Non-Autoregressive Neural Machine Translation with Consistency Regularization Optimized Variational Framework

Anonymous ACL submission

Abstract

Variational Autoencoder (VAE) is an effective way to model the interdependency for Non-autoregressive neural machine translation (NAT). LaNMT, a representative VAE-based latent-variable NAT framework achieves great improvements to vanilla models, but still suffers from two main issues which lower down the translation quality: (1) mismatch between training and inference circumstances and (2) inadequacy of latent representations. In this work, we target on addressing these issues by proposing posterior consistency regularization. Specifically, we first apply stochastic data augmentation on the input samples to better adapt the model for inference circumstance, and then perform consistency training on posterior latent variables to train a more robust posterior network with better latent representations. Experiments on En-De/De-En/En-Ro benchmarks confirm the effectiveness of our methods with about 1.3/0.7/0.8 BLEU points improvement to the baseline model with about 12.6× faster than autoregressive Transformer.

1 Introduction

Neural Machine Translation (NMT) achieves great success in recent years, and typical sequence-to-sequence frameworks like Transformer (Vaswani et al., 2017) achieved state-of-the-art performance on the task of NMT. In this framework, source sentences are translated in an autoregressive (AT) manner where each token is generated depending on previously generated tokens, inevitably, such sequential decoding strategy result in a high inference latency. To alleviate this issue, Non-autoregressive translation (NAT; Gu et al., 2018) was proposed to speed-up decoding procedure by generating target tokens in parallel. However, the translation quality of vanilla NAT is compromised, one of the most significant problem is multi-modality and it usually results in multiple translation results, duplicate or missing words in target sentences of NAT models (Gu et al., 2018). This situation results from the conditional independence proposed by NAT, since models are trained to maximize the log-probability of target tokens at each position while the interdependency is omitted.

The key to alleviate the multi-modality issue is performing dependency reduction (Gu and Kong, 2021) by modeling the target dependency information implicitly or explicitly so decoder can ease the difficulty of learning and capturing the information between target tokens and generate more accurate translations. For example, Ghazvininejad et al. (2019) (2020b) and Guo et al. (2020) model the target dependency by providing observed target tokens in training and performing iterative inference. Ran et al. (2021) generates intermediate representations by permuting the source sentences in the target order. Libovickỳ and Helcl (2018) aligns model outputs with target tokens implicitly by applying Connectionist Temporal Classification (CTC; Graves et al., 2006).

Previous works have validated the effectiveness of applying Variational Autoencoder (VAE) on AT (Zhang et al. 2016; McCarthy et al. 2019; Su et al. 2018) and NAT (Kaiser et al. 2018; Shu et al. 2020) frameworks to alleviate multi-modality issue. A representative NAT model is LaNMT[1](Shu et al., 2020) which encodes the source and target tokens into intermediate Gaussian distribution latent variables and outperforms vanilla NAT with about 5.0 BLEU points on WMT14 En-De task with 12.5× speedup to base Transformer. However, there exists a slight lag behind the state-of-the-art fully NAT models. It may be attributed to two reasons: (1) The inadequate representations of latent variables which are low in dimensions (4 to 32 is recommended). This is significantly lower than the model's hidden size (512) while high-capacity latent variables conversely deteriorate the performance because the minimization between prior

[1] https://github.com/zomux/lanmt

and posterior becomes difficult (Shu et al., 2020). (2) The mismatch between training and inference circumstances that the posterior module receives the gold sentence as inputs during training but imperfect initial translation instead during inference. Thus, in this paper, we aim to improve the robustness of the latent representation and move the training circumstance close to inference circumstance.

To this end, we apply consistency regularization over the posterior network to improve its robustness for better latent representations since the posterior is the key module that both encoder and decoder are relying on its latent representations during training. To cooperate with consistency regularization, and simultaneously, close the gap between training and inference circumstances for better refinement from imperfect initial translations during inference, four data augmentation methods are adopted to work together. Specifically, we first apply stochastic data augmentation methods *e.g.* Cutoff (Shen et al., 2020) to inject stochastic noises in posterior inputs x and y to get two different views. Both views are then forwarded to the posterior network for two latent variables z_1, z_2. As these two latent variables are derived from the same pair of input x and y, the gap between them is trained to be minimised by consistency regularization. Meanwhile, posterior module receives noisy views instead of gold samples during training, it is more adaptive to the inputs with imperfect initial translations in inference.

We verified the performance and effectiveness of our methods on WMT14 En-De, De-En and WMT16 En-Ro benchmarks. Our methods outperform the latent variable baseline with about 1.3/0.7/0.8 BLEU points improvement on three benchmarks. With these improvements, we achieve the comparable performance to the state-of-the-art fully NAT approaches: 25.47/30.23/31.56 BLEU scores on WMT14 En-De/De-En/WMT16 En-Ro with similar decoding speed, and it can be improved further with latent search. The contributions of our work can be summarized as follows:

- To achieve better latent representations, we propose posterior consistency regularization on the posterior latent variables, which improves the translation quality by training a more robust posterior network.
- To alleviate the mismatch between training and inference circumstances and cooperate with posterior consistency regularization, we apply four data augmentation methods where all of them benefit to the translation quality.
- We show our strategy is capable of improving the translation quality of the base latent-variable NAT model to be comparable with the state-of-the-art fully NAT frameworks.

2 Background

2.1 Non-Autoregressive Translation

Traditional sequence-to-sequence NMT models generate target sentences in an autoregressive manner. Specifically, given a source sentence x, AT frameworks model the conditional probability of $y = \{y_1, y_2, \cdots, y_{l_y}\}$ by the following form:

$$\log p(y|x) = \sum_{i=1}^{l_y} \log p(y_i|y_{<i}, x) \qquad (1)$$

where $y_{<i}$ indicates the target tokens already generated before y_i. Hence, the target tokens are generated sequentially which results in a high decoding latency. To alleviate this issue, vanilla NAT (Gu et al., 2018) breaks the conditional dependency by conditional independence assumption so that all tokens can be generated independently. Following its probability form:

$$\log p(y|x) = \sum_{i=1}^{l_y} \log p(y_i|x) \qquad (2)$$

where each target token y_i now only depends on the source sentence x. Benefit from the parallel computing capability of hardware accelerators like GPU or TPU, all tokens can be generated with one iteration in an ideal circumstance.

2.2 Latent-Variable Model

We mainly focus on performing optimization on the variational NAT framework proposed by Shu et al. (2020). The network architecture is constructed by four main components. An encoder $p_\omega(z|x)$ encodes the source representation of input x and computes the prior latent variable. An approximate posterior network $q_\phi(z|x, y)$ accepts both the source sentence x and target sentence y as the input and computes the posterior latent variable. A length predictor $p(l_y|z)$ predicts the length of target sentence y, and finally a decoder $p_\theta(y|x, z, l_y)$ with a length transform module to transform the latent variables z to the target length l_y at first and reconstruct y from z with the source representations of x. Note that the l_y here is the gold length in training. Hence, the training objective is aiming to maximize the evidence lowerbound (ELBO):

$$\mathcal{L}(x, y; \phi, \theta, \omega) = \mathbb{E}_{z \sim q_\phi}[\log p_\theta(y|x, z, l_y) + \log p(l_y|z)] - KL\left[q_\phi(z|x, y)||p_\omega(z|x)\right] \quad (3)$$

where the latent variables z is constrained with the same length as x and the value is modeled as spherical Gaussian distribution. KL denotes Kullback-Leibler divergence.

2.3 Consistency Regularization

Consistency regularization is considered as an effective method on semi-supervised learning to capture the potential features from unlabeled samples (Sajjadi et al., 2016; Laine and Aila, 2017; Tarvainen and Valpola, 2017; Xie et al., 2020). It is also utilized as a complementary regularization tool with other regularization methods to prevent model from overfitting (Liang et al., 2021). In a nutshell, consistency regularization assumes a well trained model should be robust enough to any small changes in the input samples or hidden states and generate invariant outputs (Xie et al., 2020). To this end, it regularizes model's final outputs to be invariant to input samples with small stochastic noises injected by minimizing the gap between two augmented views of one sample.

In this paper, we focus on a sub-module of the variational model and apply consistency regularization on it instead of the whole network. Along with data augmentation for noise injection, consistency regularization is capable to improve the representation of this module and result in better translation quality.

3 Approach

The posterior module is considered to train with consistency regularization and data augmentation for better translation quality. In this section, we will introduce the details of our method, including the overall network architecture, the objective and procedure of training with consistency regularization, four data augmentation methods and three decoding strategies applied for inference.

3.1 Model Architecture

We follow the variational model architecture proposed by Shu et al. (2020) with four main components: encoder, posterior, length predictor and decoder module. Since we apply consistency regularization on the posterior, an additional stochastic data augmentation module is added for noise injection on posterior input samples. With two augmented views derived from one sample, each sample thus appears twice in a training batch. Figure 1 shows the brief model architecture and training pipeline of our work. The part in the dashed box is the major difference to the base model.

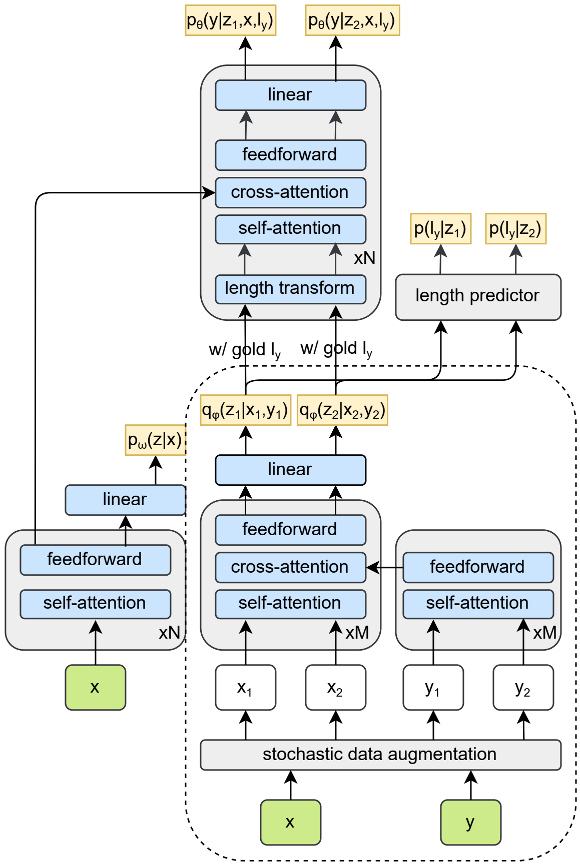

Figure 1: The overall pipeline of training with posterior consistency regularization

3.2 Posterior Consistency Regularization

As discussed above, consistency regularization is applied on the posterior module to improve its robustness. Given a training sample with a pair of source sentence $x = \{x_1, x_2, \cdots, x_{l_x}\}$, and target sentence $y = \{y_1, y_2, \cdots, y_{l_y}\}$, we first apply data augmentation on both x and y twice to inject stochastic noises and obtain two different views. Both views are forwarded to the posterior network $q_\phi(z|x, y)$ to predict the mean and variance vectors of two latent variables z_1 and z_2. Since the latent variables derive from the same input sample, the consistency regularization method tries to minimize the difference between these two latent vari-

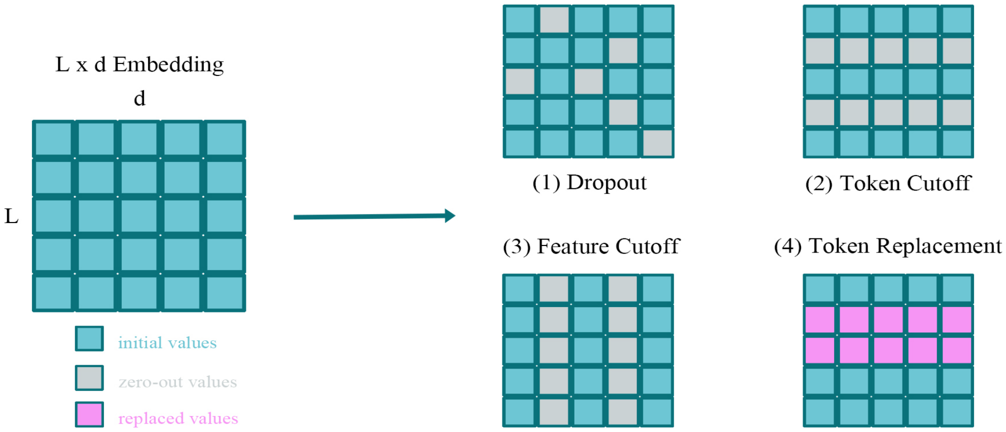

Figure 2: Four stochastic data augmentation methods we used for noise injection

ables by measuring bidirectional KL-divergence as follows:

$$\mathcal{L}_{cons} = \frac{1}{2}(KL(z_1||z_2) + KL(z_2||z_1)), \quad (4)$$

Combining with the basic negative log-likelihood (NLL) objective on the decoder, since there are two different z for the same sample, it is evaluated by averaging them:

$$\mathcal{L}_{nll} = -\frac{1}{2}(\log p_\theta(y|x, z_1, l_y) + \log p_\theta(y|x, z_2, l_y)) \quad (5)$$

Note that the gold length l_y of target sentence y is used which is known during training. Similarly, the objective of the length predictor is calculated by:

$$\mathcal{L}_{len} = -\frac{1}{2}(\log p(l_y|z_1) + \log p(l_y|z_2)) \quad (6)$$

To back propagate the gradient information from the decoder and length predictor to posterior, reparameterization trick is applied to sample z from q_ϕ where $z = \mu + \theta * \mathcal{N}(0, 1)$ in Eq.(5) and (6). Here, μ and θ indicate mean and variance vector. For encoder, it not only generates representations of source sentence x but also computes the prior latent variables. Thus, we close the KL-divergence between prior and two posterior latent variables by:

$$\mathcal{L}_{prior} = \frac{1}{2}(KL(z_1||z_p) + KL(z_2||z_p)), \quad z_p = p_\omega(z|x) \quad (7)$$

Finally, to achieve the similar goal of maximizing (3), we minimize the loss function by combining (4), (5), (6) and (7) as follows:

$$\mathcal{L}_{loss} = \mathcal{L}_{nll} + \mathcal{L}_{len} + \mathcal{L}_{prior} + \alpha\mathcal{L}_{cons} \quad (8)$$

where α here is the only hyperparameter to weight the consistency regularization loss.

3.3 Data Augmentation Methods

Given an embedding matrix $\mathbb{R}^{L\times d}$ with L tokens embedded into d-dimensions vectors, to generate different views of each sample for the posterior network inputs and perform consistency regularization on the posterior network, as well as to close the gap between training and inference circumstances, we explore four data augmentation methods for this purpose including dropout, feature cutoff, token cutoff and replacement as presented in Figure 2.

Dropout Dropout (Srivastava et al., 2014) is widely used as a regularization method to prevent neural networks from overfitting. But in this paper, we found that it is also an effective data augmentation method for noise injection. Specifically, we randomly choose values on token embeddings by a specific proportion and force them to zero.

Cutoff This is a simple but effective augmentation method proposed by Shen et al. (2020). The cutoff methods we adopted include token cutoff and feature cutoff. For token cutoff, a specific proportion of tokens are chosen from the token dimension L and dropped by setting the vectors to zero. For feature cutoff, the dropped values are chosen from feature dimension d instead.

Replacement This is similar to the token replacement adopted by BERT pre-training (Devlin et al., 2019) where the chosen token vectors are replaced by the embedding of new tokens that randomly selected from the vocabulary instead of setting them to zero or any special tokens directly.

3.4 Decoding Strategies

Non-refinement For this strategy, we completely follow the original design (Shu et al., 2020) where the posterior network is discarded since the target sentence y is unknown during inference. The foremost step is to obtain the representations of x and the prior latent variable z from encoder with source input x. The latent variable is then used to determine the target length and generate target sentence. Note that to avoid randomness during inference, z is set to its mean value μ instead of reparameterization sampling. This can be summarized as follows:

$$\begin{aligned} \mu_0 &= \mathbb{E}_{p_\omega(z|x)}[z], \\ l_{y_0} &= argmax_{l_y} \log p(l_y|z = \mu_0), \\ y_0 &= argmax_y \log p_\theta(y|x, l_{y_0}, z = \mu_0) \end{aligned} \quad (9)$$

Deterministic Refinement The posterior network q_ϕ can be reused to take refinement on the initial output y_0 above. However, its original design allows iterative refinement with multiple steps which sacrifices huge cost in decoding speed for a tiny quality improvement. Thus, we consider refinement for one step only in this paper:

$$\begin{aligned} \mu_1 &= \mathbb{E}_{q_\phi(z|x,y_0)}[z], \\ l_{y_1} &= argmax_{l_y} \log p(l_y|z = \mu_1), \\ y_1 &= argmax_y \log p_\theta(y|x, l_{y_1}, z = \mu_1) \end{aligned} \quad (10)$$

Here the y_1 is the final output after refinement.

Latent Search Since reparameterization is disabled in above two strategies to generate deterministic results, it is also capable to search the best latent variable from Gaussian distribution. Specifically, m prior latent variables are sampled by reparameterization and decoded in parallel, result in m target candidates for each source sentence. To get the best result, we select the candidate with the highest score by averaging the log-probability of tokens as the final output. This is different from Shu et al. (2020) or Noisy Parallel Decoding (NPD; Gu et al. 2018) which rescore the candidates by autoregressive teacher and cuts the decoding speed by half, our no-rescoring strategy is still effective and much faster.

4 Experiments

In this section, we will introduce the settings of our experiments, report the main results and compare our model to the representative NAT frameworks. Our experiments mainly focus on (1) the improvement benefit from our optimization to former VAE-based NAT model. (2) The effectiveness of consistency regularization and different data augmentation methods.

4.1 Experimental Setup

Dataset Three of the commonly used machine translation benchmarks are adopted to evaluate our proposed method: WMT14 English<->German[2] (En-De and De-En, 4.5M) and WMT16 English->Romanian[3] (En-Ro, 610K). We follow previous works' data preprocessing configurations to preprocess the data (En-De: Shu et al., 2020, En-Ro: Ghazvininejad et al. 2019). To learn the subword vocabulary, we apply SentencePiece (Kudo and Richardson, 2018) to generate joint subword vocabulary of 32K tokens for each dataset respectively.

Knowledge Distillation Following previous studies on NAT that models are trained on distilled data generated by autoregressive teacher, we also apply sentence-level knowledge distillation for all datasets to obtain less noisy and more deterministic data. In this work, Transformer (Vaswani et al., 2017) with base settings is adopted and reproduced as the teacher model for data distillation.

Implementation Details The model is trained by the objective function illustrated on Eq.(8). To avoid posterior collapse, freebits annealing (Chen et al., 2017) is applied on KL terms in Eq.(7) to keep a distance between prior and posterior. Its threshold is fixed to 1 for the first half training steps, and linearly decay to 0 on the second half. For both dataset, we train the model with a batch size of approximate 40K tokens for overall 100K steps on four Tesla V100 GPUs and continue to fine-tune it for additional 20K steps with freebits annealing disabled.

For network settings, we use 6 layers encoder and decoder with $d_{model}/d_{feedforward} = 512/2048$. Following Shu et al. (2020), the posterior network contains 3 transformer layers and the dimension of latent variable is set to 8. We set dropout between attention layers with rate of 0.1/0.3 for WMT14 En<->De and WMT16 En->Ro respectively and label smoothing rate $\epsilon = 0.1$ on the target tokens. Models are trained by Adam (Kingma and Ba, 2015) with settings of $\beta = (0.9, 0.98)$ and $\epsilon = 1e-4$. We use the same strategy as Vaswani et al. (2017) to schedule the learning rate and set warm-up steps to 4000. To obtain the final model, we average 5 best checkpoints chosen by validation BLEU score. By default, we set rate of 0.3/0.2/0.1/0.2 for four data augmenta-

[2] https://www.statmt.org/wmt14/
[3] https://www.statmt.org/wmt16/

	Models	Iter.	WMT14 En-De	WMT14 De-En	WMT16 En-Ro	Speed
AT	Transformer (Vaswani et al., 2017)	N	27.30	/	/	/
	Transformer (ours)	N	27.18*	31.28*	33.73*	1.0×
Iterative NAT	NAT-IR (Lee et al., 2018)	10	21.61	25.48	29.32	1.5×
	CMLM (Ghazvininejad et al., 2019)	10	27.03	30.53	33.08	1.7×
	LevT (Gu et al., 2019)	Adv.	27.27	/	/	4.0×
	JM-NAT (Guo et al., 2020)	10	27.69	32.24	33.52	5.7×
Fully NAT	Vanilla-NAT (Gu et al., 2018)	1	17.69	21.47	27.29	15.6×
	Imitate-NAT (Wei et al., 2019)	1	22.44	25.67	28.61	18.6×
	FlowSeq (Ma et al., 2019)	1	23.72	28.39	29.73	1.1×
	NAT-DCRF (Sun et al., 2019)	1	23.44	27.22	/	10.4×
	BoN (Shao et al., 2020)	1	20.90	24.60	28.31	10.7×
	AXE (Ghazvininejad et al., 2020a)	1	23.53	27.90	30.75	/
	GLAT (Qian et al., 2021)	1	**25.21**	**29.84**	31.19	15.3×
	Reorder-NAT (Ran et al., 2021)	1	22.79	27.28	29.30	16.1×
	SNAT (Liu et al., 2021)	1	24.64	28.42	**32.87**	22.6×
Baselines and Ours	LT (Kaiser et al., 2018)	/	19.80	/	/	3.8×
	LaNMT (Shu et al., 2020)	1	22.20	26.76*	29.21*	22.2×
	+ refinement	2	24.10	29.47*	30.76*	12.5×
	+ latent search w/ rescoring	2	25.10	/	/	6.8×
	Ours, decode w/o refinement	1	23.67	27.39	29.90	25.6×
	+ latent search (m=9) w/o rescoring	1	24.89	30.11	31.40	21.1×
	+ latent search (m=19) w/o rescoring	1	25.20	30.70	31.65	17.6×
	decode w/ refinement	2	25.47	30.23	31.56	12.6×
	+ latent search (m=9) w/o rescoring	2	**26.02**	**31.23**	**32.50**	11.0×

Table 1: BLEU scores and speedup rates for performance comparison on WMT14 En-De, De-En and WMT16 En-Ro benchmarks without rescoring. We report the best scores here among all tested combinations of data augmentation methods with consistency regularization. **Iter.** denotes the number of iterations during inference. **Adv.** means adaptive. / denotes the value is not reported, * denotes the results obtained by our implementation.

tion methods: dropout, feature cutoff, token cutoff and token replacement respectively with the weight term $\alpha = 0.1$ at Eq.(8).

Evaluation For all benchmarks, we use sacreBLEU[4] (Post, 2018) to evaluate BLEU score of translation results. Following Lee et al. (2018) and Shu et al. (2020), repetition tokens are removed before generating the final outputs for evaluation. The results of latent search is obtained by the mean score of 5 independent runs on the test set of each benchmark to get more precise measures since reparameterization causes randomness in decoding.

To evaluate the decoding speed, following previous works (e.g. Gu et al. 2018, Lee et al. 2018), models are run on WMT14 En-De test set with batch size of 1 under the environment with one GPU only. The mean value of decoding latency among all samples is collected and represent as the decoding speed. Meanwhile, base Transformer is reproduced and evaluated on the same machine to obtain the speed up rates.

Baselines We set former VAE based NAT frameworks proposed by Kaiser et al. (2018) and Shu et al. (2020) as the main baselines to present the improvement of our method. We also compare our model with other representative NAT and AT frameworks. The performance measures including BLEU score and speedup rate of other models are directly obtained from the figures reported on their original paper, while some unreported measures are obtained by our implementation.

4.2 Results and Analysis

The main results on the benchmarks are illustrated on Table 1, we report the best scores of our experi-

[4]https://github.com/mjpost/sacrebleu

ments among different tested combinations of data augmentation methods with consistency regularization. As the performance measure shown in Table 1, our methods significantly outperform former VAE-based baselines, with about 5.6 BLEU points improvement to the discrete latent variable model (Kaiser et al., 2018) and 1.4/1.3, 0.6/0.7, 0.7/0.8 points improvement on non-refinement/refinement decoding to continuous latent variable baseline (Shu et al., 2020) on WMT14 En-De, De-En and WMT16 En-Ro benchmarks without latent search. All measures indicate that our posterior consistency regularization method greatly enhances the robustness of the VAE-based model and results in an improved translation quality.

Comparing to other representative AT and NAT models, our method shows the superiority of decoding speed to AT and iterative NAT models while there are only about 2 BLEU points lag behind. With the refinement decoding, our model also achieves a comparable translation quality to the state-of-the-art fully-NAT approaches with similar decoding latency.

The results of latent search is encouraging. Benefit from the parallel computing capability of GPU, latent search sacrifices very small decoding speed to achieve about 0.5/1.0/0.9 BLEU improvements for refinement decoding and 1.2/2.7/1.5 BLEU improvements for non-refinement decoding on WMT14 En-De / De-En / WMT16 En-Ro benchmarks with $m = 9$.

Effectiveness of Data Augmentation Methods In this work, we adopt four different data augmentation strategies as the stochastic noise injection method to cooperate with consistency regularization. To evaluate their effectiveness and the impact for translation quality, all data augmentation methods are tested with the default configurations on all of the benchmarks. The results are reported on Table 2. The method we adopt combining posterior consistency regularization with data augmentation is effective and capable to achieve higher BLEU scores than the baseline. Specifically, token replacement achieves the highest score on all of benchmarks with refinement decoding since the posterior network is trained on sentences with incorrect tokens, this is more similar to the inference circumstance. With the non-refinement decoding, non of the methods can dominate all benchmarks since the posterior is discarded.

	Method	En-De	De-En	En-Ro
w/ refinement	**Baseline**	24.10	29.47*	30.76*
	Dropout	25.08	29.74	30.85
	Token Cutoff	25.06	30.05	31.34
	Feat. Cutoff	25.13	29.58	30.95
	Token Repl.	**25.33**	**30.23**	**31.56**
w/o refinement	**Baseline**	22.20	26.76*	29.21*
	Dropout	23.25	26.93	29.40
	Token Cutoff	**23.67**	27.18	29.55
	Feat. Cutoff	23.51	26.92	**29.90**
	Token Repl.	22.98	**27.39**	29.68

Table 2: BLEU scores for baseline and our models with different data augmentation methods. * denotes the results obtained by our implementation. **Baseline** indicates Shu et al. (2020)

Method	$\alpha = 0$	0.1	0.2
Baseline		24.10	
Dropout	24.76	**25.08**	24.84
Token Cutoff	24.82	25.06	**25.17**
Feature Cutoff	24.82	25.13	**25.14**
Token Repl.	25.05	25.33	**25.47**

Table 3: BLEU scores on WMT14 En-De for baseline and our methods with different weight α for consistency regularization objective. Specially, $\alpha = 0$ indicates training with consistency regularization disabled.

Effectiveness of Consistency Regularization Consistency regularization should work together with stochastic data augmentation which is widely known as a trick to train robust neural networks (Shorten and Khoshgoftaar 2019; Shen et al. 2020). Thus, to confirm that the model is not just benefit from data augmentation only but the contribution of posterior consistency regularization, we disable the consistency regularization module by setting $\alpha = 0$ at Eq.(8) and train the model with four data augmentation methods respectively on WMT14 En-De dataset. The results illustrate on Table 3. Without consistency regularization, the data augmentation methods still result in improvement to baseline, but a slight lag is exist behind the model with consistency regularization enabled. In other words, consistency regularization can improve the translation quality further. Thus, it is confirmed that consistency regularization is effective and capable to train a more robust latent representation in this work. Besides, with different weights for consistency regularization objective term, the best

α for cutoff and replacement is 0.2 and dropout is 0.1 on WMT14 En-De in our experiments.

Effect of Augmentation Rate To investigate the impact of augmentation rate, we train the models by different augmentation rates with default $\alpha = 0.1$ on WMT14 En-De dataset. Results are illustrated on Table 4. The best augmentation rate is different for each augmentation methods. According to this experiment, 0.1/0.2 (or 0.3) is the best for token and feature cutoff. Token replacement behaves similarly to token cutoff (both are token-level augmentation) but the best rate is completely different. It could be attribute to the mechanism that model can potentially learn from the incorrect tokens and revise them, which mostly benefits to the inference where there are massive incorrect tokens from initial translations on refinement decoding. However, token cutoff simply zero-out the tokens during training, since there is no blank token in initial decoding outputs, a higher rate may conversely enlarge the mismatch between training and inference.

Method	$rate = 0.1$	0.2	0.3
Token Cutoff	**25.06**	24.98	24.54
Feature Cutoff	24.93	**25.13**	**25.13**
Token Repl.	25.26	**25.33**	25.22

Table 4: Effect of the rate for augmentation methods

Tradeoff between Speed and Quality The tradeoff between the speedup rate and translation quality on WMT14 En-De dataset is shown in Figure 3. We draw the scatter points by evaluating the proposed model on various number of candidates sampled for latent search. It can be observed that both decoding with or without refinement can benefit from latent search while the decoding speed remains acceptable. Specifically, the non-refinement decoding with more latent candidates can reach the level of refinement approach. However, refinement decoding can achieve further improvements and reaches the peak of about 26.2 BLEU points.

Summary Summarize from the experiments and corresponding results illustrated on Table 2, 3 and 4, the mechanism of data augmentation and consistency regularization in this paper can be explained in two ways: firstly, data augmentation methods help the posterior network learn the capability of encoding correct latent variables from incomplete, incorrect or noisy sentences, which narrows the gap between training and inference circumstances. Thus, our posterior network can do better refinement on the initial translation y_0 from Eq.(9) which is relatively noisy and imperfect. Secondly, consistency regularization helps the posterior network learn to be more consistent on latent variables under the impact of noises in input samples, this potentially improves the robustness of posterior network and latent representations which result in further improvements. Both strategies cooperate together and maximize the overall translation quality.

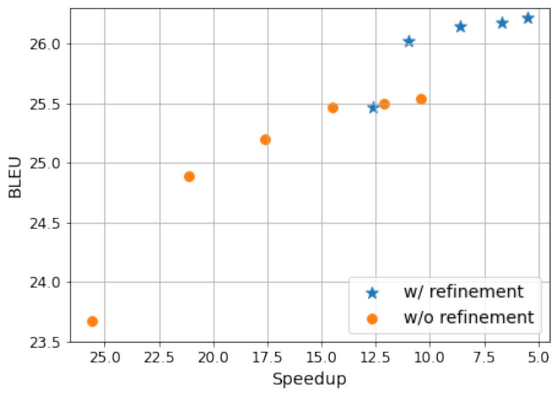

Figure 3: Tradeoff between decoding speed and translation quality on WMT14 En-De benchmark.

Conclusion

In this work, we introduce posterior consistency regularization along with a series of data augmentation methods on the posterior module of a variational NAT model to improve its performance of translation quality. This method trains the posterior network to be consistent to stochastic noises in inputs and potentially improves its representations. Meanwhile, data augmentation closes the gap between training and inference circumstances. Both are highly benefit to decoding and refinement step. Experiments on WMT14 En-De, De-En and WMT16 En-Ro benchmarks show that our approach achieves a significant improvement to the baseline model and a comparable translation quality to other state-of-the-art fully NAT models with fast decoding speed. As the effectiveness of consistency regularization and data augmentation is verified by our experiments, it is promising to be applied on other models and tasks in the future.

References

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2017. Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

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

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020a. Aligned cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 3515–3523. PMLR.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 6111–6120. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. 2020b. Semi-autoregressive training improves mask-predict decoding. *CoRR*, abs/2001.08785.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 120–133. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *Advances in Neural Information Processing Systems*, 32:11181–11191.

Junliang Guo, Linli Xu, and Enhong Chen. 2020. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 376–385.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pages 2390–2399. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. 2021. R-drop: Regularized dropout for neural networks. *CoRR*, abs/2106.14448.

Jindřich Libovickỳ and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021.

Ye Liu, Yao Wan, Jianguo Zhang, Wenting Zhao, and Philip Yu. 2021. Enriching non-autoregressive transformer with syntactic and semantic structures for neural machine translation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1235–1244, Online. Association for Computational Linguistics.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. 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 4282–4292, Hong Kong, China. Association for Computational Linguistics.

Arya D McCarthy, Xian Li, Jiatao Gu, and Ning Dong. 2019. Improved variational neural machine translation by promoting mutual information. *arXiv preprint arXiv:1909.09237*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1993–2003, Online. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13727–13735.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29:1163–1171.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 198–205.

Dinghan Shen, Mingzhi Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *CoRR*, abs/2009.13818.

Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8846–8853.

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.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhi-Hong Deng. 2019. Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3011–3020.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

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.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1304–1312.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33.

Biao Zhang, Deyi Xiong, Jinsong Su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas. Association for Computational Linguistics.