Diffusion Directed Acyclic Transformer for Non-Autoregressive Machine Translation

Anonymous ACL submission

Abstract

Non-autoregressive transformers (NATs) predict entire sequences in parallel to reduce decoding latency, but they often encounter performance challenges due to the multi-modality problem. A recent advancement, the Directed Acyclic Transformer (DAT), addresses this issue by capturing multiple translation modalities to paths in a Directed Acyclic Graph (DAG). However, the collaboration with the latent variable introduced through the Glancing training (GLAT) is crucial for DAT to attain state-of-the-art performance. In this paper, we introduce Diffusion Directed Acyclic Transformer (Diff-DAT), which serves as an alternative to GLAT as a latent variable introduction for DAT. Diff-DAT offers two significant benefits over the previous approach. Firstly, it establishes a stronger alignment between training and inference. Secondly, it facilitates a more flexible tradeoff between quality and latency.

1 Introduction

The Transformer architecture (Vaswani et al., 2017a) has gained immense popularity, particularly in sequence-to-sequence learning problems like machine translation. Conventional Transformers employ an autoregressive approach to generation, yielding robust results but proving inefficient at inference due to its sequential decoding. To address this issue, Non-autoregressive Transformers (NATs) (Gu et al., 2018) was introduced, significantly boosting the decoding speed by generating all output tokens simultaneously. This advantage often comes with a trade-off in translation quality due to the challenging multi-modality problem (Gu et al., 2018), wherein a single source sentence may have multiple translations in the target language.

Numerous approaches have been proposed to address this challenge, primarily by introducing additional latent variables to reduce the number of translation modalities given the latent variables. Among them, the Directed Acyclic Transformer (DAT) (Huang et al., 2022b) emerges as the most promising approach. In DAT, translation modalities are assigned to paths in a Directed Acyclic Graph (DAG), enabling the model to capture multiple translation modalities. Although DAT enhances translation quality and diversity, it still requires additional context from the target as a latent variable to perform effectively. Huang et al. (2022b) demonstrated that the latent variable from Glancing training (GLAT) (Qian et al., 2021a) significantly improves DAT's performance.

In this work, we aimed to enhance the capabilities of DAT by introducing latent variables through a diffusion process. We integrated diffusion models into DAT, resulting in a novel model called Diffusion Directed Acyclic Transformer (Diff-DAT). We discovered that the diffusion model can effectively replace GLAT as a latent variable introduction mechanism, enabling DAT to function optimally. This integration offers two significant advantages. Firstly, diffusion models establish a stronger alignment between training and inference. Secondly, they facilitate a more flexible tradeoff between quality and speed by allowing decoding through multiple iterations. Results on multiple machine translation benchmarks demonstrate that our approach not only improves the performance of DAT on fully non-autoregressive decoding but also improves its iterative decoding performance without a significant increase in decoding latency.

2 Preliminaries

DA-Transformer replaces the traditional Transformer's decoder with a directed acyclic decoder. This decoder organizes its outputs as a directed acyclic graph (DAG), where each path corresponds to a specific translation modality. Given a bilingual pair, $X = \{x^1, \ldots, x^N\}$ and $Y = \{y^1, \ldots, y^M\}$, DAT sets the decoder length to $L = \lambda \cdot N$ and

models the translation probability by marginalizing out the paths in the DAG.

$$P_\theta(Y|X) = \sum_A P_\theta(Y|A,X)P_\theta(A|X), \quad (1)$$

where $A = \{a^1, ..., a^M\}$ represents a path with vertex indexes satisfying $1 = a^1 < ... < a^M = L$. $P_\theta(A|X)$ and $P_\theta(Y|A,X)$ denote the probability of path A and the probability of target sentence Y conditioned on path A, respectively. The DAG factorizes the path probability $P_\theta(A|X)$ based on the Markov assumption:

$$P_\theta(A|X) = \prod_{i=1}^{M-1} P_\theta(a^{i+1}|a^i, X) = \prod_{i=1}^{M-1} \mathbf{E}_{a^i,a^{i+1}}, \quad (2)$$

where $\mathbf{E} \in \mathbb{R}^{L\times L}$ is a row-normalized transition matrix. The DAG's unidirectional nature masks the lower triangular part of $\mathbf{E}$ to zeros. Once the path A is determined, token y^i is generated conditioned on the decoder hidden state with index a^i:

$$P_\theta(Y|A,X) = \prod_{i=1}^{M} P_\theta(y^i|a^i, X), \quad (3)$$

where $P_\theta(y^i|a^i, X)$ represents the translation probability of word y^i on the position a^i of decoder. To enhance the performance of DAT, GLAT was incorporated as an additional latent variable, denoted as Z. This variable is a randomly masked target and serves as an extra input for the decoder. The final training objective is to maximize the log-likelihood with the additional latent variable:

$$\log P_\theta(Y|X) = \mathbb{E}_{Q(Z|Y,\hat{A})} \log P_\theta(Y|X,Z), \quad (4)$$

since the decoder input is longer than the target sentence, (Huang et al., 2022b) first finds the most probable path $\hat{A} = \arg\max_A P_\theta(Y, A|X)$ and uses it to assign the masked target to vertices in the DAG.

3 Methodology

Training objective: The latent variable Z in GLAT is only used during training when a target is provided and is omitted during inference, forcing the model to rely solely on X to predict the target. This mismatch between training and inference can damage the model's generalizability. This motivates us to incorporate diffusion models into DAT.

Diffusion models (Sohl-Dickstein et al., 2015) aim to predict the target Y_0 through a sequence of latent variables $Y_{1:T} = Y_1, Y_2, \ldots, Y_T$. The forward process gradually adds noise to the target Y_0 over T steps to get Y_T, the final latent variable that follows a prior noise distribution. The backward process optimizes a neural network to denoise the noisy latent variables, reversing the forward process to recover Y_0. When the step size is infinitesimally small, the forward and backward processes have the same functional form.

While diffusion models provide a strong theoretical justification for aligning training and inference, the large number of time steps (T) hinders their practical application in NATs, where decoding latency is a critical concern. Therefore, we utilize absorbing discrete diffusion (Austin et al., 2021) that uses the absorbing state ($[M]$) as noise to add to the target sentence at each step, until all tokens are masked (noise distribution Y_T). This approach reduces the number of forward steps T required to reach the noise distribution.

We compute the translation probability by marginalizing out the paths in the DAG and latent variables from the diffusion backward process.

$$\begin{aligned} P_\theta(Y_0|X) &= \sum_{Y_{1:T}} \sum_A P_\theta(Y_{0:T}, A|X) \\ &= \sum_{Y_{1:T}} \sum_A P(Y_T) \prod_{t=1}^{T} P_\theta(Y_{t-1}, A|Y_t, X) \end{aligned} \quad (5)$$

Diff-DAT maximizes the variational lower bound (VLB) of the log-likelihood:

$$\begin{aligned} &\log P_\theta(Y_0|X) = \log \sum_{Y_{1:T}} \sum_A P_\theta(Y_{0:T}, A|X) \\ &\geq \sum_A \mathbb{E}_{Q(Y_{1:T}|Y_0,A)} \log \frac{P_\theta(Y_{0:T}, A|X)}{Q(Y_{1:T}|Y_0, A)} \\ &\approx \mathbb{E}_{Q(Y_{1:T}|Y_0,\hat{A})} \sum_A \log \frac{P_\theta(Y_{0:T}, A|X)}{Q(Y_{1:T}|Y_0, \hat{A})} \\ &= \sum_{t=1}^{T} \mathcal{L}_t + \text{const}, \end{aligned} \quad (6)$$

Where $Q(Y_{1:T}|Y_0, \hat{A})$ represents the forward process transition probabilities. Following Huang et al. (2022b), we use the most probable path $\hat{A}$ to sample the latent variable Y_t to avoid performing multiple forward passes through the neural network for all paths in the DAG in order to compute the objective.

The objective at time step t is

$$\mathcal{L}_t = \mathbb{E}_{Q(Y_t|Y_0,\hat{A})} \sum_A \Big[\log P_\theta(A|Y_t, X) - \mathrm{KL}[Q(Y_{t-1}|Y_t, Y_0, A) \| P_\theta(Y_{t-1}|Y_t, A, X)] \Big].$$

We compute the objective at time step t as follows:

$$\mathcal{L}_t = \sum_A \sum_{i=1}^{M} \gamma_t b_t^i \log \mathbf{P}_{a_i, y_0^i} \sum_{i=2}^{M} \mathbf{E}_{a_{i-1}, a_i}$$

where $b_t^i = \begin{cases} 1 & \text{if } y_t^i = [M] \\ 0 & \text{otherwise} \end{cases}$, and γ_t is the hyper-parameters defined by the forward process transition probability. Please refer to Appendix C for detailed derivation. This objective can be optimized using a dynamic programming algorithm similar to the approach in (Huang et al., 2022b). The key differences between Diff-DAT and DAT lie in the sampling of latent variables and the computation of the objective function based on the sampled latent variables at each step.

Inference: Diff-DAT can reuse various decoding strategies from DAT, such as greedy, lookahead, and joint-Viterbi, to perform decoding in a single iteration. However, unlike DAT, we can perform flexible iterative decoding based on the diffusion backward process. Given the model prediction from the previous step, we can continuously sample the latent variable for the next time step and perform denoising. Unlike the fixed T value during training, the diffusion can predict multiple steps at a time, resulting in a more flexible number of decoding steps during inference.

4 Experiments

4.1 Experimental Setup

Experimental Setup. We conduct experiments on multiple public NMT datasets: IWSLT14 En-De/De-En (Cettolo et al., 2014), WMT14 En-De (Bojar et al., 2014), and WMT16 Ro-En/En-Ro (Bojar et al., 2016). To ensure a fair comparison, we used the same settings as previous works (Ghazvininejad et al., 2019; Huang et al., 2022c) and reported the test performance in BLEU score (Papineni et al., 2002a) (Appnedix A).

Main results: Table 1 demonstrates that Diff-DAT significantly outperforms the baselines while maintaining low decoding latency. Compared to DAT, Diff-DAT shows improvements even in the first iteration, implying that diffusion improves

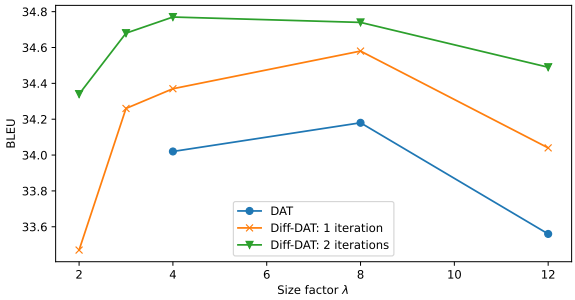

Figure 1: Effects of λ on IWSLT14 De-En. The graph size of DAT is λ times the of source length. We use Join-Viterbi decoding to evaluate BLEU.

DAT's ability to find the optimal path for each reference, effectively addressing multi-modality. We observed a more pronounced improvement between the first and second steps in the WMT16 En-Ro and IWSLT14 De-En datasets, where we used a smaller graph size ($\lambda = 4$). Given that these datasets exhibit lower multi-modality due to the smaller training data and the search space in the graph is limited, the second step of Diff-DAT remarkably boosts translation quality by capturing more dependencies between tokens. We also observed that the improvement between the two steps is more pronounced when using the "Lookahead" decoding strategy. We argue that this occurs because the path found by "Lookahead" is less optimal than "Joint-Viterbi". As a result, the improvement relies more on the better dependencies captured between tokens. Diff-DAT's second decoding step outperforms FA-DAT, demonstrating the capability of iterative decoding in addressing the monotonic assumption in DAT. Diff-DAT outperforms all other single-step decoding methods and achieves comparable or better performance than other multistep decoding methods while maintaining a minimal trade-off in decoding latency.

Although FA-DAT seems more effective in terms of BLEU score and decoding latency, its objective is based on the n-gram count, which is specifically designed to align with the BLEU score metric. Ma et al. (2023) showed that FA-DAT reduces the diversity of translations, making the model unable to capture the various translation modalities present in the data. We conduct further experiments to compare our proposed method with FA-DAT on other metrics, such as COMET in Appendix B, to demonstrate that FA-DAT only biases toward high BLEU scores but not other evaluation metrics.

Ablation study on the graph size: Figure 1 illus-

Table 1: Results on WMT14 En-De, WMT16 En-Ro, WMT17 Zh-En and IWSLT14 En-De/De-En datasets. The best performance of the NAT methods is **bolded**. * denotes our implementation.

Model		Iter.	Speedup	IWSLT14 De-En	WMT14 En-De	WMT16 En-Ro	WMT17 Zh-En
Transformer (Vaswani et al., 2017b)		M	1.0×	34.66	27.60	34.16	23.70
CMLM (Ghazvininejad et al., 2019)		10	2.2×	31.80	24.61	32.86	-
CMLM+SMART (Ghazvininejad et al., 2020b)		10	2.2×	30.74	25.10	32.71	-
DiSCo (Kasai et al., 2020)		≈ 4	3.5×	-	25.64	-	-
Imputer (Saharia et al., 2020)		8	2.7×	-	-	25.00	-
CMLMC (Huang et al., 2022c)		10	1.7×	34.28	26.40	34.14	-
DAT *(Huang et al., 2022b)	+ Lookahead	1	14.0×	33.79	26.52	33.46	22.42
	+ Joint-Viterbi	1	13.2×	34.02	26.67	33.65	23.00
FA-DAT * (Ma et al., 2023)	+ Lookahead	1	14.0×	34.65	27.29	33.72	22.73
	+ Joint-Viterbi	1	13.2×	34.66	**27.31**	33.74	22.87
Diff-DAT	+ Lookahead	1	14.0×	34.21	26.34	33.65	22.60
		2	11.8×	34.68	26.83	**34.01**	22.82
	+ Joint-Viterbi	1	13.2×	34.37	26.94	33.72	23.60
		2	9.2×	**34.77**	27.12	34.00	**23.78**

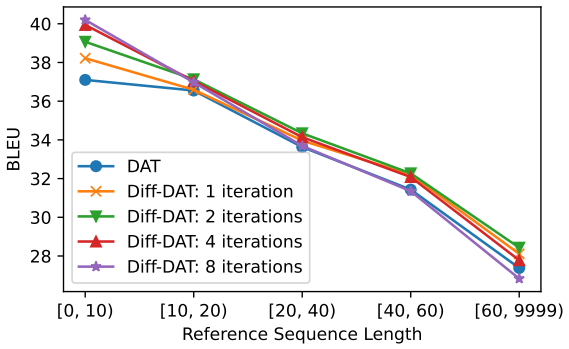

Figure 2: The effect of sequence length.

trates the results of DAT and Diff-DAT with varying graph sizes. As noted by Huang et al. (2022b), larger graph sizes complicate transition predictions, leading to a performance decline for both DAT and Diff-DAT. Nonetheless, Diff-DAT consistently outperforms DAT across all graph sizes. The graph size also influences the iterative decoding performance of Diff-DAT. Both transition predictions and iterative refinement contribute to capturing token dependencies in the generated sentence. When the graph size is small, iterative decoding significantly aids due to the limitations in transition predictions.

Ablation study on the number of decoding steps: We examine the impact of graph size on iterative decoding. Figure 2 displays BLEU scores categorized by reference length on IWSLT14 DE-EN dataset, results on other datasets are shown in the Appendix. Contrasting trends were observed in different length intervals: in the first interval (length [0, 40)), the BLEU score increases as the number of decoding iterations increases; in the second interval (length [40, ∞)), the BLEU score decreases as the number of decoding iterations increases. Since sequence length directly influences graph size, we conclude that once the graph size reaches a certain threshold, it negatively affects iterative decoding. Larger graph sizes make transition probability prediction more challenging. Additionally, each decoding iteration generates sub-paths in the graph that the model must navigate, further complicating the prediction of transition probabilities. These factors make it increasingly difficult for the model to predict transitions with each iteration. Errors from previous iterations affect subsequent ones, further damaging the model's ability to capture dependencies through transition predictions. This intriguing challenge is left for future work.

5 Conclusion

Our study introduces Diff-DAT, a novel approach for enhanced non-autoregressive machine translation. Through a fusion of diffusion models and DAT objectives and the integration of various decoding schemes, Diff-DAT effectively addresses the multi-modality problem, achieving superior translation quality while maintaining fast decoding speed. Extensive experiments across diverse benchmarks demonstrate the effectiveness of Diff-DAT, establishing a new state-of-the-art in non-autoregressive translation. Our work bridges the gap between decoding efficiency and translation quality, advancing the field of sequence-to-sequence learning.

6 Limitations

Although the optimal trade-off between decoding latency and performance occurs with 2-step decoding, increasing the number of decoding iterations does not lead to consistent performance improvements. Instead, it degrades the performance, particularly for long sentences, as demonstrated in our ablation study. This presents an ongoing challenge for iterative decoding in DAT, which remains an area for future research.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pages 131–198. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2849–2859. PMLR.

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

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China. Association for Computational Linguistics.

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

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

Shangtong Gui, Chenze Shao, Zhengrui Ma, Yunji Chen, Yang Feng, et al. 2024. Non-autoregressive machine translation with probabilistic context-free grammar. *Advances in Neural Information Processing Systems*, 36.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Chenyang Huang, Hao Zhou, Osmar R. Zaïane, Lili Mou, and Lei Li. 2022a. Non-autoregressive translation with layer-wise prediction and deep supervision. *The Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. 2022b. Directed acyclic transformer for non-autoregressive machine translation. In *International Conference on Machine Learning*, pages 9410–9428. PMLR.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. 2022c. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5144–5155. PMLR.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zhengrui Ma, Chenze Shao, Shangtong Gui, Min Zhang, and Yang Feng. 2023. Fuzzy alignments in directed acyclic graph for non-autoregressive machine translation. In *International Conference on Learning Representations*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for

sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002a. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002b. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

L. Qian, H. Zhou, Y. Bao, M. Wang, L. Qiu, W. Zhang, Y. Yu, and L. Li. 2021a. Glancing transformer for non-autoregressive neural machine translation. In *Meeting of the Association for Computational Linguistics*.

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

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1098–1108, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Chenze Shao, Zhengrui Ma, and Yang Feng. 2022. Viterbi decoding of directed acyclic transformer for non-autoregressive machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4390–4397, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need. *Advances in neural information processing systems*, 30.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

A Details Experimental Setup

Table 2: The number of sentence pairs of the training, validation, and test sets in each dataset

	IWSLT14 De-En	WMT14 En-De	WMT16 En-Ro	WMT17 Zh-En
Train	160k	4.5M	610k	20M
Validation	6.75k	3k	2k	2k
Test	6.75k	3k	2k	2k

Table 3: Hyper-parameters

Hyper-parameters	IWSLT14 De-En	WMT14 En-De	WMT16 En-Ro	WMT17 Zh-En
learning rate	0.0005	0.0005	0.0005	0.0005
warmup	30k	10k	15k	10k
dropout	0.3	0.1	0.3	0.1
updates	200k	300k	200k	300k
batch size	8k tokens	64k tokens	32k tokens	64k tokens
Size factor λ	4	8	4	8

Datasets. We conduct experiments on four popular machine translation dataset WMT14 English-German (En-De, 4.5M sentence pairs), WMT17 Chinese-English (Zh-En, 20M sentence pairs) and WMT16 English-Romanian (En-Ro, 610k sentence pairs) and IWSLT14 German-English (De-En 160k sentence pairs). The details size of each dataset is given in table 2. We apply BPE (Sennrich et al., 2015) to learn a joint subword vocabulary for En-De, En-Ro, and De-En and separate vocabularies for Zh-En on the tokenized data.

Baselines. We compare our Diff-DAT against leading NAR baselines, including CMLM (Ghazvininejad et al., 2019) and its variants, CMLM+SMART (Ghazvininejad et al., 2020b), CMLM+AXE (Ghazvininejad et al., 2020a) and CMLM+OaXE (Du et al., 2021), DisCo (Kasai et al., 2020), Imputer (Saharia et al., 2020), GLAT (Qian et al., 2021b), DSLP (Huang et al., 2022a), CMLMC (Huang et al., 2022c), PCFG-NAT (Gui et al., 2024), DAT (Huang et al., 2022b), and FA-DAT (Ma et al., 2023).

Metrics. For fair comparisons with previous work, we use tokenized BLEU (Papineni et al., 2002b) for all benchmarks. The decoding speedup is measured with a batch size of 1.

Implementation details. All baselines and our proposed method are implemented using the open-source toolkit Fairseq (Ott et al., 2019). BLEU scores are evaluated on the validation set, and the final model is obtained by averaging the best 5 checkpoints. To ensure fair comparisons with previous work, we adhere to the training hyper-parameters set by (Huang et al., 2022b,c), as detailed in Table 3. On the IWSLT14 dataset, we employ the Transformer-*small* configuration 512-1024-4, whereas on the WMT datasets, we utilize the Transformer-*base* configuration 512-2048-8 for both the encoder and decoder in our autoregressive baseline. These numerical values correspond to the embedding dimension, FFN layer size, and number of attention heads, respectively. Our model architecture strictly adheres to the settings of DAT, where we set the decoder length to 8 times the source length ($\lambda = 8$) and incorporate graph positional embeddings as decoder inputs unless otherwise specified. Additionally, we set the number of time steps adaptively equal to the length of the source sentence. We assess BLEU scores on the validation set and average the best 5 checkpoints to obtain the final model. In cases where DAT performance is not reported for the WMT16 dataset, we independently train the model using the original code and maintain the same settings in our Diff-DAT approach. Throughout all experiments, we utilize the Adam optimizer (Kingma and Ba, 2014) with default settings and conduct training on 4 Nvidia A100-80G GPUs.

method	IWSLT14 De-En	IWSLT14 De-En	WMT16 En-Ro	WMT16 En-Ro
	BLEU	COMET	BLEU	COMET
DAT	34.02	0.7617	33.65	0.7608
FA-DAT	34.66	0.7637	33.74	0.7618
Diff-DAT - 1 iter	34.37	0.7642	33.72	0.7619
Diff-DAT - 2 iters	34.77	0.7720	34.00	0.7717
FA-Diff-DAT - 1 iter	34.73	0.7655	34.13	0.7645
FA-Diff-DAT - 2 iters	35.06	0.7727	34.27	0.7751

Table 4: Comparison between Diff-DAT, DAT, and FA-DAT on COMET metric

B Experiments with COMET metric

We further provide evaluations using COMET. As shown in the table 4, our improvements are even more remarkable when evaluated with COMET, particularly in the context of iterative decoding. While FA-DAT demonstrates significant improvements in BLEU scores, its enhancements under COMET evaluation are relatively marginal.

C Derivations of the Variational Lower Bound for Diff-DAT

The following provides the derivation for the loss objective of Diff-DAT:

$$
\begin{aligned}
&\log P_\theta(Y_0|X) = \log \sum_{Y_{1:T}} \sum_{A} P_\theta(Y_{0:T}, A|X) \\
&= \log \sum_{A} \mathbb{E}_{Q(Y_{1:T}|Y_0,A)} \frac{P_\theta(Y_{0:T}, A_{1:T}|X)}{Q(Y_{1:T}|Y_0, A)} \\
&\geq \sum_{A} \mathbb{E}_{Q(Y_{1:T}|Y_0,A)} \log \frac{P_\theta(Y_{0:T}, A_{1:T}|X)}{Q(Y_{1:T}|Y_0, A)} \\
&= \sum_{A} \mathbb{E}_{Q(Y_{1:T}|Y_0,A)} \log \frac{P_\theta(Y_T|X) \prod_{t=1}^{T} P_\theta(A|Y_t, X) P_\theta(Y_{t-1}|Y_t, A, X)}{Q(Y_T|Y_0) \prod_{t=2}^{T} Q(Y_{t-1}|Y_t, Y_0, A)} \\
&= \sum_{t=2}^{T} \mathbb{E}_{Q(Y_t|Y_0,A)} \sum_{A} \Big[\log P_\theta(A|Y_t, X) - \mathrm{KL}[Q(Y_{t-1}|Y_t, Y_0, A) \| P_\theta(Y_{t-1}|Y_t, A, X)] \Big] \\
&+ \mathbb{E}_{Q(Y_1|Y_0)} \sum_{A} \Big[\log P_\theta(A|Y_1, X) + \log P_\theta(Y_0|Y_1, A, X) \Big] + \text{const}
\end{aligned}
$$

Forward process: At time step t, for each path A_t in the DAG, we sample $Y_t \sim Q(Y_t|Y_{t-1}, A_t)$ by applying the following forward transition probabilities independently for each token y^i_{t-1} in the sequence Y_{t-1}:

$$
\begin{cases}
Q(y^i_t = [M]|y^i_{t-1} = [M], a_i \in A) &= 1 \\
Q(y^i_t = y^i_0|y^i_{t-1} = y^i_0, a_i \in A) &= \beta_t \\
Q(y^i_t = [M]|y^i_{t-1} = y^i_0, a_i \in A) &= 1 - \beta_t \\
Q(y^i_t = y^i_0|y^i_{t-1} = [M], a_i \in A) &= 0 \\
Q(y^i_t = [M]|y^i_{t-1}, a_i \notin A) &= 1 \\
Q(y^i_t \neq [M]|y^i_{t-1}, a_i \notin A) &= 0
\end{cases} \tag{7}
$$

For each token, if it is part of the current path A_t, it has a probability of transitioning to the absorbing state $[M]$, or it may remain unchanged. Tokens that are not part of the path, or were already masked in previous steps, will always remain in the absorbing state. We can also perform a t-step marginal, $Q(Y_t|A_t, Y_0)$, to directly sample Y_t from Y_0:

$$\begin{cases} Q(y_t^i = y_0^i | y_0^i, a_i \in A) &= \alpha_t \\ Q(y_t^i = [M] | y_0^i, a_i \in A) &= 1 - \alpha_t \\ Q(y_t^i = [M] | y_0^i, a_i \notin A) &= 1 \\ Q(y_t^i = y_0^i | y_0^i, a_i \notin A) &= 0 \end{cases} \tag{8}$$

where $\alpha_t := \prod_{i=1}^{t} \beta_t$ is specified to decrease from 1 to 0

Backward process: We then compute the posterior at time $t-1$ as $Q(Y_{t-1}|Y_t, Y_0, A) = \frac{Q(Y_t|Y_{t-1},A)Q(Y_{t-1}|Y_0,A)}{Q(Y_t|Y_0,A)}$. The backward transition probabilities for each token are then calculated as follows:

$$\begin{cases} Q(y_{t-1}^i = y_0^i | y_t^i = [M], y_0^i, a_i \in A) &= \gamma_t \\ Q(y_{t-1}^i = [M] | y_t^i = [M], y_0^i, a_i \in A) &= 1 - \gamma_t \\ Q(y_{t-1}^i = [M] | y_t^i = y_0^i, y_0^i, a_i \in A) &= 0 \\ Q(y_{t-1}^i = [M] | a_i \notin A) &= 1 \end{cases} \tag{9}$$

where $\gamma_t = \frac{\alpha_{t-1} - \alpha_t}{1 - \alpha_t}$. Instead of training a neural network $f_\theta(Y_t, X)$ to directly predict the logits of the distribution $P_\theta(Y_{t-1}|Y_t, A, X)$, we follow the approach in (Ho et al., 2020) and focus on learning a neural network $f_\theta(Y_t, X)$ to predict the logits of the distribution $\tilde{P}_\theta(\tilde{Y}_0|Y_t, A, X)$. We then combine this with $Q(Y_{t-1}|Y_t, Y_0, A)$ and sum over the one-hot representations of Y_0 to obtain the following parameterization: $P_\theta(Y_{t-1}|Y_t, A, X) = \sum_{\tilde{Y}_0} \tilde{P}_\theta(\tilde{Y}_0|Y_t, A, X) Q(Y_{t-1}|Y_t, \tilde{Y}_0, A)$ The parameterized backward transition probabilities for each token are then derived as follows:

$$\begin{cases} P_\theta(y_{t-1}^i = y_0^i | y_t^i = [M], a_i \in A, X) = \gamma_t \mathbf{P}_{a_i, y_0^i} \\ P_\theta(y_{t-1}^i = [M] | y_t^i = [M], a_i \in A, X) = 1 - \gamma_t \\ P_\theta(y_{t-1}^i = [M] | y_t^i = y_0^i, a_i \in A, X) = 0 \\ P_\theta(y_{t-1}^i = [M] | a_i \notin A, X) = 1 \end{cases} \tag{10}$$

where $\mathbf{P} = \tilde{P}_\theta(\tilde{y}_0^i | Y_t, X) = f_\theta(Y_t, X) \in \mathbb{R}^{L \times |V|}$ is the matrix containing the token distributions on the L vertices, and $\tilde{P}_\theta(\tilde{y}_0^i = y_0^i | a_i, Y_t, X) = \mathbf{P}_{a_i, y_0^i}$. Since the forward and backward processes are factorized as conditionally independent over the image or sequence elements, the KL divergence between Q and P_θ can be computed by simply summing over all possible values of each random variable, which is given by the following:

$$\begin{aligned} &\mathrm{KL}(Q(y_{t-1}^i | y_t^i, y_0^i, A_t) \| P_\theta(y_{t-1}^i | y_t^i, A, X)) = \\ &\begin{cases} -\gamma_t \log \mathbf{P}_{a_i, y_0^i} & \text{if} \quad y_t^i = [M] \text{ and } a_i \in A \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

Put it into $\mathcal{L}_t$ in (6), we have the objective:

$$\mathcal{L}_t = \sum_A \mathbb{E}_{Q(Y_t|Y_0,A)} \sum_{i=1}^{M} \gamma_t b_t^i \log \mathbf{P}_{a_i, y_0^i} \sum_{i=2}^{M} \mathbf{E}_{a_{i-1}, a_i}$$

where $b_t^i = \begin{cases} 1 & \text{if} \quad y_t^i = [M] \\ 0 & \text{otherwise} \end{cases}$. Optimizing requires multiple forward passes of the neural network each time we sample $Y_t \sim Q(Y_t|Y_0, A)$ given a path A. To simplify this, we condition only on the path with the highest probability $\hat{A} = \arg\max_A P_\theta(Y_0, A|X)$, and sample $Y_t \sim Q(Y_t|Y_0, \hat{A})$. Consequently, $\mathcal{L}_t$ becomes:

$$\mathcal{L}_t = \mathbb{E}_{Q(Y_t|Y_0,\hat{A})} \sum_A \sum_{i=1}^{M} \gamma_t b_t^i \log \mathbf{P}_{a_i, y_0^i} \sum_{i=2}^{M} \mathbf{E}_{a_{i-1}, a_i}$$

We can efficiently compute $\mathcal{L}_t$ using dynamic programming, similar to the approach in DAT. During training, we sample a time step t and update the model parameters to maximize the objective in (**??**).

D Fuzzy Alignment in Diff-DAT

FA-DAT (Ma et al., 2023) addresses the strict monotonic alignment in DAT by fine-tuning it using a fuzzy alignment objective as follows:

$$p_n(\tilde{Y}_0, Y_0) = \frac{\sum_{g \in G_n(Y_0)} \min(C_g(\tilde{Y}_0), C_g(Y_0)}{\sum_{g \in G_n(\tilde{Y}_0)} C_g(\tilde{Y}_0)} \tag{12}$$

$$p_n(\theta, Y_0) = \mathbb{E}_{P_\theta(A|X)} p_n(P_\theta(\tilde{Y}_0|A, X), Y_0) \tag{13}$$

Due to the high complexity, they optimize a more tractable approximation of (13) as follows:

$$\tilde{p}_n(\theta, Y_0) = \frac{\sum_{g \in G_n(Y_0)} \min(\mathbb{E}_{P_\theta(A|X)} C_g(P_\theta(\tilde{Y}_0|A, X)), C_g(Y_0)}{\mathbb{E}_{P_\theta(A|X)} \sum_{g \in G_n(\tilde{Y}_0)} C_g(P_\theta(\tilde{Y}_0|A, X))} \tag{14}$$

(14) can be efficiently computed using Dynamic Programming as outlined in (Ma et al., 2023). We extend this objective for fine-tuning Diff-DAT as follows:

$$\tilde{p}_n(\theta, Y_0, t) = \frac{\sum_{g \in G_n(Y_0)} \min(\mathbb{E}_{Q(Y_t|Y_0, \hat{A}, X)} \mathbb{E}_{P_\theta(A|X)} C_g(P_\theta(\tilde{Y}_0|Y_t, A, X)), C_g(Y_0)}{\mathbb{E}_{Q(Y_t|Y_0, \hat{A}, X)} \mathbb{E}_{P_\theta(A|X)} \sum_{g \in G_n(\tilde{Y}_0)} C_g(P_\theta(\tilde{Y}_0|Y_t, A, X))} \tag{15}$$

Similar to Diff-DAT, at each gradient descent step, we sample $t \sim \text{Uniform}(T)$ and $Y_t \sim Q(Y_t|Y_0, A^0, X)$ to compute the objective (15).

We conducted experiments to compare Diff-DAT with a Fuzzy Alignment objective (FA-Diff-DAT) against other baselines, as shown in Table 5. When combined with the Fuzzy Alignment objective, FA-Diff-DAT further improves performance, surpassing FA-DAT in both 1 and 2 iterations of decoding.

Table 5: Comparison between Diff-DAT, FA-Diff-DAT, DAT, and FA-DAT. The highest-performing NAT methods are highlighted in **bold**. * indicates our implementation.

Model		**Iter.**	**Speedup**	**IWSLT14 De-En**	**WMT16 En-Ro**
Transformer (Vaswani et al., 2017b)		M	1.0×	34.66	34.16
DAT *(Huang et al., 2022b)	+ Lookahead	1	14.0×	33.79	33.46
	+ Joint-Viterbi	1	13.2×	34.02	33.65
	+ Greedy	1	14.2×	33.68	33.28
FA-DAT * (Ma et al., 2023)	+ Lookahead	1	14.0×	34.65	33.72
	+ Joint-Viterbi	1	13.2×	34.66	33.74
	+ Greedy	1	14.2×	34.64	33.69
Diff-DAT	+ Lookahead	1	14.0×	34.21	33.65
		2	11.8×	34.68	34.01
	+ Joint-Viterbi	1	13.2×	34.37	33.72
		2	9.2×	34.77	34.00
	+ Greedy	1	14.2×	33.96	33.47
		2	12.2×	34.51	33.87
FA-Diff-DAT	+ Lookahead	1	14.0×	34.71	33.74
		2	11.8×	35.04	33.97
	+ Joint-Viterbi	1	13.2×	34.73	34.13
		2	9.2×	**35.06**	**34.27**
	+ Greedy	1	14.2×	34.69	33.72
		2	12.2×	35.03	33.94

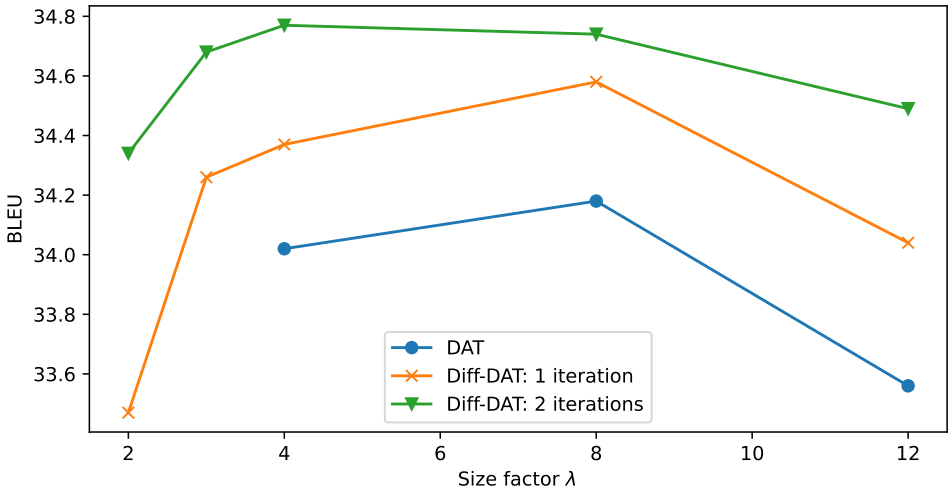

Figure 3: Effects of λ on IWSLT14 De-En. The graph size of DAT is λ times the of source length. We use Join-Viterbi decoding to evaluate BLEU.

E Ablation Study on the Impact of Size Factor λ

Figure 3 illustrates the results of DAT and Diff-DAT with varying graph sizes. Previous work (DAT) indicates that larger graph sizes complicate transition predictions, leading to a performance decline for both DAT and Diff-DAT. Nonetheless, Diff-DAT consistently outperforms DAT across all graph sizes. The graph size also influences the iterative decoding performance of Diff-DAT. Both transition predictions and iterative refinement contribute to capturing token dependencies in the generated sentence. When the graph size is small, iterative decoding significantly aids due to the limitations in transition predictions.

F Analysis of the impact of varying the number of decoding steps

We conducted another analysis to examine the impact of graph size on iterative decoding. Figures 4, 5, 6, and 7 display the BLEU scores for each dataset, categorized by reference length. A consistent trend was observed that in the results of each dataset there are two intervals: in the first interval, the BLEU score increases as the number of decoding iterations increases; in the second interval, the BLEU score decreases as the number of decoding iterations increases. Since sequence length directly influences graph size, it can be concluded that once the graph size reaches a certain threshold, it negatively affects iterative decoding. Larger graph sizes make it more challenging to predict transition probabilities. Additionally, each decoding iteration generates sub-paths in the graph that the model must navigate, further complicating the prediction of transition probabilities. These factors combined make it increasingly difficult for the model to predict transitions with each iteration. Moreover, errors from previous iterations affect subsequent ones, leading to a negative impact on the model's ability to capture dependencies within the sentence through transition predictions. This intriguing challenge inspires us to address it in future work.

G Inference

In inference, we adopt the translation of DAG with *Greedy*, *Lookahead* (Huang et al., 2022b), and *Joint-Viterbi* (Shao et al., 2022) decoding.

Greedy The most straightforward approach involves selecting the most probable transitions and tokens. In essence, we conduct simultaneous argmax operations to determine the most probable transition and token for every vertex. Subsequently, we construct the translation by gathering the predicted tokens along the selected path. This greedy decoding method is remarkably efficient, requiring only two parallel operations, as illustrated in Algorithm 1.

Lookahead Lookahead decoding enhances the efficacy of the greedy approach by jointly considering both transitions and tokens. Specifically, we rearrange $P_\theta(Y, A|X)$ into the following sequential decision problem:

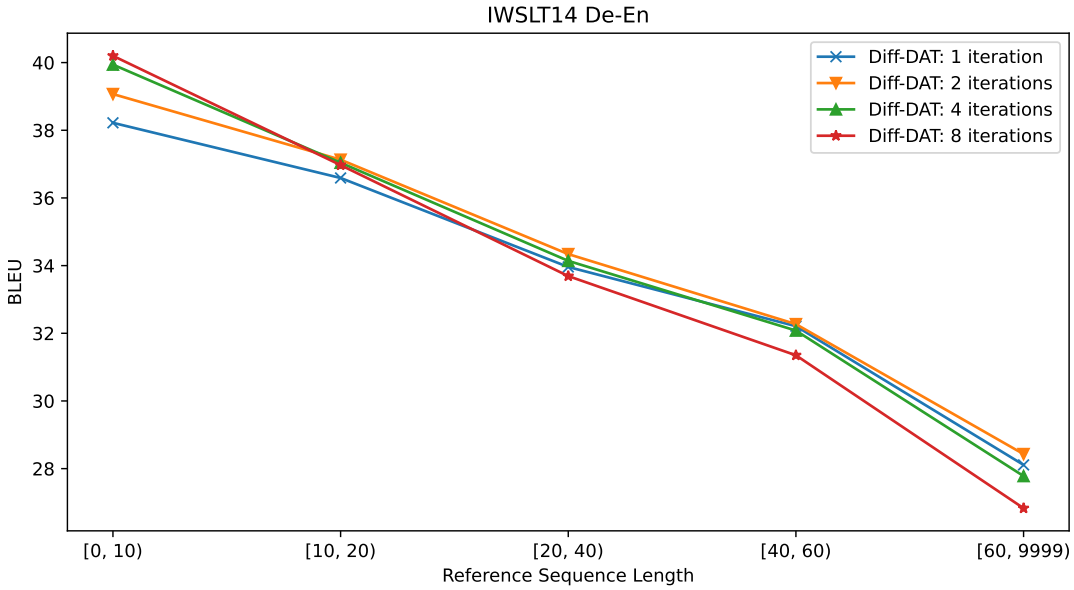

Figure 4: The BLEU score on IWSLT14 De-En bucketed by the reference length.

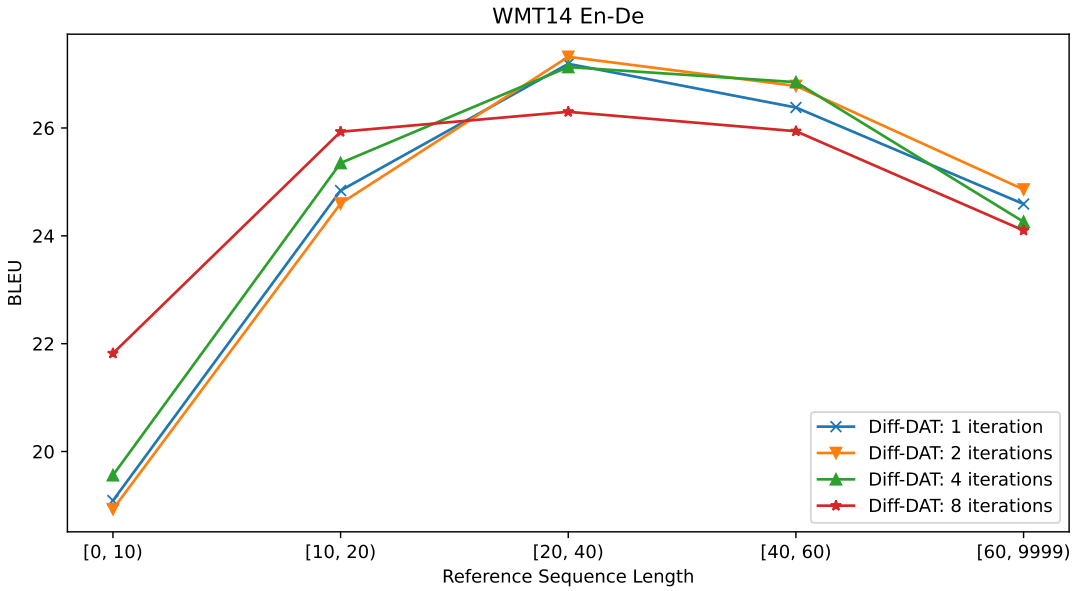

Figure 5: The BLEU score on WMT14 En-De bucketed by the reference length.

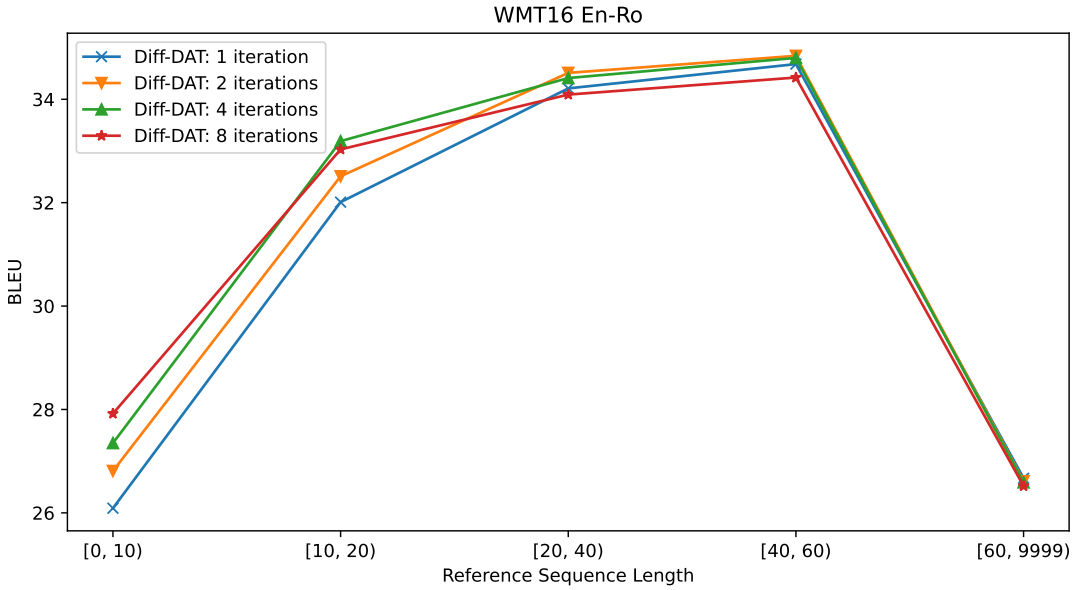

Figure 6: The BLEU score on WMT16 En-Ro bucketed by the reference length.

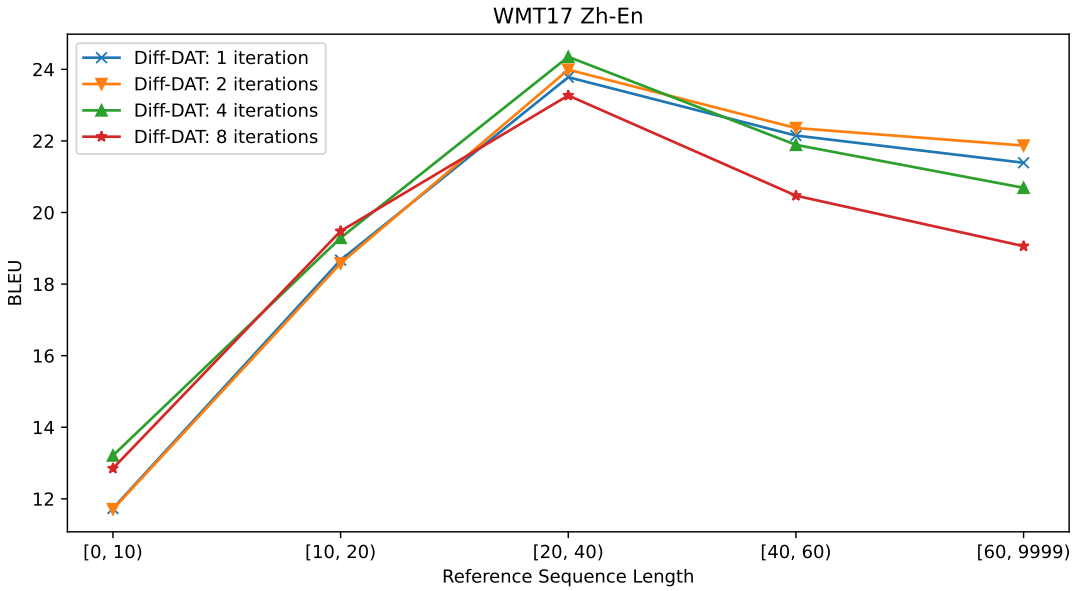

Figure 7: The BLEU score on WMT17 Zh-En bucketed by the reference length.

Algorithm 1 Greedy / Lookahead Decoding in Pytorch-like Parallel Pseudocode

Input: Graph Size L, Transition Matrix $\mathbf{E} \in \mathbb{R}^{L \times L}$,
if Using Lookahead **then**
  $\mathbf{E} := \mathbf{E} \otimes [\mathbf{P}.\text{MAX(dim=1).UNSQUEEZE(dim=0)}]$
  # $\mathbf{E}$ now jointly considers $\mathbf{P}$ and $\mathbf{E}$
  # $\otimes$ is element-wise multiplication
end if
tokens := $\mathbf{P}$.ARGMAX(dim=1) # shape: (L)
edges := $\mathbf{E}$.ARGMAX(dim=1) # shape: (L)
$i := 1$, *output* := [*tokens*[1]]
repeat
  $i :=$ *edges*[i] # jumping along the transition
  output.APPEND(*tokens*[i])
until $i = L$

$$P_\theta(y_1|a_1, X) \prod_{i=2}^{M} P_\theta(a_i|a_{i-1}, X) P_\theta(y_i|a_i, X), \tag{16}$$

This formulation transforms the task into choosing a_i and y_i sequentially. We concurrently derive:

$$y_i^*, a_i^* = \arg\max_{y_i, a_i} P_\theta(y_i|a_i, X) P_\theta(a_i|a_{i-1}, X), \tag{17}$$

This can still be executed in parallel with minimal overhead, as outlined in Algorithm 1.

Joint-Viterbi We additionally utilize Joint-Viterbi decoding (Shao et al., 2022) to determine the global joint optimum of translation and path within a predefined length constraint. Subsequently, we reevaluate these candidates through length normalization to refine their ranking.

It is notable that Joint-Viterbi decoding can be seen as enhancements to Greedy decoding and Lookahead decoding, respectively. While both Greedy and Lookahead decoding methods focus on the immediate probability and select the next token using $\arg\max_{a_i} P_\theta(a_i|X, a_{i-1})$ and $\arg\max_{y_i, a_i} P_\theta(y_i|a_i, X) P_\theta(a_i|a_{i-1}, X)$, respectively, Joint-Viterbi algorithm consider the entire decoding path. They ensure the discovery of the globally optimal solution $\arg\max_A P_\theta(A|X)$ and $\arg\max_{A,Y} P_\theta(A, Y|X)$, respectively.