NON-ITERATIVE PARALLEL TEXT GENERATION VIA GLANCING TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Although non-autoregressive models with one-iteration generation achieves remarkable inference speed-up, they still falls behind their autoregressive counterparts in prediction accuracy. The non-autoregressive models with best accuracy currently rely on multiple decoding iterations, which largely sacrifice the inference speed of non-autoregressive models. Inspired by the way of learning word dependencies in autoregressive and iterative-decoding models, we propose Glancing Transformer (GLAT) with glancing language model, which learns to capture the word dependency in a gradual fashion. Experiments on three benchmarks demonstrate that our approach can significantly improve the accuracy of non-autoregressive models without multiple decoding iterations. In particular, GLAT achieves state-of-the-art results among non-iterative models and even outperforms top iterative counterparts in some specific benchmarks.

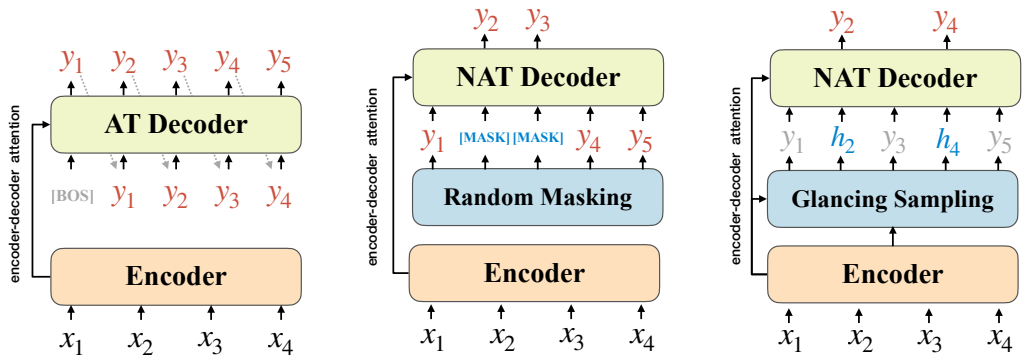
1 INTRODUCTION

Non-autoregressive transformer (NAT) has attracted wide attention in neural machine translation (Gu et al., 2018), which generates sentences simultaneously rather than sequentially. To enable parallel decoding, NAT impose a *conditional independence assumption* among words in the output sentences, which leads to significantly faster speed (almost a dozen times speed-up) than the autoregressive Transformer (Vaswani et al., 2017). However, NAT still falls behind autoregressive Transformer (AT) in the quality of output sentences (e.g. BLEU (Papineni et al., 2002) for machine translation). We blame it for the imposed conditions independence assumption, which prevents the NAT model to explicitly learn the *dependency of words* in the output sentence. Note that such word dependency is crucial, and is explicitly learned in the AT model through the *autoregressive language models* (left-to-right, see Figure 1a).

Recently, Ghazvininejad et al. (2019); Gu et al. (2019) propose to employ the *mask language models* (MLM) (Devlin et al., 2019) for including the explicit word dependency modeling in NAT (see Figure 1b), therefore giving quite competitive accuracy performances compared to AT. Specifically, during training, these NAT models randomly masking words in reference and predict the masked words conditioned on the unmasked ones. In this manner, these models are trained to capture the dependencies between mask words and unmask words explicitly. Because of the poor performance of one iteration, these models always perform decoding with multiple iterations during inference, and each decoding iteration is conditioned on the generated outputs of the last iteration. And due to the iterative process, these approaches partly sacrifice the speed merit of NAT, which is not necessarily the best. How to abandon the iterative process but still explicitly model word dependencies in NAT is still an open problem.

In this paper, we answer the above question by proposing a non-iterative parallel text generation model called *Glancing Transformer* (GLAT), which includes the explicit word dependency modeling in NAT via a proposed *Glancing Language Model* (GLM). Note that our proposed GLAT works in a one-shot way instead of an iterative way, which only improves the training of the vanilla NAT model, while *not changing* the inference part. In such a case, GLAT thoroughly enjoys the speed merits of parallel generation.

Generally, the GLM in our proposed GLAT differs from MLM in two aspects. First, by introducing a *Glancing Sampling Module*, GLM works by gradual training instead of iterative inference (see Figure 1c). We argue that decoding with only one iteration hardly works well because the training of



(a) Left-to-right LM in AT (b) Masked LM in Iterative NAT (c) GLM in GLAT

Figure 1: Different language modeling approaches of different text generation models.

NAT is very challenging, and iteration is needed to polish the outputs progressively. Usually, one NAT model cannot predict the sentence fragment correctly, let alone predict the whole sentence. To this end, we borrow the idea of curriculum learning (Bengio et al., 2009) and propose the *glancing sampling* strategy to control the training adaptively. Specifically, our model performs decoding twice in training. After the *first decoding*, GLAT will know how well it predicts the sentence by glancing at the reference. Then GLAT will sample some words from the reference and learning to predict (here is the second decoding) the remaining words not sampled from reference. Note that only the second decoding will update the model parameters, and the first one is used to decide whether the current reference sentence is hard for the model to predict. The glancing sampling strategy will sample more words if the predictions for the reference are inaccurate, and the number of sampled words will decrease when GLAT predicts the reference more accurately along the training process. More sampled words will make the learning easier, making sure the training of GLAT in a curriculum learning way.

Second, GLM replaces the `[MASK]` token in the MLM with signals from the encoder, thus not only the target word dependency, but also the dependencies between the input and target are strengthened. This works quite well in practice. Equipped with GLM, GLAT can boost the performance of NAT with one-iteration generation and make multiple decoding iterations no longer necessary.

Experimental results show that GLAT obtains significant improvements (about 5 BLEU) on EN-DE/DE-EN compared to the vanilla NAT without losing inference speed-up. GLAT outperforms the iterative decoding model mask-predict (Ghazvininejad et al., 2019) on WMT14 DE-EN and WMT16 RO-EN and closes gaps to less than 0.5 BLEU on WMT14 EN-DE and WMT16 EN-RO, but with $5\times$ speed-up. Considering fully NAT models with one-iteration simultaneous generation, GLAT achieves the best BLEU scores. When compared to standard AT models, GLAT can still close the performance gap within 1 BLEU point, while keeps $7.9\times$ speed-up. And empirically we even find GLAT outperforms AT in some cases when the output sentence length is less than 20. We think GLAT is superior than AT to some extent, because GLAT's adopted language model is bidirectional in nature, while AT's is unidirectional.

2 Text Generation via Conditional Language Modeling

In this section, we compare the different language models employed in different text generation models. Formally, consider a sequence to sequence model (Cho et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017) for predicting $Y = \{y_1, y_2, ..., y_T\}$ given the input sentence $X = \{x_1, x_2, ..., x_N\}$. In the AT model, the training objective is maximizing the log-likelihood with autoregressive decomposition:

$$\log p_\theta(Y|X) = \prod_{t=1}^{T} \log p_\theta(y_t|y_{<t}, X), \tag{1}$$

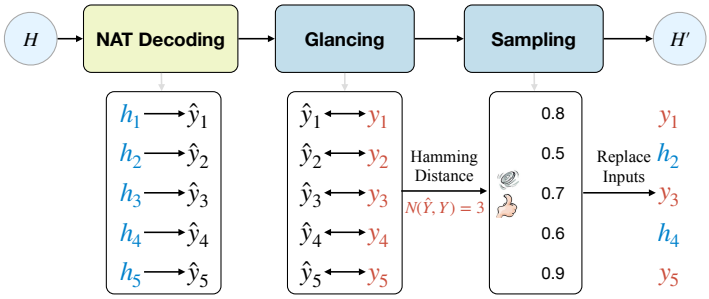


Figure 2: Illustration of the procedure of glancing sampling

where the word y_t is conditioned on the target prefix $y_{<t} = \{\texttt{[BOS]}, y_1, ..., y_{t-1}\}$ and the source input X. AT models the language from left-to-right, dependencies are learned in a unidirectional way. However, in NAT, learning is under the conditional independence assumption:

$$\log p_\theta(Y|X) = \prod_{t=1}^{T} \log p_\theta(y_t|X). \tag{2}$$

NAT learns dependencies implicitly, since there is no explicit supervision between tokens in the target sentence. Only supervision of mapping from X to Y is explicitly given. To leverage the past output predictions, Mask-Predict introduces mask language models (MLM) to predict tokens conditioned on part of the target inputs:

$$\log p(\overline{Y}|\tilde{Y}_{\text{mask}}, X; \theta) = \prod_{t=1}^{T} r_t \log p(y_t|\tilde{Y}_{\text{mask}}, X; \theta). \tag{3}$$

In MLM, the input $\tilde{Y}_{\text{mask}}$ is constructed by replacing part of the tokens in Y with the `[MASK]` token, and training objective to predict the replaced target tokens $\overline{Y}$ with $\tilde{Y}_{\text{mask}}$. r_t equals 1 if the tokens at position t is replaced, otherwise 0.

Though MLM can be used to learn dependency explicitly, it still does not give good results within one iteration. We argue that it is because the training of NAT is very challenging, and iterative editing is needed to polish the outputs progressively. In the next section, we will introduce our proposed GLAT, which includes a new glancing language models to boost the results without iteration during inference.

3 The Proposed Glancing Transformer

In this section, we introduce our proposed GLAT in detail. Generally, GLAT differs from vanilla NAT by employ the GLM in the training process, while they are exactly the same in inference. The GLM in GLAT gradually learns longer sentence fragments for simultaneous generation and introduces explicit dependency modeling with the mix of source and target inputs, which better captures dependencies under the one-iteration generation setting.

Generally, GLAT performs the two-pass decoding of NAT during training. At the first pass, we get the output prediction and compare with the reference (ground truth sentence), then sample target words by glancing at the reference according to how well this reference is predicted. At the second pass, we feed the sampled words to the decoder at their corresponding positions, then train the model to predict the remaining words via maximum likelihood estimation. In our proposed GLAT, more reference words for inaccurate sentence predictions tend to be sampled as decoding inputs. As the model can better predict the reference along the training process, the model glances less reference words and the target fragments to learn grows gradually. GLAT could also introduce explicit dependency learning in the process of predicting remaining words with some target words as input and strengthen the decoder input with source representation.

3.1 TRAINING OBJECTIVE

Formally, given the training data $D = \{X, Y\}_{i=1}^{L}$ for predicting $Y = \{y_1, y_2, ..., y_T\}$ with the input sentence $X = \{x_1, x_2, ..., x_N\}$, the training objective of GLAT is:

$$\mathcal{L}_{\text{GLAT}} = -\sum_{\{X,Y\}\in D} \sum_{y_t \in \{Y \setminus \mathbb{GS}(Y,\hat{Y})\}} \log p(y_t|\, \mathbb{GS}(Y,\hat{Y}), X; \theta). \tag{4}$$

Here, $\hat{Y}$ is the predicted sentences in the first decoding pass, and $\mathbb{GS}(Y, \hat{Y})$ is the set of sampled target words by the sampling strategy of $\mathbb{GS}$. The sampling strategy will sample more words from reference Y if $\hat{Y}$ is less accurate compared to Y, and sampling less words otherwise. Additionally, $\{Y \setminus \mathbb{GS}(Y, \hat{Y})\}$ is the difference set, representing the remaining words except these sampled words.

In the following sections, we will describe the whole training process and the sampling strategy in detail, respectively.

3.2 GLANCING LANGUAGE MODEL

Our training procedure with two-pass decoding for GLAT is illustrated in Figure 2. In the first decoding pass, given the encoder $\mathbb{F}_{\text{encoder}}$ and decoder $\mathbb{F}_{\text{decoder}}$, $H = \{h_1, h_2, ..., h_T\}$ is the encoded source embedding sequence gathered from the input X, and $\hat{Y} = \mathbb{F}_{\text{decoder}}(H, \mathbb{F}_{\text{encoder}}(X; \theta); \theta)$ is the predicted sentence in the first decoding pass. With the predicted sentence $\hat{Y}$, $\mathbb{GS}(Y, \hat{Y})$ is the set of sampled words from reference Y, according to our sampling strategy $\mathbb{GS}$, which will be introduced in the next section. Note that we use the attention mechanism to form the decoder inputs with the input X. Previous work adopt *Uniform Copy* (Gu et al., 2018) or *SoftCopy* (Wei et al., 2019) instead. But empirically, we find that they give almost the same results in our setting.

In the second decoding pass, we cover the original decoding inputs H by the embeddings of words from $\mathbb{GS}(Y, \hat{Y})$ to get the new decoding inputs $H' = \mathbb{F}_{\text{cover}}(E_{y_t \in \mathbb{GS}(Y,\hat{Y})}(y_t), H)$, where $\mathbb{F}_{\text{cover}}$ covers according to the corresponding positions. Namely, if we have a sampled word at one position, we just use its word embedding to replace the original decoding input at the same position. Here the word embeddings are obtained from the softmax embedding matrix of the decoder. With the covered decoding inputs H', the probabilities of remaining words on each position $p(y_t|\, \mathbb{GS}(Y, \hat{Y}), X; \theta)$ are computed by $\mathbb{F}_{\text{decoder}}(H', \mathbb{F}_{\text{encoder}}(X; \theta), t; \theta)$. Finally, only the loss for the remaining target words are computed, and we use maximum likelihood estimation to learn our model θ according to the Equation 4.

Intuitively, the formulation of our proposed glancing language model could be maximizing the likelihood of remaining words in a gradual fashion, with the mix of encoding signals and reference words from glancing sampling H' as inputs.

3.3 THE GLANCING SAMPLING STRATEGY

We employ a two-step sampling strategy. The first step decides the sampling number and the second step sample the reference words based on the number.

Formally, given the input X, its predicted sentence $\hat{Y}$ and its reference Y, the goal of reference glancing function $\mathbb{GS}(Y, \hat{Y})$ is to obtain a set of sampled words S, where S is a subset of Y. In the first step, we compute a sampling number $N(Y, \hat{Y})$, which decides the number of sampled words. In the second step, we sample the set S with $\mathbb{GS}(Y, \hat{Y})$, which should meet the constrain $|S| = N(Y, \hat{Y})$.

Adaptive Sampling Number With a similar idea with curriculum learning (Bengio et al., 2009), we propose a curriculum sampling strategy to decide the number that aims at reducing more difficulty from the start and gradually approaching the complete simultaneous generation setting as the model is trained better. For sentences hard to generate, we should reduce more learning difficulty and enlarge the sampling number. Thus, the demand of target words for different training samples also varies.

Specifically, given the input X, its translation $\hat{Y}$ and its reference Y, we use the distance $d(Y, \hat{Y})$ to measure how Y is well generated given X, where $d(\cdot)$ is the distance function. The less words in the

target sentence can the model generate, the more additional target words are needed to reduce the difficulty of simultaneous generation. Considering only the same words in the output can match the target, we adopt Hamming distance (Hamming, 1950) in this work, that is $d(Y, \hat{Y}) = \sum_{t=1}^{T}(y_t \neq \hat{y}_t)$, where t is the word position. To better control the reference sampling, we further employ a ratio function f_{ratio} to adjust the sampling number more flexibly. With the distance and the ratio function, the sampling number is determined by $N(Y, \hat{Y}) = f_{\text{ratio}} d(Y, \hat{Y})$, which adaptively adjusts the learning procedure.

Random Sampling Strategy Given the sampling number $N(Y, \hat{Y})$, we then decide which words should be chosen. We use simple random sampling as our strategy, that is, randomly choose $N(Y, \hat{Y})$ words from the reference Y. Random sampling is the most straightforward approach to getting samples. We choose this strategy for the following reasons. First, it samples from an almost unlimited space, which is unbiased and vital for drawing conclusions. Second, random sampling is not necessarily the best candidates, while it is very powerful in most cases. Many previous studies enjoy its simple implementation and desirable performance, such as BERT (Devlin et al., 2019).

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Datasets To validate the effectiveness of our method, we conduct experiments on three machine translation benchmarks: WMT14 EN-DE (4.5M translation pairs), WMT16 EN-RO (610k translation pairs) and IWSLT16 DE-EN (150K translation pairs). The datasets are tokenized and segmented into subword units using BPE encodings (Sennrich et al., 2016). We preprocess WMT14 EN-DE following the data preprocessing in Vaswani et al. (2017) and use the processed data provided in Lee et al. (2018) for WMT16 EN-RO and IWSLT16 DE-EN.

Distillation Following previous work (Gu et al., 2018; Lee et al., 2018; Wang et al., 2019), we also use sequence-level knowledge distillation for all datasets. We employ base autoregressive transformers in Vaswani et al. (2017) for distilling data. Then, we train our models on data distilled from transformer for all the datasets.

Inference GLAT only modifies the training procedure and performs one-iteration non-autoregressive generation as the base model in Gu et al. (2018). We also consider the common practice of noise parallel decoding (Gu et al., 2018; Lee et al., 2018; Guo et al., 2019a; Wang et al., 2019), which generates several decoding candidates in parallel and selects the best via re-scoring with a pre-trained autoregressive model. For GLAT, we first predict m target length candidates, then generate output sequences with argmax decoding for each target length candidate, which is also called length parallel decoding (LPD) (Wei et al., 2019). Then we use the pre-trained transformer to rank these sequences and identify the best overall output as the final output.

Implementation We adopt the vanilla model which copies source input uniformly in Gu et al. (2018) as our base model (NAT-base) and replace the *UniformCopy* with attention mechanism using positions. For WMT datasets, we follow the hyperparameters of base Transformer in Vaswani et al. (2017). And we choose a smaller setting for IWSLT16 considering that IWSLT16 is a smaller dataset. For IWSLT16, we use 5 layers for encoder and decoder and set the size of model input/output d_{model} to 256. We train the model with batches of 64k/8k tokens for WMT/IWSLT datasets, respectively. We use Adam optimizer (Kingma & Ba, 2014) with $\beta = (0.9, 0.999)$. For WMT datasets, the learning rate warms up to $5e-4$ in 4k steps and gradually decays according to inverse square root schedule in Vaswani et al. (2017). As for IWSLT16 DE-EN, we adopt linear annealing (from $3e-4$ to $1e-5$) as in Lee et al. (2018). For the ratio function f_{ratio}, we use a linear decreasing function from 0.5 to 0.3 for WMT datasets and a fixed ratio of 0.5 for IWSLT16. The final model is created by averaging the 5 best checkpoints.

Competitors We compare our method with strong representative baselines, including fully non-iterative models: the NAT with fertility (Gu et al., 2018, NAT-FT), our vanilla NAT-base model, the NAT imitating AT (Wei et al., 2019, imit-NAT), the Flow-based NAT (Ma et al., 2019, Flowseq),

Table 1: Performance on WMT14 EN-DE/DE-EN and WMT16 EN-RO/RO-EN benchmarks. I_{dec} is the number of decoding iterations and m is the number of reranking candidates.

Models		I_{dec}	WMT14 EN-DE	WMT14 DE-EN	WMT-2016 EN-RO	WMT-2016 RO-EN	Speed Up
AT Models	Transformer (Vaswani)	N	27.30	/	/	/	/
	Transformer (ours)	N	27.48	31.27	33.70	34.05	1.0×
Iterative NAT	NAT-IR	10	21.61	25.48	29.32	30.19	1.5×
	LevT	6+	27.27	/	/	33.26	4.0×
	Mask-Predict	10	27.03	30.53	**33.08**	**33.31**	/
	JM-NAT	10	**27.31**	**31.02**	/	/	5.7×
Non-iterative NAT	NAT-FT	1	17.69	21.47	27.29	29.06	15.6×
	imitate-NAT	1	22.44	25.67	28.61	28.90	18.6×
	NAT-HINT	1	21.11	25.24	/	/	30.2×
	Flowseq	1	23.72	28.39	29.73	30.72	/
	NAT-base (ours)	1	20.36	24.81	28.47	29.43	15.3×
	GLAT (ours)	1	**25.21**	**29.84**	**31.19**	**32.04**	15.3×
Non-iterative NAT w/ Reranking	NAT-FT (m=100)	1	19.17	23.20	29.79	31.44	2.4×
	imit-NAT (m=7)	1	24.15	27.28	31.45	31.81	9.7×
	NAT-HINT (m=9)	1	25.20	29.52	/	/	17.8×
	Flowseq (m=30)	1	25.31	30.68	32.20	32.84	/
	NAT-DCRF (m=9)	1	26.07	29.68	/	/	6.1×
	GLAT (m=7, ours)	1	**26.55**	**31.02**	**32.87**	**33.51**	7.9×

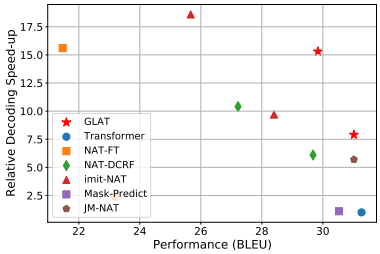


Figure 3: The trade-off between speed-up and BLEU on WMT14 DE-EN

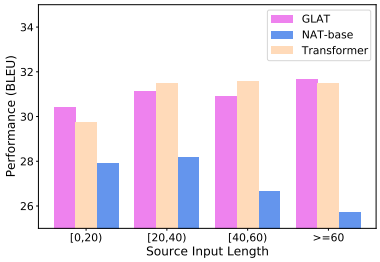


Figure 4: Performance under different source input length on WMT14 DE-EN

the NAT with hint-based training (Li et al., 2019, NAT-HINT), the NAT with CRF (Lafferty et al., 2001) decoding (Sun et al., 2019, NAT-DCRF), and the NAT with iterative refinement: NAR-IR (Lee et al., 2018), LevT (Gu et al., 2019), Mask-Predict (Ghazvininejad et al., 2019), and JM-NAT (Guo et al., 2020). For all our tasks, we obtain the performance of other NAT models by directly using the performance figures reported in their papers if they are available on our datasets.

4.2 RESULTS

The main results on the benchmarks are presented in Table 1. Clearly, our method significantly improves the translation quality and outperforms strong baselines by a large margin. Our method introduces explicit dependency learning for the decoder and gradually learns simultaneous generation of the whole sequence, enables the model to better capture the underlying data structure. Compared to models with iterative decoding, our method completely maintains the inference efficiency advantage of fully non-autoregressive models, since GLAT generate with only one-iteration. Compared with the competitors, we will highlight our empirical advantages:

- The performance of GLAT is surprisingly good. Compared with the vanilla NAT-base models, GLAT obtains significant improvements (about 5 BLEU) on EN-DE/DE-EN. Additionally, GLAT also surpasses other fully non-autoregressive models with a big margin (almost +3 BLEU score on average). The results are even very close to those of the AT model, which shows great potential.

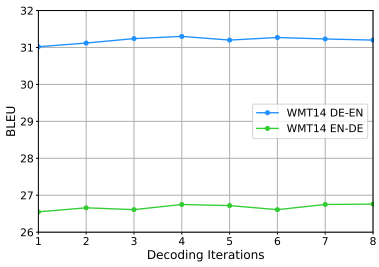


Figure 5: The BLEU scores of GLAT with different decoding iterations

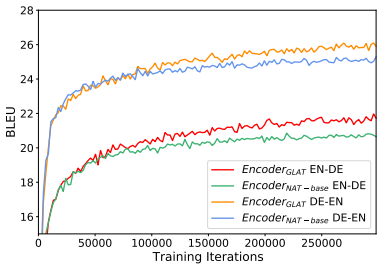


Figure 6: Training NAT with different initialized encoder

- GLAT is simple and can be applied to other NAT models flexibly, as we only modify the training process by reference glancing while keep inference unchanged. For comparison, imitate-NAT introduces additional AT models as teachers; NAT-DCRF utilizes CRF to generate sequentially; NAT-IR and Mask-Predict models need multiple decoding iteration.

We also see in Table 1 that BLEU and speed-up are more or less contradictory. Therefore, we present the scatter plot in Figure 3, displaying the trend of speed-up and BLEU scores with different NAT models. It is shown that the point of GLAT is located on the top-right of the competing methods. Obviously, GLAT outperforms our competitors in BLEU if speed-up is controlled, and in speed-up if BLEU is controlled. This indicates that GLAT outperforms previous state-of-the-art NAT methods. Although iterative models like Mask-Predict achieves competitive BLEU scores, they only maintain minor speed advantages over AT. In contrast, fully non-autoregressive models remarkably improve the inference speed.

4.3 Analysis

Effect of Source Input Length To analyze the effect of source input length on the performance of different models, we split the source sentences into different intervals by length after BPE and compute the BLEU score for each interval. The histogram of results is presented in Figure 4. The performance of NAT-base drops sharply for long sentences, while the gradual learning process enables GLAT to boost the performance by a large margin, especially for long sentences. We also find that GLAT outperforms autoregressive Transformer when the source input length is smaller than 20.

GLAT Achieves Strong Results without Multiple Iterations We conduct experiments of GLAT with more than one decoding iteration in inference. We adopt the inference algorithm in Ghazvininejad et al. (2019) for multiple-iterations decoding. The results are shown in Figure 5. We find that GLAT can achieve decent performances with only one decoding iteration, further iterations only obtain minor improvements of 0.2$\sim$0.3 BLEU.

GLAT Learns Better Encoder To validate whether GLAT strengthens encoders, we conduct experiments to present GLAT learns better encoder. We train NAT models with different initialization for comparison, the encoders of NAT models are initialized with the parameter of trained GLAT and trained NAT-base respectively and the parameters of encoders are fixed during training. The models are all trained using the traditional MLE loss and the results are presented in Figure 6. Obviously, models initialized with encoders of GLAT outperform those with NAT-base encoders, indicating that GLAT learns better encoders.

Effectiveness of the Adaptive Sampling Strategy To validate the effectiveness of the adaptive sampling strategy for the sampling number $N(Y, \hat{Y})$, we also propose two fixed approaches for comparison. The first one decides the sampling number with $\lambda * T$, where T is the length of Y, and λ is the constant ratio. The second one is relatively flexible, which sets a start ratio λ_s and end ratio λ_e, and linearly reduce the sampling number from $\lambda_s * T$ to $\lambda_e * T$ with the training process.

As shown in Table 2 and Table 3, clearly, our adaptive approach (Adaptive in the table) outperforms the competitors with big margins. The results confirm our intuition that the sampling schedule affects

Table 2: Performances on IWSLT16 with fixed sampling ratio.

Sampling Method	λ	BLEU
Fixed	0.0	24.66
	0.1	24.91
	0.2	27.12
	0.3	24.98
	0.4	22.96
Adaptive	-	**29.61**

Table 3: Performances on IWSLT16 with decreasing sampling ratio.

Sampling Method	Schedule λ_s	Schedule λ_e	BLEU
Fixed	0.5	0	27.80
	0.5	0.1	28.21
	0.5	0.2	27.15
	0.5	0.3	23.37
Adaptive	-		**29.61**

the translation performance of our NAT model. The sampling strategy, which first offering relatively easy generation problems and then turns harder, benefits the final performance.

More specifically, the random sampling strategy is robust. Even with the simplest constant ratio, GLAT still achieves remarkable results. When set $\lambda = 0.2$, it even outperforms the baseline $\lambda = 0.0$ by 2.5 BLEU score.

The experiments potentially support our suggestion that it is beneficial to learn simple patterns at the start and gradually transfer to more complex patterns. The method of flexible decreasing ratio works better than that of the constant one, and our proposed adaptive approaches achieve the best results.

5 Related Work

Fully Non-Autoregressive Models A line of work introduces various forms of latent variables to reduce the model's burden of dealing with dependencies among output words (Gu et al., 2018; Ma et al., 2019; Bao et al., 2019; Ran et al., 2019). Another branch of work considers transferring the knowledge from autoregressive models to non-autoregressive models (Wei et al., 2019; Li et al., 2019; Guo et al., 2019b). Besides, there are also some work that apply different training objectives to train non-autoregressive models (Libovický & Helcl, 2018; Shao et al., 2020; Ghazvininejad et al., 2020) or add regularization terms (Wang et al., 2019; Guo et al., 2019a).

Non-Autoregressive Models with Structured Decoding In order to model the dependencies between words, Sun et al. (2019) introduces a CRF inference module in NAT and performs additional sequential decoding after the non-autoregressive computation in inference. Deng & Rush (2020) proposes cascaded CRF decoding. Since GLAT only performs one-iteration non-autoregressive generation, our approach is orthogonal to the method proposed in Sun et al. (2019). We can also combine our approach with the structured decoding method.

Non-Autoregressive Models with Iterative Refinement There is a series of work devotes to semi-autoregressive models which combine the strength of both types of models by iteratively refining the former outputs. Lee et al. (2018) proposed a method of iterative refinement based on denoising autoencoder. Gu et al. (2019) utilized insertion and deletion to refine the generation. Ghazvininejad et al. (2019) trained the model in the way of masked language model and the model iteratively replaces the mask tokens with new outputs. Despite the relatively better accuracy, the multiple decoding iterations largely reduces the inference efficiency of non-autoregressive models.

6 Conclusion

In this paper, we propose Glancing Transformer with glancing language model to improve the performance of NAT with one-iteration generation. With the glancing language model, the model reduces learning difficulty by gradually learning with growing targets for simultaneous generation and explicitly learns the target language modeling. Experimental results show that our approach significantly improves the performance of non-autoregressive machine translation with one-iteration generation. As non-autoregressive models are efficient and have a large potential in multiple tasks, we plan to apply our approach to other tasks.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

Yu Bao, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and Lei Li. Non-autoregressive transformer by position learning. arXiv preprint arXiv:1911.10677, 2019.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In ICML, pp. 41–48, 2009.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In EMNLP, pp. 1724–1734, 2014.

Yuntian Deng and Alexander M Rush. Cascaded text generation with markov transformers. arXiv preprint arXiv:2006.01112, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, pp. 4171–4186, 2019.

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

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. arXiv preprint arXiv:2004.01655, 2020.

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

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In NeurIPS, pp. 11179–11189, 2019.

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, pp. 3723–3730, 2019a.

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, 2019b.

Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In ACL, pp. 376–385, 2020.

Richard W Hamming. Error detecting and error correcting codes. The Bell system technical journal, 29(2):147–160, 1950.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In ICML, pp. 282–289, 2001.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In EMNLP, pp. 1173–1182, 2018.

Zhuohan Li, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. Hint-based training for non-autoregressive translation. In EMNLP-IJCNLP, 2019.

Jindřich Libovický and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In EMNLP, pp. 3016–3021, 2018.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In EMNLP-IJCNLP, pp. 4273–4283, Hong Kong, China, November 2019. doi: 10.18653/v1/D19-1437.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In ACL, pp. 311–318, 2002.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. Guiding non-autoregressive neural machine translation decoding with reordering information. arXiv preprint arXiv:1911.02215, 2019.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In ACL, pp. 1715–1725, 2016.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In AAAI, pp. 198–205, 2020.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. Fast structured decoding for sequence models. In NeurIPS, pp. 3016–3026, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, pp. 5998–6008, 2017.

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

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. Imitation learning for non-autoregressive neural machine translation. In ACL, 2019.