The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Fine-Tuning by Curriculum Learning for Non-Autoregressive Neural Machine Translation

Junliang Guo,[†] Xu Tan,[‡] Linli Xu,[†*] Tao Qin,[‡] Enhong Chen,[†] Tie-Yan Liu[‡]
[†]Anhui Province Key Laboratory of Big Data Analysis and Application,
School of Computer Science and Technology, University of Science and Technology of China
[‡]Microsoft Research
[†]guojunll@mail.ustc.edu.cn, {linlixu, cheneh}@ustc.edu.cn, [‡]{xuta, taoqin, tyliu}@microsoft.com

Abstract

Non-autoregressive translation (NAT) models remove the dependence on previous target tokens and generate all target tokens in parallel, resulting in significant inference speedup but at the cost of inferior translation accuracy compared to autoregressive translation (AT) models. Considering that AT models have higher accuracy and are easier to train than NAT models, and both of them share the same model configurations, a natural idea to improve the accuracy of NAT models is to transfer a well-trained AT model to an NAT model through fine-tuning. However, since AT and NAT models differ greatly in training strategy, straightforward fine-tuning does not work well. In this work, we introduce curriculum learning into fine-tuning for NAT. Specifically, we design a curriculum in the fine-tuning process to progressively switch the training from autoregressive generation to non-autoregressive generation. Experiments on four benchmark translation datasets show that the proposed method achieves good improvement (more than 1 BLEU score) over previous NAT baselines in terms of translation accuracy, and greatly speed up (more than 10 times) the inference process over AT baselines.

1 Introduction

Neural machine translation (NMT) (Bahdanau, Cho, and Bengio 2014; Gehring et al. 2017; Shen et al. 2018; Vaswani et al. 2017; He et al. 2018; Hassan et al. 2018) has made rapid progress in recent years. The dominant approaches for NMT are based on autoregressive translation (AT), where the generation of the current token in the target sentence depends on the previously generated tokens as well as the source sentence. The conditional distribution of sentence generation in AT models can be formulated as:

$$P(y|x) = \prod_{t=1}^{T_y} P(y_t|y_{<t}, x), \tag{1}$$

where T_y is the length of the target sentence which is implicitly decided by predicting the `[EOS]` token, and $y_{<t}$ represents all generated target tokens before y_t and x represents the source sentence. Since AT model generates the target tokens sequentially, the inference speed is a natural bottleneck for real-world machine translation systems.

Recently, non-autoregressive translation (NAT) models (Gu et al. 2017; Kaiser et al. 2018; Lee, Mansimov, and Cho 2018; Guo et al. 2019; Wang et al. 2019; Li et al. 2019) are proposed to reduce the inference latency by generating all target tokens independently and simultaneously. Instead of conditioning on previously generated target tokens, NAT models generate target tokens by taking other target-independent signals as the decoder input. In this way, the generation of y can be written as:

$$P(y|x) = P(T_y|x) \cdot \prod_{t=1}^{T_y} P(y_t|z, x), \tag{2}$$

where $P(T_y|x)$ is the explicit length prediction process for NAT models, and z represents the decoder input which is generated conditionally independent of y. As a result, the inference speed can be significantly boosted. However, the context dependency within the target sentence is sacrificed at the same time, which leads to a large degradation of the translation quality of NAT models. Therefore, improving the accuracy of NAT models becomes a critical research problem.

Considering that 1) NAT is a harder task than AT due to that the decoder in the NAT model has to handle the translation task conditioned on less and weaker target-side information; 2) AT models are of higher accuracy than NAT models; 3) NAT models (Gu et al. 2017; Guo et al. 2019; Wang et al. 2019) usually share the same encoder-decoder framework with AT models (Vaswani et al. 2017), it is very natural to fine-tune a well-trained AT model for NAT, in order to transfer the knowledge learned in the AT model, especially the ability of target language modeling and generation in the decoder. However, AT and NAT models differ a lot in training, and thus directly fine-tuning a well-trained AT model does not lead to a good NAT model in general.

To effectively transfer an AT model and obtain a good NAT model, we first note that there are two major differences between NAT and AT models, as shown in Figure 1.

- **Decoder input**: The decoder in AT models leverages the previous tokens as input while the decoder in NAT models

[*]Corresponding Author.

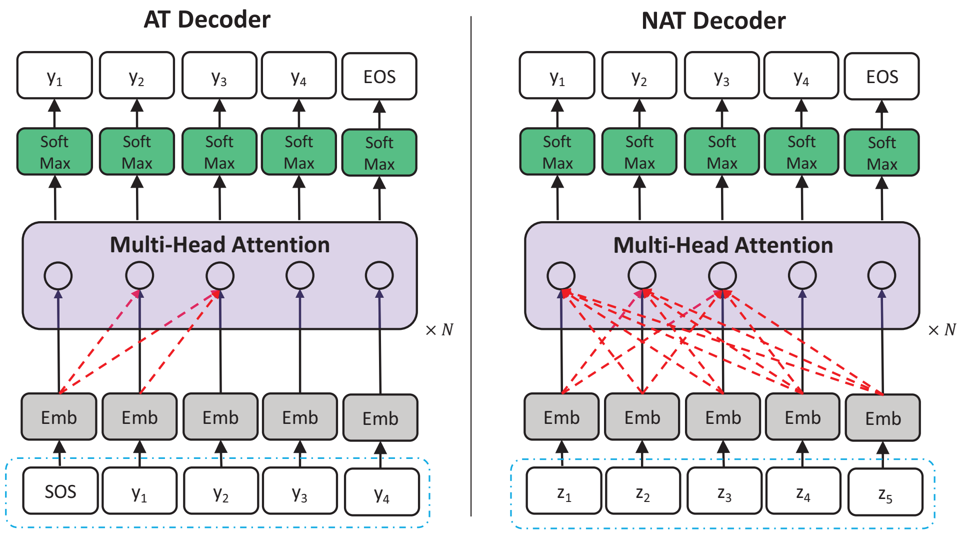

Figure 1: The comparison between the decoders of AT models and NAT models. The red dashed line indicates the attention mask, and we only draw the masks of the first three tokens for simplicity. The blue dashed box indicates the decoder input. Best view in color.

takes target-independent signals as input. Specifically, Gu et al. (2017) and Wang et al. (2019) take a copy of the source sentence x as the decoder input.

- **Attention mask**: Each token can only attend to the tokens in its previous positions in AT models, while each token can attend to the tokens in all positions in NAT models.

In order to handle the differences between the AT and NAT models during the fine-tuning process, we introduce the idea of curriculum learning (Bengio et al. 2009) to make the transfer smooth and progressive. Specifically, we propose two kinds of curriculums for the transfer from an AT model to an NAT model:

- **Curriculum for the decoder input**: We first feed the target sentence as AT models do, and then randomly substitute a number of tokens by the tokens in the copied source sentence, where the number of substituted tokens depends on a probability that is monotonically increasing w.r.t the training step.
- **Curriculum for the attention mask**: We first train the model with the attention mask of AT models and switch to that of NAT models entirely after a pre-defined training step.

In this way, we first train the translation model in an easier autoregressive generation, and gradually transfer to a harder non-autoregressive generation. We conduct experiments on four translation datasets including WMT14 English-German, WMT14 German-English and IWSLT14 German-English to verify the effectiveness of the proposed method, and our model outperforms all non-autoregressive baselines on these tasks. Specifically, we outperform the best NAT baseline (Wang et al. 2019) by 1.87 BLEU on the IWSLT14 De-En task and 1.14 BLEU on the WMT14 En-De task.

2 Related Work

2.1 Non-Autoregressive Neural Machine Translation

As shown by Equation 2, NAT models generate target tokens conditioned on the source sentence x and the decoder input z, and some previous works concentrate on the design of z. Gu et al. (2017) introduce a fertility predictor to guide how many times a source token is copied to the decoder input. Lee, Mansimov, and Cho (2018) define z by iteratively refining the target sentences generated by NAT. Guo et al. (2019) enhance the decoder input with target-side information by either utilizing auxiliary information or introducing extra parameters. Ren et al. (2019) directly use the expanded hidden sequence from the source side as z in text to speech problem. Besides trying different designs of z, Wang et al. (2019) and Li et al. (2019) propose auxiliary loss functions to solve the problem that NAT models tend to translate missing and duplicating words.

Another line of related works focuses on finding a tradeoff between high inference speed and good translation performance. Traditional AT models take $O(n)$ iterations to generate a sentence with length n during inference. Kaiser et al. (2018) takes intermediate discrete variables with length $m = \frac{n}{8}$ as z, which are generated autoregressively. These methods can result in a generation complexity of $O(\frac{n}{8})$, and similar complexity also holds for Wang, Zhang, and Chen (2018) and Stern, Shazeer, and Uszkoreit (2018). Recently, some works (Stern et al. 2019; Welleck et al. 2019) propose to change the generation order from the traditional left-to-right manner to a tree-based manner, resulting in a complexity of $O(\log n)$.

In this paper, we focus on NAT models with generation complexity of $O(1)$ and propose a new perspective of the training paradigm. We consider the training of AT and NAT models easier and harder tasks respectively, and utilize the strategy of fine-tuning by curriculum learning to train the model, i.e., smoothly transferring from the training of AT models to the training of NAT models.

2.2 Transfer Learning

Transfer learning has been extensively studied in machine learning and deep learning (Pan and Yang 2009). For example, the model pre-trained on ImageNet is widely used as the initialization in downstream tasks such as object detection (Girshick et al. 2014) and image segmentation (Long, Shelhamer, and Darrell 2015). On NLP tasks, the pre-trained model such as BERT (Devlin et al. 2018) and MASS (Song et al. 2019) are fine-tuned in many language understanding and generation tasks. In this paper, we find that the pre-training task (AT) and fine-tuning task (NAT) are quite different in the training strategy, where directly fine-tuning results in sub-optimal performance. Therefore, we propose using curriculum learning in the transfer process to achieve a soft landing of the AT models on NAT.

2.3 Curriculum Learning

Humans usually learn better when the curriculums are organized from easy to hard. Inspired by that, Bengio et al. (2009) propose curriculum learning, a machine learning training strategy that feeds training instances to the model from easy to hard. Most works on curriculum learning focus on determining the order of data (Lee and Grauman 2011; Sachan and Xing 2016) or tasks (Pentina, Sharmanska, and Lampert 2015; Sarafianos et al. 2017). In our setting, we design curriculums for neither data samples nor tasks, but the training mechanisms. This way, we make the fine-tuning process smoother and ensure a soft landing from the AT models to NAT models.

3 Fine-Tuning by Curriculum Learning for NAT

In this section, we introduce the proposed method, Fine-tuning by Curriculum Learning for Non-Autoregressive Translation (FCL-NAT). We start with the problem definition, and then introduce the methodology as well as some discussions on our proposed method.

3.1 Problem Definition

Given a source sentence $x \in \mathcal{X}$ and target sentence $y \in \mathcal{Y}$, we consider autoregressive translation (AT) as the source task $\mathcal{T}_S = \{\mathcal{Y}, P(y|z_{\text{AT}}, x)\}$[1], and non-autoregressive translation (NAT) as the target task $\mathcal{T}_T = \{\mathcal{Y}, P(y|z_{\text{NAT}}, x)\}$, where z_{AT} and z_{NAT} are the decoder input of AT and NAT models respectively. Given a bilingual sentence pair (x, y), the conditional probability $P(y|z, x)$ can be written as

$$P(y|z,x) = \prod_{t=1}^{T_y} P(y_t|z,x) = \prod_{t=1}^{T_y} P(y_t|z,x;\theta_{\text{enc}},\theta_{\text{dec}}), \tag{3}$$

[1]We follow the task definition in Pan and Yang (2009), where the tuple consists of a label space $\mathcal{Y}$ and a prediction function $P(y|z_{\text{AT}}, x)$.

where T_y is the length of the target sentence, θ_{enc} and θ_{dec} denote the parameters of the encoder and decoder. For AT models, we denote the decoder input as $z_{\text{AT}} = (y_0, ..., y_{t-1})$, which is the left-shifted target sentence in teacher forcing (Williams and Zipser 1989) training. For NAT models, we denote the decoder input as $z_{\text{NAT}} = (\tilde{x}_1, ..., \tilde{x}_{T_y})$, which is obtained from copying the source sentence x. Note that we do not follow (Gu et al. 2017) which introduces a learnable neural network based fertility predictor to guide the copying process, but utilize a simple and efficient hard copy method which has been used in several previous works (Wang et al. 2019; Guo et al. 2019; Li et al. 2019).

Our objective is to learn an NAT model $\Theta = (\theta_{\text{enc}}, \theta_{\text{dec}})$, utilizing the knowledge learned in the source task $\mathcal{T}_S$ to facilitate learning in the target task $\mathcal{T}_T$, with a curriculum learning way to fine-tune from

$$L_{\text{AT}}(x,y;\Theta) = -\sum_{t=1}^{T_y} \log P(y_t|z_{\text{AT}},x) \tag{4}$$

to

$$L_{\text{NAT}}(x,y;\Theta) = -\sum_{t=1}^{T_y} \log P(y_t|z_{\text{NAT}},x). \tag{5}$$

3.2 Methodology

Generally, the transfer learning procedure for our NAT model can be divided into three stages: 1) AT training, where the model is trained autoregressively as the traditional AT model, and this is equivalent to initializing our model with a pre-trained AT model; 2) Curriculum learning, which is the main stage we focus on and will be introduced with details in this section; 3) NAT training, where we train the model non-autoregressively until convergence. We denote the training steps of the corresponding stages as I_{AT}, I_{CL} and I_{NAT}.

As introduced in Section 1, decoder input and attention mask are the two major differences between AT and NAT models. We describe the curriculums on the two components respectively.

Curriculum for the decoder input For the decoder input, we implement the smooth transfer from z_{AT} to z_{NAT} by progressively substituting the tokens in z_{AT} with the tokens at the same positions of z_{NAT}. Specifically, at the i-th training step, given the AT decoder input z_{AT} and the NAT decoder input z_{NAT}, we first calculate the substitution rate $\alpha_i = f_{\text{sub}}(i) \in [0,1]$, where $f_{\text{sub}}(i)$ is the substitution function which is increased monotonically from 0 to 1 w.r.t the training step i. Then, we randomly select $n_i = \lfloor \alpha_i \cdot T_y \rfloor$ tokens to substitute. We use a binary vector $P_T \in \{0,1\}^{T_y}$ to represent the positions of selected tokens, where $\|P_T\|_1 = n_i$. If the j-th element of P_T equals 1, it indicates the j-th token will be substituted. Then, the decoder input z_i after substitution can be computed formally as:

$$z_i = (1 - P_T) \odot z_{\text{AT}} + P_T \odot z_{\text{NAT}}, \tag{6}$$

where $\odot$ is the element-wise multiplication.

The substitution function $f_{\text{sub}}(i)$ controls how fast the decoder input will be transfered from z_{AT} to z_{NAT}, and we

Pacing Functions	Description
Ladder-like	$f_{\text{ladder}}(i) = \frac{\lfloor \frac{i+1}{K} \rfloor \cdot K}{\mathrm{I_{CL}}}$
Linear	$f_{\text{linear}}(i) = \frac{i+1}{\mathrm{I_{CL}}}$
Logarithmic	$f_{\log}(i) = \frac{\log(i+1)}{\log(\mathrm{I_{CL}})}$

Table 1: The proposed different pacing functions and their definitions.

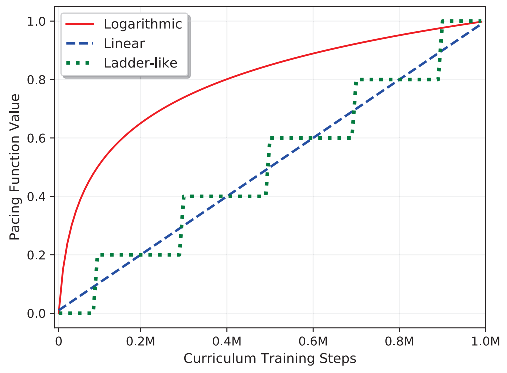

Figure 2: Illustration of the proposed pacing functions. We set $\mathrm{I_{CL}} = 1\mathrm{M}$ here, and $K = 20$ for the ladder-like function, i.e., divide the training stage into 5 sub-stages.

Algorithm 1: Fine-tuning by curriculum learning for NAT (FCL-NAT)

```
Input: The translation model $\Theta = (\theta_{\text{enc}}, \theta_{\text{dec}})$; the
       training set $(\mathcal{X}, \mathcal{Y})$; the AT/NAT decoder input
       $z_{\text{AT}}$ and $z_{\text{NAT}}$, the attention mask $M_{\text{AT}}$ and
       $M_{\text{NAT}}$; the substitution function $f_{\text{sub}}(i)$ which is
       chose from aforementioned functions; the
       maximum curriculum training step $\mathrm{I_{CL}}$; the
       mask switching rate $\alpha_{\mathrm{M}}$.
1  Set the attention mask $M = M_{\text{AT}}$ ;
2  Pretrain the model autoregressively following the loss
     function Equation 4 ;
3  for $i = 1 \ldots I_{CL}$ do
4      Draw a mini-batch of training pairs from $(\mathcal{X}, \mathcal{Y})$ ;
5      Compute the substitution rate $\alpha_i = f_{\text{sub}}(i)$ and
         construct the substitution mask $P_T$ ;
6      if $\alpha_i > \alpha_M$ then
7          Set the attention mask $M = M_{\text{NAT}}$ ;
8      Compute the substitution result $z_i$ following
         Equation 6 ;
9      Train the model with the loss
         $L_{\text{CL}}(x, y; \Theta) = -\sum_{t=1}^{T_y} \log P(y_t | z_i, x)$ on this
         mini-batch ;
10 end
11 Train the model in purely non-autoregressive manner
     until convergence following Equation 5.
```

term it as the pacing function because it works similar to pacing strategies introduced in previous curriculum learning works (Kumar, Packer, and Koller 2010; Hacohen and Weinshall 2019). In this paper, we define three different pacing functions named as ladder-like, linear and logarithmic to make a smooth transformation from z_{AT} to z_{NAT} and verify the effectiveness of the proposed model.

The definition and illustration of these pacing functions are shown in Table 1 and Figure 2 respectively. Correspondingly, the ladder-like function divides the curriculum learning stage into $\frac{\mathrm{I_{CL}}}{K}$ sub-stages, and the substitution rate is fixed within each sub-stage but increased when switching to the next sub-stage. As this is a discrete pacing strategy w.r.t the training step i, we propose the other two continuous functions for comparison. According to their definitions, the difference between linear and logarithmic pacing functions is that $f_{\text{linear}}(i)$ does not show any preference on easier or harder stages but keeps a steady increasing pace, and $f_{\log}(i)$ concentrates more on the harder stage. As different substitution functions reflect different pacing strategies in curriculum learning, we compare and analyze the proposed functions in experiments.

Curriculum for the attention mask The attention mask can be formulated as a zero-one matrix $M \in \{0, 1\}^{T_y \times T_y}$. For AT models, M_{AT} is an upper triangle matrix to prevent the model from attending to future words during training. For NAT models, as the decoder input is the copied source sentence and all target words are generated conditionally independently, and thus the matrix M_{NAT} is a matrix with all ones. We find that there does not exist a natural intermediate state between these two types of attention masks, therefore we choose to directly switch the mask from AT to NAT when the substitution rate exceeds a pre-defined threshold α_{M}.

We summarize the whole procedure of fine-tuning by curriculum learning in Algorithm 1. If we remove the curriculum learning part from line 3 to line 10, it yields the traditional fine-tuning strategy, which will be compared as a baseline in our experiments.

3.3 Discussion

Why token-level substitution? A straightforward idea of implementing the curriculum learning procedure from AT to NAT models is to conduct sentence-level substitution, i.e., we can directly replace the decoder input from z_{AT} to z_{NAT} at the sentence level with probability α_i at the i-th training step. However, the sentence-level substitution is only alternating between two different training strategies, without explicitly providing a transfer to sufficiently leverage the information contained in the intermediate states between AT and NAT models. Our preliminary experiments verify our statement by showing that the performance of sentence-level substitution is inferior to that of token-level substitution under the same setting, and detailed results are listed in the section of experiments.

Why directly switch the attention mask? The decoder input of AT models z_{AT} is the left-shifted target sequence, e.g., the $(i + 1)$-th token is the label of the i-th token, and M_{AT}

prevents the i-th token from seeing its label. However, the attention mask of NAT decoder enables the i-th position to see the tokens in all positions. Therefore, at the early stage of curriculum learning where AT tokens are dominant in the substituted results, if we follow the same token-level substitution mechanism for the attention mask, then the substituted position will see the next AT token which is supposed to be the label of the current position, making the model learn to copy instead of learning to translate. Therefore, we use the attention mask of AT models M_{AT} in the early stage of curriculum learning, and switch to utilize M_{NAT} when there are enough NAT tokens in the substitution results.

4 Experiments and Results

4.1 Experimental Setup

Datasets We evaluate our method on four widely used benchmark datasets: IWSLT14 German to English translation (IWSLT14 De-En) and WMT14 English to German/German to English translation (WMT14 En-De/De-En)[2]. We strictly follow the dataset configurations of previous works (Gu et al. 2017; Guo et al. 2019). Specifically, for the IWSLT14 De-En task, we have $153k/7k/7k$ parallel bilingual sentences in the training/dev/test sets respectively. WMT14 En-De/De-En has a much larger dataset which contains 4.5M training pairs, where `newstest2013` and `newstest2014` are used as the validation and test set respectively. For each dataset, we tokenize the sentences by Moses (Koehn et al. 2007) and segment each word into subwords using Byte-Pair Encoding (BPE) (Sennrich, Haddow, and Birch 2015), resulting in a 32k vocabulary shared by source and target languages.

Model Configurations We follow (Gu et al. 2017; Guo et al. 2019) for the basic configuration of our model, which is based on the Transformer (Vaswani et al. 2017) architecture that consists of multi-head attention and feed-forward networks. We also utilize the multi-head positional attention proposed by (Gu et al. 2017). For WMT14 datasets, we use the hyperparameters of a `base` transformer ($d_{\text{model}} = d_{\text{hidden}} = 512$, $n_{\text{layer}} = 6$, $n_{\text{head}} = 8$). For IWSLT14 datasets, we utilize smaller architectures ($d_{\text{model}} = d_{\text{hidden}} = 256$, $n_{\text{layer}} = 5$, $n_{\text{head}} = 4$) for IWSLT14. Please refer to (Vaswani et al. 2017; Gu et al. 2017) for more detailed settings.

Training and Inference Following previous works (Gu et al. 2017; Lee, Mansimov, and Cho 2018; Guo et al. 2019), we also utilize sequence-level knowledge distillation (Kim and Rush 2016) during training. We first train an AT teacher model which has the same architecture as the NAT student model, then we use the translation results of each source sentence generated by the teacher model as the new ground truth to formulate a new training set. The distilled training set is more deterministic and less noisy, and thus makes the training of NAT models much easier (Gu et al. 2017). We set the beam size to be 4 for the teacher model. While the performance of the AT teacher may influence the performance of the NAT student (Wang et al. 2019), to ensure a fair comparison, we use the autoregressive models of the same performance with that in (Wang et al. 2019) as our teacher models for all datasets. We train the NAT model on $8/1$ Nvidia M40 GPUs for WMT/IWSLT datasets respectively, and follow the optimizer setting in Transformer (Vaswani et al. 2017). We adopt the logarithmic pacing function for the main results. For the three training stages introduced in Section 3.2, we list their settings in Table 2, which are determined by the model performance on the validation sets. We set $\alpha_{\text{M}} = 0.6$ for all tasks. We implement our model on Tensorflow[3], and we have released our code[4].

	WMT14		IWSLT14
	En−De	De−En	De−En
I_{AT}	119k	138k	55k
I_{CL}	0.5M	0.5M	1.0M
I_{NAT}	1.5M	1.5M	2.0M

Table 2: Training steps for the three training stages.

During inference, we utilize Noisy Parallel Decoding (NPD) to generate multiple samples and select the best translation from them, which is also a common practice in previous NAT models (Wang et al. 2019; Guo et al. 2019; Li et al. 2019). Specifically, as we do not know the lengths of target sentences during inference, we generate multiple translation candidates with different target lengths in $T_y \in \big[\lfloor \beta \cdot T_x - B \rfloor, \lfloor \beta \cdot T_x + B \rfloor\big]$ where β is the average ratio between target and source sentence lengths calculated in the training set, and B is half of the searching window of the target length. For example, $B = 0$ represents greedy search. For $B \geq 1$, we first generate $2B + 1$ translation candidates, and then utilize the AT teacher model to score and select the best translation as our final result. As this scoring procedure is fully parallelizable, it will not hurt the non-autoregressive property of the model. In our experiments, we set $\beta = 1.1$ for all English to German tasks, and $\beta = 0.9$ for all German to English tasks. We test with $B = 0$ and $B = 4$ to keep consistent with our baselines (Wang et al. 2019; Guo et al. 2019). We use tokenized case-sensitive BLEU (Papineni et al. 2002) for the WMT14 datasets, and tokenized case-insensitive BLEU for the IWSLT14 dataset, which are all common practices in the literature (Gu et al. 2017; Wang et al. 2019; Guo et al. 2019). For the inference latency, we report the per-sentence decoding latency on the `newstest2014` test set of the WMT14 En-De task, i.e., set the batch size to 1 and calculate the average translation time over all sentences in the test set, which is conducted on a single Nvidia P100 GPU to ensure a fair comparison with baselines (Gu et al. 2017; Wang et al. 2019; Guo et al. 2019).

[2]https://www.statmt.org/wmt14/translation-task

[3]https://github.com/tensorflow/tensor2tensor

[4]https://github.com/lemmonation/fcl-nat

Models	WMT14 En−De	WMT14 De−En	IWSLT14 De−En	Latency	Speedup
Transformer (Vaswani et al. 2017)	27.30	31.29	33.52	607 ms	1.00×
NAT-FT (Gu et al. 2017)	17.69	21.47	20.32[†]	39 ms	15.6×
NAT-FT (NPD 10)	18.66	22.41	21.39[†]	79 ms	7.68×
NAT-FT (NPD 100)	19.17	23.20	24.21[†]	257 ms	2.36×
NAT-IR (Lee, Mansimov, and Cho 2018)	21.61	25.48	23.94[†]	404[†] ms	1.50×
ENAT (Guo et al. 2019)	20.65	23.23	25.09	24 ms	25.3×
ENAT (NPD 9)	24.28	26.67	28.60	49 ms	12.4×
NAT-Reg (Wang et al. 2019)	20.65	24.77	23.89	22 ms	27.6×
NAT-Reg (NPD 9)	24.61	28.90	28.04	40 ms	15.1×
Direct Transfer	20.23	23.16	23.05	21 ms	28.9×
FCL-NAT	21.70	25.32	26.62	21 ms	28.9×
FCL-NAT (NPD 9)	**25.75**	**29.50**	**29.91**	38 ms	16.0×

Table 3: The BLEU scores of our proposed FCL-NAT and the baseline methods on the WMT14 En-De, WMT14 De-En and IWSLT14 De-En tasks. "†" indicates that the result is provided by Wang et al. (2019), and "/" indicates the corresponding result is not reported in the original paper. We report the best results for baseline methods and also list the inference latency as well as the speedup w.r.t autoregressive models. NPD 9 indicates results of noisy parallel decoding with 9 candidates, i.e., $B = 4$, otherwise $B = 0$.

4.2 Results

We compare our model with non-autoregressive baselines including NAT with Fertility (NAT-FT) (Gu et al. 2017), NAT with Iterative Refinement (NAT-IR) (Lee, Mansimov, and Cho 2018), NAT with Enhanced Decoder Input (ENAT) (Guo et al. 2019) and NAT with Auxiliary Regularization (NAT-Reg) (Wang et al. 2019). For NAT-IR, we report their best results with 10 refinement iterations. For ENAT and NAT-Reg, we report their best results when $B = 0$ and $B = 4$ correspondingly. We take Direct Transfer (DT) as another baseline, where we omit the curriculum learning strategy from line 3 to line 10 in Algorithm 1, and train the model in a non-autoregressive manner for extra I_{CL} steps to ensure a fair comparison.

The main results of this paper are listed in Table 3. Our method FCL-NAT achieves significant improvements over all NAT baselines on different tasks. Specifically, note that although we do not introduce any auxiliary loss functions or new parameters, we outperform NAT-Reg and ENAT with a large margin, which demonstrates the superiority of the proposed fine-tuning by curriculum learning method. Compared with Direct Transfer, FCL-NAT brings a large improvement on translation accuracy, demonstrating the importance of the progressive transfer between two tasks with curriculum learning. As for the inference efficiency, we achieve a 16.0 times speedup, which is comparable with NAT-Reg and ENAT.

4.3 Analyses

Comparison with Direct Transfer We compare the training curve of our proposed FCL-NAT with Direct Transfer (DT). We evaluate FCL-NAT with different pacing functions proposed in Table 1, as well as DT on the validation set of the IWSLT14 German-to-Engish task, and plot the training curves in Figure 3. We can find that the NAT models trained with different curriculum mechanisms (pacing functions) achieve better translation accuracy than the model trained with DT.

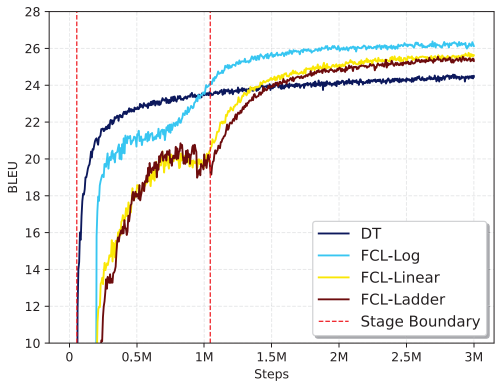

Figure 3: The comparison of BLEU scores on the validation set of IWSLT14 De-En task among different pacing functions proposed in Table 1 as well as the direct transfer (DT) baseline. We set $B = 0$ here. The two red dashed vertical lines indicate the boundary of three training stages, i.e., AT training, curriculum learning, and NAT training from left to right.

As described in Section 3.2, the training process of the proposed method can be divided into three stages: AT training, curriculum learning and NAT training. An interesting finding from Figure 3 is that although the accuracy of our method is worse than DT in the first two stages, it finally becomes higher than DT in the NAT training stage. The worse performance in the first two stages is due to that the train-

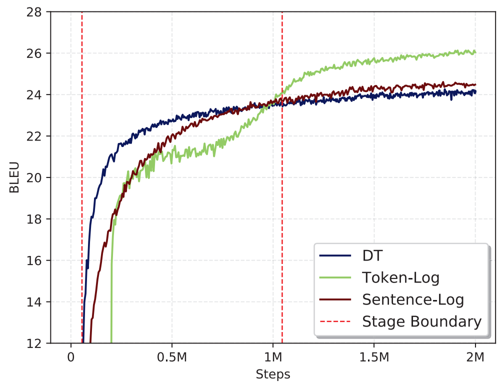

Figure 4: The comparison of BLEU scores on the validation set of the IWSLT14 De-En task between token-level and sentence-level substitution strategy as well as the direct transfer (DT) baseline. We choose the logarithmic pacing function for both substitution strategies and term them as Token-Log and Sentence-Log. The two red dashed vertical lines indicate the boundary of the three training stages, and we set $B = 0$ here.

ing in the first two stages of our method is not consistent with the NAT inference, while there is no such issue for DT. However, our method has a higher increasing rate in the final NAT training stage and eventually outperforms DT. Clearly, the proposed transfer learning by curriculum learning strategy helps the model find a better initial point for NAT training. Due to the difference between AT and NAT models, direct initializing by AT model is far from enough for an NAT model. Our method is able to achieve a smooth transfer between AT and NAT models.

Comparison of Different Pacing Functions The performance of different pacing functions can also be found in Figure 3. For the ladder-like pacing function, we set $K = 10$ to divide the curriculum learning stage to 10 substages. We have two observations: 1) the ladder-like and linear pacing functions result in similar accuracy curves, and 2) the logarithmic pacing function outperforms the above two functions. These observations indicate that 1) whether the pacing function is discrete or continuous does not influence the performance much in a constant pacing strategy, and 2) the logarithmic pacing function results in more training steps with larger substitution rates, which achieves a good trade-off between leveraging the information of AT models and training non-autoregressively, and thus demonstrates better performance.

Study on Sentence-Level Substitution As stated in Section 3.3, a straightforward way to implement our idea is to directly replace the decoder input from z_{AT} to z_{NAT} when conducting substitution, termed as sentence-level substitution. We compare token-level and sentence-level substitution strategies on the validation set of the IWSLT14 De-En task, and keep other settings aligned, i.e., we choose the logarithmic pacing function and set $\mathrm{I}_{\mathrm{AT}} = 55\mathrm{k}$, $\mathrm{I}_{\mathrm{CL}} = 1.0\mathrm{M}$, $\mathrm{I}_{\mathrm{NAT}} = 0.5\mathrm{M}$ in both settings. The results are shown in

NAT-FT	NAT-Reg	FCL-NAT
2.30	0.90	**0.57**

Table 4: The comparison on the average number of per-sentence repetitive tokens on the validation set of the IWSLT14 De-En task.

Figure 4. Both token-level and sentence-level substitution strategies outperform the direct transfer baseline, which further demonstrates the efficacy of the proposed fine-tuning by curriculum learning methodology. In addition, the token-level substitution outperforms the sentence-level substitution by a large margin, showing that it is crucial to leverage the information provided by the intermediate states during transferring from a task to another.

Study on Repetitive Tokens As pointed out by Wang et al. (2019), a typical translation error of the basic NAT model is translating repetitive tokens, and they propose an auxiliary regularization function to explicitly address the problem. While our proposed FCL-NAT is not specifically designed to deal with this issue, we find it also alleviates this problem as a byproduct. We calculate the average number of consecutive repetitive tokens in a sentence on the validation set of IWSLT14 De-En, and the results are listed in Table 4. We observe that without an explicit regularization, our model is still able to reduce repetitive tokens more effectively.

5 Conclusion

In this paper, we propose a novel fine-tuning by curriculum learning method for non-autoregressive neural machine translation, which progressively transfers the knowledge learned in AT models into NAT models. We consider AT training as a source and easier task and NAT training as a target and harder task, and designed a curriculum to gradually substitute the decoder input and attention mask in an AT decoder with that in an NAT decoder. Experiments on four benchmark datasets demonstrate the effectiveness of our proposed method for non-autoregressive translation.

In the future, we will extend our idea to other tasks such as text-to-speech and image-to-image translation. As long as there exists a smooth transformation between the source task and the target task, similar ideas can be applied to leverage the intermediate states between the two tasks. In addition, it is also interesting to explore the theoretical explanation of the proposed model.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (No. 61673364, No. U1605251) and the Fundamental Research Funds for the Central Universities (WK2150110008). The authors would like to thank Information Science Laboratory Center of USTC for the hardware and software services.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 41–48. ACM.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 580–587.

Gu, J.; Bradbury, J.; Xiong, C.; Li, V. O.; and Socher, R. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Guo, J.; Tan, X.; He, D.; Qin, T.; Xu, L.; and Liu, T.-Y. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *AAAI*.

Hacohen, G., and Weinshall, D. 2019. On the power of curriculum learning in training deep networks. *arXiv preprint arXiv:1904.03626*.

Hassan, H.; Aue, A.; Chen, C.; Chowdhary, V.; Clark, J.; Federmann, C.; Huang, X.; Junczys-Dowmunt, M.; Lewis, W.; Li, M.; et al. 2018. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

He, T.; Tan, X.; Xia, Y.; He, D.; Qin, T.; Chen, Z.; and Liu, T.-Y. 2018. Layer-wise coordination between encoder and decoder for neural machine translation. In *NIPS*.

Kaiser, Ł.; Roy, A.; Vaswani, A.; Pamar, N.; Bengio, S.; Uszkoreit, J.; and Shazeer, N. 2018. Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Kim, Y., and Rush, A. M. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on interactive poster and demonstration sessions*, 177–180.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. In *NIPS*.

Lee, Y. J., and Grauman, K. 2011. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, 1721–1728.

Lee, J.; Mansimov, E.; and Cho, K. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Li, Z.; Lin, Z.; He, D.; Tian, F.; Qin, T.; Wang, L.; and Liu, T.-Y. 2019. Hint-based training for non-autoregressive translation. *arXiv preprint arXiv:1909.06708*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

Pan, S. J., and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 311–318.

Pentina, A.; Sharmanska, V.; and Lampert, C. H. 2015. Curriculum learning of multiple tasks. In *CVPR*, 5492–5500.

Ren, Y.; Ruan, Y.; Tan, X.; Qin, T.; Zhao, S.; Zhao, Z.; and Liu, T.-Y. 2019. Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*.

Sachan, M., and Xing, E. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, 453–463.

Sarafianos, N.; Giannakopoulos, T.; Nikou, C.; and Kakadiaris, I. A. 2017. Curriculum learning for multi-task classification of visual attributes. In *ICCV*, 2608–2615.

Sennrich, R.; Haddow, B.; and Birch, A. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Shen, Y.; Tan, X.; He, D.; Qin, T.; and Liu, T.-Y. 2018. Dense information flow for neural machine translation. In *NAACL-HLT 2018*, 1294–1303.

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T.-Y. 2019. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*.

Stern, M.; Chan, W.; Kiros, J.; and Uszkoreit, J. 2019. Insertion transformer: Flexible sequence generation via insertion operations. *arXiv preprint arXiv:1902.03249*.

Stern, M.; Shazeer, N.; and Uszkoreit, J. 2018. Blockwise parallel decoding for deep autoregressive models. In *NIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Wang, Y.; Tian, F.; He, D.; Qin, T.; Zhai, C.; and Liu, T.-Y. 2019. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*.

Wang, C.; Zhang, J.; and Chen, H. 2018. Semi-autoregressive neural machine translation. *arXiv preprint arXiv:1808.08583*.

Welleck, S.; Brantley, K.; Daumé III, H.; and Cho, K. 2019. Non-monotonic sequential text generation. *arXiv preprint arXiv:1902.02192*.

Williams, R. J., and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.