LayerGLAT: a Flexible Non-Autoregressive Transformer for Single-Pass and Multi-Pass Prediction

Shijie Li[1], Inigo Jauregi Unanue[1,2], and Massimo Piccardi[1] (✉)

[1] University of Technology Sydney
Shijie.Li@student.uts.edu.au
Massimo.Piccardi@uts.edu.au
[2] RoZetta Technology
Inigo.Jauregi@rozettatechnology.com

Abstract. Non-autoregressive transformers (NATs) have made substantial progress in recent years, improving their predictive accuracy while achieving speed-ups of an order of magnitude compared to their conventional, autoregressive counterparts. However, the performance gap between NATs and autoregressive transformers (ATs) is still significant, which has triggered the development of "iterative" NATs which predict through multiple passes, targeting a trade-off between accuracy and speed. Notwithstanding the manifest benefits of both fully and iterative NATs, research seems to have overlooked the possibility of integrating them effectively, so as to deliver both strong single- and multi-pass prediction while retaining the highest possible speed-up. To bridge this gap, this paper introduces LayerGLAT, a hybrid model that combines the strengths of both fully and iterative NATs, achieving competitive performance in both single-pass and iterative prediction. The key idea of the proposed approach is a layer-wise training strategy that is able to emulate the generating conditions of both single-pass and multi-pass generation, leading to strong performance in both cases. The experimental results over three machine translation datasets have given evidence to the remarkable performance of the proposed model, which has been able to outperform leading NATs in accuracy and speed and near the accuracy of ATs.[3]

Keywords: Non-Autoregressive Transformer · Multi-Pass Prediction · Layer-Wise Training.

1 Introduction

In the field of natural language processing, fast and efficient generation is of paramount importance [8,27]. In contrast to standard autoregressive transformers (ATs) that generate tokens sequentially in a left-to-right manner, non-autoregressive transformers (NATs) are able to generate all tokens simultaneously in a single forward pass [6]. NAT's generation scheme may offer a significant speed-up over AT models; e.g., 15.6 times was reported by Gu et al. [6]. However, this advantage comes at the cost of reduced performance metrics, typically in the range of several percentage points in BLEU score [6]. As a result, recent efforts have focused on enhancing the performance of the basic,

[3] Our code is publicly available at https://github.com/lsj72123/layer-GLAT.

"vanilla" NAT, concentrating primarily in two directions: *fully* NATs and *iterative* NATs [30].

Fully NATs are designed to improve the performance of the vanilla NAT, still constrained to a single forward pass during inference. A main strategy for boosting their performance is to reduce inter-token dependencies in the training phase [7]. For example, Gu et al. [6] suggested utilizing sequence-level knowledge distillation [14], which simplifies the training corpus by replacing it with the predictions of an AT teacher model. Ghazvininejad et al. [4] and Saharia et al. [23] recommended using latent alignment in the training objective to alleviate the strict positional alignment demands of the standard cross entropy. Additionally, Qian et al. [21] introduced GLAT, a scheduled sampling strategy that partially incorporates target (i.e., reference, ground-truth) tokens during training. All these approaches solely impact the training stage, allowing these models to retain the speed advantage of the vanilla NAT while improving performance. However, a noticeable performance gap remains in several tasks when compared to their AT counterparts [10].

In contrast to fully NATs, iterative NATs achieve a balance between quality and efficiency by employing multiple, iterative refinement passes during inference [5,12]. In these models, the output of each subsequent pass is contingent on the output of the preceding one. Prominent examples include the conditional masked language model (CMLM) of Ghazvininejad et al. [5] and its follow-up works [11,12,24]. In general, iterative NATs have achieved impressive accuracy and, with unbounded iterations, can in principle reach the same accuracy as ATs. However, the performance in their initial pass typically falls short of that achieved by fully NATs, simply because their design is deliberately optimized for iterative use. On the whole, this is rather undesirable since users typically expect to attain the highest possible accuracy for any given number of iterations. For instance, in a hypothetical online translation service, users may opt for a specific price point, which will be roughly proportional to the computational effort (and, in the case of NATs, the number of iterations), expecting to achieve the highest possible accuracy for that price.

In light of the challenges posed by both fully and iterative NATs, this paper introduces *LayerGLAT*, a model designed to synergize their strengths and mitigate their weaknesses. This is achieved by amalgamating two leading-edge models for the field: GLAT for the fully NATs and CMLM for the iterative NATs. The rationale behind this choice lies in their strong performance, prominence in the literature, and remarkable similarities in training approach, which not only facilitates their integration, but also allows incorporating advanced variants such as CMLMC [11], latent-GLAT [1], and DSLP [9], that use either CMLM or GLAT as their base. We detail our integration and original contributions in Section 2.3. The proposed model has been evaluated over three widely-used machine translation datasets with varying training corpus sizes, and the experimental results show that the proposed approach with just a single iteration has been able to achieve a similar or higher performance than the iterative baselines, and outperform the compared fully NATs. In addition, increasing the number of iterations has delivered mild or marked performance improvements in all cases.

2 Methodology

In this section, we present a probabilistic description of both fully and iterative NATs, discuss the complementarity of GLAT and CMLM, and detail their integration for building our model.

2.1 Non-Autoregressive Generative Models

Contemporary machine translation systems predominantly employ autoregressive sequence modelling [19]. Given a source sentence, $X_{1:M} = \{x_1, \ldots, x_M\}$, these systems model the target sentence $Y_{1:N} = \{y_1, \ldots, y_N\}$ as a series of conditional distributions, with each token y_i based on its preceding tokens, $Y_{1:i-1}$, and the entire input, $X_{1:M}$:

$$p_{AT}(Y_{1:N}) = \prod_{i=1}^{N} p(y_i|Y_{1:i-1}, X_{1:M}) \tag{1}$$

At inference time, token predictions take place sequentially, and are typically concluded by the generation of a special end-of-sentence (EOS) token. In contrast, NAT models enable parallel generation by assuming conditional independence among target tokens and relying only on the source sentence as input:

$$p_{NAT}(Y_{1:N}) = \prod_{i=1}^{N} p(y_i|X_{1:M}) \tag{2}$$

Unlike AT models, this assumption prevents the confident generation of an EOS token since its precise position cannot be inferred due to the conditional independence of the model. As a result, the determination of the output sentence's length in NAT models relies on either a predefined latent mechanism, such as connectionist temporal classification [16], or an additional length prediction module jointly trained alongside the NAT model itself [6]. Moreover, given that the predictions cannot depend on previous tokens, all the inputs to the transformer's decoder are typically presented as masked tokens.[4] Overall, the assumption of complete conditional independence among target tokens is arguably too simplistic for many tasks [7]. Therefore, *iterative* NATs reintroduce dependencies by conditioning each token at the T-th iteration on the output of the $(T-1)$-th iteration:

$$p_{NAT}(Y^T_{1:N}) = \prod_{i=1}^{N} p(y^T_i|Y^{T-1}_{1:N}, X_{1:M}) \tag{3}$$

By controlling the number of iterations, iterative NATs are able to strike a balance between generation quality and speed. While slower than fully NATs, they can near the accuracy of AT models while avoiding their sequence-dependent decoding time, which is a significant advantage, particularly for long sentences [13].

[4] Alternatives are possible, such as for instance using the encoder's hidden states as input for the decoder [6]. However, Schmidt et al. [25] have shown that using a fully-masked sentence generally yields better performance.

2.2 GLAT and CMLM Training Strategies

Despite GLAT and CMLM focusing on different aspects of NAT performance, their training strategies exhibit significant similarities. The key idea of GLAT's training [21] is to use a certain number of target tokens, denoted as Y_{gt}, as input to the decoder. This changes its training objective from the unconditional probabilities of Equation 2 to the conditional probabilities of Equation 3, which are generally easier to model. As the training progresses, the proportion of Y_{gt} (noted as λ) is gradually decreased to zero, to progressively align the training objective to the operating conditions during the inference. To precisely control λ, GLAT breaks it down into two terms, defined as $\lambda = \lambda_1 \times \lambda_2$ [21]. The first term, λ_1, is a predefined hyperparameter in the $[0, 1]$ range that sets the upper bound for λ. The second term, λ_2, also ranges in $[0, 1]$ and acts as a dynamic scaling factor based on the model's current performance, which is quantified as the Hamming distance between the predicted and the target sentences. When the predictive performance is strong (i.e., low distances), the percentage of target tokens is correspondingly reduced (low values of λ_2). Therefore, as the model increasingly improves its predictions of the training set, λ_2, and thereby λ, decrease towards 0. While this scheduled training approach has allowed GLAT to perform remarkably, it only reflects the operating conditions of a single-pass prediction rather than those of iterative models.

Like GLAT, CMLM, too, incorporates target tokens during training, albeit with the distinct aim of aligning training to an iterative inference process. To start with, the iterative inference of CMLM operates in this way: during the t-th pass, CMLM retains the most confident t/T predicted tokens as input for the next pass, masks the less confident tokens, and re-predicts them in the next pass. As such, the t-th pass divides up $Y_{1:N}^{t-1}$ in Equation 3 into two groups: Y_{mask}^{t-1} and Y_{preds}^{t-1}. As its training strategy, CMLM aims to create training samples that mirror all the different passes of the inference. To this aim, the Y_{preds}^{t-1} are replaced by Y_{gt}, and a different λ value is uniformly sampled in $[0, 1]$ for every training sample to achieve varying proportions of Y_{gt} in the training objective. Although the CMLM's training objective covers the case of unconditional generation for values of $\lambda \sim 0$, in Section 5.2 we show that the large amount of Y_{gt} in the training objective significantly impairs the model's first-pass performance. As such, CMLM is intrinsically designed for use with multiple iterations.

2.3 The Proposed Model: LayerGLAT

The goal of the proposed model is to bring together the advantages of both GLAT and CMLM, i.e. GLAT's strong single-pass performance and CMLM's effective multi-pass performance for any arbitrary number of iterations. To this aim, our model employs a strategy similar to that of GLAT for tightly controlling the amount of Y_{gt} during training, utilizing two sampling rates to collaboratively establish the overall sampling rate, λ. However, our implementation departs from GLAT's significantly: while in GLAT parameter λ_2 is determined by the predictions of the last decoder layer, in our model it is based on the predictions of a *randomly-selected decoder layer*. The reason for using predictions from the various layers is that they vary wildly in accuracy, mimicking the operating conditions of both single-pass and multi-pass predictions. Higher layers

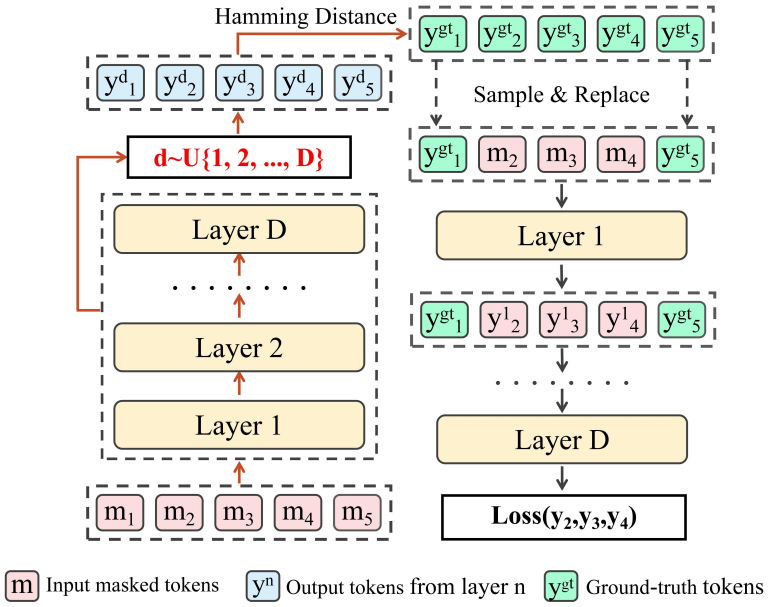

Fig. 1. Illustration of our model's training pipeline, which includes two distinct phases: a dedicated inference phase that determines the proportion of target tokens to be incorporated in the subsequent training phase, and the training phase itself, where the loss function is computed and the model's parameters are updated. Left: the red arrows (→) show the inference phase of our training approach, where predictions are provided from a randomly-chosen decoder layer. The predictions are then compared with the targets, and a proportion of target tokens is selected for input to the training phase. Right: the black arrows (→) show the substitution of mask tokens with target tokens in the first layer's input, and the substitution of the remaining mask tokens in the following layers with the predictions from the previous layer.

typically enjoy higher accuracy, which reduces the number of target tokens required in input and aligns with the operating conditions of a first-pass prediction (i.e., all masked tokens in input). Conversely, lower layers are typically less accurate, requiring a significant amount of target tokens in input and aligning with the operating conditions of the later iterations (i.e., many, correct predictions in input). To enable accurate predictions from every layer in the decoder, we also introduce an individual training signal for each layer. The loss function for the first decoder layer can be concisely noted as:

$$\mathcal{L}_1 = -\sum_{i \in msk} \log p_1(y_i | Y_{gt}, X_{1:M}) \tag{4}$$

where the probability of the first decoder layer has been noted as p_1, the tokens in input at the unmasked positions (omitted for brevity) as Y_{gt}, and the target tokens at the masked positions as $y_i, i \in msk$. In turn, the loss function for each of the following

layers can be noted as:

$$\mathcal{L}_{1<d\leq D} = -\sum_{i\in msk} \log p_d(y_i|Y_{gt}, Y^{d-1}_{preds}, X_{1:M}) \tag{5}$$

where the probability of the d-th decoder layer has been noted as p_d, the tokens in input at the unmasked and masked positions as Y_{gt} and Y^{d-1}_{preds}, respectively (where the predictions from the previous layer, Y^{d-1}_{preds}, have replaced the masked tokens), and the total number of layers as D. The overall training loss is computed as the average across all the layers. Aside from the improved accuracy, an additional benefit of explicitly training every decoder layer is the flexibility that this may offer at inference time: that is, we can opt to use only the first few layers for prediction and completely discard the remaining top layers. This adaptability spontaneously allows our model to meet variable memory constraints by simply adjusting the number of decoder layers employed. Fig. 1 shows an illustration of our model's entire training pipeline. The leftmost part shows the predictions from the decoder, which are used to compute the Hamming distance from the target tokens, and the consequent proportion of mask input tokens to be replaced with target tokens. The decoder's layer for the prediction, $d \in [1 \dots D]$, is sampled from a uniform distribution. Note that when the sampled layer is the last (D), this process is equivalent to GLAT's. In turn, the rightmost part shows the input and output to the various decoder layers used to compute the training objective. In the first layer's input, a proportion of the mask tokens is replaced with target tokens (y^{gt}_1 and y^{gt}_5 in the example). For the following layers' input, the remaining mask tokens are replaced by the predictions from the previous layer (y^{d-1}_{2-4} in the example). The loss function is computed over the log-likelihood (4), (5) of all the layers (with the connections omitted for simplicity). Appendix B provides a detailed training algorithm. With our experimental results, we show that our model is able to effectively handle both single-pass and multi-pass generation.

We note that the incorporation of layer-wise training signal in NATs was originally proposed by DSLP [9]. However, the focus of DSLP is different from ours, in that DSLP aims to enhance the performance of fully NATs, while we aim to improve iterative NATs in both single-pass and multi-pass prediction. As stated above, our rationale for using different decoder layers is to precisely control the percentage of Y_{gt} in the training, differing significantly from Huang et al. [9]. However, we gratefully acknowledge the inspiration from DSLP and we utilise it as a benchmark for comparative analysis.

3 Experimental Set-up

3.1 Datasets

To probe the performance of the proposed model, we have conducted experiments using three machine translation datasets of different sizes, in five directions in total: IWSLT14 De→En (small, with $160K$ sentence pairs), WMT16 En↔Ro (medium, with $610K$ sentence pairs) and WMT14 En↔De (large, with $4.5M$ sentence pairs). For the IWSLT14 dataset, we have followed the data preprocessing guidelines outlined in the

Table 1. Comparison of our model with other baseline models. T: number of iterations; m: size of length beam.

Models	WMT14		WMT16		IWSLT14†	Speed-up‡
	En→	De→	En→	Ro→	De→En	(GPU)
Autoregressive						
Transformer (ours, teacher)	27.1	32.6	33.8	33.8	34.6	1×
+ KD	27.4	32.8	33.6	33.8	34.9	1×
Transformer [11]	27.7	31.1	34.2	34.5	34.7	1×
Fully NAT						
Vanilla NAT [6]	17.7	21.5	27.3	29.1	-	-
GLAT [21]	25.2	29.8	31.2	32.0	31.5	13.5×
+ DSLP [9]	25.7	29.9	32.4	33.0	31.7	13.0×
Iterative NAT						
CMLM ($m = 5$) [5]	18.1	21.8	27.3	28.2	28.7	13.6×
+ $T = 4$	25.9	29.9	32.5	33.2	33.2	5.2×
+ $T = 10$	27.0	30.5	33.1	33.3	33.6	2.3×
Disco ($m = 5$) [12]	-	-	-	-	26.4	13.2×
+ $T =$ Adaptive*	27.3	31.3	33.2	33.3	32.9	3.9×
Proposed approach						
Vanilla NAT re-implementation	22.3	27.9	28.9	29.7	27.6	13.5×
LayerGLAT	26.3	31.8	32.1	32.5	32.3	13.1×
+ $T = 2$	26.8	31.9	32.7	32.7	33.2	7.9×
+ $T = 4$	27.1	32.1	33.0	32.7	33.6	4.5×
+ $T = 10$	27.3	32.4	33.4	32.9	34.1	2.0×
LayerGLAT ($m = 3$)	27.3	32.3	32.9	33.2	33.1	12.5×
+ $T = 2$	27.4	32.3	33.3	33.2	33.9	7.8×
+ $T = 4$	27.6	32.4	33.2	33.0	34.1	4.4×
+ $T = 10$	27.6	32.4	33.4	33.0	34.3	1.9×

†: Results of NAT models in this column are based on our implementation.
‡: Speedups are calculated using the IWSLT14 test set with a batch size of 1.
$\star$: $T \approx 3.7$ in the IWSLT14 test set based on our implementation.

fairseq library[5]. For the two WMT datasets, we have utilized the preprocessed data from Lee et al. [15] to ensure a consistent and fair comparison with prior studies.

3.2 Models and Settings

For all the experiments, we have utilized the base models and hyperparameters from the *fairseq* library [17]. For the WMT datasets, we have employed the base transformer configuration detailed in Vaswani et al. [29], consisting of 6 layers each for the encoder and decoder, 8 attention heads, 512 hidden size, and 2048 feedforward hidden size. For the smaller IWSLT dataset, we have adopted a scaled-down configuration with 6 layers in the encoder and decoder, 4 attention heads, 512 hidden size, and 1024 feedforward

[5] `https://github.com/facebookresearch/fairseq/blob/main/examples/translation/prepare-iwslt14.sh`

hidden size. We have initialized all models with the BERT scheme [2] and trained them with the Adam optimizer, setting $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-8}$, and a weight decay of 0.01. The learning rate initially increases over the first $10K$ steps and then decreases following an inverse square root manner. Training is completed in $200K$ updates with a mixed precision [18]. Other dataset-specific hyperparameters are listed in Appendix A. We have evaluated the model's performance with the tokenized BLEU score [20], and obtained the final model by averaging the top 5 checkpoints based on the validation set performance.

3.3 Knowledge Distillation

To train our NAT model, we have employed sequence-level knowledge distillation, wherein the target training sentences are replaced by the translations generated by a teacher AT model [14]. This technique has consistently proved to enhance the performance of both fully NATs and iterative NATs, making it a standard practice in the field [6,31]. For our experiments, the distilled dataset was created using a beam size of 5. For comparison, we have also trained an equivalent AT model on the same distilled dataset.

3.4 Noisy Parallel Decoding

As discussed in Section 2.1, in NAT inference the sentence length is determined by the length prediction module. To this aim, we have adopted Noisy Parallel Decoding (NPD) [6], which samples multiple-length candidates (denoted by their length, m) and generates outputs of various lengths simultaneously. These generated sentences are then re-ranked, either by the pretrained AT model or directly by the NAT model (as done in all our experiments). Importantly, sentences of different lengths are inputted in parallel, thereby preserving the inherent efficiency of the inference process.

4 Main Results

In Table 1, we present a comparison of our model against other AT and NAT baseline models. For the WMT datasets, our comparisons are based directly on the results reported in the literature. However, for the IWSLT14 dataset we report results from our own re-implementations to ensure a fair comparison of decoding efficiency across the different models. For reference, the BLEU scores that we have obtained with our re-implementations align closely with those reported in prior studies [1,11]. The speed-up calculations have been conducted on the IWSLT14 test set under identical hardware conditions, and averaged over three runs to minimize variance. For the AT models, we have utilized a caching mechanism to accelerate their decoding [17]. This approach has led to much more challenging speed-up comparisons for the NAT models, intentionally preventing the overstatement of their benefits [13,25].

4.1 Performance of Single-Pass Generation

We start by examining the performance of all models in the first, unconditional generation pass (i.e., $T = 1$). As expected, the two selected iterative baseline models, CMLM

and Disco [12], have exhibited significant performance degradation across all five translation tasks compared to the $T = 10$ case. For instance, on the IWSLT14 dataset, the BLEU score for CMLM has dropped from 33.6 to 28.7, and for Disco from 32.9 to 26.4. On the WMT14 dataset, CMLM has experienced a drop of almost 9 pp in BLEU score. In contrast, our LayerGLAT has produced a strong single-pass performance, outperforming the competitive fully NAT baseline (GLAT + DSLP) on the WMT14 and IWSLT14 datasets, and achieving comparable results on the WMT16 dataset. Furthermore, by employing a length beam size of $m = 3$, our model with a single pass has matched the performance of the two iterative baseline models with $T = 10$. Notably, on the WMT14 dataset, the performance of our model has even neared that of the AT baseline model. These findings validate the remarkable single-pass generation capabilities of LayerGLAT.

4.2 Iterative Performance

As shown in Table 1, the iterative baseline models have been able to achieve incremental performance improvements with increasing iterations, leading to a gain of up to 9 BLEU pp compared to $T = 1$, and eventually surpassing the fully NAT baselines. Our model has also demonstrated remarkable iterative generation capabilities; however, its relative improvements have been smaller, given its very strong first-pass performance. In general, its gains have been larger where the first-pass performance had left more margin for improvement: for instance, on the IWSLT14 dataset, our model ($m = 3$) has achieved a first-pass BLEU score of 33.1, lower than the AT baseline's score by 1.5 pp. This has allowed room for improving the score by 1.2 pp when moving from $T = 1$ to $T = 10$. However, in the larger WMT14 De→En task, our model had already scored 32.3 in its first pass, only 0.3 pp below the AT baseline. As such, increasing the number of iterations in this case has led to only notional improvements. These results suggest that the proposed model, while iterative, may be able to achieve a competitive performance in its first pass, dispensing with the need for further iterations, or reducing it substantially.

4.3 Inference Speed-up

Table 1 also presents the relative speed-up of various NAT models compared to the AT baseline (rightmost column). Despite employing an accelerating caching mechanism for the AT model, the NAT models have still proved up to 13.5 faster. Notably, noisy parallel decoding has proved more efficient as an improvement than iterative generation, as also reported by Gu et al. [7]. For example, on the IWSLT14 dataset, our model has achieved a score of 32.3 with $T = 1$ and $m = 1$. Increasing m to 3 or T to 2 has raised the score to very similar values (33.1 and 33.2, respectively, but the former has still achieved a strong speed-up of 12.5, whereas the latter has dropped it to 7.9. This finding validates that the first-pass performance of iterative NATs is crucial for better quality-speed trade-offs. However, the extent of improvement achievable through noisy parallel decoding is limited [6], as it still relies solely on unconditional generation. In such scenarios, iterative generation continues to be a useful approach for achieving additional improvements.

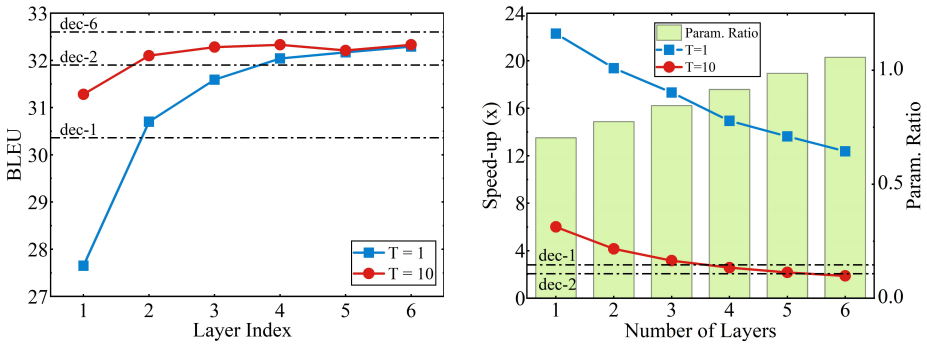

Fig. 2. Layer-wise BLEU scores (left) and the speed-up and parameter ratios of our model with different numbers of decoder layers (right) on the WMT14 De→En test set with $m = 3$.

5 Ablation and Sensitivity Analysis

In this section, we investigate the performance of the proposed approach with models of reduced size (5.1), different sampling rate (5.2), alternative sampling strategy (5.3), and by training with the original, "raw" data (5.4).

5.1 Performance With Reduced Model Size

As explained in Section 2.3, our training strategy allows LayerGLAT to perform effective predictions at inference time from any of its decoder layers. Given that the decoding speed is directly correlated with the number of layers, this capability allows the model to be deployed with fewer layers, potentially meeting diverse requirements for speed, quality, and model size. To illustrate this versatility, Fig. 2 (left) shows the layer-wise BLEU scores obtained by performing inference from each decoder layer of our model on the WMT14 De→En test set, for both single-pass ($T = 1$) and iterative ($T = 10$) prediction. Given that the decoder layers above that employed for inference become completely unnecessary at inference time, Fig. 2 (right) also shows the speed-up of our model deployed with any number of decoder layers, and its parameter ratio compared to the full teacher AT model (note that in the case of 6 layers, the ratio is slightly greater than one because our linear blocks are mildly larger). These results are compared with the teacher AT model, which features a 6-layer decoder (dec-6), and with two additional AT models with 1-layer (dec-1) and 2-layer (dec-2) decoders, respectively. In single-pass prediction, our 1-layer model has not matched the performance of the 1-layer AT model. However, its performance has significantly improved with $T = 10$, outperforming the 1-layer AT model and reaching levels comparable to those of the 6-layer NAT model, while $6\times$ faster and with only 70% of the parameters of the full AT model. Moreover, by increasing the number of layers, both its single-pass and iterative performance have consistently improved, with the single-pass performance with 6 layers nearing that of the full teacher AT model. We ascribe this desirable behavior

to our training strategy, which provides direct training signals to all the layers, not just the last one. As a major benefit, our model could be effectively deployed with different size-quality trade-offs.

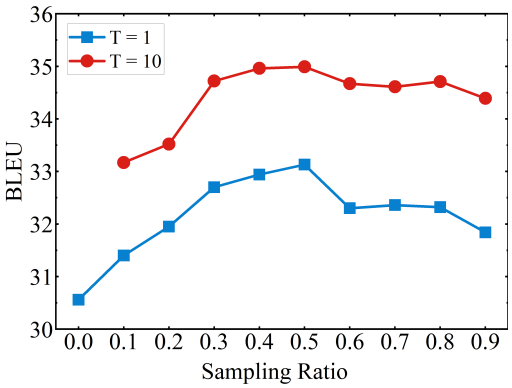

Fig. 3. BLEU scores with different sampling rate λ_1 on the IWSLT14 validation set.

5.2 Sampling Rate

To explore the impact of the set sampling rate, λ_1 on our model, we have conducted an experiment with variable λ_1 values on the IWSLT14 validation set. The results for both single-pass and multi-pass ($T = 10$) inference are shown in Fig. 3. For $\lambda_1 = 0$, our model has achieved its lowest single-pass performance and has shown a collapse of its iterative performance, evidenced by the decrease in BLEU score from 30.6 with $T = 1$ to 10.0 with $T = 10$ (out of scale; not plotted). Interestingly, introducing even a small number of ground-truth tokens during training ($\lambda_1 = 0.1$) has substantially boosted both single-pass and multi-pass performance. For increasing λ_1 values, the model's single-pass performance has initially increased, but then decreased, showing that an excess of ground-truth tokens during training is detrimental to its first-pass performance. Also the model's iterative performance has reached a peak for $\lambda_1 = 0.5$, but it has remained steadier also for larger values, probably because of the complementary impact of the adaptive learning rate, λ_2. Overall, these findings show that the carefully-dosed use of ground-truth tokens during training is the key component of the effectiveness of the proposed model.

5.3 Sampling Strategy

To better illustrate the advantages of our random layer selection during training over the original method, GLAT, proposed by Qian et al. [21], we have conducted a comparative experiment by fixing the selected layer to be the last layer, with all the other settings left unchanged. Fig. 4 plots the BLEU scores for predictions from the different layers

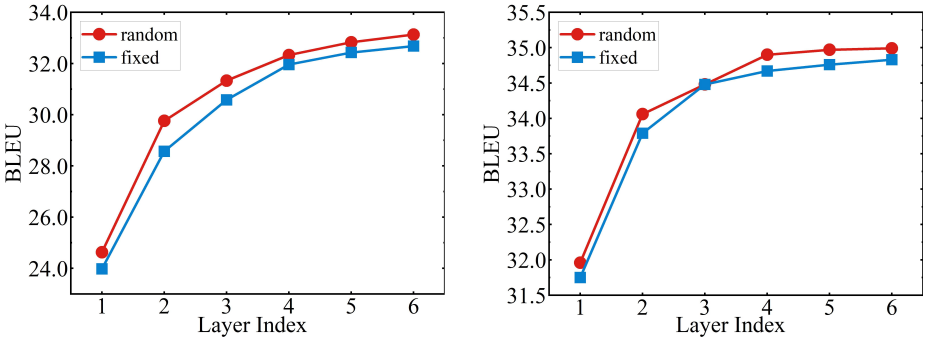

Fig. 4. BLEU scores from the different decoder layers for the original GLAT approach [21] ("fixed") and the proposed approach with random selection of the layer ("random") on the IWSLT14 validation set. (left: $T = 1$; right: $T = 10$).

$(1 - 6)$, showing that the proposed model has consistently outperformed the original glancing method in both single and iterative generation scenarios, including from the final layer.

Table 2. Test-set BLEU scores for the proposed model and CMLM trained on the raw training sets.

Iterations	WMT14 EN→DE		WMT16 EN→RO	
	CMLM	LayerGLAT	CMLM	LayerGLAT
$T = 1$	10.64	20.75	21.22	30.82
$T = 2$	-	22.74	-	31.58
$T = 4$	22.25	23.76	31.40	32.04
$T = 10$	24.61	24.70	32.86	32.32

5.4 Performance with Raw Data Training

We have also conducted an evaluation of the proposed model by training with the raw data (i.e., without knowledge distillation), with the aim of exploring the robustness of single-pass performance for our model in this case as well. The results are reported in Table 2, where we also compare with CMLM [5] in the same scenario. Table 2 shows that a model such as CMLM has experienced a very substantial drop in its single-pass performance compared to $T = 10$ (-12.97 pp for WMT14 En→De and -11.64 pp for WMT16 En→Ro). Conversely, our model has shown a much more limited performance degradation when decreasing the number of iterations from $T = 10$ to $T = 1$ (-3.95 pp and -1.50 pp, respectively), proving more robust to the undistilled training data. In addition, our model's single-pass performance has outperformed that of CMLM by large margins.

6 Related Work

Recent advancements in iterative NAT models have sought to achieve a better trade-off between quality and speed, positioning their performance between that of fully NATs and ATs. However, we believe that a common oversight for these models is the insufficient attention paid to the performance of the first generation pass ($T = 1$). For instance, Huang et al. [11] have enhanced NAT training by substituting some target tokens with the model's own predictions, acknowledging that during inference they will differ from the target tokens. Similarly, Savinov et al. [24] have introduced a step-unrolled diffusion model that iteratively generates tokens from random initial states. However, their approach requires up to $T = 16$ iterations to achieve satisfactory results. Both these approaches show a tendency to prioritize iterative refinement over first-pass performance, which seems rather limiting, especially considering that single-pass NAT models have made substantial progress in recent years, rivalling the quality of AT models while drastically speeding up generation [22,7]. In light of these developments, enhancing the first-pass performance of iterative NATs seems crucial to retain their relevance.

In addition to the approaches covered by our study, other recent works have contributed to improving NATs' performance, particularly when the models are trained on raw datasets. For example, Sun et al. [28] and Shao et al. [26] have optimized training objectives other than the cross entropy, and h et al. [3]ave proposed a positionally-relaxed cross entropy objective. We may explore these approaches in future work.

7 Conclusion

This paper has proposed LayerGLAT, an iterative non-autoregressive transformer that aims to achieve strong performance in both single-pass and multi-pass generation. The key idea behind the proposed approach is a layer-wise training strategy that aims to replicate the operating conditions of both single-pass and multi-pass prediction, which is achieved by carefully injecting target tokens during training and applying a specific training objective to each individual layer of the transformer's decoder. The experiments over three machine translation datasets of different sizes have shown a remarkable number of outcomes: 1) the BLEU scores of the proposed approach with a single iteration have equalled or surpassed those of two state-of-the-art iterative NATs (CMLM and Disco) with $T = 10$ iterations; 2) the proposed approach has comprehensively outperformed a non-iterative NAT (GLAT) in single-pass generation; and 3) the single-pass speed-up over an equivalent AT baseline (12.5x for beam size $m = 3$) has proved comparable to that of less performing NAT baselines. A further benefit of the proposed approach is that the trained model is capable of strong predictive performance from any of the decoder layers, thus allowing it to be deployed in fields with any subset of the layers. This can offer a flexible trade-off between model speed, quality, and size.

Ethics Considerations

In our self-assessment, our work does not raise any ethical issues other than those shared with the whole field of natural language generation. The overarching risk of all generative models is that they may generate text that is inappropriate for their users under

several dimensions (correctness, respectfulness etc). However, such a risk can be mitigated by models of adequate accuracy and proper training. Overall, the quest for more efficient transformer models such as NATs can be regarded as beneficial for society, as it aims to stem the costs and resources associated with the operation of these models.

Acknowledgments. The first author is funded by the China Scholarship Council (CSC) from the Ministry of Education of the P. R. of China.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bao, Y., Zhou, H., Huang, S., Wang, D., Qian, L., Dai, X., Chen, J., Li, L.: latent-GLAT: Glancing at latent variables for parallel text generation. ArXiv **abs/2204.02030** (2022)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
3. Du, C., Tu, Z., Jiang, J.: Order-agnostic cross entropy for non-autoregressive machine translation. In: Proc. of ICML (2021)
4. Ghazvininejad, M., Karpukhin, V., Zettlemoyer, L., Levy, O.: Aligned cross entropy for non-autoregressive machine translation. In: International Conference on Machine Learning (2020)
5. Ghazvininejad, M., Levy, O., Liu, Y., Zettlemoyer, L.: Mask-predict: Parallel decoding of conditional masked language models. In: EMNLP (2019)
6. Gu, J., Bradbury, J., Xiong, C., Li, V.O.K., Socher, R.: Non-autoregressive neural machine translation. ArXiv **abs/1711.02281** (2018)
7. Gu, J., Kong, X.: Fully non-autoregressive neural machine translation: Tricks of the trade. ArXiv **abs/2012.15833** (2021)
8. Heafield, K., Zhu, Q., Grundkiewicz, R.: Findings of the wmt 2021 shared task on efficient translation. In: Conference on Machine Translation (2021)
9. Huang, C., Zhou, H., Zaiane, O.R., Mou, L., Li, L.: Non-autoregressive translation with layer-wise prediction and deep supervision. In: AAAI (2022)
10. Huang, F., Tao, T., Zhou, H., Li, L., Huang, M.: On the learning of non-autoregressive transformers. In: ICML (2022)
11. Huang, X., Pérez, F., Volkovs, M.: Improving non-autoregressive translation models without distillation. In: International Conference on Learning Representations (2022)
12. Kasai, J., Cross, J., Ghazvininejad, M., Gu, J.: Non-autoregressive machine translation with disentangled context transformer. In: International Conference on Machine Learning (2020)
13. Kasai, J., Pappas, N., Peng, H., Cross, J., Smith, N.A.: Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In: ICLR (2021)
14. Kim, Y., Rush, A.M.: Sequence-level knowledge distillation. In: EMNLP (2016)
15. Lee, J., Mansimov, E., Cho, K.: Deterministic non-autoregressive neural sequence modeling by iterative refinement. In: EMNLP (2018)
16. Libovický, J., Helcl, J.: End-to-end non-autoregressive neural machine translation with connectionist temporal classification. ArXiv **abs/1811.04719** (2018)
17. Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., Auli, M.: fairseq: A fast, extensible toolkit for sequence modeling. In: NAACL (2019)
18. Ott, M., Edunov, S., Grangier, D., Auli, M.: Scaling neural machine translation. In: WMT (2018)

19. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L.E., Simens, M., Askell, A., Welinder, P., Christiano, P.F., Leike, J., Lowe, R.J.: Training language models to follow instructions with human feedback. ArXiv **abs/2203.02155** (2022)
20. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: ACL (2002)
21. Qian, L., Zhou, H., Bao, Y., Wang, M., Qiu, L., Zhang, W., Yu, Y., Li, L.: Glancing transformer for non-autoregressive neural machine translation. In: ACL (2021)
22. Qian, L., Zhou, Y., Zheng, Z., Zhu, Y., Lin, Z., Feng, J., Cheng, S., Li, L., Wang, M., Zhou, H.: The volctrans glat system: Non-autoregressive translation meets wmt21. In: WMT (2021)
23. Saharia, C., Chan, W., Saxena, S., Norouzi, M.: Non-autoregressive machine translation with latent alignments. In: EMNLP (2020)
24. Savinov, N., Chung, J., Binkowski, M., Elsen, E., van den Oord, A.: Step-unrolled denoising autoencoders for text generation. ArXiv **abs/2112.06749** (2021)
25. Schmidt, R.M., Pires, T.J.P., Peitz, S., Lööf, J.: Non-autoregressive neural machine translation: A call for clarity. ArXiv **abs/2205.10577** (2022)
26. Shao, C., Zhang, J., Feng, Y., Meng, F., Zhou, J.: Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In: AAAI Conference on Artificial Intelligence (2019)
27. Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., Wei, F.: Retentive network: A successor to transformer for large language models. ArXiv **abs/2307.08621** (2023)
28. Sun, Z., Li, Z., Wang, H., Lin, Z., He, D., Deng, Z.: Fast structured decoding for sequence models. In: NeurIPS (2019)
29. Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. ArXiv **abs/1706.03762** (2017)
30. Xiao, Y., Wu, L., Guo, J., Li, J., Zhang, M., Qin, T., Liu, T.Y.: A survey on non-autoregressive generation for neural machine translation and beyond. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**, 11407–11427 (2022)
31. Zhou, C., Neubig, G., Gu, J.: Understanding knowledge distillation in non-autoregressive machine translation. ArXiv **abs/1911.02727** (2020)

A Dataset-Specific Hyperparameters

Table A.1 shows the dataset-specific hyperparameters used for training our model. The "batch size" in the table is calculated by multiplying the number of tokens per GPU, the number of GPUs, and the number of gradient accumulations before committing a gradient update [18].

Table A.1. Dataset-specific hyperparameters used for training our model.

Parameters	IWSLT14	WMT16	WMT14
learning rate	$5 * 10^{-4}$	$5 * 10^{-4}$	$7 * 10^{-4}$
dropout	0.3	0.3	0.1
batch size	8K	32K	64K

B Training Algorithm

Algorithm B shows the key steps of our training procedure. Steps 2-6 describe the generation of the sequence of binary masks that is used to select tokens in input to the various decoder layers. The binary masks are obtained by sampling a Bernoulli distribution of parameter λ. Steps 7-15 describe the built training loss. For the first layer (Step 10), the input tokens are selected as a combination of ground-truth tokens and mask tokens. For the subsequent layers (Step 12), the input tokens are selected as a combination of ground-truth tokens and predictions from the previous layer.

Algorithm 1 Training Pipeline of LayerGLAT

Require: Ground-truth sequence: Y^{gt}: $\{y_1^{gt}, y_2^{gt}, \ldots, y_N^{gt}\}$
Require: Mask token: M
1: **for** i in range(max_training_steps) **do**
2: $d = \text{Uniform}\{1, 2, \ldots, D\}$, D: number of decoder layers
3: Predictions Y^d: $\{y_1^d, y_2^d, \ldots, y_N^d\} = \text{Decoder}_d(M_{1:N})$
4: λ_2 = HammingDistance(Y^{gt}, Y^d)
5: $\lambda = \lambda_1 * \lambda_2$
6: Masking sequence S: $\{s_1, s_2, \ldots, s_N\}$, $s_i \sim$ Bernoulli(λ)
7: TrainingLoss = 0
8: **for** d in range(D) **do**
9: **if** $d = 1$ **then**
10: Predictions Y^d: $\text{Decoder}_d(s_i * y_i^{gt} + (1 - s_i) * M, i \in 1, \ldots, N)$
11: **else**
12: Predictions Y^d: $\text{Decoder}_d(s_i * y_i^{gt} + (1 - s_i) * y_i^{d-1}, i \in 1, \ldots, N)$
13: **end if**
14: TrainingLoss += Cross_Entropy(Y^d, Y^{gt}) / D
15: **end for**
16: TrainingLoss.backward()
17: **end for**