Task-Level Curriculum Learning for Non-Autoregressive Neural Machine Translation

Jinglin Liu[1*], **Yi Ren**[1*], **Xu Tan**[2], **Chen Zhang**[1], **Tao Qin**[2], **Zhou Zhao**[1†] and **Tie-Yan Liu**[2]
[1]Zhejiang University
[2]Microsoft Research Asia
{jinglinliu,rayeren,zc99,zhaozhou}@zju.edu.cn,
{xuta,taoqin,tyliu}@microsoft.com

Abstract

Non-autoregressive translation (NAT) achieves faster inference speed but at the cost of worse accuracy compared with autoregressive translation (AT). Since AT and NAT can share model structure and AT is an easier task than NAT due to the explicit dependency on previous target-side tokens, a natural idea is to gradually shift the model training from the easier AT task to the harder NAT task. To smooth the shift from AT training to NAT training, in this paper, we introduce semi-autoregressive translation (SAT) as intermediate tasks. SAT contains a hyperparameter k, and each k value defines a SAT task with different degrees of parallelism. Specially, SAT covers AT and NAT as its special cases: it reduces to AT when $k = 1$ and to NAT when $k = N$ (N is the length of target sentence). We design curriculum schedules to gradually shift k from 1 to N, with different pacing functions and number of tasks trained at the same time. We called our method as task-level curriculum learning for NAT (TCL-NAT). Experiments on IWSLT14 De-En, IWSLT16 En-De, WMT14 En-De and De-En datasets show that TCL-NAT achieves significant accuracy improvements over previous NAT baselines and reduces the performance gap between NAT and AT models to 1-2 BLEU points, demonstrating the effectiveness of our proposed method.

1 Introduction

Neural Machine Translation (NMT) has witnessed rapid progress in recent years [Bahdanau *et al.*, 2015; Gehring *et al.*, 2017; Vaswani *et al.*, 2017]. Typically, NMT models adopt the encoder-decoder framework [Bahdanau *et al.*, 2015], and the decoder generates a target sentence in an autoregressive manner [Bahdanau *et al.*, 2015; Vaswani *et al.*, 2017], where the generation of the current token depends on previous tokens and the source context from the encoder. While the accuracy of NMT models achieve human parity, they usually suffer from high inference latency due to autoregressive generation. Therefore, non-autoregressive translation (NAT) [Gu *et al.*, 2018; Guo *et al.*, 2019a; Wang *et al.*, 2019; Ma *et al.*, 2019; Ren *et al.*, 2019] has been proposed to generate target tokens in parallel, which can greatly speed up the inference process.

However, the accuracy of NAT models still lag behind that of the autoregressive translation (AT) models, due to the previous target tokens are removed from conditional dependency. A variety of works have tried to improve the accuracy of NAT, including enhanced decoder input with embedding mapping [Guo *et al.*, 2019a], generative flow [Ma *et al.*, 2019], and iterative refinement [Ghazvininejad *et al.*, 2019; Lee *et al.*, 2018], etc. However, none of these works leverage the task relationship between AT and NAT when designing their methods. As AT models are more accurate and easier to train than NAT models due to the explicit dependency on previous tokens, a natural idea is to first train the model with easier AT, and then continue to train it with harder NAT.

AT and NAT can be regarded as two tasks that are far different from each other, which makes it less beneficial to directly shift to NAT training right after AT training. How to smoothly shift the model training from AT to NAT is critical for the final accuracy. In this paper, we introduce semi-autoregressive translation (SAT) [Wang *et al.*, 2018], which only generates a part of the tokens in parallel at each decoding step, as intermediate tasks to bridge the shift process from AT to NAT. Specifically, we define a parameter k to represent the degree of parallelism for each task, and view different tasks under a unified perspective: $k = 1$ represents AT, $k = N$ represents NAT where N is the length of target sentence, and $1 < k < N$ represents SAT. Intuitively, a task with smaller k is easier to train and achieves higher accuracy, while that with larger k is harder to train and results in worse accuracy [Wang *et al.*, 2018], which forms a good curriculum to train the model from easy to hard.

Inspired by this, we propose a task-level curriculum learning for non-autoregressive translation (TCL-NAT), which trains the model with sequentially increased k. We divide the training procedure into three phases: AT training ($k = 1$), SAT training ($1 < k < N$) and NAT training ($k = N$). SAT training consists of multiple stages, where we shift k gradually and exponentially as $k = 2, 4, 8, ..., 16$. To find the best schedule strategy to shift k, we design different pacing func-

[*]Equal contribution.
[†]Corresponding author

tions to control the training steps for each k, including linear, logarithmic and exponential functions. On the other hand, to smooth the shift process and reduce the gap between different stages, we further introduce a parameter called task window w, which represents the number of tasks training at the same time in each stage. For example, when $w = 2$, we train the model with $k = 1, 2$ for the first stage and $k = 2, 4$ for the second stage, and so on.

We implement TCL-NAT on Transformer model [Vaswani *et al.*, 2017]. In order to support different k in the same model, we introduce a causal-k self-attention mechanism in the Transformer decoder. We conduct experiments on four translation datasets including IWSLT14 German-English (De-En), IWSLT16 English-German (En-De), WMT14 English-German (En-De) and WMT14 German-English (De-En) to demonstrate the effectiveness of our method. The experiment results show that our method can achieve significant improvement over NAT baselines and also outperform state of the art NAT models, without sacrificing the inference speed. Specifically, we outperform the state of art NAT model [Guo *et al.*, 2019b] by 1.88 BLEU on the IWSLT14 De-En task, and reduce the accuracy gap between AT and NAT models to nearly 1 BLEU point on IWSLT16 En-De and WMT14 En-De tasks.

2 Related Work

In this section, we first introduce the related works on neural machine translation, including autoregressive translation (AT), non-autoregressive translation (NAT) and semi-autoregressive translation (SAT), and then describe three learning paradigms: transfer learning, multitask learning and curriculum learning, which are related to our method.

2.1 Neural Machine Translation (AT/NAT/SAT)

An autoregressive translation (AT) model takes source sentence s as input and then generates the tokens of target sentence y one by one during the inference process [Bahdanau *et al.*, 2015; Sutskever *et al.*, 2014; Vaswani *et al.*, 2017], which causes much inference latency. To improve the inference speed of AT models, a series of works develop non-autoregressive translation (NAT) models based on Transformer [Gu *et al.*, 2018; Lee *et al.*, 2018; Li *et al.*, 2019; Wang *et al.*, 2019; Guo *et al.*, 2019a], which generate all the target tokens in parallel. Several works introduce auxiliary components or losses to improve the accuracy of NAT models: 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 tokens; Guo et al. [2019a] try to enhance the decoder input with target-side information by leveraging auxiliary information; Ma et al. [2019] introduce generative flow to directly model the joint distribution of all target tokens simultaneously. While NAT models achieve faster inference speed, the translation accuracy is still worse than AT model. Some works aim to balance the translation accuracy and inference latency between AT and NAT by introducing semi-autoregressive translation (SAT) [Wang *et al.*, 2018], which generates multiple adjacent tokens in parallel during the autoregressive generation.

Different from the above works, we leverage AT, SAT and NAT together and schedule the training in a curriculum way to achieve better translation accuracy for NAT.

2.2 Transfer/Multitask/Curriculum Learning

Our proposed TCL-NAT actually leverages the knowledge from easier and more accurate tasks $k < N$ to help the task $k = N$, but uses a curriculum schedule and trains multiple tasks at a stage. In general, our work is related to three different learning paradigms: transfer learning, multitask learning and curriculum learning.

Transfer learning has been a common approach for NLP tasks. Pre-trained models such as BERT [Devlin *et al.*, 2019] and MASS [Song *et al.*, 2019] are fine-tuned on many language understanding and generation tasks for better accuracy. Many NAT works [Gu *et al.*, 2018; Lee *et al.*, 2018; Guo *et al.*, 2019a; Guo *et al.*, 2019b] employ sequence level data distillation to transfer the knowledge from AT teacher model to NAT student model which has proved to be effective.

Multitask learning [Caruana, 1997] has found extensive usage in NLP tasks. Dong et al. [2015] use multitask learning for multiple language translation. Anastasopoulos and Chiang [2018] explore multitask models for neural translation of speech and find that jointly trained models improve performance on the tasks of low-resource speech transcription and translation. Garg et al. [2019] leverage extracted discrete alignments in a multi-task framework to optimize towards translation and alignment objectives.

Inspired by the human learning process, curriculum learning [Bengio *et al.*, 2009] is proposed as a machine learning training strategy by feeding training instances to the model from easy to hard. Most of the works on curriculum learning focus on the determining the orders of data [Lee and Grauman, 2011; Sachan and Xing, 2016]. Later, some works explore the curriculum learning strategies in task level. Previous work [Sarafianos *et al.*, 2017] in computer visual domain splits the tasks into groups according to the correlation and transfers the acquired knowledge from strongly correlated task to weakly correlated one.

Guo et al. [2019b] propose a fine-tuning method to transfer a well-trained AT model to a NAT model by designing a curriculum in the shift process between two kinds of models, which is perhaps the most similar work to ours. However, the training strategy during their curriculum learning process is not a natural task, but just some hand-crafted training strategies, which could affect the final transfer accuracy and the total training time. In contrast, each intermediate task during our curriculum learning process is a standard translation task and is empirically verified to be helpful to the consequent tasks.

3 Task-Level Curriculum Learning For NAT

In this section, we introduce our proposed task-level curriculum learning for NAT (TCL-NAT) in detail. First, we propose a unified perspective to represent different tasks including AT, SAT and NAT with a parameter k. Second, we empirically demonstrate the task with smaller k can help the task with

bigger k. Third, we introduce the task-level curriculum learning mechanism based on the unified perspective. Finally, we describe the design of our model architecture for TCL-NAT.

3.1 A Unified Perspective for AT/SAT/NAT

We propose a new perspective to view AT, SAT and NAT as to generate target tokens in an autoregressive manner during the whole sentence translation, but generate k adjacent tokens in parallel at a time. Specifically, given a source and target sentence pair $(x, y) \in (\mathcal{X}, \mathcal{Y})$, we factorize the conditional probability $P(y|x)$ according to the chain rule:

$$P(y|x) = \prod_{t=0}^{\lfloor N/k \rfloor} \prod_{j=1}^{k} P(y_{tk+j}|y_{<tk+1}, x; \theta), \tag{1}$$

where N is the length of the target sequence, k denotes the number of tokens that generated in parallel in a decoding step, $\lfloor \cdot \rfloor$ denotes the floor operation, θ denotes the parameters of the model. In the above equation, y_t where $t < 1$ or $t > N$ represents invalid tokens, which is introduced to make our formulation simple.

Under this perspective, we regard each k as an individual task. As special cases, when $k = 1$, the equation becomes:

$$P(y|x) = \prod_{t=0}^{N} P(y_{t+1}|y_{<t+1}, x; \theta), \tag{2}$$

which is exactly the conditional probability for AT; when $k = N$, the equation becomes:

$$P(y|x) = \prod_{j=1}^{N} P(y_j|x; \theta), \tag{3}$$

which is the conditional probability for NAT; when $1 < k < N$, the equation represents the conditional probability for SAT.

3.2 A Preliminary Study

Based on this perspective of k, we train multiple models[1] with task $k = 1, 2, 4, 8, 16, N$ respectively and test them on the test set of IWSLT14 De-En dataset with different k. We have some analyses and observations:

- The task with smaller k is easier to train and achieves higher accuracy, while that with larger k is harder to train and achieves slightly worse accuracy [Wang *et al.*, 2018], which forms a good curriculum way that shifts the task from easier to harder. The italic numbers in Table 1 show that when training and testing the models with the same k, larger k leads to worse accuracy.
- The model trained with task $k < N$ can bring advantages to the model training of task $k = N$, which can be supported in our experiments: we train multiple models with task $k = 1, 2, 4, 8, 16$ respectively, and then test the translation accuracy of these models with task $k = N$ (NAT). We found most of the models (trained with $k = 4, 8, 16$) can achieve reasonable accuracy on NAT, as shown in Table 1.
- When testing the accuracy of task $k = N$, the model trained with task $k = k' < N$ brings more advantages than that trained with task $k < k'$. Similarly, when testing the accuracy of task $k = k'$, the model trained with task $k = k'' < k'$ also provides a better initialization than that trained with task $k < k''$. Similar results can be got for smaller k recursively. We can see from the bold numbers in Table 1 that when testing the accuracy of task $k = N$, the model trained with $k = 16$ achieves the best, when testing that of task $k = 16$, the model trained with $k = 8$ achieves the best and so on.

Train / Test	k=1	k=2	k=4	k=8	k=16	k=N
k'=N	0.28	1.35	6.39	19.38	**23.78**	*24.89*
k'=16	0.00	0.68	11.17	**20.11**	*24.97*	/
k'=8	0.17	1.54	**12.87**	*28.6*	/	/
k'=4	0.34	**4.07**	*31.27*	/	/	/
k'=2	0.86	*33.2*	/	/	/	/
k'=1	*34.8*	/	/	/	/	/

Table 1: The BLEU scores on the test set of IWSLT14 De-En task. The model is trained with k for 80k steps but test with another k'. The italic numbers show the accuracy of models that train and test with the same k. Row 1 shows that models trained with task $k = 4, 8, 16$ can achieve reasonable accuracy on NAT. The bold numbers show that models trained with task $k = k'' < k'$ can achieve better scores than that trained with task $k < k''$ when testing the accuracy of task $k = k'$.

3.3 Task-Level Curriculum Learning

Based on the unified perspective and the observations in the last subsections, we propose task-level curriculum learning for NAT (TCL-NAT). Our method consists of three phases:

- AT training. We first train a model with $k = 1$ (AT) as the initial model.
- SAT training. We then continue to train the model with increasing k sequentially. For simplicity, we shift k gradually and exponentially as $k = 2, 4, 8, 16$ considering the length distribution of the dataset.
- NAT training. We continue to train the model with $k = N$ till convergence to obtain the final NAT model.

Next, we elaborate the curriculum learning mechanism from two aspects: the pacing function for curriculum schedule and task window. Inspired by Guo et al. [2019b], the pacing function is used to depict the curriculum scheduling strategy introduced in previous curriculum learning work, which controls the training step for each stage in SAT training phase. The task window represents the number of tasks trained at each stage, which is regarded as an extension to our method that only trains the model with one task at each stage.

Pacing Functions. Specifically, at the i-th training step, we choose $k = f(i) \in \{2, 4, 8, 16\}$, where $f(i)$ is the pacing function w.r.t the training step i. We define three different pacing functions: linear $f_{\text{linear}}(i)$, logarithmic $f_{\log}(i)$ and exponential $f_{\exp}(i)$ to make a smooth transformation from

[1] We use TCL-NAT model setting which is introduced in Section 3.4 for this preliminary study.

AT to NAT. These pacing functions divide the SAT training phase into several stages and there are clear differences among them: linear pacing function is the simplest pacing function where each stage is trained with same steps; logarithmic pacing function results in more training steps with tasks whose k is larger and exponential pacing function shows more preference on tasks with smaller k. Since different pacing functions reflect different curriculum learning strategies, we conduct experiments to compare and analyze these proposed pacing functions. The detailed definitions and analysis of these pacing functions are shown in Section 4.3.

Curriculum Learning with Task Window. If we just use the above mechanism for training, although the tasks between stages are similar and close, there still exists a gap between the tasks in neighboring stages. In order to reduce the gap between the stages, we try to extend the task-level curriculum learning with task window. Specifically, we define a task window w to represent the number of tasks training at the same time at each stage. The situation we discussed before is corresponding to $w = 1$. When $w = 2$, we train the model with two tasks $k = 1, 2$ at the same time in the first stage, then with two tasks $k = 2, 4$ at the same time in the second stage, and so on until the last stage train with $k = 16, N$. When $w = 3$, our model is trained with three tasks $k = 1, 2, 4$ in the first stage, and so on. In this way, the training tasks in the two neighboring training stages have overlaps when $w > 1$, which is smoother for task shift than $w = 1$ intuitively.

3.4 Model Structure for TCL-NAT

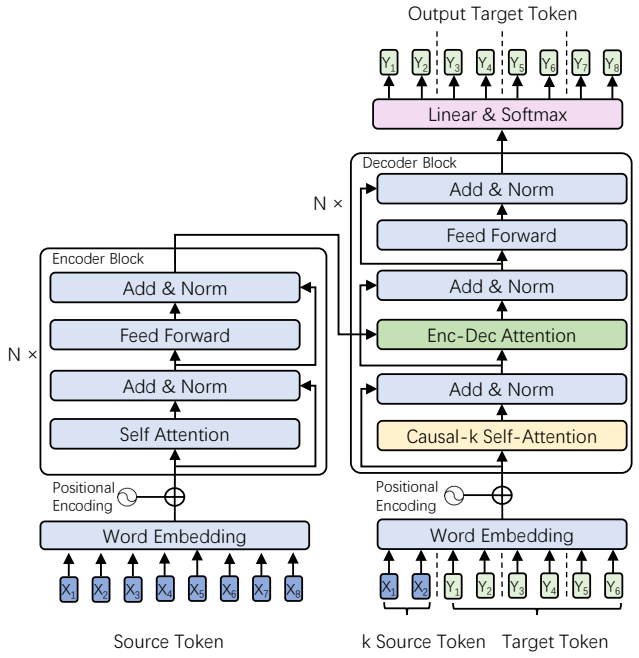

Figure 1: The overview of the model structure for TCL-NAT. This figure shows the case with $k = 2$.

To support different k in the same model, we leverage Transformer model with a causal-k self-attention mechanism in the decoder [Wang *et al.*, 2018]. Note that although we only apply the task-level curriculum learning technique in Transformer-based model, it can also be easily applied to other non-autoregressive architectures such as CNN. The whole model architecture of TCL-NAT is shown in Figure 1. The encoder of TCL-NAT is exactly the same as the basic structure of Transformer [Vaswani *et al.*, 2017], which uses stacked self-attention and fully connected layers, shown in the left panel of the Figure 1. For the decoder, we introduce a causal-k self-attention mechanism which can generate k successive tokens in parallel. As shown in the right panel of Figure 1, our decoder is similar with the decoder in SAT [Wang *et al.*, 2018], except that we feed the model with the first k source tokens $(x_1, ...x_k)$ rather than special tokens to predict $(y_1, ..., y_k)$ in parallel at the beginning of decoding, in order to keep consistent with NAT model when $k = N$. Then $(y_1, ..., y_k)$ are fed to the model to predict $(y_{k+1}..., y_{2k})$ in parallel. As a result, the decoder input can be denoted as $(x_1, ..., x_k, y_1, ...y_{N-k})$. We also adopt a causal-k mask in the decoder self-attention following Wang et al. [2018], as shown in Figure 2.

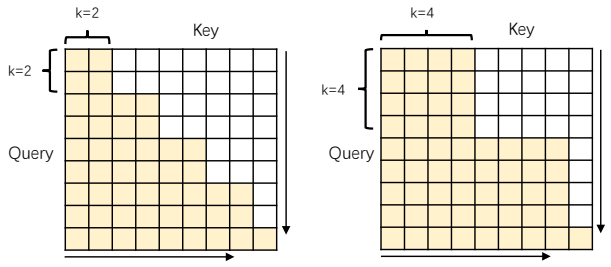

Figure 2: Causal-k self-attention mask for an intuitive understanding. Yellow grids denote elements 1 and white grids denote elements 0 in the decoder self-attention mask. The left subfigure shows the case with $k = 2$ and the right one shows the case with $k = 4$.

Under this model structure, we can utilize the first k source tokens $(x_1, ..., x_k)$ to predict sub-sentence $(y_1, ...y_k)$ in parallel, and then utilize them to predict sub-sentence $(y_{k+1}, ..., y_{2k})$ in parallel and so on. Then it comes the conditional probability in Equation 1. As the k increases, the model's dependency on target tokens decreases. Specially, when $k = 1$ the decoder is an autoregressive decoder, and when k is large enough, the decoder becomes a non-autoregressive decoder which generates all outputs simultaneously depending on the source tokens only. In the inference stage, we set k to N and make the decoder run in the NAT mode.

4 Experiments and Results

4.1 Experiments Settings

Datasets. We evaluate our method on three standard translation datasets: IWSLT14 German-to-English (De-En) dataset[2], IWSLT16 English-to-German (En-De) dataset[3] and WMT14 English-to-German (En-De) dataset[4]. Following Li et al. [2019], we reverse WMT14 English-to-German to

[2]https://wit3.fbk.eu/mt.php?release=2014-01
[3]https://wit3.fbk.eu/mt.php?release=2016-01
[4]https://www.statmt.org/wmt14/translation-task.html

get WMT14 German-to-English (De-En) dataset. In details, IWSLT14 dataset contains $153k/7k/7k$ parallel bilingual sentences for training/dev/test set respectively; IWSLT16 dataset contains $195k/1k/1k$ parallel bilingual sentences for training/dev/test set and WMT14 dataset contains 4.5M parallel sentence pairs for training sets, where newstest2014 and newstest2013 are used as test and validation set respectively, following previous works [Gu *et al.*, 2018; Guo *et al.*, 2019b]. We split each token into subwords using Byte-Pair Encoding (BPE) [Sennrich *et al.*, 2016] and set $10k$, $10k$ and $40k$ as the vocabulary size for IWSLT14, IWSLT16 and WMT14 respectively. The vocabulary is shared by source and target languages in those datasets.

Model Configuration. We adopt the basic NAT model configuration [Gu *et al.*, 2018; Guo *et al.*, 2019b] based on Transformer [Vaswani *et al.*, 2017], which is composed by multi-head attention modules and feed forward networks. We follow Guo et al. [2019b] for configuration hyperparameters: 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 and IWSLT16 datasets, we utilize a `small` Transformer ($d_{\text{model}} = d_{\text{hidden}} = 256$, $n_{\text{layer}} = 6$, $n_{\text{head}} = 4$).

Training. Following previous works [Gu *et al.*, 2018; Guo *et al.*, 2019a; Wang *et al.*, 2019], we employ sequence-level knowledge distillation [Kim and Rush, 2016] during training to reduce the difficulty of training and boost the accuracy through constructing a more deterministic and less noisy training set: first we train an AT teacher model which has the same architecture as the TCL-NAT model, and then we use the translation results of each source sentence generated by the teacher model as the new ground truth to construct a new training set for further training. We design three different pacing functions mentioned in Section 3.3 and detail their definitions in Table 3. We set task window w to 2 by default, which is determined by the model performance on the validation sets. We train all models using Adam following the optimizer settings and learning rate schedule in Transformer [Vaswani *et al.*, 2017]. We run the training procedure on 8 NVIDIA Tesla P100 GPUs for WMT and 2 NVIDIA 2080Ti GPUs for IWSLT datasets respectively. The training steps of each phase are listed in Table 2. We implement our model on Tensor2Tensor [Vaswani *et al.*, 2018].

Phases	IWSLT14 De-En	IWSLT16 En-De	WMT14 De-En	WMT14 En-De
AT	0.08M	0.08M	0.16M	0.16M
SAT	0.24M	0.24M	0.32M	0.32M
NAT	0.32M	0.32M	0.64M	0.64M

Table 2: The training steps of TCL-NAT for different datasets for each phase.

Inference and Evaluation. For inference, we adopt the common method of noisy parallel decoding (NPD) [Gu *et al.*, 2018], which generates a number of decoding candidates in parallel and selects the best translation by AT teacher model re-scoring. In our work, we generate multiple translation candidates by predicting different target lengths $N \in [M - B, M + B]$ (M is the length of the source sentence), which results in $2B + 1$ candidates. We test with $B = 0$ and $B = 4$ (denoted as NPD 9) to keep consistent with our baselines [Wang *et al.*, 2019; Guo *et al.*, 2019a; Guo *et al.*, 2019b]. We evaluate the translation quality by tokenized case sensitive BLEU [Papineni *et al.*, 2002] with multi-bleu.pl[5]. Inference and evaluation are run on 1 Nvidia P100 GPU for WMT14 En-De datasets in order to keep in line with previous works for testing latency.

Name	Description
Linear	$f_{\text{linear}}(i) = \min(2^{\lfloor 4i/S_{\text{SAT}} \rfloor + 1}, 16)$
Logarithmic	$f_{\log}(i) = \min(2^{\lfloor log_{1.5}(4i/S_{\text{SAT}} + 1) \rfloor + 1}, 16)$
Exponential	$f_{\exp}(i) = \min(2^{\lfloor 1.5^{4i/S_{\text{SAT}}} \rfloor}, 16)$

Table 3: The proposed different curriculum pacing functions and their definitions. S_{SAT} denotes the total steps in SAT training phase. We choose constants empirically to meet the actual training situation.

4.2 Results

We compare TCL-NAT with non-autoregressive baselines including NAT-FT [Gu *et al.*, 2018], NAT-IR [Lee *et al.*, 2018], ENAT [Guo *et al.*, 2019a], NAT-Reg [Wang *et al.*, 2019], FlowSeq [Ma *et al.*, 2019] and FCL-NAT [Guo *et al.*, 2019b]. For NAT-IR, we report their best results when refinement steps is 10. For ENAT, NAT-Reg and FCL-NAT, we report their best results with B = 0 and B = 4 correspondingly. For FlowSeq, we report their results without NPD. It is worth noting that we mainly compare our method with existing methods that have similar speed-up, so Mask-Predict [Ghazvininejad *et al.*, 2019], LevT [Gu *et al.*, 2019] and FlowSeq-large are not included into discussion.

We list the main results of our work in Table 4. We can see that TCL-NAT achieves significant improvements over all NAT baselines on different datasets. Specifically, we outperform ENAT and NAT-Reg with a notable margin. In addition, compared with NAT-Reg, we do not introduce any auxiliary loss functions in training stage and compared with ENAT, we just copy the source sentence as the decoder input, which does not add the extra workload in inference stage. Compared with FlowSeq, our method (without NPD) achieves better scores on most datasets with a much larger speedup. We also outperform FCL-NAT on most datasets with a less training steps. As for the inference efficiency, we achieve a 16.0 times speedup (NPD 9), which is comparable with state of the art methods (FCL-NAT and ENAT).

4.3 Analyses

Comparison with Direct Transfer. We take Direct Transfer (DT) as another baseline, where we omit the SAT stage in Section 3.3, and train the model in a non-autoregressive manner for the same steps as TCL-NAT to ensure a fair comparison. We test DT model on the test set of IWSLT14 De-En task and obtain the BLEU score of 27.00, while our method

[5]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

Models	IWSLT14 De-En	IWSLT16 En-De	WMT14 De-En	WMT14 En-De	Latency	Speedup
Autoregressive Models (AT Teachers)						
Transformer [Vaswani *et al.*, 2017]	33.90	30.32	31.38	27.30	607 ms	1.00 ×
Non-Autoregressive Models						
NAT-FT [Gu *et al.*, 2018]	/	26.52	21.47	17.69	39 ms	15.6 ×
NAT-FT (NPD 10)	/	27.44	22.41	18.66	79 ms	7.68 ×
NAT-IR [Lee *et al.*, 2018]	/	27.11	25.48	21.61	404 ms	1.50 ×
ENAT [Guo *et al.*, 2019a]	25.09	/	23.23	20.65	24 ms	25.3 ×
ENAT (NPD 9)	28.60	/	26.67	24.28	49 ms	12.4 ×
NAT-Reg [Wang *et al.*, 2019]	23.89	23.14	24.77	20.65	22 ms	27.6 ×
NAT-Reg (NPD 9)	28.04	27.02	28.90	24.61	40 ms	15.1 ×
FlowSeq-base [Ma *et al.*, 2019]	27.55	/	26.16	21.45	/	5.94 ×
FCL-NAT [Guo *et al.*, 2019b]	26.62	/	25.32	21.70	21 ms	28.9 ×
FCL-NAT (NPD 9)	29.91	/	29.50	**25.75**	38 ms	16.0 ×
TCL-NAT	28.16	26.01	25.62	21.94	22 ms	27.6 ×
TCL-NAT (NPD 9)	**31.79**	**29.30**	**29.60**	25.37	38 ms	16.0 ×

Table 4: The BLEU scores of our proposed TCL-NAT and the baseline methods on the IWSLT14 De-En, IWSLT16 En-De, WMT14 De-En and WMT14 En-De tasks. NPD 9 indicates results of noisy parallel decoding with 9 candidates, i.e., B = 4, otherwise B = 0.

achieves 28.16 BLEU score. We can see that compared with DT, TCL-NAT gains a large improvement on translation accuracy, demonstrating the importance of the progressive transfer between two tasks with curriculum learning.

Analysis on Pacing Functions. We compare the accuracy of models trained with different pacing functions. We evaluate TCL-NAT with different pacing functions shown in Table 3. From the table, we can see that the model trained with exponential function slightly outperforms those with other functions and the logarithmic function performs the worst. As we mentioned in Section 3.3, exponential function shows more preference on easier stages while logarithmic function focuses more on harder stages, and therefore we can conclude that showing more preference on easier tasks is beneficial to the NAT model training, and thus obtain a better score.

Pacing Functions	Linear	Logarithmic	Exponential
TCL-NAT	27.89	27.76	**27.96**
TCL-NAT (NPD 9)	31.51	31.45	**31.71**

Table 5: The comparison of BLEU scores on the test set of IWSLT14 De-En task among different pacing functions.

Analysis on Task Window. We compare the accuracy of models trained with different task windows, as mentioned in Section 3.3. The results are listed in Table 6. From the table, we can see that the model trained with $w = 2$ achieves the best score in IWSLT14 De-En task, which proves that an appropriate task window w can help reduce the gap between neighboring stages, and thus help model training.

5 Conclusion

In this work, we proposed a novel task-level curriculum learning method to improve the accuracy of non-autoregressive neural machine translation. We first view autoregressive, semi-autoregressive and non-autoregressive translation as individual tasks with different k, and propose a task-level curriculum mechanism to shift the training process from $k = 1$ to N, where N is the length of the target sentence. Experiments on several benchmark translation datasets demonstrate the effectiveness of our method for NAT.

In the future, we will extend the task-level curriculum learning method to other sequence generation tasks such as non-autoregressive speech synthesis, automatic speech recognition and image captioning, where there exists smooth transformation between autoregressive and non-autoregressive generation using semi-autoregressive generation as bridges. We expect task-level curriculum learning could become a general training paradigm for a broader range of tasks.

Task Window	$w = 1$	$w = 2$	$w = 3$	$w = 4$
TCL-NAT	27.89	**28.16**	28.00	27.96
TCL-NAT (NPD 9)	31.51	**31.79**	31.44	31.40

Table 6: The comparison of BLEU scores on the test set of IWSLT14 De-En task among different task windows.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (Grant No.2018AAA0100603), Zhejiang Natural Science Foundation (LR19F020006), National Natural Science Foundation of China (Grant No.61836002), National Natural Science Foundation of China (Grant No.U1611461), National Natural Science Foundation of China (Grant No.61751209), and Microsoft Research Asia.

References

[Anastasopoulos and Chiang, 2018] Antonios Anastasopoulos and David Chiang. Tied multitask learning for neural speech translation. In *NAACL*, pages 82–91, June 2018.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48. ACM, 2009.

[Caruana, 1997] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[Dong *et al.*, 2015] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *ACL-IJCNLP*, pages 1723–1732, 2015.

[Garg *et al.*, 2019] Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. Jointly learning to align and translate with transformer models. In *EMNLP-IJCNLP*, pages 4452–4461, November 2019.

[Gehring *et al.*, 2017] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252, 2017.

[Ghazvininejad *et al.*, 2019] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP-IJCNLP*, pages 6114–6123, 2019.

[Gu *et al.*, 2018] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *ICLR*, 2018.

[Gu *et al.*, 2019] Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In *Advances in Neural Information Processing Systems*, pages 11179–11189, 2019.

[Guo *et al.*, 2019a] Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. Non-autoregressive neural machine translation with enhanced decoder input. In *AAAI*, volume 33, pages 3723–3730, 2019.

[Guo *et al.*, 2019b] Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. *arXiv preprint arXiv:1911.08717*, 2019.

[Kim and Rush, 2016] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *EMNLP*, pages 1317–1327, 2016.

[Lee and Grauman, 2011] Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR 2011*, pages 1721–1728. IEEE, 2011.

[Lee *et al.*, 2018] Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*, pages 1173–1182, 2018.

[Li *et al.*, 2019] Zhuohan Li, Zi Lin, Di He, Fei Tian, QIN Tao, WANG Liwei, and Tie-Yan Liu. Hint-based training for non-autoregressive machine translation. In *EMNLP-IJCNLP*, pages 5712–5717, 2019.

[Ma *et al.*, 2019] Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP-IJCNLP*, pages 4273–4283, 2019.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[Ren *et al.*, 2019] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*, 2019.

[Sachan and Xing, 2016] Mrinmaya Sachan and Eric Xing. Easy questions first? a case study on curriculum learning for question answering. In *ACL*, volume 1, pages 453–463, 2016.

[Sarafianos *et al.*, 2017] Nikolaos Sarafianos, Theodore Giannakopoulos, Christophoros Nikou, and Ioannis A Kakadiaris. Curriculum learning for multi-task classification of visual attributes. In *ICCV*, pages 2608–2615, 2017.

[Sennrich *et al.*, 2016] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, pages 1715–1725, 2016.

[Song *et al.*, 2019] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. In *ICML*, pages 5926–5936, 2019.

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

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 6000–6010, 2017.

[Vaswani *et al.*, 2018] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. Tensor2tensor for neural machine translation. In *AMTA*, pages 193–199, 2018.

[Wang *et al.*, 2018] Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *EMNLP*, pages 479–488, 2018.

[Wang *et al.*, 2019] Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. Non-autoregressive machine translation with auxiliary regularization. In *AAAI*, 2019.