Multi-Agent Mutual Learning at Sentence-Level and Token-Level for Neural Machine Translation

Baohao Liao **Yingbo Gao** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
baohao.liao@rwth-aachen.de
{ygao|ney}@cs.rwth-aachen.de

Abstract

Mutual learning, where multiple agents learn collaboratively and teach one another, has been shown to be an effective way to distill knowledge for image classification tasks. In this paper, we extend mutual learning to the machine translation task and operate at both the sentence-level and the token-level. Firstly, we co-train multiple agents by using the same parallel corpora. After convergence, each agent selects and learns its poorly predicted tokens from other agents. The poorly predicted tokens are determined by the acceptance-rejection sampling algorithm. Our experiments show that sequential mutual learning at the sentence-level and the token-level improves the results cumulatively. Absolute improvements compared to strong baselines are obtained on various translation tasks. On the IWSLT'14 German-English task, we get a new state-of-the-art BLEU score of 37.0. We also report a competitive result, 29.9 BLEU score, on the WMT'14 English-German task.

1 Introduction

Neural machine translation (NMT) has achieved significant progress over recent years (Sutskever et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017; Vaswani et al., 2017; Edunov et al., 2018). Conventional training of the NMT models with hard targets limits the models' generalization ability (Szegedy et al., 2016; Pereyra et al., 2017). This has led to a rapid growth of research in developing more regularized models. Teacher-student (T/S) learning (Li et al., 2014; Hinton et al., 2015; Meng et al., 2019) is an effective method to handle this problem. It has been widely applied in many cases, e.g. model compression (Li et al., 2014; Hinton et al., 2015), domain adaptation (Li et al., 2017; Meng et al., 2018) and low-resource machine translation (Chen et al., 2017).

T/S learning is a strategy that trains a student model with both hard targets and soft posteriors produced by a pre-trained teacher model (Li et al., 2014). Because training with soft targets provides smoother output distribution, T/S learning could outperform the single model training (Li et al., 2014; Hinton et al., 2015; Meng et al., 2018).

However, does a teacher always outperform a student? In order to evaluate the pros and cons of different models, we conduct experiments on two different architectures. Table 1 shows the translation quality from various models. Arch_1 outperforms Arch_2 for the translation tense, whereas Arch_2 outperforms Arch_1 for certain word translation. Besides, the same architecture but initialized differently also has tiny translation differences. This phenomenon shows that a certain architecture may not always be suitable to be a teacher.

In this paper, we propose a two-step multi-agent mutual learning scheme, where one agent learns from other agents at the sentence-level as the first step, which we call sentence-wise mutual learning. When it becomes "smart", as the second step it will compare its own predicted tokens with other agents and only learn those poorly predicted tokens, which we call token-wise mutual learning. Mutual learning is first proposed by Zhang et al. (2018) for image classification tasks. Compared to T/S learning, there is no fixed teacher model. The co-trained agents are teachers to one another.

For sentence-wise mutual learning, each agent learns from other agents at sentence-level. When all of the agents converge after the first step, they continue to learn from other agents at the token-level. Each agent selects and learns the tokens that it predicts poorly. The poorly predicted tokens are determined by acceptance-rejection sampling (Chib and Greenberg, 1995). For every two agents, the target tokens can be divided into two subsets, where one agent performs poorly in one subset and

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1715–1724
November 16 - 20, 2020.

Src	die evolution ist ein andauerndes thema hier auf der ted-konferenz gewesen, aber heute möchte ich ihnen die ansicht eines zu dem thema geben.
Ref	evolution has been a perennial topic here at the ted conference, but i want to give you today one doctor's take on the subject.
Arch_1 (Init_1)	evolution has been a serious topic here at the ted conference, but i want to give you today the view of an ark on the subject.
Arch_1 (Init_2)	evolution has been a severe subject here at the ted conference, but today i want to give you the view of a doctor.
Arch_2	now, evolution is a continuous topic in the ted conference today, but today i want to give you the view of a doctor on the subject.

Table 1: Arch_1 and Arch_2 denote Transformer (Vaswani et al., 2017) and ConvS2S (Gehring et al., 2017), respectively. Init_1 and Init_2 denote two random initialization. The models with different architectures tend to translate diversely. The models with the same architecture but initialized differently have tiny differences.

learn those tokens from the other agent.

We train our agents on small-scale IWSLT'14 German-English and IWSLT'14 Dutch-English, middle-scale WMT'16 Romanian-English and large-scale WMT'14 English-German datasets. We obtain significant improvements compared to strong baselines. Up to $+2.3$, $+2.2$, $+2.0$ and $+1.6$ absolute BLEU scores are achieved on these four tasks.

To the best of our knowledge, this is the first work using multi-agent mutual learning for NMT tasks. The token-level knowledge distillation is also applied for the first time.

Our contributions are summarized as follows:

- We extend mutual learning to MT tasks and develop a sentence-level and token-level training scheme. Performance is improved significantly and consistently on various MT tasks.
- We compare our method with the similar training method, i.e. T/S learning and conditional T/S learning (Meng et al., 2019), and provide theoretical insights and practical evidences why our method performs well.
- We further delve into the effect of various factors, including the architecture diversity, different methods for interpolation weight and the number of co-trained agents.

2 Related Work

T/S Learning Knowledge distillation is first introduced by Buciluǎ et al. (2006) and re-gains popularity due to Li et al. (2014) and Hinton et al. (2015). Currently, T/S learning and its variants can be roughly divided into two paradigms: a fixed pre-trained teacher model (Li et al., 2014; Hinton et al., 2015; Meng et al., 2019) and a dynamic co-trained teacher model (Zhang et al., 2018; Bi et al., 2019). For the former, the student learns from both hard targets and soft posteriors generated by a fixed teacher. For the latter, multiple co-trained students are considered as a teacher to one another, also known as mutual learning (Zhang et al., 2018). Alternatively, an ensemble integrated by multiple co-trained agents can also be treated as a teacher to all agents (Bi et al., 2019).

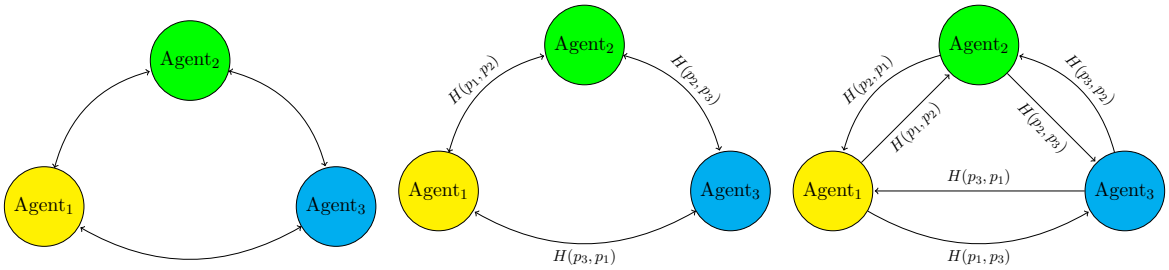
Dual Learning Dual learning (He et al., 2016) or multi-agent dual learning (Wang et al., 2018) is to leverage the duality between the primal task and the dual task. The source and target domains for these two tasks are opposite. Even though both dual learning and mutual learning introduce extra models compared to traditional training method, the source and target sentences for all agents in mutual learning stay the same. There is only one task for mutual learning, i.e. translation from source sentences to target sentences.

MT at Sentence-Level and Token-Level Chen et al. (2017) propose a training method at sentence-level and token-level for pivot-based zero-resource NMT. However, we have different definitions for the sentence-level and token-level translation. Both the sentence-level and token-level translation in Chen et al. (2017) are considered as the sentence-level translation in our work. The token-level translation in this paper means one agent only learns the poorly predicted tokens from other agents.

3 Approach

3.1 General Mutual Learning

We consider a parallel sentence pair: a source sentence f_1^J with sentence length J, a target sentence



(a) General Mutual Learning (b) Sentence-wise Mutual Learning (c) Token-wise Mutual Learning

Figure 1: Two-step mutual learning with three agents. The direction of arrow denotes the direction of knowledge distillation. (a): General mutual learning schematic, each agent learns from all other agents. (b): Sentence-wise mutual learning is the first training step. Without loss of generality, we assume Agent_1 performs best and Agent_3 performs worst for a certain training step here. Bidirectional arrows denote that these two agents distill knowledge to each other for the whole sentence with the same cross entropy loss, i.e. symmetric SML. (c): The second training step is token-wise mutual learning. Each agent learns its poorly predicted tokens from other agents. Unidirectional arrows denote that one agent is only a teacher for another agent in one subset of the tokens.

e_1^I with sentence length I. Indexed target token e_i takes value from $\{1, 2, ..., V\}$ in the target vocabulary, whose size is V. The probability of the token $\hat{e}_i$ being generated is conditioned on the whole source sentence f_1^J and the previously generated tokens $\hat{e}_1^{i-1}$:

$$p(\hat{e}_i) := p(\hat{e}_i|\hat{e}_1^{i-1}, f_1^J) \tag{1}$$

The conventional training criterion is cross entropy. For a given sentence pair, we minimize the cross entropy loss between the empirical distribution and the model distribution p, which can be written as:

$$\mathcal{L} = -\sum_{i=1}^{I}\sum_{v=1}^{V} \mathbb{1}\{e_i = v\} \log p(\hat{e}_i = v) \tag{2}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function.

The objective function only takes care of the probabilities of target tokens and omit the probabilities of rival tokens according to Equation (2), where no explicit regularization is introduced. One could make the model generalize better by discounting a certain probability mass from the one-hot target distribution and interpolate with a uniform prior over the vocabulary (Szegedy et al., 2016; Pereyra et al., 2017; Gao et al., 2020a,b), which is also known as label smoothing. Then the loss function is:

$$\mathcal{L} = -\sum_{i=1}^{I}\sum_{v=1}^{V} pr(e_i) \log p(\hat{e}_i = v) \tag{3}$$

with:

$$pr(e_i) = \begin{cases} 1-\alpha & , \text{if } e_i = v \\ \frac{\alpha}{V-1} & , \text{otherwise} \end{cases} \tag{4}$$

with the discounted probability mass α, where $0 \leq \alpha \leq 1$. Empirically, we choose $\alpha = 0.1$. Compared to using hard targets, label smoothing assigns some probability mass to the rival labels.

Apart from label smoothing, mutual learning (ML) (Zhang et al., 2018) is another method to regularize the models. Multi-agent ML with three agents is illustrated in Figure 1a. Some studies have shown that one agent could perform better by learning soft posteriors from other agents (Li et al., 2014; Hinton et al., 2015; Meng et al., 2018). This is because soft posteriors provide smoother distribution than hard targets.

Building on top of ML, we propose a two-step ML method: sentence-wise mutual learning and token-wise mutual learning. Firstly, we co-train multiple agents with sentence-wise mutual learning until convergence. Each agent learns from both hard targets and soft posteriors at sentence-level. The agents are then again co-trained with token-wise mutual learning until convergence. Even though they still learn from both hard and soft targets, they only learn their poorly predicted tokens from other agents. With both sentence-wise mutual learning and token-wise mutual learning, we can improve the performance cumulatively.

3.2 Sentence-wise Mutual Learning

Sentence-wise mutual learning (SML) with three agents is illustrated in Figure 1b. The cross entropy loss between empirical distribution and model distributions, and among different model distributions are minimized together with different interpolation weights. So each agent learns from both hard targets and soft posteriors.

Each agent sees the same source sentence and the same target sentence for each step. Suppose we have K agents (with the same or different architectures) and $p_1, p_2, ..., p_K$ are the probability distributions of a certain time step for each agent. $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_K$ are the label smoothing losses (see Equation (3)) for each agent. We introduce an extra loss between different agents:

$$\mathcal{L}_{k,l}^{\text{SML}} = \begin{cases} H(p_k, p_l) & , \text{ if } \mathcal{L}_k \leq \mathcal{L}_l \\ H(p_l, p_k) & , \text{ otherwise} \end{cases} \quad (5)$$

with cross entropy loss $H(\cdot, \cdot)$:

$$H(f, g) = -\sum_{i=1}^{I}\sum_{v=1}^{V} f(\hat{e}_i = v)\log g(\hat{e}_i = v) \quad (6)$$

So the better performing agent is always used as f and serves as a teacher. The overall loss function of k^{th} agent is:

$$\mathcal{L}_{\text{total}}^{k} = \lambda\mathcal{L}_k + (1-\lambda)\frac{1}{K-1}\sum_{l=1,l\neq k}^{K}\mathcal{L}_{k,l}^{\text{SML}} \quad (7)$$

with interpolation weight λ, where $0 \leq \lambda \leq 1$. The interpolation weight could be a static hyper-parameter that stays the same for the whole SML procedure or a dynamic hyper-parameter which decreases for each epoch:

$$\lambda = 0.5 + \beta\frac{0.5}{n} \quad (8)$$

with the number of training epochs n and the decreasing rate β, where $0 < \beta \leq 1$.

The interpolation weight λ is always larger than 0.5 and decreases for the whole SML procedure according to Equation (8). So the agent learning focuses more on the soft posteriors as the training progresses. The motivation for this is that soft posteriors from agents contain little useful knowledge about the data at the beginning of training. As training goes on, they learn information from hard targets and preserve more useful information. For the static interpolation weight λ, we suggest to set it larger than 0.5, so the agents can learn more from hard targets than from other agents.

Each agent learns from all other agents, even though some agents perform worse than it (see Equation (5) and (7)). Zhang et al. (2018) propose an asymmetric learning method for image classification, i.e. other agents are always used as f in Equation (6). However, learning from better agents and from worse agents is symmetric in our work, i.e. the better performing agent is always used as f and the worse performing agent is used as g. Empirically, we obtain better results with such symmetric learning for machine translation tasks.

3.3 Token-wise Mutual Learning

After SML, each agent becomes "smart". Instead of learning from other agents at sentence-level, they only learn the poorly predicted tokens from other agents. The token-wise mutual learning (TML) scheme is illustrated in Figure 1c.

Algorithm 1 Acceptance-Rejection Sampling

Input: Parallel sentence (f_1^J, e_1^I)

for $i \leftarrow 1$ to I **do**
  $\gamma_i \leftarrow \frac{p_k(\hat{e}_i = e_i)}{c \cdot p_l(\hat{e}_i = e_i)}$
  $u_i \sim U(0, 1)$
  if $u_i \leq \gamma_i$ **then**
    $i \in S_{l,k}$
  else
    $i \in S_{k,l}$
  end if
end for

Inspired by the acceptance-rejection sampling method (Chib and Greenberg, 1995), the poorly predicted tokens are determined by the probability ratios, γ_i, of the target tokens between two agents as in Algorithm 1. If value u_i obtained by uniform sampling fulfills $u_i \leq \gamma_i$, we consider agent_k to be performing better on the target token e_i than agent_j. Then for this time step, agent_j needs to learn from agent_k. Normally, we set the scale factor c as:

$$c = \max_{i \in \{1,2,...,I\}} \frac{p_k(\hat{e}_i = e_i)}{p_l(\hat{e}_i = e_i)} \quad (9)$$

With the acceptance-rejection sampling method, we obtain two target token subsets for each parallel sentence pair between agent_k and agent_l: $S_{k,l}$ and $S_{l,k}$. Agent_k predicts poorly in the subset $S_{k,l}$ and needs to learn these tokens from agent_l. Agent_l predicts poorly in the subset $S_{l,k}$ and needs to learn these tokens from agent_k.

Different from SML, each agent only learns its poorly predicted tokens from other agents for TML. Other agents are always used as f in Equation (6).

	De-En	Nl-En	Ro-En	En-De
# train	153k	153k	612k	4.5M
# val.	7k	7k	2k	3k
# test	7k	5k	2k	3k
# voc.	10k	10k	20k	32k

Table 2: The amount of parallel sentence pairs and vocabulary sizes for IWSLT'14 De-En, IWSLT'14 Nl-En, WMT'16 Ro-En and WMT'14 En-De. Val. denotes validation set. Voc. denotes vocabulary size.

The extra loss function is defined as:

$$\mathcal{L}_{k,l}^{\mathrm{TML}} = -\sum_{i \in S_{k,l}} \sum_{v=1}^{V} p_l(\hat{e}_i = v) \log p_k(\hat{e}_i = v) \tag{10}$$

The overall loss definition for agent_k at this step stays the same as Equation (7) for SML (replace $\mathcal{L}_{k,l}^{\mathrm{SML}}$ by $\mathcal{L}_{k,l}^{\mathrm{TML}}$). Since all agents preserve much more useful information than hard targets after the convergence for SML, they simply need to be fine-tuned for some iterations with TML. All agents are also more stable and reliable after SML, we only need to use static interpolation weight and set $\lambda < 0.5$. So the learning focus more on the soft posteriors than the hard targets.

4 Experimental Setup

Datasets In this paper, we run experiments on multiple benchmark MT datasets to evaluate the effectiveness of the proposed method, including IWSLT'14 German-English (De-En), IWSLT'14 Dutch-English (Nl-En), WMT'16 Romanian-English (Ro-En) and WMT'14 English-German (En-De). The amount of parallel sentence pairs for different MT tasks is shown in Table 2.

Following Edunov et al. (2017), we split IWSLT'14 De-En, IWSLT'14 Nl-En and WMT'16 Ro-En datasets into various amounts of parallel sentence pairs for training, validation and testing. On WMT'14 En-De, we choose the WMT'16 training set and sample 4.5M parallel sentence pairs for training (Ott et al., 2018), employ newstest 2013 as the validation set and use newstest 2014 as the testing set (Vaswani et al., 2017). For all language pairs, we use byte-pair encoding (Sennrich et al., 2015) with shared vocabularies.

Model Architecture We mainly employ three types of the Transformer model (Vaswani et al., 2017), i.e. Transformer_small, Transformer_base and Transformer_big, implemented in the fairseq-py toolkit (Ott et al., 2019). Transformer_base and Transformer_big stay the same as Vaswani et al. (2017). The difference between Transformer_small and Transformer_base is that each encoder and decoder layer in Transformer_small employs a word representation size of 512, a feed-forward layer dimension of 1024 and 4 attention heads.

Transformer_small is used for the small-scale IWSLT'14 De-En and IWSLT'14 Nl-En datasets with a dropout rate of 0.3. Transformer_base is applied for the middle-scale WMT'16 Ro-En and large-scale WMT'14 En-De with a dropout rate of 0.1. Transformer_big is also employed for the large-scale WMT'14 En-De with a dropout probability of 0.3. We also train a convolutional sequence to sequence network (ConvS2S) (Gehring et al., 2017) and a seven encoder and decoder layer Transformer_small (Transformer_small7) on IWSLT'14 De-En and IWSLT'14 Nl-En as our baselines.

Optimization and Evaluation We use the same settings for the optimization and the learning rate decay rule as Vaswani et al. (2017) for Transformer_small, Transformer_small7, Transformer_base and Transformer_big with an initial learning rate of 5e-4. We use a batch size (the number of tokens) of 4K for Transformer_small and Transformer_small7, a batch size of 25K for Transformer_base and Transformer_big. If the batch size can not be set as the number above because of memory limit, we accumulate gradients to match it. We use the same settings for the optimization and learning rate as Gehring et al. (2017) for ConvS2S with a batch size of 4K. We use beam search with a beam size of five and length penalty of 0.6 to generate translations for all of the models. The evaluation metric is BLEU (Papineni et al., 2002).

For IWSLT'14 De-En and IWSLT'14 Nl-En, we use a single Nvidia GTX 1080Ti GPU to train 2, 3 and 4 co-trained Transformer_small for 1.5, 2 and 3 days, respectively. For WMT'16 Ro-En, we use a single GPU to train 2, 3 and 4 co-trained Transformer_base for 2, 3.5 and 4.5 days, respectively. For WMT'14 En-De, we use four GPUs to train 2, 3 and 4 Transformer_base for 10, 13 and 18 days, respectively. For WMT'14 En-De, we also use four GPUs to train 2, 3 and 4 Transformer_big for 12, 18 and 21 days, respectively.

Method	IWSLT'14	
	De-En	Nl-En
Transformer_small	34.7	37.7
T/S	35.0	38.1
Conditional T/S	34.2	37.0
Asymmetric SML	35.3	38.4
Symmetric SML	**36.2**	**38.9**

Table 3: BLEU scores for different training methods with two agents. Both agents are Transformer_small. T/S and conditional T/S: The scores are from the student model. Asymmetric and symmetric SML: The scores are from the best co-trained agent.

Model	IWSLT'14	
	De-En	Nl-En
a. ConvS2S	32.8	35.5
b. Transformer_small	34.7	37.7
c. Transformer_small7	34.9	37.8
a / b	30.8/34.5	33.8/35.5
b / c	34.4/33.9	35.4/34.9
b $\times$ 2	36.0/**36.2**	**38.9**/38.4

Table 4: BLEU scores for two-agent SML with different or identical architectures.

5 Results

SML and TML are conducted sequentially. Firstly, we train our multiple agents at the sentence-level until convergence. Secondly, we train again the agents with the best performance from SML at the token-level till convergence.

5.1 Results for SML

Different Training Methods T/S learning and its variants, i.e. Conditional T/S learning (Meng et al., 2019) and asymmetric ML (Zhang et al., 2018), perform well on various tasks. We reproduce these methods on our tasks to study the effectiveness of SML. As shown in Table 3, our proposed method (symmetric SML) outperforms other methods and achieves +1.5 and +1.2 BLEU scores on IWSLT'14 De-En and IWSLT'14 Nl-En, respectively.

T/S and conditional T/S employ a fixed pre-trained teacher model. Empirically, the size of the teacher model need to be much bigger than the student model to obtain a better student model. In our case, we observe there is no significant improvement when the teacher and student share the same architecture for T/S and conditional T/S (see Table 3). Compared to asymmetric SML, symmetric SML obtain +0.9 and +0.5 BLEU scores, which is different from the results reported in Zhang et al. (2018), where they obtain similar results from asymmetric and symmetric ML on image classification tasks.

Agents with Different or Identical Architectures We assess the impact of the architecture diversity of agents. From Table 4, we observe that the agents with the identical architecture outperform the agents with different architectures. For the co-training of the agents with different architectures, similar architectures (Transformer_small and Transformer_small7) or completely different architectures (ConvS2S and Transformer_small), the results even get worse than single model training. This is different from Bi et al. (2019), where better results are reported for the teacher model integrated by students with different architectures. We believe the reason is: Bi et al. (2019) first pre-train multiple students independently and co-train them as the second step. All of the student models have converged after the first step. They are only fine-tuned with the second step. In this paper, we co-train multiple agents from scratch. It is difficult to balance their performance for each iteration when the architectures are different. The agents with the identical architecture but initialized differently obtain at most +1.4 and +1.2 BLEU scores on IWSLT'14 De-En and IWSLT'14 Nl-En.

Figure 2 shows the training procedure of these models. When the architectures vary significantly, the performance of them become far away from each other. When the architectures are similar, the performance of them is close at the beginning and the end of the training. When these agents share the same architecture, the performance of them are always close. These phenomena imply that the agents with different architectures could not effectively distill knowledge to one another since different architectures have various learning capabilities. One could obtain better results with independent learning scheduling for these two agents, like the training of generative adversarial networks (Goodfellow et al., 2014). This is our future work. The results below are obtained by training the agents sharing the identical architecture.

Static or Dynamic Interpolation Weight We employ both static and dynamic interpolation weights (see Equation 8). Figure 3 shows that the

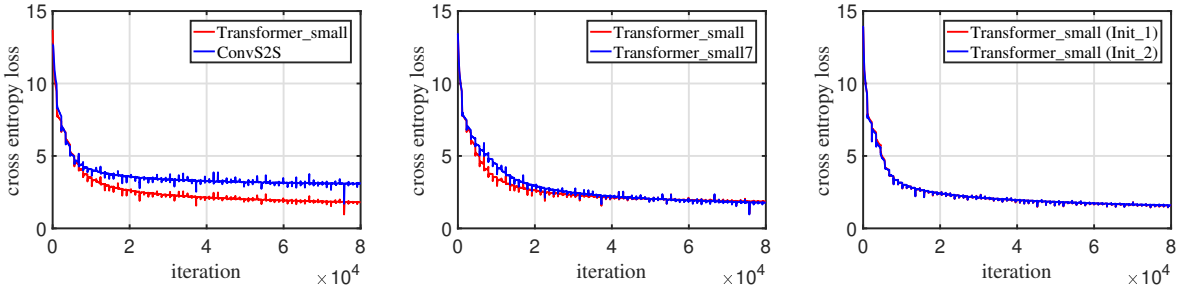


Figure 2: Cross entropy loss (see Equation 2) of two co-trained agents for SML on IWSLT'14 De-En. Left: Two models with different architectures. Middle: Two models with similar architectures. Right: Two models with the same architecture but initialized differently.

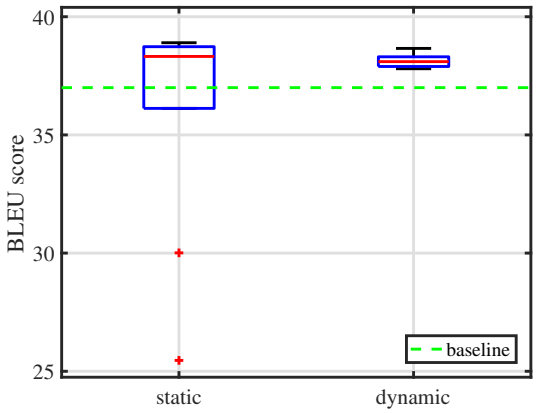


Figure 3: Box plots with 95% confidence interval for two-agent SML with static or dynamic interpolation weights on IWSLT'14 Nl-En. The static interpolation weight λ = 0.1, 0.2, ..., 0.9. The decreasing rate for the dynamic interpolation weight β=0.1,0.2,...,0.9.

performance of the agent is less sensitive to the dynamic interpolation weight. For all values of the decreasing rate β (=0.1,0.2,...,0.9), all of the results are better than the baseline. We believe the reason is: the dynamic interpolation weight gets smaller and smaller with increasing number of epochs. Compared to the scale of the epoch number, the difference of β does not matter significantly. Besides, the amount of distilled knowledge becomes less and less with increasing number of epochs. The agent does not learn much from other agents when it becomes "smart".

Even though the results are sensitive to the static interpolation weight, we can obtain the best result from the fine-tuning of it. Empirically, We can get better results with a static interpolation weight $\lambda = 0.6$ or 0.7 or with a decreasing rate $\beta = 1.0$ or 0.7 for the dynamic interpolation weight.

Number of Agents We also investigate the influence of the number of co-trained agents for SML. As shown in Table 5, we can obtain better results on IWSLT'14 De-En, IWSLT'14 Nl-En and WMT'14 En-De with increasing number of agents. However, the improvement might become saturated, e.g. the best result on WMT'16 Ro-En is obtained from three-agent SML. Overall, we achieve at most +2.0, +1.8, +1.5 and +1.5 BLEU scores on IWSLT'14 De-En, IWSLT'14 Nl-En, WMT'16 Ro-En and WMT'14 En-De with only SML compared to strong baselines, respectively.

5.2 Results for SML + TML

Number of Agents After SML, the agents become "smarter". There is only slight difference between them (see Figure 2). For further improvement, they only learn their poorly predicted tokens from one another. As shown in Table 5, training with both SML and TML consistently obtains better scores on various MT tasks compared to the scores from SML. Similar to SML, this improvement can be saturated, with only slight improvement on WMT'14 En-De as the number of agents increases to 4. Overall, compared to strong baselines, we obtain at most +2.3, +2.2, +2.0 and +1.6 BLEU scores on IWSLT'14 De-En, IWSLT'14 Nl-En, WMT'16 Ro-En and WMT'14 En-De, respectively.

Effect of Beam Size Figure 4 illustrates the sensitivity of the agent against beam size. Without ML, the agent obtains better result with increasing beam size. After the two-step ML, the performances of the agent are less sensitive to the beam size. The line tends to be stable after beam size equals to three.

Effect of Ensemble Figure 5 and Table 5 show the performances of independent ensembles and ensembles with ML. We observe that ensembles with ML consistently outperform independent en-

Model	Method	IWSLT'14 De-En	IWSLT'14 Nl-En	WMT'16 Ro-En	WMT'14 En-De base	WMT'14 En-De big
Vaswani et al. (2017)	-	-	-	-	27.3	28.4
Bi et al. (2019)	-	36.3	-	-	-	29.7
1 agent [1]	-	34.7	37.7	34.1	27.1	28.3
2 agents	SML	36.2	38.9	34.8	27.8	28.9
	SML+TML	36.6	39.5	35.3	28.3	29.4
3 agents	SML	36.4	39.2	35.6	28.1	29.4
	SML+TML	36.8	**39.9**	35.9	28.4	29.8
4 agents	SML	36.7	39.5	35.5	28.2	29.8
	SML+TML	**37.0**	39.8	**36.1**	**28.7**	**29.9**
	Independent ensemble	36.8	38.2	35.8	28.6	29.2
	ML ensemble	**38.0**	**40.3**	**36.4**	**28.8**	**30.1**

Table 5: BLEU scores for two-step training of agents with the same architecture. Transformer_smalls are trained on IWSLT'14 De-En and IWSLT'14 Nl-En. Transformer_bases are trained on WMT'16 Ro-En and WMT'16 En-De. Transformer_bigs are trained on WMT'16 En-De. The agents with the best performance from SML are further trained at the token-level. Cumulative improvements are obtained with TML following SML. Compared to independent ensemble, better results are reported by ensemble with ML.

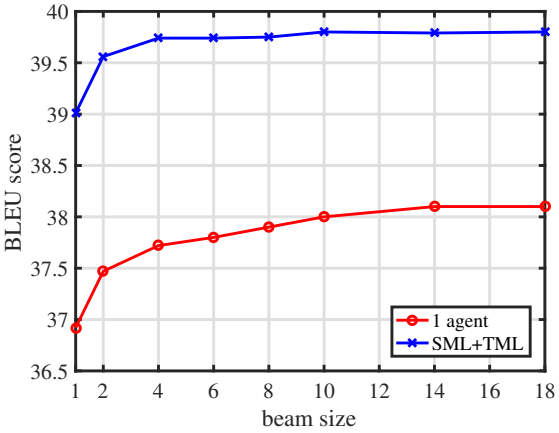


Figure 4: BLEU scores against beam size on IWSLT'14 Nl-En. The result of the red line is from the best agent of three-agent SML+TML.

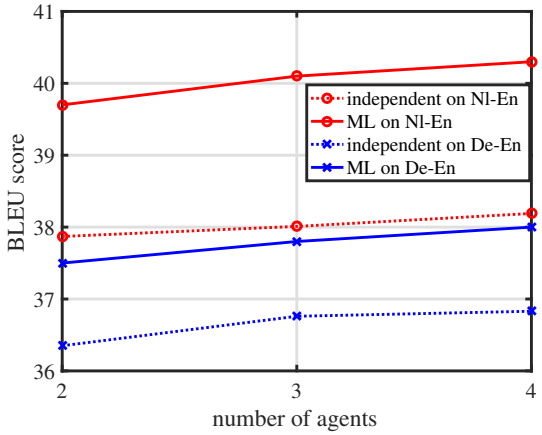


Figure 5: BLEU scores for model ensemble on IWSLT'14 De-En and IWSLT'14 Nl-En.

sembles. Compared to four co-trained agents with ML, the ensembling results are improved less significantly for larger datasets.

6 Conclusion

We extend mutual learning to machine translation tasks at both the sentence-level and the token-level, where multiple agents are co-trained and distill knowledge to one another throughout the training procedure. Firstly, the agents learn the whole sentence from one another. After convergence, they only learn the poorly predicted tokens from other agents. Sampling of poorly predicted tokens is done with acceptance-rejection sampling.

With our two-step mutual learning, agents could learn from one another at different levels and are improved cumulatively. Extensive experiments show significant improvements for both steps. We improve the state-of-the-art for IWSLT'14 German-English from 36.3 (Bi et al., 2019) to 37.0 points without using additional data. On

[1]The results in this row are obtained with the average checkpoint from top 10 checkpoints. In this way, we can have strong baselines. The other results are obtained from the best checkpoint of the best agent. The trick of checkpoint averaging does not improve the results for ML.

WMT'14 English-German, we report 28.7 and 29.9 vs. 27.3 and 28.4 (Vaswani et al., 2017) with Transformer_base and Transformer_big, respectively.

We plan to extend the work by looking into more sophisticated training schedules for the agents with different architectures and applying back-translation to ML.

Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG. Experiments were partially performed with computing resources granted by RWTH Aachen under project nova0003.

References

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*.

Tianchi Bi, Hao Xiong, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Multi-agent learning for neural machine translation. *arXiv preprint arXiv:1909.01101*.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Yun Chen, Yang Liu, Yong Cheng, and Victor OK Li. 2017. A teacher-student framework for zero-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1925–1935.

Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. *arXiv preprint arXiv:1711.04956*.

Yingbo Gao, Baohao Liao, and Hermann Ney. 2020a. Unifying input and output smoothing in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics in Barcelona*.

Yingbo Gao, Weiyue Wang, Christian Herold, Zijian Yang, and Hermann Ney. 2020b. Towards a better understanding of label smoothing in neural machine translation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in neural information processing systems*, pages 820–828.

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

Jinyu Li, Michael Seltzer, Xi Wang, Rui Zhao, and Yifan Gong. 2017. Large-scale domain adaptation via teacher-student learning. pages 2386–2390.

Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size dnn with output-distribution-based criteria. In *Fifteenth annual conference of the international speech communication association*.

Zhong Meng, Jinyu Li, Yifan Gong, and Biing-Hwang Juang. 2018. Adversarial teacher-student learning for unsupervised domain adaptation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5949–5953. IEEE.

Zhong Meng, Jinyu Li, Yong Zhao, and Yifan Gong. 2019. Conditional teacher-student learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6445–6449. IEEE.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

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

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

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

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.

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 5998–6008.

Yiren Wang, Yingce Xia, Tianyu He, Fei Tian, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2018. Multi-agent dual learning.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328.