Layer-Wise Multi-View Learning for Neural Machine Translation

Qiang Wang[1]*, **Changliang Li**[2], **Yue Zhang**[3]†, **Tong Xiao**[4,5], **Jingbo Zhu**[4,5]
[1]Machine Intelligence Technology Lab, Alibaba DAMO Academy, Hangzhou, China
[2]Kingsoft AI Lab, Beijing, China
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study, Hangzhou, China
[4] School of Computer Science and Engineering, Northeastern University, Shenyang, China
[5]NiuTrans Research, Shenyang, China
zhiniao.wq@alibaba-inc.com, lichangliang@kingsoft.com
zhangyue@westlake.edu.cn, {xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract

Traditional neural machine translation is limited to the topmost encoder layer's context representation and cannot directly perceive the lower encoder layers. Existing solutions usually rely on the adjustment of network architecture, making the calculation more complicated or introducing additional structural restrictions. In this work, we propose layer-wise multi-view learning to solve this problem, circumventing the necessity to change the model structure. We regard each encoder layer's off-the-shelf output, a by-product in layer-by-layer encoding, as the redundant view for the input sentence. In this way, in addition to the topmost encoder layer (referred to as the primary view), we also incorporate an intermediate encoder layer as the auxiliary view. We feed the two views to a partially shared decoder to maintain independent prediction. Consistency regularization based on KL divergence is used to encourage the two views to learn from each other. Extensive experimental results on five translation tasks show that our approach yields stable improvements over multiple strong baselines. As another bonus, our method is agnostic to network architectures and can maintain the same inference speed as the original model.

1 Introduction

Neural Machine Translation (NMT) adopts the encoder-decoder paradigm to model the entire translation process (Bahdanau et al., 2015). Specifically, the encoder finds a multi-layer representation of the source sentence, and the decoder queries the topmost encoding representation to produce the target sentence through a cross-attention mechanism (Wu et al., 2016; Vaswani et al., 2017). However, such over-reliance on the topmost encoding layer is problematic in two aspects: (1) Prone to over-fitting, especially when the encoder is under-trained, such as in low-resource tasks (Wang et al., 2018); (2) It cannot make full use of representations extracted from lower encoder layers, which are syntactically and semantically complementary to higher layers (Peters et al., 2018; Raganato and Tiedemann, 2018).

Researchers have proposed many methods to make the model aware of various encoder layers besides the topmost to mitigate this issue. Almost all of them resort to the adjustment of network structure, which can be further divided into two categories. The first is to merge the feature representations extracted by distinct encoder layers before being fed to the decoder (Wang et al., 2018; Dou et al., 2018; Wang et al., 2019b). The differences between them lie in the design of the merge function: through self-attention (Wang et al., 2018), recurrent neural network (Wang et al., 2019b), or tree-like hierarchical merge (Dou et al., 2018). Moreover, the second makes each decoder layer explicitly align to a parallel encoder layer (He et al., 2018) or all encoder layers (Bapna et al., 2018). However, the above methods either complicate the original model (Wang et al., 2018; Dou et al., 2018; Wang et al., 2019b; Bapna et al., 2018) or limit the model's flexibility, such as requiring the number of the encoder layers to be the same as the decoder layers (He et al., 2018).

Instead, in this work, we propose layer-wise multi-view learning to address this problem from the perspective of model training, without changing the model structure. Our method's highlight is that

*Work done during Ph.D. study at Northeastern University.
†Corresponding author.

only the training process is concerned, while the inference speed is guaranteed to be the same as that of the standard model. The core idea is that we regard the off-the-shelf output of each encoding layer as a view for the input sentence. Therefore, it is straightforward and cheap to construct multiple views during a standard layer-by-layer encoding process. Further, in addition to the output of the topmost encoder layer used in standard models (refer to the *primary view*), we also incorporate an intermediate encoder layer as the *auxiliary view*. We feed the two views to a partially shared decoder for independent predictions. An additional regularization loss based on prediction consistency between views is used to encourage the auxiliary view to mimic the primary view. Thanks to the co-training on the two views, the gradients during back-propagation can simultaneously flow into the two views, which implicitly realizes the knowledge transfer.

Extensive experimental results on five translation tasks (Ko$\rightarrow$En, IWSLT'14 De$\rightarrow$En, WMT'17 Tr$\rightarrow$En, WMT'16 Ro$\rightarrow$En, and WMT'16 En$\rightarrow$De) show that our method can stably outperform multiple baseline models (Vaswani et al., 2017; Wang et al., 2018; Dou et al., 2018; Bapna et al., 2018). In particular, we have achieved new state-of-the-art results of 10.8 BLEU on Ko$\rightarrow$En and 36.23 BLEU on IWSLT'14 De$\rightarrow$En. Further analysis shows that our method's success lies in the robustness to encoding representations and dark knowledge (Hinton et al., 2015) provided by consistency regularization.

2 Approach

In this section, we will take the Transformer model (Vaswani et al., 2017) as an example to show how to train a model by our multi-view learning. We first briefly introduce Transformer in § 2.1, then describe the proposed multi-view Transformer model (called `MV-Transformer`) and its training and inference in detail in § 2.2. Finally, we discuss why our method works in § 2.3. See Figure 1 for an overview of the proposed approach.

2.1 Transformer

The Transformer systems follow the encoder-decoder paradigm. On the encoder side, there are M identical stacked layers. Each of them comprises a self-attention-network (SAN) sub-layer and a feed-forward-network (FFN) sub-layer. To easy optimization, layer normalization (LN) (Ba et al., 2016) and residual connections (He et al., 2016) are used between these sub-layers. There are two ways to incorporate them, namely `PreNorm` Transformer and `PostNorm` Transformer (Wang et al., 2019a). Without loss generalization, here we only describe the implementation of PreNorm Transformer, but we test our method in both cases[1]. The l-th encoder layer of PreNorm Transformer is:

$$\begin{aligned} \dot{H}^{(l)} &= H^{(l-1)} + \text{SAN}\big(\text{LN}(H^{(l-1)})\big) \\ H^{(l)} &= \dot{H}^{(l)} + \text{FFN}\big(\text{LN}(\dot{H}^{(l)})\big) \end{aligned} \tag{1}$$

where $\dot{H}$ denotes the intermediate encoding state after the first sublayer. Besides, there is an extra layer normalization behind the topmost layer to prevent the excessive accumulation of the unnormalized output in each layer, i.e. $H^* = \text{LN}(H^{(M)})$, where H^* denotes the final encoding result. Likewise, the decoder has another stack of N identical layers, but an additional cross-attention-network (CAN) sub-layer is inserted between SAN and FFN compared to the encoder layer:

$$\ddot{Z}^{(l)} = \dot{Z}^{(l)} + \text{CAN}\big(\text{LN}(\dot{Z}^{(l)}), H^*\big) \tag{2}$$

CAN$(\cdot)$ is similar to SAN$(\cdot)$ except that its *key* and *value* are composed of the encoding output H^* instead of *query* itself. Resemble H^*, the last extracted feature vector by decoder is $Z^* = \text{LN}(Z^{(N)})$. Thus, given a sentence pair of $\langle \mathbf{x}, \mathbf{y} \rangle$, where $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, we can train the model parameters θ by minimizing the negative log-likelihood:

$$\mathcal{L}_{nll}(\theta) = -\sum_{j=1}^{n} \text{log} p_\theta(y_j | \mathbf{x}, y_{<j}) \tag{3}$$

[1]The basic form of PostNorm Transformer is: $y = \text{LN}(x + \text{SubLayer}(x))$, and there is no additional layer normalization on the top of encoder/decoder.

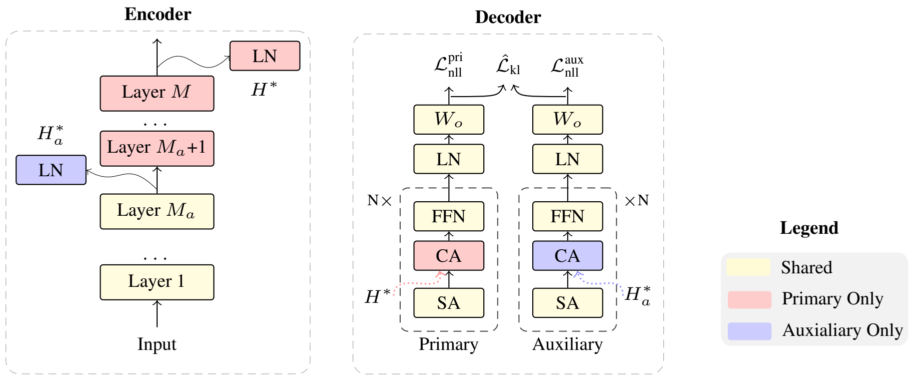


Figure 1: An overview of proposed layer-wise multi-view learning for PreNorm Transformer. We maintain a two-stream decoder during training by partially sharing.

where $p_\theta(y_j|\mathbf{x}, y_{<j}) = \text{Softmax}(W_o Z^*_j + b_o)$, W_o and b_o are the parameters in the output layer.

2.2 Multi-View Transformer

Multi-view. Multi-view learning has achieved great success in conventional machine learning by exploiting the redundant views of the same input data (Xu et al., 2013), where one of the keys is view construction. In our scenario, a view is the hidden representation of the input sentence (an array of hidden vectors for each token, e.g., H^*). In this work, we further propose to take the off-the-shelf output of each encoder layer (i.e., $H^{(l)}$ [2]) to construct the redundant views. In NLP, previous implementations of view construction generally require the model to recalculate on the reconstructed input, such as using different orders of n-grams in the bag-of-word model (Matsubara et al., 2005), randomly masking the input tokens (Clark et al., 2018). As opposed to them, our method is very cheap as the by-product of the standard layer-by-layer encoding process. According to the definition of a view, we can regard the vanilla Transformer as a single-view model since only the topmost encoder layer (also called *primary view*) is fed to the decoder. In contrast, MV-Transformer additionally contains an intermediate layer M_a ($1 \le M_a < M$) as the *auxiliary view* [3]. The choice of M_a can be arbitrary, and we discuss its effect in § 4.2. Our goal is to learn a better single model with the help of the auxiliary view.

Partially shared parameters. In the encoder, except for the last layer normalization, all other parameters are shared in the two views to obtain the corresponding view representations by encoding only once. However, the situation is different for the decoder. We empirically find that a fully shared decoder has no sufficient capacity to be compatible with two different views simultaneously, especially in medium or large translation tasks (see § 4.4). On the other hand, it can be seen that the difference between the two views only directly affects the CANs in the decoder and has nothing to do with other sublayers (i.e., SANs, FFNs). Therefore, using a separate decoder for each view will cause an enormous waste of decoder model parameters. To trade-off, we extend the decoder network by using independent CANs for each view but share all SANs and FFNs (see Figure 1) [4].

Two-stream decoder. Given the two views, we use a two-stream decoder during training. Like Eq. 2, the auxiliary view is queried as:

$$\ddot{Z}^{(l)}_a = \dot{Z}^{(l)}_a + \text{CAN}_a\big(\text{LN}_a(\dot{Z}^{(l)}_a), H^*_a\big) \tag{4}$$

[2]To be precise, there is a newly added layer normalization following $H^{(l)}$ to keep consistent with H^*.

[3]In this work, we only consider the case of one auxiliary view. More auxiliary views may be more helpful, while it requires more training costs. We leave this issue in future work.

[4]In earlier experiments, we tried to add *view embedding* to make the decoder aware of identity of each view, i.e. $H^* + \mathcal{E}_{pri}$ for primary view, where $\mathcal{E}_{pri} \in \mathcal{R}^d$, but it did not work well.

where the subscript a indicates used for auxiliary view. In each decoding step, one stream queries $\dot{Z}^{(l)}$ from the primary view H^* like a standard Transformer, while the other stream queries $\dot{Z}_a^{(l)}$ from the auxiliary view H_a^* through separate CAN sublayers. In this way, each stream yields distinct predictions based on different context semantics in the views. Here we use $p_{pri}(\cdot)$ and $p_{aux}(\cdot)$ to denote the prediction distribution by the primary view and the auxiliary view, respectively.

Training. To jointly train the two views and transfer the knowledge between them, the training objective of MV-Transformer consists of two items. The first item $\hat{\mathcal{L}}_{nll}$ is similar to the negative log-likelihood in Eq. 3, but additionally considers the log-likelihood of the auxiliary view prediction:

$$\hat{\mathcal{L}}_{nll} = \frac{1}{2} \times (\mathcal{L}_{nll}^{pri} + \mathcal{L}_{nll}^{aux}) \tag{5}$$

where $\mathcal{L}_{nll}^{pri}$, $\mathcal{L}_{nll}^{aux}$ are based on the distribution of $p_{pri}(\cdot)$ and $p_{aux}(\cdot)$ respectively, and $1/2$ is used to numerically scale $\hat{\mathcal{L}}_{nll}$ to $\mathcal{L}_{nll}$. The second item $\hat{\mathcal{L}}_{cr}$ is the consistency regularization loss between views, where we use Kullback–Leibler (KL) divergence to let the student (played by the auxiliary view) imitate the prediction of the teacher (played by the primary view):

$$\hat{\mathcal{L}}_{cr} = -\sum_{j=1}^{n} \mathrm{KL}(p_{aux}^{(j)} || p_{pri}^{(j)}) = -\sum_{j=1}^{n} \sum_{v \in \mathcal{V}} p_{pri}^{(j)}(v) \log \frac{p_{pri}^{(j)}(v)}{p_{aux}^{(j)}(v)} \tag{6}$$

where $p^{(j)}(v)$ is the probability of generating token v at step j[5]. We note that our consistency regularization is different from traditional knowledge distillation, where a typical implementation is to detach the teacher's prediction $p_{pri}^{(j)}$ as a constant (Hinton et al., 2015). On the contrary, our method takes $p_{pri}^{(j)}$ as a variable that requires gradients during back-propagation. To this end, the entire model parameters are optimized to give good predictions in two views instead of considering only one, which implicitly makes the model learn from different encoder layers. Some people may say that it is enough for the student to learn from the teacher, but the reverse is unreasonable. However, we believe that the information in different views is complementary, so the potential for mutual learning of views may be greater than one-way learning. And our empirical comparison in § 3.3 also confirms this assumption. Finally, we can interpolate these two losses with the hyper-parameter α to obtain the overall loss function for multi-view learning:

$$\hat{\mathcal{L}} = (1-\alpha) \times \hat{\mathcal{L}}_{nll} + \alpha \times \hat{\mathcal{L}}_{cr} \tag{7}$$

Intuitively, when α is low, the loss degrades into Eq. 5, which only focuses on the ground-truth labels. On the contrary, a high α overemphasizes the consistency of the entire vocabulary between the two views, resulting in neglecting to learn from the provided ground-truth. We discuss α's effect in § 4.2.

Inference. Instead of maintaining both views like training, we can shift to any single view at inference time. Considering the primary view as an example: We can straightforwardly discard all the modules attached to the auxiliary view, including CAN$_a$ and LN$_a$ in the decoder as well as the newly added layer normalization in the encoder. It makes the decoding speed to be precisely the same as that of the standard model. Likely, we can also switch to the auxiliary view composed of fewer encoder layers for slightly faster speed, but with the risk of performance degradation.

2.3 Discussion

In this section, we discuss why our method works from two aspects: robustness to encoding representation and dark knowledge. See § 4.1 for more experimental analysis.

Robustness to encoding representation. Over-reliance on the top encoding layer (primary view) makes the model easier to over-fit (Wang et al., 2018). Our method attempts to reduce the sensitivity

[5]We use temperature τ=1 in all experiments, e.g. $p(i) = \exp(z_i/\tau)/\sum_{j \in \mathcal{V}} \exp(z_j/\tau)$, z is the logit.

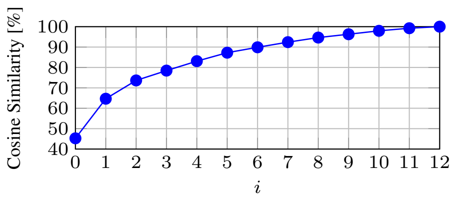


Figure 2: Cosine similarity between the i-th encoder layer and the topmost encoder layer in a pre-trained PreNorm Transformer with a 12-layer encoder. The results were measured on the IWSLT'14 De→En validation set. i=0 indicates the embedding layer.

to the primary view by feeding an auxiliary view. Figure 2 shows that the vector similarity between the i-th encoder layer and the topmost layer grows as the increase of i. Therefore, we can regard the middle layer's auxiliary view as a noisy version of the primary view. Training with noises has been widely proven to effectively improve the model's generalization ability, such as dropout (Srivastava et al., 2014), adversarial training (Miyato et al., 2017; Cheng et al., 2019) etc. We also experimentally confirm that our model is more robust than the single view model when injecting random noises into the encoding representation.

Dark knowledge. Typically, the prediction target in $\mathcal{L}_{nll}$ is a one-hot distribution: Only the gold label is 1, while the others are 0. A better alternative is label smoothing (Szegedy et al., 2016), which reduces the probability of gold label by ϵ and redistributes ϵ to all non-gold labels on average. However, label smoothing ignores the relationship between non-gold labels. For example, if the current ground-truth is "improve", then "promote" should have high probability than "eat". In contrast, in our method, the target in the auxiliary view is the primary view's prediction, which contains more information about non-gold labels, also known as `dark knowledge` (Hinton et al., 2015).

3 Experiments

3.1 Setup

Datasets. We conducted experiments on five translation tasks: Korean→English (Ko→En, 96k)[6], IWSLT'14 German→English (De→En, 160k), WMT'17 Turkish→English (Tr→En, 205k), WMT'16 Romanian→English (Ro→En, 601k)[7] and WMT'16 English→German (De→En, 4.5M)[8]. We use the officially provided development sets and test sets for all tasks. We pre-process and tokenize all the data sets using the Moses toolkit.

Models and hyperparameters. We tested all models in shallow networks based on PostNorm and deep networks based on PreNorm, respectively. Concretely, for shallow models, we use M=N=3 for the small-scale Ko-En task [9], while M=N=6 for other tasks. We use *Small* configuration (embed=512, ffn=1024, head=4) for {Ko, De, Tr}-En, and *Base* configuration (embed=512, ffn=2048, head=8) for Ro-En and En-De. As for deep models, we double the encoder depth as the corresponding PostNorm counterparts. E.g., suppose we use a 6-layer encoder in vanilla Transformer. In that case, we turn it to 12-layer in deep Transformer. For MV-Transformer, we use 1/3/6-th encoder layer as the auxiliary view when the encoder depth is 3/6/12, respectively. Following Vaswani et al. (2017), we use the *inverse_sqrt* learning rate schedule with warm-up and label smoothing of 0.1. Some training hyperparameters are distinct across tasks due to the different data sizes. Detailed hyperparameters are listed in Appendix A.

Decoding and evaluation. To compare with previous works, we use the beam size of 4 and average last 5 checkpoints on De→En, while for other tasks, we use the beam size of 5 and the best checkpoint according to the best BLEU score on the development set. For evaluation, except that Ko→En

[6]`https://sites.google.com/site/koreanparalleldata/`
[7]`https://github.com/nyu-dl/dl4mt-nonauto`
[8]`https://drive.google.com/uc?export=download&id=0B_bZck-ksdkpM25jRUN2X2UxMm8`
[9]We failed to train with M=N=6 in early experiments.

	Model	Ko→En	De→En	Tr→En	Ro→En	En→De
	Shallow networks with PostNorm					
Aux.	Transformer†	8.7	34.10	14.32	33.07	32.80
	MLRF†	9.1	34.73	**14.97**	33.7	33.21
	HieraAgg†	N/A	N/A	N/A	N/A	N/A
	TA†	8.8	34.23	14.51	33.15	32.92
	MV-Transformer	**10.2**	**35.25**	14.74	**34.24**	**33.38**
	Δ	+1.5	+1.15	+0.42	+1.17	+0.58
Pri.	Transformer†	9.6	34.77	14.78	33.20	33.06
	MLRF†	10.0	33.53	15.18	33.79	33.17
	HieraAgg†	N/A	34.98	14.75	34.09	33.36
	TA†	9.1	34.58	15.06	33.49	32.97
	MV-Transformer	**10.4**	**35.49**	**15.25**	**34.45***	**33.75**
	Δ	+0.8	+0.72	+0.47	+1.25	+0.69
	Deep networks with PreNorm					
Aux.	deep Transformer†	9.1	35.38	14.27	33.15	33.61
	MLRF†	9.5	35.32	14.71	33.55	33.51
	HieraAgg†	9.4	34.77	14.89	33.13	33.51
	TA†	8.7	35.14	14.84	33.26	33.32
	deep MV-Transformer	**10.8**	**35.95**	**15.71**	**33.90**	**34.10**
	Δ	+1.7	+0.57	+1.44	+0.75	+0.49
Pri.	deep Transformer†	9.7	35.75	15.03	33.55	34.06
	MLRF†	10.0	35.99	15.0	33.24	**34.57***
	HieraAgg†	8.9	35.03	14.48	32.57	33.82
	TA†	9.0	34.63	14.85	33.78	33.88
	deep MV-Transformer	**10.8***	**36.23***	**15.74***	**34.05**	**34.57***
	Δ	+1.1	+0.48	+0.71	+0.50	+0.51

Table 1: BLEU scores on five translation tasks. For (deep) Transformer, `Aux./Pri.` denotes the independently trained model with M_a/M-layer encoder respectively. For (deep) MV-Transformer, `Aux./Pri.` denotes the used view at inference time. Δ denotes the improved BLEU score over the Transformer baseline when using multi-view learning at the same encoder depth. † denotes our implementation. Boldface and * represent local and global best results, respectively. All the MV-Transformer results are significantly better ($p<0.01$) than the Transformer counterparts, measured by paired bootstrap resampling (Koehn, 2004).

uses *sacrebleu* [10], all other datasets are evaluated by *multi-bleu.perl*. Only De→En is reported by case insensitive BLEU.

3.2 Main results

In addition to Transformer, we also re-implemented three previously proposed models that incorporate multiple encoder layers: multi-layer representation fusion (MLRF)(Wang et al., 2018), hierarchical aggregation (HieraAgg) (Dou et al., 2018), and transparent attention (TA) (Bapna et al., 2018). Table 1 shows the results of the five translation tasks on PostNorm and PreNorm. First, our MV-Transformer outperforms all baselines across the board. Specifically, for PostNorm models, with the helper of multi-view learning, both views can improve the Transformer baselines by about 0.4-1.5 BLEU points. Consistent improvements of 0.5-1.7 BLEU points are also obtained even in the stronger PreNorm baselines that benefit from the encoder's increased depth. And we achieve the new state-of-the-art of 10.8 and 36.23 on Ko→En and De→En, respectively [11]. Note that these five tasks include both low-resource scenarios (Ko→En) and rich-resource scenarios (En→De), which indicates that our method has a good generalization for the scale of data size.

[10]BLEU+c.mixed+l.ko-en+#.1+s.exp+tok.13a+v.1.2.21

[11]The previous state-of-the-art is 10.3 (Sennrich and Zhang, 2019) on Ko→En and 35.6 (Zhang et al., 2019) on De→En.

Model	Speed	De→En	Ro→En
Baseline (3L)	+0.3%	34.10	33.07
Baseline	reference	34.77	33.20
Oneway-KD	+0.8%	35.07	33.78
Seq-KD (3L)	+4.7%	34.13	32.45
Ensemble	-45.3%	**35.77**	34.15
MV (3L)	+1.6%	35.25	34.24
MV	+0.3%	35.49	**34.45**

Table 2: Compare multi-view learning (MV) to oneway knowledge distillation (Oneway-KD), sequence-level knowledge distillation (Seq-KD) and model ensemble (Ensemble) on IWSLT'14 De→En and WMT'16 En→Ro test sets. Speed is the average of 5 runs with `batch`=32, `beam`=5.

ID	Gold	Dark	BLEU
#1	$\surd$	All	35.49
#2	$\surd$	-	35.02 (-0.47)
#3	-	All	35.44 (-0.05)
#4	-	[1,100]	35.35 (-0.14)
#5	-	[100,]	34.79 (-0.70)

Table 3: Dark knowledge in our multi-view learning. *Gold* is the ground-truth label, while *Dark* denotes the set of all non-ground-truth labels. $[a, b]$ denotes the top-a to top-b predicted labels (exclude ground-truth).

3.3 Compare to knowledge distillation and model ensemble

MV-Transformer can be thought of as consisting of two models: A large model as the primary view, and a small model (with shallower encoder) as the auxiliary view. Here we compare with the other three methods of integrating multiple models:

- **Oneway-KD.** Similar to Eq. 7 but detach the teacher's prediction, i.e., gradients of the teacher's prediction is not tracked, posing a one-way transfer from primary view to auxiliary view.
- **Seq-KD.** Train the large model first and then translate the original training set by beam search to construct the distilled training set for the small model (Kim and Rush, 2016).
- **Ensemble.** Independently train the two models and combine their predictions at inference time, e.g., by algorithmic average.

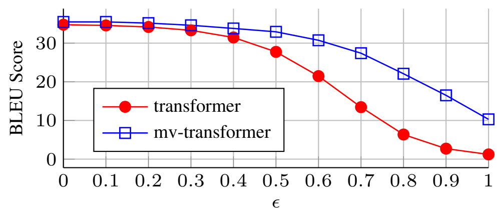
Experiments are done on IWSLT'14 De→En, where the small model has a 3-layer encoder. As shown in Table 2, we can see that: (1) *Oneway-KD* suffers from severe degradation than *MV* when detaching the primary view, which indicates that making mutual learning between the primary view and auxiliary view is critical; (2) *Seq-KD* is almost useless or even badly hurts the performance (vs. *Baseline (3L)*), which is against the previous belief that *Seq-KD* helps the small model a lot by learning from the teacher. We suspect that the reason is that our student's performance has already been closed to the teacher; (3) *Ensemble* can achieve significant performance improvement than a single model but at the cost of almost twice the slower decoding speed. However, our approach can achieve comparable or even better results than the model ensemble but maintain the decoding speed as a single model.

4 Analysis

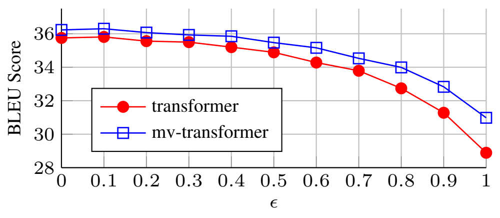
4.1 Why multi-view learning works?

Robustness to encoding noises. In § 2.3, we suppose that MV-Transformer is less sensitive to noise encoding representations due to the introduction of the auxiliary view. To verify it, we add random Gaussian noises sampled from $\mathcal{N}(0, \epsilon)$ into the normalized input in the last layer normalization of the encoder[12]. As shown in Figure 3, we can see that while both models degrade performance along with stronger noises, MV-Transformer is less sensitive, e.g., the maximum gap is 15.72/2.09 when ϵ=0.8/1.0 for the 6/12-layer encoder respectively. It indicates that our model has a better generalization even if the test distribution is largely different from the training distribution. We also observed that the PreNorm-style Transformer

[12]Original layer normalization is $y = g \odot N(x) + b$, while our noised version is $y = g \odot \big(N(x) + \epsilon\big) + b$. Our purpose is to avoid different scales of g and b between models.



(a) 6-layer encoder



(b) 12-layer encoder

Figure 3: BLEU scores on IWSLT'14 De→En test set w.r.t injecting noises sampled from $\mathcal{N}(0, \epsilon)$ on the encoding representation. (a) 6-layer encoder with PostNorm; (b) 12-layer encoder with PreNorm.

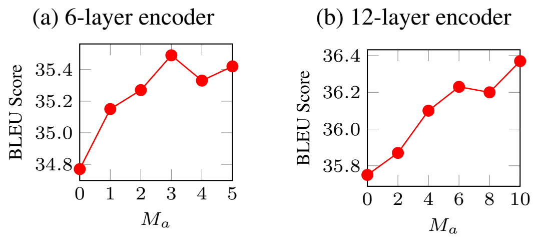


Figure 4: BLEU scores against the position of auxiliary view (M_a) on IWSLT'14 De→En. M_a=0: no multi-view.

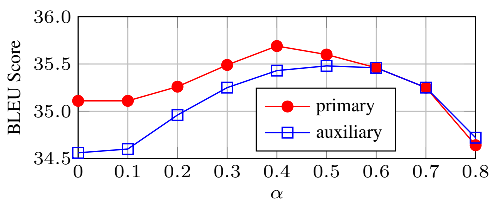


Figure 5: BLEU scores against α for multi-view learning on IWSLT'14 De→En test set. M_a=3, M=6.

is less sensitive than the PostNorm counterpart, e.g., when ϵ=1.0, the PostNorm Transformer decreases 33.55 BLEU points, while the PreNorm one only decreases 6.86.

Dark knowledge. As shown in Table 3, we study the effect of dark knowledge in our multi-view learning. First, we test the case where we only use gold knowledge (ground-truth label, #2) and dark knowledge (non-ground truth labels, #3). Obviously, without the help of dark knowledge, multi-view learning fails to boost the performance. Going further along this line, we study which part of dark knowledge is the most important (#4-5). Specifically, we split all non-gold labels into two parts from high to low according to their probability: [1,100] and [100,]. We can see that our approach's success lies in those non-gold labels at the top, not the long tail part.

4.2 Hyperparameter sensitivity

Position of auxiliary view. In Figure 4, we plot the BLEU score curves against the auxiliary view position L_a on IWSLT'14 De→En. In general, we can obtain better performance when L_a is closer to L_e, but this is not always true. It is intuitive: When $L_a \ll L_e$, the auxiliary view is numerically far different from the primary view, which is difficult for a partially shared decoder to learn. On the contrary, if L_a is too close to the top, the slight difference may not bring too many learnable signals. In this work, we use the middle layer as the auxiliary view, while it should be noted that we could obtain better results if we tune L_a more carefully (e.g., L_a=10 vs. L_a=6 in the 12-layer encoder).

Interpolation coefficient. In Figure 5, we show the curve of BLEU score against the hyperparameter α on IWSLT'14 De→En test set. First of all, we can see that the BLEU score is improved along with the increase of α, while it starts to decrease when α is too large (e.g., $\alpha > 0.6$). In particular, we failed to train the model when α=1.0. On the other hand, a large α can reduce the gap between the two views as expected. We note that even though it is difficult to know the optimal α in advance, we empirically found that $\alpha \in [0.3, 0.5]$ is robust across these distinct tasks.

Model	De→En
DynamicConv (Wu et al., 2019)	35.2
DynamicConv[†] (4 layers)	35.13
DynamicConv[†] (7 layers)	35.21
MV-DynamicConv (auxiliary)	35.59
MV-DynamicConv (primary)	**35.79**

Table 4: Apply multi-view learning to DynamicConv with M_a=4 and M=7 on IWSLT'14 De→En test set. † denotes our implementation.

ID	Model	DE→EN	RO→EN
#1	Baseline	34.77	33.20
#2	MV-3-6	35.49	34.45
#3	MV-3-6 (shared)	35.51	33.93
#4	MV-6-6	35.05	33.86

Table 5: Ablation study. `MV-#1-#2` denotes M_a=#1 and M=#2, `(shared)` indicates the use of shared CAN sublayers in decoder.

4.3 Transparency to network architecture

In addition to the Transformer, we also test our method on the recently proposed DynamicConv (Wu et al., 2019). Original DynamicConv is composed of a 7-layer encoder and a 6-layer decoder. Our method takes the topmost layer as the primary view as before and uses the 4-th layer as the auxiliary view. The results on IWSLT'14 De→En task are listed in Table 4. It can be seen that DynamicConv with multi-view is stably higher than that of a single-view model by about 0.5 BLEU score, which indicates that our method is transparent to network architecture and has the potential to be widely used.

4.4 Ablation study

We did ablation studies to understand the effects of (a) separate CANs and (b) using a lower encoder layer as auxiliary view. Experimental results are listed in Table 5. We can see that: (1) Under almost the same parameter size, sharing CANs (#3) obtains +0.7 BLEU in both tasks compared to the baseline (#1), which indicates the improvement comes from our multi-view training instead of the increased parameters; (2) Using separate CANs is more helpful than sharing CANs when the size of training data is large enough (#3 vs. #2); (3) Thanks to separate CANs, the decoder can obtain distinguishable context representations even if the auxiliary view is the same as the primary view (#4 vs. #1); (4) The auxiliary view with a high layer (#4, M_a=6) performs worse than that of a low layer (#2, M_a=3), which strongly indicates the diversity between views is more important than the quality of the auxiliary view.

5 Related Work

Multi-view learning. In multi-view learning, one of the most fundamental problems is view construction. Most previous works study random sampling in the feature spaces (Ho, 1998), feature vector transformation by reshaping (Wang et al., 2011). For natural language processing, Matsubara et al. (2005) obtain the multiple views of one document by taking different grams as terms in the bag-of-word model. Perhaps the most related work in this topic is Clark et al. (2018), which randomly mask input tokens to generate different sequences. Different from Clark et al. (2018), we take the off-the-shelf outputs of the encoder layers as views which is more general for multi-layer networks without any construction cost.

Consistency regularization. Knowledge distillation (KD) is a typical application of consistency regularization, which achieves knowledge transfer by letting the student model imitate the teacher model (Hinton et al., 2015). There are many ways to construct the student model. For example, the student is the peer model as the teacher in Zhang et al. (2018), and Lan et al. (2018) take one branch as the student in their multi-branch network architecture. As for us, our student model consists of a shallow teacher network in a partially shared manner. Another important application scenario of consistency regularization is semi-supervised learning, such as Temporal Ensembling (Laine and Aila, 2017), Mean Teacher (Tarvainen and Valpola, 2017), Virtual Adversarial Training (Miyato et al., 2017) etc. However, our method works in supervised learning without the requirement of unlabeled data.

6 Conclusion

We studied to incorporate different encoder layers through multi-view learning in neural machine translation. In addition to the primary view from the topmost layer, the proposed model introduces an auxiliary view from an intermediate encoder layer and encourages the transfer of knowledge between the two views. Our method is agnostic to network architecture and can maintain the same inference speed as the original model. We tested our method on five translation tasks with multiple strong baselines: Transformer, deep Transformer, and DynamicConv. Experimental results show that our multi-view learning method can stably outperform the baseline models. Our models have achieved new state-of-the-art results in Ko$\rightarrow$En and IWSLT'14 De$\rightarrow$En tasks.

Acknowledgements

This work was supported in part by the National Science Foundation of China (Nos. 61876035 and 61732005), the National Key R&D Program of China (No. 2019QY1801). Yue Zhang was funded by the machine translation project of Alibaba DAMO Academy.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/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.*

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.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy, July. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium, October-November.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 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*, pages 7955–7965.

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

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *Proc. International Conference on Learning Representations (ICLR).*

Xu Lan, Xiatian Zhu, and Shaogang Gong. 2018. Knowledge distillation by on-the-fly native ensemble. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7528–7538. Curran Associates Inc.

Edson Takashi Matsubara, Maria Carolina Monard, and Gustavo E. A. P. A. Batista. 2005. Multi-view semi-supervised learning: An approach to obtain different views from text datasets. In *Proceedings of the 2005 Conference on Advances in Logic Based Intelligent Systems*, pages 97–104.

Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *5th International Conference on Learning Representations, ICLR 2017*.

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)*, volume 1, pages 2227–2237.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium, November.

Rico Sennrich and Biao Zhang. 2019. Revisiting Low-Resource Neural Machine Translation: A Case Study. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 211–221, Florence, Italy, July.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1195–1204. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Zhe Wang, Songcan Chen, and Daqi Gao. 2011. A novel multi-view learning developed from single-view patterns. *Pattern Recognition*, 44(10):2395–2413.

Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li, and Jingbo Zhu. 2018. Multi-layer representation fusion for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3015–3026.

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, July.

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, July.

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

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *7th International Conference on Learning Representations, ICLR 2019*.

Chang Xu, Dacheng Tao, and Chao Xu. 2013. A survey on multi-view learning. *arXiv preprint arXiv:1304.5634*.

Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *CVPR*.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. 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 898–909, Hong Kong, China, November. Association for Computational Linguistics.

Appendix A. Hyper-parameters setting

Table 6 shows the hyper-parameter list used in our experiments. It can be seen that these training hyper-parameters change across tasks according to model architecture, data scale, and learning difficulty. From the perspective of the model, the Transformer and deep Transformer uses layer normalization in post-norm/pre-norm form, respectively. When using a deep Transformer, we use a deeper encoder and double the batch size to reduce gradient variance with a larger learning rate for fast divergence. Also, we half the number of updates to guarantee the same quantity of training dataset seen for a fair comparison. Note that the decoder depth keeps the same.

Model	**Hyperparameter**	**Ko-En**	**De-En**	**Tr-En**	**Ro-En**	**En-De**
(MV-)Transformer	encoder layer	3	6	6	6	6
	decoder layer	3	6	6	6	6
	batch size	4096	4096	4096	4096*8	4096*8
	learning rate	0.0003	0.0005	0.0002	0.0005	0.0007
	warmup	16k	4k	16k	4k	4k
	update	100k	50k	100k	100k	100k
	dropout	0.3	0.3	0.3	0.3	0.1
	attention dropout	0.1	0.0	0.1	0.0	0.0
	α	0.5	0.3	0.3	0.4	0.4
deep (MV-)Transformer	encoder layer	6	12	12	12	12
	decoder layer	3	6	6	6	6
	batch size	4096	4096*2	4096*2	4096*16	4096*16
	learning rate	0.0004	0.0015	0.001	0.0015	0.002
	warmup	16k	16k	16k	16k	16k
	update	100k	25k	50k	50k	50k
	dropout	0.3	0.3	0.3	0.3	0.1
	attention dropout	0.1	0.1	0.1	0.1	0.1
	α	0.5	0.3	0.3	0.4	0.3

Table 6: Hyper-parameter list for (MV-)Transformer and deep (MV-)Transformer.