep MSC nets.

standard deviation of prediction loss of sentence $\mathbf{y}$ given sentence $\mathbf{x}$, and the introduction of which aims to prevent training oscillations from affecting complicated sentences identification.

We adopt a shallow plain net (`small`, 6 layers) to assign the prediction loss $s(\mathbf{x}, \mathbf{y})$ to each sentence pair. Further, we split the IWSLT En→De test set into 4 equal parts according to the prediction losses, which are pre-defined to have "Simple", "Ordinary", "Difficult" and "Challenging" translation difficulties, respectively.[4] Results on these fine-grained test sets are shown in Figure 3. First of all, all methods yield minor BLEU improvements over the baseline on the first sub-set that containing sentences with little difficulties to be translated. However, when the translation difficulty increases, the improvements of the deep MSC nets are expanded to around 2 BLEU. These results indicate that our MSC framework deals with sentences which are difficult to be translated well.

[4]The fine-grained test sets are publicly available at https://github.com/pemywei/MSC-NMT/tree/master/IWSLT_En2De_Split_Test.

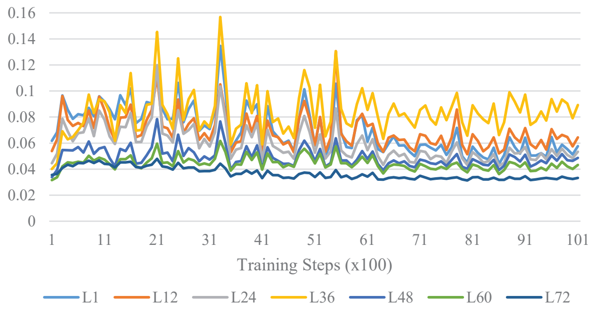


Figure 5: Gradient norm (y-axis) of each encoder layer in 72-layer MSC over the fist 10k training steps. "*Li*" denotes the i-th encoder layer. The MSC framework helps balance the gradient norm between top and bottom layers during training

We also visualize the attention weights from the top-most layer of the decoder of both shallow and deep MSC nets in Figure 4. As shown in Figure 4(a), when generating the next token of "tun", the shallow MSC attends to diverse tokens, such as "to", "that", "." and "eos", which causes the generation of "eos" and the phrase "be able to" is mistakenly untranslated. Remarkably, the deep MSC (Figure 4(b)) mostly focuses on the source tokens "be", "able" and "to", and translates this complicated sentence successfully. More cases can be found in Appendix C. This kind of cases show the advantages of constructing extremely deep models for translating semantic-complicated sentences.

Analysis of Error Propagation. To understand the propagation process of training signals, we collect the gradient norm of each encoder layer during training. Results in Figure 5 show that with the MSC framework each layer enjoys a certain value of gradient for parameter update, and the error signals traverse along the depth of the model without hindrance. MSC helps balance the gradient norm between top and bottom layers in deep models.

Ablation Study. We conduct ablation study to investigate the performance of each component of our model. The results are reported in Table 5: (1) We use simple element-wise addition for feature fusion instead of using a gated combination as introduced in Section 3.2. This method achieves a 29.45 BLEU, which is lower than the best result. We additionally modify the implementation of the contextual collaboration cell $\mathcal{Q}(\cdot)$ as FFN$(\cdot)$, which shows that the performance is reduced by 0.5 BLEU. (2) Removing CXT-ENC ATTENTION and/or contextual collaboration makes the BLEU score drop by $\sim$0.7, which suggests that multiscale collaboration helps in constructing extremely deep models. (3) Considering that the deep MSC introduces more parameters, we also train another two MSC models with about the same or double number of parameters: with 18/36 layers, embedding size 512 and FFN layer dimension 1024. These models underperform the deeper 72-layer model, which shows that the number of parameters is not the key to the improvement.

Model	**BELU**
MSC, 72 layers	29.67
- feature fusion with addition	29.45
- implement $\mathcal{Q}(\cdot)$ in Eq. (9) as FFN$(\cdot)$	29.17
- remove CXT-ENC ATTENTION	28.99
- remove contextual collaboration	28.94
MSC, 18 layers (emb=512, ffn=1024)	29.08
MSC, 36 layers (emb=512, ffn=1024)	29.41

Table 5: Ablation study on IWSLT14 En→De task.

5 Related Work

Researchers have constructed deep NMT models that use linear connections to reduce the gradient propagation length inside the topology (Zhou et al., 2016; Wang et al., 2017; Zhang et al., 2018b) or read-write operations on stacked layers of memories (Meng et al., 2015). Such work has been conducted on the basis of the conventional RNN architectures and may not be fully applicable to the advanced Transformer.

Recently, Bapna et al. (2018) introduced a transparent network into NMT models to ease the optimization of models with deeper encoders. To improve gradient flow they let each decoder layer find an unique weighted combination of all encoder layer outputs, instead of just the top encoder layer. Wang et al. (2019a) found that adopting the proper use of layer normalization helps to learn deep encoders. A method was further proposed to combine layers and encourage gradient flow by simple shortcut connections. Zhang et al. (2019a) introduced a depth-scaled initialization to improve norm preservation and proposed a merged attention sublayer to avoid the computational overhead for deep models. Researchers have also explored growing NMT models in two stages (Wu et al., 2019), in which shallow encoders and decoders are trained in the first stage and subsequently held constant, when another set of shallow layers are stacked on the top. In concurrent work, Xu et al. (2019) studied the effect of the computation order of residual connection and layer normalization, and proposed an parameter initialization method with Lipschitz restrictions to ensure the convergence of deep Transformers. Our method significantly differs from these methods, solving the problem by associating the decoder with the encoder with multi-granular dependencies in different space-scales.

Exploiting deep representations have been studied to strengthen feature propagation and encourage feature reuse in NMT (Shen et al., 2018; Dou et al., 2018, 2019; Wang et al., 2019b). All of these works mainly attend the decoder to the final output of the encoder stack, we instead coordinate the encoder and the decoder at earlier stage.

6 Conclusion and Future Work

In this paper, we propose a multisacle collaborative framework to ease the training of extremely deep NMT models. Specifically, instead of the top-most representation of the encoder stack, we attend the decoder to multi-granular source information with different space-scales. We have shown that the proposed approach boosts the training of very deep models and can bring improvements on translation quality from greatly increased depth. Experiments on various language pairs show that the MSC achieves prominent improvements over strong baselines as well as previous deep models.

In the future, we would like to extend our model to extremely large datasets, such as WMT'14 English-to-French with about 36M sentence-pairs. And the deeper MSC model results in high computational overhead, to address this issue, we would like to apply the average attention network (Zhang et al., 2018a) to our deep MSC models.

Acknowledgments

We would like to thank the anonymous reviewers for the helpful comments. We also thank Xingxing Zhang, Luxi Xing and Kaixin Wu for their instructive suggestions and invaluable help. This work is supported by the National Key Research and Development Program (Grant No.2017YFB0803301).

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 3159–3166.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. In *arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ankur Bapna, Mia Chen, Orhan Firat, Yuan Cao, and Yonghui Wu. 2018. Training deeper neural machine translation models with transparent attention. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3028–3033, Brussels, Belgium. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4253–4262, Brussels, Belgium. Association for Computational Linguistics.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. 2019. Dynamic layer aggregation for neural machine translation with routing-by-agreement. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 86–93.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 1 (Long and Short Papers), pages 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Salah El Hihi and Yoshua Bengio. 1996. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pages 493–499.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *arXiv:1705.03122*.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778.

Tianyu He, Xu Tan, Yingce Xia, Di He, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems 31*, pages 7944–7954.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4700–4708.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. 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*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork rnn. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. In *arXiv:1903.10318*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Fandong Meng, Zhengdong Lu, Zhaopeng Tu, Hang Li, and Qun Liu. 2015. A deep memory-based architecture for sequence-to-sequence learning. In *arXiv:1506.06442*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Belgium, Brussels. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *OpenAI Blog*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. In *Neural Computation*, pages 234–242.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yanyao Shen, Xu Tan, Di He, Tao Qin, and Tie-Yan Liu. 2018. Dense information flow for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1294–1303, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Vancouver, Canada. Association for Computational Linguistics.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F. Wong, and Lidia S. Chao. 2019a. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1810–1822, Florence, Italy. Association for Computational Linguistics.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019b. Exploiting sentential context for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6203, Florence, Italy. Association for Computational Linguistics.

Lijun Wu, Yiren Wang, Yingce Xia, Fei Tian, Fei Gao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Depth growing for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5558–5563, Florence, Italy. Association for Computational Linguistics.

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

Hongfei Xu, Qiuhui Liu, Josef van Genabith, and Jingyi Zhang. 2019. Why deep transformers are difficult to converge? from computation order to lipschitz restricted parameter initialization. In *arXiv:1911.03179*.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2403–2412.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019a. Improving deep transformer with depth-scaled initialization and merged attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 897–908, Hong Kong, China. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018b. Neural machine translation with deep attention. In *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 154–163.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019b. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy. Association for Computational Linguistics.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

A Abstractive Summarization

We further verify the effectiveness of MSC on text summarization. Automatic text summarization produces a concise and fluent summary conveying the key information in the input (e.g., a news article). We focus on abstractive summarization, a generation task aims to generate the summary of a document with rewriting. We use the

Model	R-1	R-2	R-L
Extractive Summarization			
Lead3	40.38	17.61	36.59
HIBERT$_M$	42.37	19.95	38.83
Liu (2019)	43.25	20.24	39.63
Abstractive Summarization			
PGNet	39.53	17.28	36.38
Bottom-Up	41.22	18.68	38.34
S2S-ELMo	41.56	18.94	38.47
TRANSFORMER	40.28	17.87	37.25
HIERTRANS (36 L)	41.22	18.97	38.45
MSC (36 L)	**41.96**	**19.50**	**39.07**

Table 6: Results on CNNDM summarization using ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L). "36L" is short for "36-layer encoder".

non-anonymized version of the CNN/DailyMail (CNNDM) dataset (See et al., 2017) for evaluation. We preprocessed the dataset using the scripts from the authors of See et al. (2017),[5] and the resulting dataset contains 287,226 documents with summaries for training, 13,368 for validation and 11,490 for test.

We still adopt the Transformer (Vaswani et al., 2017) as our backbone, with a embedding size of 512 and FFN layer dimension of 1024. We train our model on the training set for 30 epochs, and also use label smoothing with rate of 0.1. We set batch size to 32, and maximum length to 768. During decoding, we use beam search with beam size of 5. The input document is truncated to the first 640 tokens. We remove duplicated trigrams in beam search, and tweak the maximum summary length on the development set (Paulus et al., 2018; Edunov et al., 2019). We use the F1 version of ROUGE (Lin, 2004) as the evaluation metric.

In Table 6, we compare MSC (36 layers) against the baseline and several state-of-the-art models on CNN/DailyMail, with extractive models in the top block and abstractive models in the bottom block. Lead3 is a baseline which simply selects the first three sentences as the summary. HIBERT$_M$ (Zhang et al., 2019b) adds the large open-domain unlabeled data to pre-train hierarchical transformer encoders and fine-tune on the extractive summarization task. We also include in Table 6 the best reported extractive summarization result taken from (Liu, 2019) on the dataset. PGNet (See et al., 2017), Bottom-Up (Gehrmann et al., 2018) and S2S-ELMo (Edunov et al., 2019) are all sequence to sequence learning based models with copy and coverage modeling, bottom-up content selecting and pre-trained ELMo representations augmenting. We also implemented two baselines. One is the standard 6-layer Transformer model. We can see that the deep MSC leads to a +1.8 ROUGE improvement over TRANSFORMER. The other baseline is the hierarchical transformer summarization model (HIERTRANS), which involevs both a sentence-level and a document-level transformer encoders, as well as a standard transformer decoder. Note the settings for both encoders are the same (each of them have L=18, emb=512, ffn=1024, head=8). The deep MSC outperforms HIERTRANS by 0.5 to 0.7 ROUGE with the same depth of encoders.

[5] https://github.com/abisee/cnn-dailymail

B Derivations of Block-Scale Collaboration

In pre-norm Transformer, a general transformation can be formulated as:

$$\mathrm{H}^{l} = \mathcal{F}(\mathrm{H}^{l-1}; \boldsymbol{\Theta}^{l}) + \mathrm{H}^{l-1}, \tag{16}$$

where H^{l-1} and H^{l} are the input and output of the l-th layer. For the Block-Scale Collaboration framework, there are two channels for passing error gradients from the prediction loss $\mathcal{L}$ to encoder layers, which are from the top-most layers of the whole encoder stack H_e^{N,M_N} (identical to B_e^{N}) and the current bock H_e^{n,M_n} (identical to B_e^{n}), respectively. From the chain rule of back propagation we can obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^{n,l}} = \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^{N}} \times \frac{\partial \mathrm{B}_e^{N}}{\partial \mathrm{H}_e^{n,l}} + \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^{n}} \times \frac{\partial \mathrm{B}_e^{n}}{\partial \mathrm{H}_e^{n,l}}. \tag{17}$$

We can recursively use Eq. (16) to formulate that

$$\begin{aligned} \mathrm{B}_e^{N} =& \mathrm{H}_e^{N,M_N} \\ =& \mathrm{H}_e^{n,l} + \sum_{k=l+1}^{M_n} \mathcal{F}(\mathrm{H}_e^{n,k-1}; \boldsymbol{\Theta}_e^{n,k}) \\ &+ \sum_{i=n+1}^{N} \sum_{j=1}^{M_i} \mathcal{F}(\mathrm{H}_e^{i,j-1}; \boldsymbol{\Theta}_e^{i,j}), \end{aligned} \tag{18}$$

and

$$\begin{aligned} \mathrm{B}_e^{N} =& \mathrm{H}_e^{n,M_n} \\ =& \mathrm{H}_e^{n,l} + \sum_{k=l+1}^{M_n} \mathcal{F}(\mathrm{H}_e^{n,k-1}; \boldsymbol{\Theta}_e^{n,k}), \end{aligned} \tag{19}$$

respectively. In this way, the derivations of B_e^N and B_e^n with respect to $\mathrm{H}_e^{n,l}$ can be calculated as:

$$\frac{\partial \mathrm{B}_e^N}{\partial \mathrm{H}_e^{n,l}} = 1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathcal{F}(\mathrm{H}_e^{n,k-1}; \boldsymbol{\Theta}_e^{n,k})}{\partial \mathrm{H}_e^{n,l}} + \sum_{i=n+1}^{N} \sum_{j=1}^{M_i} \frac{\partial \mathcal{F}(\mathrm{H}_e^{i,j-1}; \boldsymbol{\Theta}_e^{i,j})}{\partial \mathrm{H}_e^{n,l}}, \quad (20)$$

$$\frac{\partial \mathrm{B}_e^n}{\partial \mathrm{H}_e^{n,l}} = 1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathcal{F}(\mathrm{H}_e^{n,k-1}; \boldsymbol{\Theta}_e^{n,k})}{\partial \mathrm{H}_e^{n,l}}.$$

Finally, we can put Eq. (20) into Eq. (17) and obtain:

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial \mathrm{H}_e^{n,l}} \\
&= \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^N} \times \frac{\partial \mathrm{B}_e^N}{\partial \mathrm{H}_e^{n,l}} + \frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^n} \times \frac{\partial \mathrm{B}_e^n}{\partial \mathrm{H}_e^{n,l}} \\
&= \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^N} \times (1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathrm{H}_e^{n,k}}{\partial \mathrm{H}_e^{n,l}} + \sum_{i=n+1}^{N} \sum_{j=1}^{M_i} \frac{\partial \mathrm{H}_e^{i,j}}{\partial \mathrm{H}_e^{n,l}})}_{(a)} \\
&\quad + \underbrace{\frac{\partial \mathcal{L}}{\partial \mathrm{B}_e^n} \times (1 + \sum_{k=l+1}^{M_n} \frac{\partial \mathrm{H}_e^{n,k}}{\partial \mathrm{H}_e^{n,l}})}_{(b)}.
\end{aligned} \quad (21)$$