Online Distilling from Checkpoints for Neural Machine Translation

Hao-Ran Wei, Shujian Huang ; Ran Wang, Xin-yu Dai and Jiajun Chen

State Key Laboratory for Novel Software Technology Nanjing University Nanjing 210023, China {weihr, huangsj, wangr, daixy, chenjj}@nlp.nju.edu.cn

Abstract

Current predominant neural machine translation (NMT) models often have a deep structure with large amounts of parameters, making these models hard to train and easily suffering from over-fitting. A common practice is to utilize a validation set to evaluate the training process and select the best checkpoint. Average and ensemble techniques on checkpoints can lead to further performance improvement. However, as these methods do not affect the training process, the system performance is restricted to the checkpoints generated in the original training procedure. In contrast, we propose an online knowledge distillation method. Our method on-the-fly generates a teacher model from checkpoints, guiding the training process to obtain better performance. Experiments on several datasets and language pairs show steady improvement over a strong self-attention-based baseline system. We also provide analysis on data-limited setting against over-fitting. Furthermore, our method leads to an improvement on a machine reading experiment as well.

1 Introduction

Neural Machine Translation (NMT) (Cho et al., 2014; Sutskever et al., 2014) has been rapidly developed during the past several years. For further performance improvement, deeper and more expressive structures (Johnson et al., 2017; Barone et al., 2017b; Gehring et al., 2017; Vaswani et al., 2017) have been exploited. However, all of these models have more than hundreds of millions of parameters, which makes the training process more challenging. During the training of NMT models, we notice the following two problematic phenomena: First, the training process is unstable. This is evidenced by the decreasing of training loss with

fluctuate performance on the validation set. Second, the performance on validation set usually begins to worsen after several epochs, while the training loss keeps decreasing, which suggests the model being at risk of over-fitting.

In order to alleviate these issues, the common practice is to periodically evaluate models on a held-out set (with each evaluated model saved as a *checkpoint*). Training is terminated when *m* consecutive checkpoints show no improvement and select the checkpoint with best evaluation score as the final model. Further improvement can be achieved by utilizing more checkpoints, by *smoothing*, which averages these checkpoints' parameters to generate more desirable parameters (Sennrich et al., 2016a); or by *ensemble*, which averages these checkpoints' output probabilities at every step during inference (Chen et al., 2017).

However, we notice that all of these methods have a limitation. Once the training process gets parameters with poor performance, selecting, smoothing or ensemble from the checkpoints in this process may have limited generalization performance as well. We impute the limitation to the "offline" property of these methods. In other words, only employing checkpoints after training cannot affect the original training process.

In this paper, we propose to utilize checkpoints to lead the training process. Our method is carried out in a *knowledge distillation* manner. At each training step, because being evaluated on the heldout validation data, the best checkpoint up to the current training step can be seen as a model with the best generalization ability so far. Therefore, we employ this checkpoint as the teacher model, and let the current training model, as the student, learn from the output probability distributions of the teacher model, as well as truth translations in the training data. Such kind of knowledge distillation is performed on-the-fly because the teacher

^{*}Corresponding Author.

model could always be updated once any latest better checkpoint is generated. We call our method Online Distillation from Checkpoints (ODC).

We conduct experiments on four translation tasks (including two low-resource tasks), and one machine reading comprehension task. All the results demonstrate that our ODC method can achieve improvement upon strong baseline systems. ODC also outperforms checkpoint smoothing and ensemble methods, without extra cost during inference. We can achieve further improvement by combining ODC with those methods.

Major contributions of our work include:

- In contrast to checkpoint smoothing and ensemble which do not affect the training process, we explore the way to distill knowledge from checkpoints to lead the training process in an on-the-fly manner(§3.1, §3.2). We obtain better performance by replacing the best checkpoint with moving average parameters at that step. (§3.3)
- 2. We conduct experiments on four translation tasks, including two low resource tasks. In all the tasks our method outperforms strong baseline systems (§4.2, §4.3). We also conduct an experiment on machine reading comprehension task and the result shows that our method can be applied to other tasks too (§4.4).
- We conduct comprehensive analysis and show that our method can significantly alleviate over-fitting issue in low-resource condition (§5.1), and help to find a wider minimum which brings better generation (§5.2).

2 Background

2.1 Neural Machine Translation

Neural Machine Translation (NMT) systems learn a conditional probability P(Y|X) for translating a source sentence $X = (x_1, ..., x_M)$ to a target sentence $Y = (y_1, ..., y_N)$, in which x_i and y_j are the *i*-th word and *j*-th word in sentence X and Y, respectively. An NMT model usually consists of an encoder (parameterized by θ_{enc}) and a decoder (parameterized by θ_{dec}). The encoder transforms a sequence of source tokens into a sequence of hidden states:

$$H(X) = (h_1, ..., h_M) = f_{\text{enc}}(X; \theta_{\text{enc}}).$$
 (1)

The decoder of NMT is usually a network computing the conditional probability of each target words y_j based on its previous words and the source sentence:

$$p(y_j|y_{< j}, X) \propto \exp(f_{\text{dec}}(y_{< j}, s_j, H(X); \theta_{\text{dec}})),$$
(2)

where s_j is the hidden state of decoder at time step j, p is the distribution of NMT model and θ is all the parameters of NMT model.

The standard way to train an NMT model is to minimize the cross-entropy between the one-hot distribution of the target sentence and the NMT model's output distribution:

$$\mathcal{L}(\theta) = -\sum_{j=1}^{N} \sum_{k=1}^{|\mathcal{V}|} \mathbb{1}\{y_j = k\}$$
(3)

$$\log p(y_j = k | y_{< j}, X; \theta),$$

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta),$$
 (4)

where $\mathbb{1}(\cdot)$ is the indicator function and \mathcal{V} is the target vocabulary.

2.2 Knowledge Distillation in Neural Machine Translation

Knowledge distillation is a class of methods which transfers knowledge from a pre-trained *teacher* model \mathcal{T} , to a *student* model S. The teacher model can be a model with large capacity (Bucila et al., 2006) or an ensemble of several models (Hinton et al., 2015). In knowledge distillation, the student model learns to match the predictions of the teacher model. Concretely, assuming that we learn a classification model (parameterized by θ) on a set of training samples in the form of (x, y) with $|\mathcal{V}|$ classes. Instead of minimizing the cross-entropy loss between one-hot label y and model's output probability $p(y|x;\theta)$, knowledge distillation uses the teacher model's distribution $q(\cdot|x)$ as "soft targets" and optimizes the loss:

$$\mathcal{L}_{\mathrm{KD}}(\theta) = -\sum_{k=1}^{|\mathcal{V}|} q(y=k|x;\theta_{\mathcal{T}}) \qquad (5)$$
$$\log p(y=k|x;\theta),$$

where $\theta_{\mathcal{T}}$ parameterizes the teacher model and $p(\cdot|x)$ is the distribution of the student model.

Kim and Rush (2016) proposed that, as the loss of NMT model (Equation 4) can be factored into minimizing cross-entropy loss between the target



Figure 1: Illustration of online distillation from checkpoints(ODC). Darker color means better performance on validation data. In validation step $T_{k'}$, ODC selects the current best checkpoints as the teacher model; while in the next validation step $T_{k'+1}$, the training generates a better checkpoint, and use it to update the teacher model. In validation step $T_{k'+2}$, training model's performance declines, and the teacher model is used to lead the training of the model. The similar situation happens during the entire training process.

words and word-level probabilities of the NMT model for every position at target side, knowledge distillation on multi-class classification can be naturally applied. They defined word-level knowledge distillation (W-KD) on a sentence as:

$$\mathcal{L}_{\text{W-KD}}(\theta) = -\sum_{i=j}^{N} \sum_{k=1}^{|\mathcal{V}|} q(y_j = k | y_{< j}, H(x); \theta_{\mathcal{T}})$$

$$(6)$$

$$\cdot \log p(y_j = k | y_{< j}, H(x); \theta),$$

where \mathcal{V} is the target vocabulary.

They further proposed sequence-level knowledge distillation (S-KD), which optimizes the student model by matching the predictions of the teacher model in the probability distribution over the space of all possible target sequences:

$$\mathcal{L}_{\text{S-KD}}(\theta) = \sum_{Y \in \tau} q(Y|X; \theta_{\mathcal{T}}) \log p(Y|X; \theta), \quad (7)$$

where τ is the space of target side sentences. As summing over exponential numbers of samples here is intractable, they proposed to train student model on samples generated by teacher model as an approximation.

3 Online Distillation from Checkpoints

3.1 Online Knowledge Distillation

Traditional knowledge distillation maintains a static teacher throughout the training process,

which not only requires one pre-training process to obtain the teacher model, but also limits the power of leading the training process.

In contrast, we are aiming at a more integrated process where the teacher model does not come from a separate training process, but from the current training routine itself. More specifically, we update the teacher along with the training process, so the distilled knowledge could be updated when a stronger model comes out. Figure 1 illustrates the paradigm of our method.

In generation tasks, the knowledge distillation could be performed at the word-level or sequencelevel. In this paper, we focus on the word-level distillation because this distillation only needs forced teaching, which could be performed efficiently together with the training of the student model compared to generating translations from teacher model. It is more computational-friendly, especially when the NMT models are built with parallelizable convolution (Gehring et al., 2017) or self-attention structures (Vaswani et al., 2017).

3.2 Online Distillation from Best Checkpoint

Observed from the training process of NMT models, performance on the validation set does not improve monotonically. When the performance of the training model on the validation set declines, we could always select the best checkpoint so far as the teacher, because it has the best generalization performance. Specially, when the best checkpoint is generated at the current time step, we only update the teacher model but perform no distillation. Figure 1 gives an illustration of this process.

The online distillation process is summarized in Algorithm 1. We use t to denote the training step and θ_t to denote the parameters at time step t. We denote T_k as the time step the k-th time when the model is evaluated on validation and ΔT as the validation interval, for which $T_{k+1} =$ $T_k + \Delta T$. Let \hat{T}_k ($\hat{T}_k \leq T_k$) be the time step when the best checkpoint is obtained up to T_k , and θ_T as the teacher's parameters to lead the following training process. If the current checkpoint is the best checkpoint so far, i.e. $\hat{T}_k = T_k$, we update the teacher to be this new checkpoint $\theta_T = \theta_{\hat{T}_k}$ (in Line 16 and 20). The loss for the training process at time step t ($T_k < t < T_{k+1}$) is defined as follows:

$$\mathcal{L}_{t}(\theta) = \begin{cases} \mathcal{L}(\theta) & \hat{T}_{k} = T_{k} \\ \mathcal{L}(\theta) + \mathcal{L}_{W-\text{KD}}(\theta) & \text{otherwise,} \end{cases}$$
(8)

where $\mathcal{L}(\theta)$ and $\mathcal{L}(\theta)_{W-KD}$ is defined in Equation 4 and 7, respectively (in Line 5-8).

3.3 Integrated with Mean Teacher

Knowledge distillation usually works better when teacher models have better performance. As Tarvainen and Valpola (2017) proposed in their work, averaging model parameters over training steps tends to produce a more accurate model that using final parameters directly. They called this method as *Mean Teacher*.

Following Tarvainen and Valpola (2017), besides updating parameters, we maintain the exponential moving average (EMA) of the model parameters as:

$$\theta_t' = \alpha \theta_{t-1}' + (1 - \alpha) \theta_t, \tag{9}$$

where t is the update step, θ is the parameters of the training model and θ' the parameters of EMA. α is the decay weight which is close to 1.0, and typically in multiple-nines range, i.e., 0.999, 0.9999. By doing so, at each timestep t, parameters of NMT model θ_t has their corresponding EMA parameters θ'_t .

Whenever we update teacher model θ_T with the current best checkpoint, we can use its EMA parameters instead (in Line 17-18). It can further improve the generalization ability of the teacher model, and bring a better performance of knowledge distillation. We will show in §4.2 that using meaning teacher indeed achieves better performance.

Algorithm 1: Online Distillation from Checkpoints

```
1 Input: validation interval \Delta T; validation count k; EMA
       decay weight \alpha; initial model parameters \theta_0
 2 Initialization: k = 0; t = 0; T_0 = -1, \hat{T}_0 = -1;
       \theta_{\mathcal{T}} = \emptyset; \ \theta'_0 = \theta_0; \ \mathcal{L}_0(\theta) = \mathcal{L}(\theta)
    while not reach stopping criteria do
 3
 4
            repeat
                  if \hat{T}_k = T_k then
 5
                         \mathcal{L}_t(\theta) = \mathcal{L}(\theta)
 6
                  else
 7
                     | \mathcal{L}_t(\theta) = \mathcal{L}(\theta) + \mathcal{L}_{W-KD}(\theta) 
 8
                  minimize \mathcal{L}_t(\theta) and update \theta_t;
 9
                  \theta_t' = \alpha \theta_{t-1}' + (1 - \alpha) \theta_t;
10
11
                  t = t + 1;
            until t \mod \Delta T == 0;
12
            T_{k+1} = t;
13
            evaluate on validation set;
14
           if get better checkpoint then
15
                  T_{k+1} = t;
16
                  if use EMA as teacher then
17
                         \theta_{\mathcal{T}} = \theta'_t;
18
                  else
19
                         \theta_{\mathcal{T}} = \theta_t;
20
21
            else
              \hat{T}_{k+1} = \hat{T}_k;
22
            k = k + 1;
23
```

4 **Experiments**

4.1 Setups

To evaluate the effectiveness of our method, we conduct experiments on four machine translation tasks: NIST Chinese-English, WMT17 Chinese-English, IWSLT15 English-Vietnamese, and WMT17 English-Turkish. We conduct experiments based on an open source implementation of Transformer (Vaswani et al., 2017) model in NJUNMT-pytorch¹. For all the translation experiments, we use SacreBLEU² to report reproducible BLEU scores.

We also present an experiment on machine reading comprehension, showing our method could also be applied to other tasks.

Datasets For NIST Chinese-English translation task, training data consists of 1.34M LDC sentence pairs³, with 40.8M Chinese words and 45.8M English words, respectively. We use NIST2003 dataset set as the validation set and NIST 2004,

```
/sockeye_contrib/sacrebleu
```

³The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

¹https://github.com/whr94621/NJUNMT-pytorch ²https://github.com/awslabs/sockeye/tree/master

SYSTEMS	NIST03	NIST04	NIST Chine NIST05	se-English NIST06	Average	Δ
RNNSearch (Zhang et al., 2018b) Transformer-base(Yang et al., 2018)	36.59 42.23	39.57 42.17	35.56 41.02	35.29		
baseline baseline + LKS baseline + BKS baseline + BKE	43.78 44.12 44.23 44.30	44.26 44.87 44.98 45.01	40.97 41.59 41.62 41.86	38.93 39.22 39.74 40.05	41.39 41.89 42.11 42.31	+0.50 +0.73 +0.92
ODC ODC + LKS ODC + BKS ODC + BKE	45.33 45.05 45.35 45.34	45.18 45.49 45.49 45.92	42.60 42.99 43.21 43.35	39.67 40.48 39.96 40.30	42.48 42.99 42.89 43.19	+1.10 +1.60 +1.50 +1.80
ODC-EMA	45.52	45.72	43.01	40.65	43.13	+1.74

Table 1: Case-insensitive BLEU scores of Chinese-English translation on NIST datasets. "Average" means average scores on NIST04, 05 and 06.

2005, 2006 as test sets. We filter out sentence pairs whose source or target side contain more than 50 words. We use BPE (Sennrich et al., 2016b) with 30K merge operations on both sides.

For WMT17 Chinese-English translation task, we use the pre-processed version released by WMT⁴. We only use CWMT part of WMT Corpus. We use newsdev2017 as the validation set and newstest2017 s the test set. We learn a BPE model with 32K merge operations and keep all the BPE tokens in the vocabulary. We limit the maximal sentence length as 100 after BPE segmentation.

For IWSLT15 English-Vietnamese translation task, we directly use the pre-processed data used in Luong and Manning (2015)⁵, which has 133K sentence pairs, with 2.70M English words and 3.31M Vietnamese words. We use the released validation and test set, which has 1553 and 1268 sentences respectively. Following the settings in Huang et al. (2017), the Vietnamese and English vocabulary size are 7,709 and 17,191, respectively.

For WMT17 English-Turkish translation task, We use the pre-processed data released by WMT17⁶. It has 207K sentence pairs, with 5.21M English words and 4.63 Turkish words. We use newstest2016 as our validation set and newstest2017 as the test set. We use joint BPE segmentation (Sennrich et al., 2017) to process the whole training data. The merge operations are 16K. **Implementation Details** Without specific statement, we follow the *transformer_base_v1* hyperparameters settings ⁷, with 6 layers in both encoder and decoder, 512 hidden units and 8 attention heads in multi-head attention mechanism and 2048 hidden units in feed-forward layers. Parameters are optimized using Adam(Kingma and Ba, 2014). The initial learning rate is set as 0.1 and scheduled according to the method proposed in Vaswani et al. (2017), with warm-up steps as 4000.

We periodically evaluate the training model on the validation set by doing translation and compute the BLEU scores. We stop training when 50 subsequent of BLEU scores on validation set do not get improvement. We use beam search with beam size as 5.

4.2 Evaluation on Chinese-English Translation Tasks

We first evaluate the capability of our method for improving performance when there are plenty of training data. We conduct experiments on both NIST and WMT17 Chinese-English Translation tasks.

Results on NIST Dataset We compare our method with several ways to utilize checkpoints⁸:

- **last-k-smoothing**: After training the baseline model, we average the parameters of the last k checkpoints as the final model.
- **best-k-smoothing**: Average the parameters of the best k checkpoints, instead of the last k,

⁴http://data.statmt.org/wmt18/translationtask/preprocessed/zh-en/

⁵https://github.com/tefan-it/nmt-en-vi

⁶http://data.statmt.org/wmt17/translation-

task/preprocessed/tr-en/

⁷https://github.com/tensorflow/tensor2tensor/blob/v1.3.0/ tensor2tensor/models/transformer.py

⁸We set k = 5 in this experiments.

as the final model. In this case, checkpoints may have better performance but higher variance which could be harmful to parameters averaging.

• **best-k-ensemble**: Do ensemble inference (average the output probabilities) with the best k checkpoints (Chen et al., 2017).

As shown in Table 1, our baseline is comparable to the other two recent published results (Zhang et al. (2018b), Yang et al. (2018)). In consistent with Chen et al. (2017), using checkpoints for smoothing or ensemble does improve the baseline system. Using EMA parameters also improve the baseline system as well, which is in consist with (Tarvainen and Valpola, 2017).

Compared to the baseline, our approach ODC brings translation improvement across different test sets and achieves 42.48 BLEU scores on average(+1.09 BLEU v.s. baseline). This result confirms that using best checkpoint as teacher indeed helps improving the performance of the translation model.

Besides, ODC is comparable to the best results among smoothing and ensemble on baseline's checkpoints (achieved by best-k-ensemble). Considering that best-k-ensemble needs to decode with k models, while ODC decodes only one, our model enjoys a better efficiency. Furthermore, we can achieve further improvement by combining these methods on checkpoints generated by ODC.

Results also show that ODC-EMA (§3.3) could achieve additional improvement from ODC itself (43.13 v.s. 42.48 BLEU), demonstrating that using EMA of the best checkpoint instead can bring better knowledge distillation performance, as it generates a better teacher model.

Results on WMT17 Dataset We present the results on WMT17 Chinese-English translation task in Table 2. We report the results of the baseline, ODC and a recent result published by Zhang et al. (2018c). To make a fair comparison, we follow the experiment setting in Zhang et al. (2018c). The experiment results show similar trends with those on the NIST datasets. Applying ODC leads to the result of 24.22 BLEU, which is 0.85 BLEU higher compared with baseline.

4.3 Evaluation on Low-resource Scenario

We also apply our method to two low resource translation tasks, i.e., IWSLT2015 English-Vietnamese

SYSTEM	newsdev2017	newstest2017		
Zhang et al. (2018c) baseline ODC	21.96 22.24	23.01 23.37 24.22		

Table 2: Case-sensitive BLEU scores on WMT17Chinese-English Translation

(EN2VI) and WMT17 English-Turkish (EN2TR). Due to the limited amount of training data, models are more likely to suffer from over-fitting. Therefore, we use a higher dropout rate of 0.2 and weight decay, another common technique against overfitting, with decay weight set as 10^{-3} as the default setting. We implement weight decay as AdamW (Loshchilov and Hutter, 2017) does.

Besides, we further experiment with grid search on the validation set for optimal hyper-parameters of dropout rate and weight decay, which may lead to better results. We adopt a simple heuristic, which first searches an optimal dropout rate, and then further searches weight decay coefficients based on this dropout. We experiment with dropout as 0.2, 0.3, 0.4, and weight decay as $10^{-1}, 10^{-2}$ and 10^{-3} .

SYSTEMS	EN2VI	EN2TR
Zhang et al. (2018a) tensor2tensor	- 28.43	12.11
baseline	28.56	12.20
baseline w/ grid-search	29.01	12.51
ODC	29.47	12.92
ODC w/ grid-search	29.59	13.18

Table 3: Case-sensitive BLEU scores on two low resource translation tasks.

As in Table 3, our baseline is comparable to two recent published results, respectively: EN2TR from Zhang et al. (2018c) and EN2VI from official release tensor2tensor problem⁹. Grid hyperparameter search does improve the baseline system. ODC leads to better results compared to the baseline, as well as the baseline with grid parameter search. ODC can achieve further improvement after searching for optimal hyper-parameters of dropout and weight decay.

⁹https://github.com/tensorflow/tensor2tensor/pull/611

4.4 Evaluation on Machine Reading Comprehension

Although our main research is focused for the task of machine translation, the idea of ODC could be applied to other tasks as well. We experiments on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), a machine reading comprehension task.

SQuAD contains 107,785 human-generated reading comprehension questions, with 536 Wikipedia articles. Each question is associated with a paragraph extracted from an article, and the corresponding answer is a span from this article. A machine reading comprehension model is designed to predict the start and end positions in the article of the answer.

The state-of-the-art machine reading comprehension system also employs a deep neural network structure, which is similar to NMT. We apply our ODC method on **BiDAF++** (Choi et al., 2018), a multi-layer SQuAD model that augments BiDAF (Seo et al., 2016) with self-attention and contextualized embeddings. We evaluate the model after each epoch and implement the knowledge distillation by teaching the student with the output distribution of answer start and end positions predicted by the best checkpoint.

For the results, ODC improves a base BiDAF++ from 76.83 to 77.40, in EM scores, showing that our method can be applied to a broader range of tasks.

5 Analysis

We conduct further analysis to probe into the reasons for the advantages of ODC. We first show that our method can significantly alleviate the overfitting issue in data-limited condition. After that, we show that parameters gained from our method tend to be wider minimums, which represents better generalization.

5.1 Effectiveness on reducing over-fitting

Taking IWSLT15 English-Vietnamese as a test-bed, we analyze whether our method could help handle the over-fitting issue. We first plot the curve of the loss on the validation set at each training step for the different models (in Figure 2, the top curve with rounds). It is easy to see that the loss curve of the baseline increases as the training goes after 50K steps, indicating a severe over-fitting. With better dropout rate and weight decay, the over-fitting is



Figure 2: Loss curves (top) and final BLEU scores (bottom) on the validation set of baseline, baseline with grid-search and ODC, respectively.

less severe; while with ODC the loss curve shows a more steady trend of decrease, and is almost always under the other two's.

The final BLEU score on the validation set (Figure 2, bottom) shows corresponding result. The grid search of hyper-parameters improves the BLEU from 26.06 to 26.42 in BLEU, while ODC achieves 26.99.

Both results indicate that our method is more effective at handling the over-fitting problem. We hold that minimizing the cross-entropy between the teacher model and the student model serves as regularization to the training of the student model, which avoids the model getting into over-fitting.

5.2 ODC Brings Wider Minimum

In the training process in Chinese-English tasks, we do not observe obvious over-fitting issue as shown in low resource translation tasks. In this section, we analyze how ODC helps the model generalization. Keskar et al. (2016) proposed that the width



Figure 3: The upper plot shows the validation losses curve along the line segment decided by parameters of baseline and ODC. The bottom plot shows the standard deviations within the neighborhood of baseline and ODC at different distances.

of the minimum in a loss surface is related to its generalization ability. Therefore, we compare the generalization capability between baseline system and our ODC method by exploring around the parameters.

We make use of the visualization technique employed in (Goodfellow and Vinyals, 2014) and analyze the results on the NIST data set. Let θ_{base} and θ_{ODC} denote the final parameters obtained from baseline and ODC. Consider the line:

$$\theta(\alpha) = \alpha \cdot \theta_{\text{ODC}} + (1.0 - \alpha) \cdot \theta_{\text{base}},$$
 (10)

which connects θ_{base} ($\alpha = 0.0$) and θ_{ODC} ($\alpha = 1.0$). We plot the value of Equation 4 as a function of α (normalized by count of words per sentence) with $\theta = \theta(\alpha)$. We draw α from -1.0 to 2.0 at an interval of 0.02. In this way, the width of θ_{base} and θ_{ODC} can be represented as the steepness of the curve nearby. To further quantitatively represent the steepness, we compute the standard deviation of values on this curve within different distances to the two parameters, respectively. We plot them in

Figure 3.

From Figure 3 we can see that the loss curve behaves steeper around the parameters of baseline than of ODC. Besides, the standard deviations of losses around the baseline model are consistently higher than ODC within all the distances. It is evident that the parameters of ODC act as a wider minimum c and explains why ODC can lead to a more generalized model.

6 Related Works

6.1 Regularization in NMT

Regularization has broad applications in training NMT models to improve performance and avoid over-fitting. There are some common regularization techniques, such as L^2 normalization and dropout (Srivastava et al., 2014). These methods are simple and easy to implement but need carefully tuning on the validation set. These methods are also orthogonal to our method.

There are also some works to exploit regularization techniques in fine tuning of NMT model. Barone et al. (2017a) proposed a tuneout method which randomly replaces columns of weight matrices of out-of-domain parameter matrices. Khayrallah et al. (2018) shared similar training object with us, as they computed the KL divergence between out-of-domain and in-domain model. Both of their works request a pre-trained teacher model, while we are work on a more general training problem which does not require such kind of model.

6.2 Online Knowledge Distillation

While traditional knowledge distillation requires a static, pre-trained teacher model, online knowledge distillation tends to overcome this problem by selecting or generating a teacher dynamically from scratch.

To the best of our knowledge, Zhang et al. (2017) is the first trial to replace the offline teacher model. They trained peer models to teach each other simultaneously. Compared to their work, our method uses the best checkpoint as the teacher, which avoids introducing extra parameters. Furlanello et al. (2018) tends to update teacher model during the training procedure iteratively, but their method needs to train the teacher model until convergence in each iteration. Instead, our method only needs one phase of training, whose overhead is relatively small.

Lan et al. (2018) using an ensemble of several branches of the model as teacher for computer vision tasks, which only needs one-phase training as well. However, their method relies heavily on the multi-branch structures of the tasks, which are not widely applicable in neural machine translation.

7 Conclusion

In this paper, we propose an online knowledge distillation method with the teacher model generated from checkpoints during the training procedure. Experiments on four machine translation tasks and a machine reading task show that our method outperforms strong baseline systems. Further analysis shows that our method can effectively alleviate the over-fitting issue, and tend to find a wider minimum.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful comments. We also thank Boxing Chen from Alibaba Group for his helpful comments. This work is supported by the National Science Foundation of China (No. 61772261, 61672277) and the Jiangsu Province Research Foundation for Basic Research (No. BK20170074). Part of this work is supported by "13th Five-Yea" All-Army Common Information System Equipment Pre-Research Project (No. 31510040201).

References

- Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017a. Regularization techniques for fine-tuning in neural machine translation. In *EMNLP 2017*, pages 1489–1494.
- Antonio Valerio Miceli Barone, Jindrich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017b. Deep architectures for neural machine translation. In *WMT 2017*.
- Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.
- Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. Checkpoint ensembles: Ensemble methods from a single training process. *CoRR*, abs/1710.03282.
- Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP 2014*.

- Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke S. Zettle-moyer. 2018. Quac: Question answering in context. In *EMNLP*.
- Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*.
- Ian J. Goodfellow and Oriol Vinyals. 2014. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.
- Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR*, abs/1706.05565.
- Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL 2017*, 5:339–351.
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836.
- Huda Khayrallah, Brian Thompson, Kevin Duh, and Philipp Koehn. 2018. Regularized training objective for continued training for domain adaptation in neural machine translation. In *WMT 2018*, pages 36–44.
- Yoon Kim and Alexander M. Rush. 2016. Sequencelevel knowledge distillation. In *EMNLP*.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- Xu Lan, Xiatian Zhu, and Shaogang Gong. 2018. Knowledge distillation by on-the-fly native ensemble. *CoRR*, abs/1806.04606.
- Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.
- Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *IWSLT 2015*, Da Nang, Vietnam.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

- Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of edinburgh's neural MT systems for WMT17. In *WMT 2017*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In ACL 2016, pages 371–376.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL 2016*.
- Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*, pages 3104–3112.
- Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS 2017*, pages 1195–1204.
- 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 *NIPS*.
- Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *NAACL-HLT*.
- Biao Zhang, Deyi Xiong, and Jinsong Su. 2018a. Accelerating neural transformer via an average attention network. In *ACL*.
- Xiangwen Zhang, Jinsong Su, Yue Qin, Yang Liu, Rongrong Ji, and Hongji Wang. 2018b. Asynchronous bidirectional decoding for neural machine translation. In *AAAI*.
- Ying Zhang, Tao Xiang, Timothy M. Hospedales, and Huchuan Lu. 2017. Deep mutdual learning. *CoRR*, abs/1706.00384.
- Zhirui Zhang, Shuangzhi Wu, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2018c. Regularizing neural machine translation by target-bidirectional agreement. *CoRR*, abs/1808.04064.