Non-autoregressive Machine Translation with Disentangled Context Transformer

Jungo Kasai [1] **James Cross** [2] **Marjan Ghazvininejad** [2] **Jiatao Gu** [2]

Abstract

State-of-the-art neural machine translation models generate a translation from left to right and every step is conditioned on the previously generated tokens. The sequential nature of this generation process causes fundamental latency in inference since we cannot generate multiple tokens in each sentence in parallel. We propose an attention-masking based model, called ***Dis**entangled **Co**ntext* (DisCo) transformer, that simultaneously generates all tokens given different contexts. The DisCo transformer is trained to predict every output token given an arbitrary subset of the other reference tokens. We also develop the *parallel easy-first* inference algorithm, which iteratively refines every token in parallel and reduces the number of required iterations. Our extensive experiments on 7 translation directions with varying data sizes demonstrate that our model achieves competitive, if not better, performance compared to the state of the art in non-autoregressive machine translation while significantly reducing decoding time on average. Our code is available at `https://github.com/facebookresearch/DisCo`.

1. Introduction

State-of-the-art neural machine translation systems use *autoregressive* decoding where words are predicted one-by-one conditioned on all previous words (Bahdanau et al., 2015; Vaswani et al., 2017). *Non-autoregressive* machine translation (NAT, Gu et al., 2018), on the other hand, generates all words in one shot and speeds up decoding at the expense of performance drop. Parallel decoding results in conditional independence and prevents the model from properly capturing the highly multimodal distribution of target translations (Gu et al., 2018). One way to remedy this fundamental problem is to refine model output iteratively (Lee et al., 2018; Ghazvininejad et al., 2019). This work pursues this iterative approach to non-autoregressive translation.[1]

In this work, we propose a transformer-based architecture with attention masking, which we call ***Dis**entangled **Co**ntext* (DisCo) transformer, and use it for non-autoregressive decoding. Specifically, our DisCo transformer predicts every word in a sentence conditioned on an arbitrary subset of the rest of the words. Unlike the masked language models (Devlin et al., 2019; Ghazvininejad et al., 2019) where the model only predicts the masked words, the DisCo transformer can predict all words simultaneously, leading to faster inference as well as a substantial performance gain when training data are relatively large.

We also introduce a new inference algorithm for iterative parallel decoding, *parallel easy-first*, where each word is predicted by attending to the words that the model is more confident about. This decoding algorithm allows for predicting all tokens with different contexts in each iteration and terminates when the output prediction converges, contrasting with the constant number of iterations (Ghazvininejad et al., 2019). Indeed, we will show in a later section that this method substantially reduces the number of required iterations without loss in performance.

Our extensive empirical evaluations on 7 translation directions from standard WMT benchmarks show that our approach achieves competitive performance to state-of-the-art non-autoregressive and autoregressive machine translation while significantly reducing decoding time on average.

2. DisCo Transformer

We introduce our DisCo transformer for non-autoregressive translation (Fig. 1). We propose a DisCo objective as an efficient alternative to masked language modeling and design an architecture that computes the objective in a single pass.

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington. Work done at Facebook AI. [2]Facebook AI. Correspondence to: Jungo Kasai <jkasai@cs.washington.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Refinement requires several sequential steps, but we abuse the term *non-autoregressive* generation to mean a broad family of methods that generate the target in parallel for simplicity.

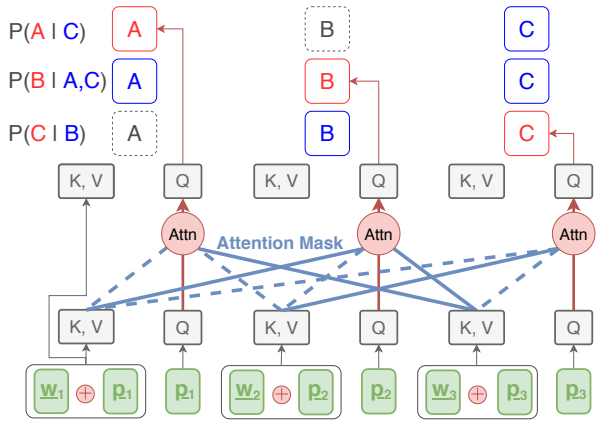

Figure 1. DisCo Transformer. W and p denote word and positional embeddings respectively. We simulate three *disentangled* contexts to predict A, B, and C (Y_n) given {C}, {A, C}, and {B} ($Y^n_{\texttt{obs}}$) respectively. **Dashed lines** indicate masked-out attention connections to $Y^n_{\texttt{mask}}$ and Y_n itself. K and V are direct projections of $w_n + p_n$ (thus contextless) for all layers to avoid leakage.

2.1. DisCo Objective

Similar to masked language models (Devlin et al., 2019), a conditional masked language model (CMLM, Ghazvininejad et al., 2019) predicts randomly masked target tokens $Y_{\texttt{mask}}$ given a source text X and the rest of the target tokens $Y_{\texttt{obs}}$. Namely, for every sentence pair in bitext X and Y,

$$P(Y_{\texttt{mask}}|X, Y_{\texttt{obs}}) = \texttt{Transformer}(X, Y_{\texttt{obs}})$$
$$Y_{\texttt{mask}} \sim \texttt{RS}(Y) \quad Y_{\texttt{obs}} = Y \setminus Y_{\texttt{mask}}$$

where RS denotes random sampling of masked tokens.[2] CMLMs have proven successful in parallel decoding for machine translation (Ghazvininejad et al., 2019), video captioning (Yang et al., 2019a), and speech recognition (Nakayama et al., 2019). However, the fundamental inefficiency with this masked language modeling objective is that the model can only be trained to predict a subset of the reference tokens ($Y_{\texttt{mask}}$) for each network pass unlike a normal autoregressive model where we predict all Y from left to right. To address this limitation, we propose a ***Dis**entangled **Co**ntext* (DisCo) objective.[3] The objective involves prediction of every token given an arbitrary (thus *disentangled*) subset of the other tokens. For every $1 \leq n \leq N$ where $|Y| = N$, we predict:

$$P(Y_n|X, Y^n_{\texttt{obs}}) = \texttt{Transformer}(X, Y^n_{\texttt{obs}})$$
$$Y^n_{\texttt{obs}} \sim \texttt{RS}(Y \setminus Y_n)$$

[2]BERT (Devlin et al., 2019) masks a token with probability 0.15 while CMLMs (Ghazvininejad et al., 2019) sample the number of masked tokens uniformly from $[1, N]$.

[3]We distinguish this from disentangled representation.

2.2. DisCo Transformer Architecture

Simply computing conditional probabilities $P(Y_n|X, Y^n_{\texttt{obs}})$ with a vanilla transformer decoder will necessitate N separate transformer passes for each $Y^n_{\texttt{obs}}$. We introduce the DisCo transformer to compute these N *contexts* in one shot:

$$P(Y_1|X, Y^1_{\texttt{obs}}), \cdots, P(Y_N|X, Y^N_{\texttt{obs}}) = \texttt{DisCo}(X, Y)$$

In particular, our DisCo transformer makes crucial use of attention masking to achieve this computational efficiency. Denote input word and positional embeddings at position n by w_n and p_n. For each position n in Y, the vanilla transformer computes self-attention:[4]

$$k_n, v_n, q_n = \texttt{Proj}(w_n + p_n)$$
$$h_n = \texttt{Attention}(K, V, q_n)$$
$$K, V = \texttt{Concat}\left(\{k_m\}_{m=1}^N\right), \texttt{Concat}\left(\{v_m\}_{m=1}^N\right)$$

We modify this attention computation in two aspects. First, we separate query input from key and value input to avoid feeding the token we predict. Then we only attend to keys and values that correspond to observed tokens ($K^n_{\texttt{obs}}, V^n_{\texttt{obs}}$) and *mask out* the connection to the other tokens ($Y^n_{\texttt{mask}}$ and Y_n itself, **dashed lines** in Fig. 1).

$$k_n, v_n = \texttt{Proj}(w_n + p_n) \quad q_n = \texttt{Proj}(p_n)$$
$$h_n = \texttt{Attention}(K^n_{\texttt{obs}}, V^n_{\texttt{obs}}, q_n)$$
$$K^n_{\texttt{obs}} = \texttt{Concat}\left(\{k_m | Y_m \in Y^n_{\texttt{obs}}\}\right)$$
$$V^n_{\texttt{obs}} = \texttt{Concat}\left(\{v_m | Y_m \in Y^n_{\texttt{obs}}\}\right)$$

2.3. Stacked DisCo Transformer

Unfortunately stacking DisCo transformer layers is not straightforward. Suppose that we compute the nth position in the jth layer from the prevous layer's output as follows:

$$k^j_n, v^j_n = \texttt{Proj}(w_n + h^{j-1}_n) \quad q^j_n = \texttt{Proj}(h^{j-1}_n)$$
$$h^j_n = \texttt{Attention}(K^{n,j}_{\texttt{obs}}, V^{n,j}_{\texttt{obs}}, q^j_n)$$

In this case, however, any cyclic relation between positions will cause information leakage. Concretely, assume that $Y = [A, B]$ and $N = 2$. Suppose also that $Y^1_{obs} = B$ and $Y^2_{obs} = A$, and thus there is a cycle that position 1 can see B and position 2 can see A. Then the output state at position 1 in the first layer h^1_1 becomes a function of B:

$$h^1_1(B) = \texttt{Attention}(k^1_2(B), v^1_2(B), q^1_1)$$

Since position 2 can see position 1, the output state at position 2 in the second layer h^2_2 is computed by

$$h^2_2 = \texttt{Attention}\left(k^2_1(h^1_1(B)), v^2_1(h^1_1(B)), q^2_2\right)$$

[4]For simplicity, here we omit fully-connected layers, layernorm, residual connections, and cross attention to the encoder.

But h_2^2 will be used to predict the token at position 2 i.e., B, and this will clearly make the prediction problem degenerate. To avoid this cyclic leakage, we make keys and values independent of the previous layer's output h_n^{j-1}:

$$k_n^j, v_n^j = \texttt{Proj}(w_n + p_n) \quad q_n^j = \texttt{Proj}(h_n^{j-1})$$
$$h_n^j = \texttt{Attention}(K_{\texttt{obs}}^{n,j}, V_{\texttt{obs}}^{n,j}, q_n^j)$$

In other words, we *decontextualize* keys and values in stacked DisCo layers.

2.4. Training Loss

We use a standard transformer as an encoder and stacked DisCo layers as a decoder. For each Y_n in Y where $|Y| = N$, we uniformly sample the number of visible tokens from $[0, N-1]$, and then we randomly choose that number of tokens from $Y \setminus Y_n$ as $Y_{\texttt{obs}}^n$, similarly to CMLMs (Ghazvininejad et al., 2019). We optimize the negative log likelihood loss from $P(Y_n|X, Y_{\texttt{obs}}^n)$ $(1 \leq n \leq N)$. Again following CMLMs, we append a special token to the encoder and project the vector to predict the target length for parallel decoding. We add the negative log likelihood loss from this length prediction to the loss from word predictions.

2.5. DisCo Objective as Generalization

We designed the DisCo transformer to compute conditional probabilities at every position efficiently, but here we note that the DisCo transformer can be readily used with other training schemes in the literature. We can train an autoregressive DisCo transformer by always setting $Y_{\texttt{obs}}^n = Y_{<n}$. XLNet (Yang et al., 2019b) is also a related variant of a transformer that was introduced to produce general-purpose contextual word representations. The DisCo transformer differs from XLNet in two critical ways. First, XLNet consists of separate context stream and query stream attention. This means that we need to double the amount of expensive attention and fully connected layer computation in the transformer. Another difference is that XLNet is only trained to predict tokens in permuted order. The DisCo transformer can be trained for the permutation objective by setting $Y_{\texttt{obs}}^n = \{Y_i | z(i) < z(n)\}$ where $z(i)$ indicates the rank of the ith element in the new order. Baevski et al. (2019) train their two tower model with the *cloze objective* again for general-purpose pretraining. We can train our DisCo transformer with this objective by setting $Y_{\texttt{obs}}^n = \{Y_{\neq n}\}$. The proposed DisCo objective provides generalization that encompasses all of these special cases.

3. Inference Algorithms

In this section, we discuss inference algorithms for our DisCo transformer. We first review *mask-predict* from prior work as a baseline and introduce a new parallelizable inference algorithm, *parallel easy-first* (Alg. 1).

3.1. Mask-Predict

Mask-predict is an iterative inference algorithm introduced in Ghazvininejad et al. (2019) to decode a conditional masked language model (CMLM). The target length N is first predicted, and then the algorithm iterates over two steps: *mask* where i_t tokens with lowest probability are masked and *predict* where those masked tokens are updated given the other $N - i_t$ tokens. The number of masked tokens i_t decays from N with a constant rate over a fixed number of iterations T. Specifically, at iteration t,

$$i_t = \left\lfloor N \cdot \frac{T-t+1}{T} \right\rfloor$$
$$Y_{\texttt{obs}}^t = \{Y_j^{t-1} | j \in \underset{n}{\text{topk}}(p_n^{t-1}, k = N - i_t)\}$$
$$Y_n^t, p_n^t = \begin{cases} (\arg)\max_w P(Y_n = w | X, Y_{\texttt{obs}}^t) \text{ if } Y_n^t \notin Y_{\texttt{obs}}^t \\ Y_n^{t-1}, p_n^{t-1} \qquad \text{otherwise} \end{cases}$$

This method is directly applicable to our DisCo transformer by fixing $Y_{obs}^{n,t}$ regardless of the position n.

3.2. Parallel Easy-First

An advantage of the DisCo transformer over a CMLM is that we can predict tokens in all positions conditioned on different context simultaneously. The mask-predict inference can only update masked tokens given the fixed observed tokens $Y_{\texttt{obs}}^t$, meaning that we are wasting the opportunity to improve upon $Y_{\texttt{obs}}^t$ and to take advantage of broader context present in $Y_{\texttt{mask}}^t$. We develop an algorithm, *parallel easy-first* (Alg. 1), which makes predictions in all positions, thereby benefiting from this property. Concretely, in the first iteration, we predict all tokens in parallel given source text:

$$Y_n^1, p_n = (\arg)\max_w P(Y_n = w | X)$$

Then, we get the *easy-first* order z where $z(i)$ denotes the rank of p_i in descending order. At iteration $t > 1$, we update predictions for all positions by

$$Y_{\texttt{obs}}^{n,t} = \left\{Y_i^{t-1} | z(i) < z(n)\right\}$$
$$Y_n^t, p_n^t = (\arg)\max_w P\left(Y_n = w | X, Y_{\texttt{obs}}^{n,t}\right)$$

Namely, we update each position given previous predictions on the *easier* positions. In a later section, we will explore several variants of choosing $Y_{\texttt{obs}}^{n,t}$ and show that this easy-first strategy performs best despite its simplicity.

3.3. Length Beam

Following Ghazvininejad et al. (2019), we apply length beam. In particular, we predict top K lengths from the distri-

Algorithm 1 Parallel Easy-First with Length Beam

Source sentence: X
Predicted lengths: $N_1, \cdots, N_K$
Max number of iterations: T
for $k \in \{1, 2, ..., K\}$ **do**
  for $n \in \{1, 2, ..., N_k\}$ **do**
    $Y_n^{1,k}, p_n^k = (\arg)\max_w P(y_n = w|X)$
  end for
  Get the easy-first order z_k by sorting p^k and let $z_k(i)$ be the rank of the ith position.
end for
for $t \in \{2, ..., T\}$ **do**
  for $k \in \{1, ..., K\}$ **do**
    for $n \in \{1, 2, ..., N_k\}$ **do**

$$Y_{\text{obs}}^{n,t} = \{Y_i^{t-1,k} | z_k(i) < z_k(n)\}$$
$$Y_n^{t,k}, p_n^{t,k} = (\arg)\max_w P\left(Y_n = w | X, Y_{\text{obs}}^{n,t}\right)$$

    end for
  end for

$$k^* = \operatorname*{argmax}_k \sum_{n=1}^{N_k} \log\left(p_n^{t,k}\right) / N_k$$

  if $Y^{t-1,k^*} = Y^{t,k^*}$ **then return** Y^{t,k^*}
end for
return Y^{T,k^*}

bution in length prediction and run parallel easy-first simultaneously. In order to speed up decoding, we terminate if the one with the highest average log score $\sum_{n=1}^{N} \log(p_n^t)/N$ converges. It should be noted that for parallel easy-first, $Y^t = Y^{t-1}$ means convergence because $Y_{obs}^{n,t} = Y_{obs}^{n,t+1}$ for all positions n while mask-predict may keep updating tokens even after because Y_{obs}^t changes over iterations. See Alg. 1 for full pseudo-code. Notice that all for-loops are parallelizable except the one over iterations t. In the subsequent experiments, we use length beam size of 5 (Ghazvininejad et al., 2019) unless otherwise noted. In Sec. 5.2, we will illustrate that length beam facilitates decoding both the CMLM and DisCo transformer.

4. Experiments

We conduct extensive experiments on standard machine translation benchmarks. We demonstrate that our DisCo transformer with the parallel easy-first inference achieves comparable performance to, if not better than, prior work on non-autoregressive machine translation with substantial reduction in the number of sequential steps of transformer computation. We also find that our DisCo transformer achieves more pronounced improvement when bitext training data are large, getting close to the performance of autoregressive models.

4.1. Experimental Setup

Benchmark datasets We evaluate on 7 directions from four standard datasets with various training data sizes: WMT14 EN-DE (4.5M pairs), WMT16 EN-RO (610K pairs), WMT17 EN-ZH (20M pairs), and WMT14 EN-FR (36M pairs, en→fr only). These datasets are all encoded into subword units by BPE (Sennrich et al., 2016).[5] We use the same preprocessed data and train/dev/test splits as prior work for fair comparisons (EN-DE: Vaswani et al., 2017; EN-RO: Lee et al., 2018; EN-ZH: Hassan et al., 2018; Wu et al., 2019; EN-FR: Gehring et al., 2017; Ott et al., 2018). We evaluate performance with BLEU scores (Papineni et al., 2002) for all directions except that we use SacreBLEU (Post, 2018)[6] in en→zh again for fair comparison with prior work (Ghazvininejad et al., 2019). For all autoregressive models, we apply beam search with $b = 5$ (Vaswani et al., 2017; Ott et al., 2018) and tune length penalty of $\alpha \in [0.0, 0.2, \cdots, 2.0]$ in validation. For parallel easy-first, we set the max number of iterations $T = 10$ and use $T = 4, 10$ for constant-time mask-predict.

4.2. Baselines and Comparison

There has been a flurry of recent work on non-autoregressive machine translation (NAT) that finds a balance between parallelism and performance. Performance can be measured using automatic evaluation such as BLEU scores (Papineni et al., 2002). Latency is, however, challenging to compare across different methods. For models that have an autoregressive component (e.g., Kaiser et al., 2018; Ran et al., 2019), we can speed up sequential computation by caching states. Further, many of prior NAT approaches generate varying numbers of translation candidates and rescore them using an autoregressive model. The rescoring process typically costs overhead of one parallel pass of a transformer encoder followed by a decoder. Given this complexity in latency comparison, we highlight two state-of-the-art iteration-based NAT models whose latency is comparable to our DisCo transformer due to the similar model structure. See Sec. 6 for descriptions of more work on NAT.

CMLM As discussed earlier, we can generate a translation with mask-predict from a CMLM (Ghazvininejad et al., 2019). We can directly compare our DisCo transformer with this method by the number of iterations required.[7] We

[5] We run joint BPE on all language pairs except EN-ZH.

[6] SacreBLEU hash: BLEU+case.mixed+lang.en-zh+numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.3.7.

[7] Caching *contextless* key and value computation in the DisCo transformer gives us a slight speedup, but it is relatively minor as compared to expensive attention and fully connected computation.

Table 1. The performance of non-autoregressive machine translation methods on the WMT14 EN-DE and WMT16 EN-RO test data. The Step columns indicate the average number of sequential transformer passes. Shaded results use a small transformer ($d_{model} = d_{hidden} = 512$). Our EN-DE results show the scores after conventional compound splitting (Luong et al., 2015; Vaswani et al., 2017).

Model	**en→de**		**de→en**		**en→ro**		**ro→en**	
n: # rescored candidates	Step	BLEU	Step	BLEU	Step	BLEU	Step	BLEU
Gu et al. (2018) ($n = 100$)	1	19.17	1	23.20	1	29.79	1	31.44
Wang et al. (2019) ($n = 9$)	1	24.61	1	28.90	–	–	–	–
Li et al. (2019) ($n = 9$)	1	25.20	1	28.80	–	–	–	–
Ma et al. (2019) ($n = 30$)	1	25.31	1	30.68	1	32.35	1	32.91
Sun et al. (2019) ($n = 19$)	1	26.80	1	30.04	–	–	–	–
Ran et al. (2019)	1	26.51	1	31.13	1	31.70	1	31.99
Shu et al. (2020) ($n = 50$)	1	25.1	–	–	–	–	–	–
Iterative NAT Models								
Lee et al. (2018)	10	21.61	10	25.48	10	29.32	10	30.19
Ghazvininejad et al. (2019) (CMLM)	4	25.94	4	29.90	4	32.53	4	33.23
	10	27.03	10	30.53	10	33.08	10	33.31
Gu et al. (2019b) (LevT)	7+	27.27	–	–	–	–	7+	33.26
Our Implementations								
CMLM + Mask-Predict	4	26.73	4	30.75	4	33.02	4	33.27
CMLM + Mask-Predict	10	**27.39**	10	31.24	10	**33.33**	10	**33.67**
DisCo + Mask-Predict	4	25.83	4	30.15	4	32.22	4	32.92
DisCo + Mask-Predict	10	27.06	10	30.89	10	32.92	10	33.12
DisCo + Easy-First	4.82	27.34	4.23	**31.31**	3.29	33.22	3.10	33.25
AT Models								
Vaswani et al. (2017) (base)	N	27.3	–	–	–	–	–	–
Vaswani et al. (2017) (large)	N	28.4	–	–	–	–	–	–
Our Implementations								
AT Transformer Base (EN-RO teacher)	N	27.38	N	31.78	N	34.16	N	34.46
AT Transformer Base + Distillation	N	28.24	N	31.54	–	–	–	–
AT Transformer Large (EN-DE teacher)	N	28.60	N	31.71	–	–	–	–

provide results obtained by running their code.[8]

Levenshtein Transformer Levenshtein transformer (LevT) is a transformer-based iterative model for parallel sequence generation (Gu et al., 2019b). Its iteration consists of three sequential steps: *deletion*, *placeholder prediction*, and *token prediction*. Unlike the CMLM with the constant-time mask-predict inference, decoding in LevT terminates adaptively under certain condition. Its latency is roughly comparable by the average number of sequential transformer runs. Each iteration consists of three transformer runs except that the first iteration skips the *deletion* step. See Gu et al. (2019b) for detail.

Hyperparameters We generally follow the hyperparameters for a transformer base (Vaswani et al., 2017; Ghazvininejad et al., 2019): 6 layers for both the encoder and decoder, 8 attention heads, 512 model dimensions, and 2048 hidden dimensions. We sample weights from $\mathcal{N}(0, 0.02)$, initialize biases to zero, and set layer normalization parameters to $\beta = 0, \gamma = 1$ (Devlin et al., 2019). For regularization, we tune the dropout rate from $[0.1, 0.2, 0.3]$ based on dev performance in each direction, and apply weight decay with 0.01 and label smoothing with $\varepsilon = 0.1$. We train batches of approximately 128K tokens using Adam (Kingma & Ba, 2015) with $\beta = (0.9, 0.999)$ and $\varepsilon = 10^{-6}$. The learning rate warms up to $5 \cdot 10^{-4}$ in the first 10K steps, and then decays with the inverse square-root schedule. We train all models for 300K steps apart from en→fr where we make 500K steps to account for the data size. We measure the dev BLEU score at the end of each epoch to avoid stochasticity, and average the 5 best checkpoints to obtain the final model. We use 16 Telsa V100 GPUs and accelerate training by mixed precision floating point (Micikevicius et al., 2018), and implement all models with `fairseq` (Ott et al., 2019).

Distillation Similar to previous work on non-autoregressive translation (e.g., Gu et al., 2018; Lee et al., 2018), we apply sequence-level knowledge distillation (Kim & Rush, 2016) by training every model

[8] https://github.com/facebookresearch/Mask-Predict

in all directions on translations produced by a standard left-to-right transformer model (transformer large for EN-DE, EN-ZH, and EN-FR and base for EN-RO). We also present results obtained from training a standard autoregressive base transformer on the same distillation data for comparison. We assess the impact of distillation in Sec. 5.1 and demonstrate that distillation is still a key component in our non-autoregressive models.

4.3. Results and Discussion

Seen in Table 1 are the results in the four directions from the WMT14 EN-DE and WMT16 EN-RO datasets. First, our re-implementations of CMLM + Mask-Predict outperform Ghazvininejad et al. (2019) (e.g., 31.24 vs. 30.53 in de→en with 10 steps). This is probably due to our tuning on the dropout rate and weight averaging of the 5 best epochs based on the validation BLEU performance (Sec. 4.1).

Our DisCo transformer with the parallel easy-first inference achieves at least comparable performance to the CMLM with 10 steps despite the significantly fewer steps on average (e.g., 4.82 steps in en→de). The one exception is ro→en (33.25 vs. 33.67), but DisCo + Easy-First requires only 3.10 steps, and CMLM + Mask-Predict with 4 steps achieves similar performance of 33.27. The limited advantage of our DisCo transformer on the EN-RO dataset suggests that we benefit less from the training efficiency of the DisCo transformer on the small dataset (610K sentence pairs). DisCo + Mask-Predict generally underperforms DisCo + Easy-First, implying that the mask-predict inference, which fixes Y^n_{obs} across all positions n, fails to utilize the flexibility of the DisCo transformer. DisCo + Easy-First also accomplishes significant reduction in the average number of steps as compared to the adaptive decoding in LevT (Gu et al., 2019b) while performing competitively. As discussed earlier, each iteration in inference on LevT involves three sequential transformer runs, which undermine the latency improvement.

Overall, our implementations compare well with other NAT models from prior work. We achieve competitive performance to the standard autoregressive models with the same transformer base configuration on the EN-DE dataset except that the autoregressive model with distillation performs comparably to the transformer large teacher in en→de (28.24 vs. 28.60). Nonetheless, we still see a large gap between the autoregressive teachers and our NAT results in both directions from EN-RO, illustrating a limitation of our remedy for the trade-off between decoding parallelism and performance.

Tables 2 and 3 show results from the EN-ZH and EN-FR datasets where the bitext data are larger (20M and 36M sentence pairs). In both cases we see similar (yet more pronounced) patterns to the EN-DE and EN-RO experiments. Particularly noteworthy is that DisCo with the parallel easy-first inference and dropout tuning yields 34.63 points, a gain of 1.4 BLEU improvement over Ghazvininejad et al. (2019) in en→zh despite the average of 5.44 steps.

Table 2. WMT17 EN-ZH test results.

Model	**en→zh**		**zh→en**	
	Step	BLEU	Step	BLEU
Ghazvininejad et al. (2019)	4	32.63	4	21.90
	10	33.19	10	23.21
Our Implementations				
CMLM + Mask-Predict	4	33.58	4	22.57
CMLM + Mask-Predict	10	34.24	10	23.76
DisCo + Mask-Predict	4	33.61	4	22.42
DisCo + Mask-Predict	10	34.51	10	23.68
DisCo + Easy-First	5.44	**34.63**	5.90	**23.83**
AT Transformer Base	N	34.74	N	23.77
+ Distillation	N	35.09	N	24.53
AT Trans. Large (teacher)	N	35.01	N	24.65

Table 3. WMT14 EN-FR test results.

Model	**en→fr**		Train
	Step	BLEU	Time
CMLM + Mask-Predict	4	40.21	53 h
CMLM + Mask-Predict	10	40.55	
DisCo + Mask-Predict	4	39.59	
DisCo + Mask-Predict	10	40.27	37 h
DisCo + Easy-First	4.29	40.66	
Vaswani et al. (2017) (base)	N	38.1	–
Vaswani et al. (2017) (large)	N	41.8	–
Ott et al. (2018) (teacher)	N	43.2	–
AT Transformer Base	N	41.27	28 h
+ Distillation	N	42.03	28 h

4.4. Decoding Speed

We saw the the DisCo transformer with the parallel easy-first inference achieves competitive performance to the CMLM while reducing the number of iterations. Here we compare them in terms of the wall-time speedup with respect to the standard autoregressive model of the same base configuration (Fig. 2). For each decoding run, we feed one sentence at a time and measure the wall time from when the model is loaded until the last sentence is translated, following the setting in Gu et al. (2019b). All models are implemented in `fairseq` (Ott et al., 2019) and run on a single Nvidia V100 GPU. We can confirm that the average number of iterations directly translates to decoding time; the average number of iterations of the DisCo transformer with $T = 10$ was 5.44 and the measured speedup lies between $T = 5, 6$ of the CMLM. Note that `fairseq` implements effcient

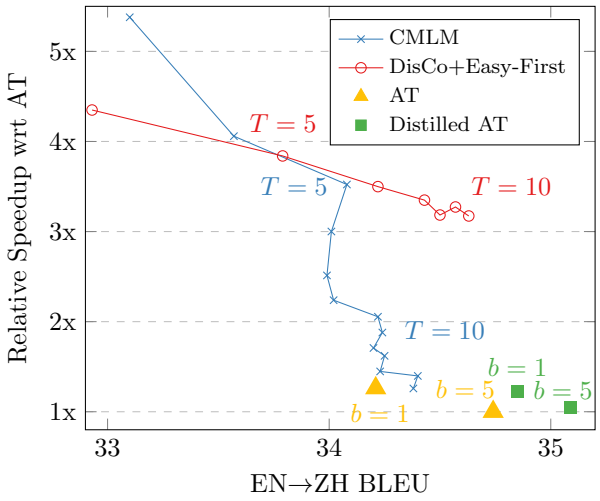

Figure 2. Relative decoding speedup on the en→zh test data with respect to the standard autoregressive model (indicated as ▲). T and b denote the (max) number of iterations and beam size respectively. The length beam size is all set to 5.

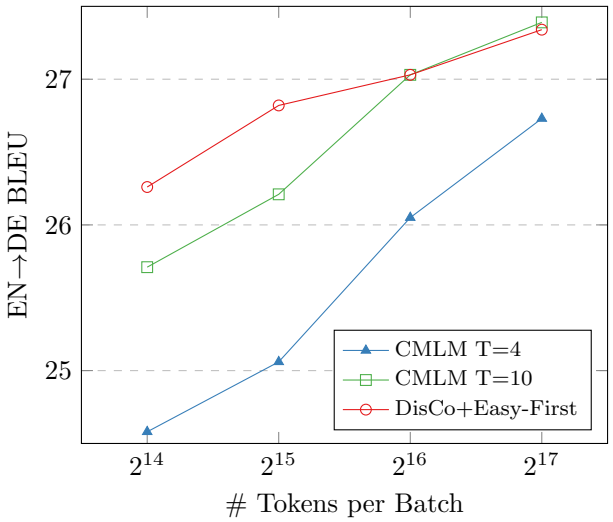

Figure 3. EN→DE test results with varying batch size.

decoding of autoregressive models by caching hidden states. The average length of generated sentences in the autoregressive model was 25.16 (4.6x steps compared to 5.44 steps), but we only gained a threefold speedup from DisCo.

5. Analysis and Ablations

In this section, we give an extensive analysis on our approach along training and inference dimensions.

5.1. Training

Training Efficiency In Sec. 2.1, we discussed the fundamental inefficiency of CMLM training—a CMLM model is trained to only predict a subset of the target words. DisCo addresses this problem by its architecture that allows for predicting every word given a randomly chosen subset of the target words. Seen in Fig. 3 are results on the en→de test data with varying batch sizes. We can see that DisCo is more robust to smaller batch sizes, supporting our claim that it provides more efficient training.

Distillation We assess the effects of knowledge distillation across different models and inference configurations (Table 4). Consistent with previous models (Gu et al., 2018; Zhou et al., 2020), we find that distillation facilitates all of the non-autoregressive models. Moreover, the DisCo transformer benefits more from distillation compared to the CMLM under the same mask-predict inference. This is in line with Zhou et al. (2020) who showed that there is correlation between the model capacity and distillation data complexity. The DisCo transformer uses contextless keys and values, resulting in reduced capacity. Autoregressive translation also improves with distillation from a large transformer, but the difference is relatively small. Finally, we can observe that the gain from distillation decreases as we incorporate more global information in inference (more iterations in NAT cases and larger beam size in AT cases).

Table 4. Effects of distillation across different models and inference. All results are BLEU scores from the dev data. T and b denote the max number of iterations and beam size respectively.

		en→de			**ro→en**		
Model	T	raw	dist.	Δ	raw	dist.	Δ
CMLM + MaskP	4	22.7	25.5	2.8	33.2	34.8	1.6
CMLM + MaskP	10	24.5	25.9	1.4	34.5	34.9	0.4
DisCo + MaskP	4	21.4	24.6	3.2	32.3	34.1	1.8
DisCo + MaskP	10	23.6	25.3	1.7	33.4	34.3	0.9
DisCo + EasyF	10	23.9	25.6	1.7	34.0	35.0	1.0
AT Base ($b = 1$)	N	25.5	26.4	0.9	–	–	–
AT Base ($b = 5$)	N	26.1	26.8	0.7	–	–	–

AT with Contextless KVs We saw that a decoder with contextless keys and values can still retain performance in non-autoregressive models. Here we use a decoder with contextless keys and values in autoregressive models. The results (Table 5) show that it is able to retain performance even in autoregressive models regardless of distillation, suggesting further potential of our approach.

Table 5. Test results (BLEU) from AT with contextless keys and values.

AT	**en→de**		**de→en**		**ro→en**
Decoder	raw	dist.	raw	dist.	raw
Contextless	27.09	27.86	30.91	31.46	34.25
Original	26.85	27.69	31.33	31.09	34.46

Easy-First Training So far we have trained our models to predict every word given a random subset of the other words. But this training scheme yields a gap between training and inference, which might harm the model. We attempt to make training closer to inference by training the DisCo transformer in the easy-first order. Similarly to the inference, we first predict the easy-first order by estimating $P(Y_n|X)$ for all n. Then, use that order to determine $Y^n_{\texttt{obs}}$.[9] The overall loss will be the sum of the negative loglikelihood of these two steps. Seen in Table 6 are the results on the dev sets of en→de and ro→en. In both directions, this easy-first training does not ameliorate performance, suggesting that randomness helps the model. Notice also that the average number of iterations in inference decreases (4.03 vs. 4.29, 2.94 vs. 3.17). The model gets trapped in a sub-optimal solution with reduced iterations due to lack of exploration.

Table 6. Dev results from bringing training closer to inference.

	en→de		ro→en	
Training Variant	Step	BLEU	Step	BLEU
Random Sampling	4.29	**25.60**	3.17	**34.97**
Easy-First Training	4.03	24.76	2.94	34.96

5.2. Inference

Alternative Inference Algorithms Here we compare various decoding strategies on the DisCo transformer (Table 7). Recall in the parallel easy-first inference (Sec. 3.2), we find the easy-first order by sorting the probabilities in the first iteration and compute each position's probability conditioned on the *easier* positions from the previous iteration. We evaluate two alternative orderings: left-to-right and right-to-left. We see that both of them yield much degraded performance. We also attempt to use even broader context than parallel easy-first by computing the probability at each position based on all other positions (*all-but-itself*, $Y^{n,t}_{obs} = Y^{t-1}_{\neq n}$). We again see degraded performance, suggesting that cyclic dependency (e.g., $Y^{t-1}_m \in Y^{n,t}_{\texttt{obs}}$ and $Y^{t-1}_n \in Y^{m,t}_{\texttt{obs}}$) breaks consistency. For example, a model can have two output candidates: "Hong Kong" and "New York" (Zhang et al., 2020). In this case, we might end up producing "Hong York" due to this cyclic dependency. These results suggest that the easy-first ordering we introduced is a simple yet effective approach.

[9] This training process can be seen as the hard EM algorithm where the easy-first order is a latent variable.

Table 7. Dev results with different decoding strategies.

	en→de		ro→en	
Inference Strategy	Step	BLEU	Step	BLEU
Left-to-Right Order	6.80	21.25	4.86	33.87
Right-to-Left Order	6.79	20.75	4.67	34.38
All-But-Itself	6.90	20.72	4.80	33.35
Parallel Easy-First	4.29	**25.60**	3.17	**34.97**
Mask-Predict	10	25.34	10	34.54

Example Translation Seen in Fig. 4 is a translation example in de→en when decoding the same DisCo transformer with the mask-predict or parallel easy-first inference. In both algorithms, iterative refinement resolves structural inconsistency, such as repetition. Parallel easy-first succeeds in incorporating more context in early stages whereas mask-predict continues to produce inconsistent predictions ("my my activities") until more context is available later, resulting in one additional iteration to land on a consistent output.

Length Beam Fig. 5 shows performance of the CMLM and DisCo transformer with varying size of length beam. All cases benefit from multiple candidates with different lengths to a certain point, but DisCo + Easy-First improves most. This can be because parallel easy-first relies on the easy-first order as well as the length, and length beam provides opportunity to try multiple orderings.

Iterations vs. Length We saw that parallel easy-first inference substantially reduced the number of required iterations. We hypothesize that the algorithm effectively adapts the number of iterations based on the difficulty, which is reminiscent of a dynamic halting mechanism (Graves, 2016; Dehghani et al., 2019). To see this, we compare the number of required iterations and the generated sentence length as a proxy for difficulty (Fig. 6). Similarly to the experiments above, we set the max iteration and length beam to be 10 and 5 respectively. While the number of required iterations vary to a certain degree, we see that long sentences tend to require more iterations to converge.

6. Related and Future Work

In addition to the work discussed above, prior and concurrent work on non-autoregressive translation developed ways to mitigate the trade-off between decoding parallelism and performance. As in this work, several prior and concurrent work proposed methods to iteratively refine (or insert) output predictions (Mansimov et al., 2019; Stern et al., 2019; Gu et al., 2019a; Chan et al., 2019a;b; Ghazvininejad et al., 2020b; Li et al., 2020; Saharia et al., 2020). Other approaches include adding a lite autoregressive module to parallel decoding (Kaiser et al., 2018; Sun et al., 2019; Ran et al., 2019), partially decoding autoregressively (Stern et al., 2018; 2019), rescoring output candidates autoregressively (e.g., Gu et al., 2018), mimicking hidden states of an autoregressive teacher (Li et al., 2019), training with different objectives than vanilla negative log likelihood (Libovický & Helcl, 2018; Wang et al., 2019; Shao et al., 2020;

X: Als Präsident würde ich meine Unternehmungen jedoch ganz einstellen.
Y: And if I became president, then I would end my business activity.

t	Mask-Predict	t	Parallel Easy-First
1	As Pres. as Pres. , I would stop stop my doing altogether .	1	As Pres. as Pres. , I would stop stop my doing altogether .
2	However , as Pres. , I would stop my my doing altogether .	2	As , , Pres. , I would stop doing my activities altogether .
3	However , as Pres. , I would stop my my activities altogether .	3	However , as however , I would stop doing my business altogether .
4	However , as Pres. , I would stop my my activities altogether .	4	However , as Pres. , I would stop doing my business altogether
5	However , as Pres. , I would stop doing my activities altogether .		
1	As Pres. as Pres. , I would stop stop my doing altogether .	1	As Pres. as Pres. , I would stop stop my doing altogether .
2	However , as Pres. , I would stop my my doing altogether .	2	As , , Pres. , I would stop doing my activities altogether .
3	However , as Pres. , I would stop my my activities altogether .	1	As Pres. as Pres. , I would stop stop my doing altogether .
4	However , as Pres. , I would stop my my activities altogether .	2	As , , Pres. , I would stop doing my activities altogether .

Figure 4. An example of inference iterations in de→en from the dev set when max iteration T is 5. (*Pres.* stands for *President*). We show how each of the underscored words were generated in the bottom section. Prediction is conditioned on highlighted tokens.

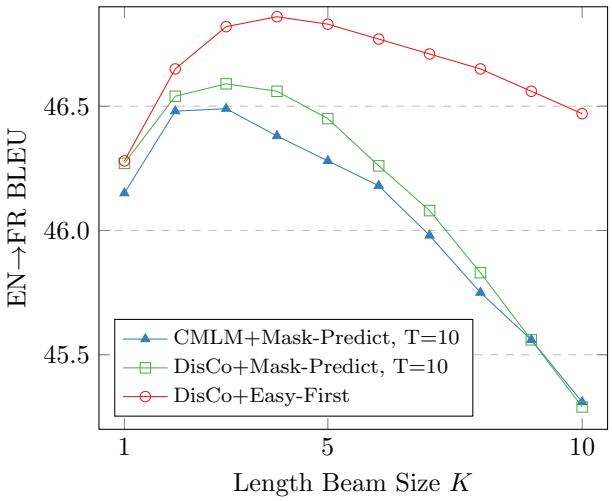

Figure 5. EN→FR dev results with varying length beam size.

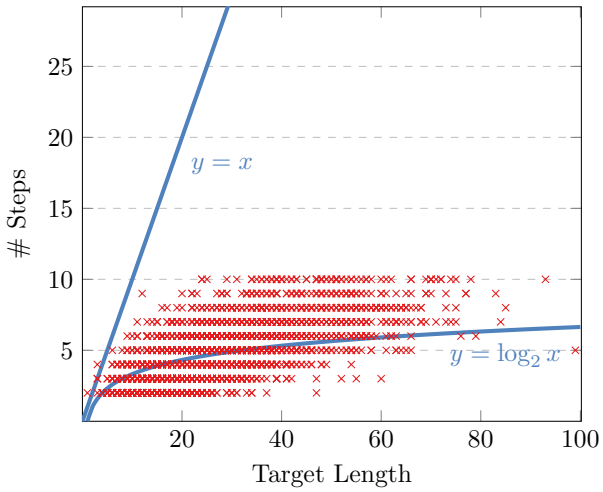

Figure 6. # Refinement steps vs. target length on the WMT14 en→de test data.

Ghazvininejad et al., 2020a; Li et al., 2020), reordering input sentences (Ran et al., 2019), generating with an energy-based inference network (Tu et al., 2020), training on additional data from an autoregressive model (Zhou & Keung, 2020), and modeling with latent variables (Ma et al., 2019; Shu et al., 2020).

While this work took iterative decoding methods, our DisCo transformer can be combined with other approaches for efficient training. For example, Li et al. (2019) trained two separate non-autoregressive and autoregressive models, but it is possible to train a single DisCo transformer with both autoregressive and random masking and use hidden states from autoregressive masking as a teacher. We leave integration of the DisCo transformer with more approaches to non-autoregressive translation for future.

We also note that our DisCo transformer can be used for general-purpose representation learning. In particular, Liu et al. (2019) found that masking different tokens in every epoch outperforms static masking in BERT (Devlin et al., 2019). Our DisCo transformer would allow for making a prediction at every position given arbitrary context, providing even more flexibility for large-scale pretraining.

7. Conclusion

We presented the DisCo transformer that predicts every word in a sentence conditioned on an arbitrary subset of the other words. We developed an inference algorithm that takes advantage of this efficiency and further speeds up generation without loss in translation quality. Our results provide further support for the claim that non-autoregressive translation is a fast viable alternative to autoregressive translation. Nonetheless, a discrepancy still remains between autoregressive and non-autoregressive performance when knowledge distillation from a large transformer is applied to both. We will explore ways to narrow this gap in the future.

Acknowledgements

We thank Tim Dettmers, Hao Peng, Mohammad Rasooli, William Chan, and Qinghong Han as well as the anonymous reviewers for their helpful feedback on this work.

References

Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L. S., and Auli, M. Cloze-driven pretraining of self-attention networks, 2019. URL https://arxiv.org/abs/1903.07785.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*, 2015. URL https://arxiv.org/abs/1409.0473.

Chan, W., Kitaev, N., Guu, K., Stern, M., and Uszkoreit, J. KERMIT: Generative insertion-based modeling for sequences, 2019a. URL https://arxiv.org/abs/1906.01604.

Chan, W., Stern, M., Kiros, J. R., and Uszkoreit, J. An empirical study of generation order for machine translation, 2019b. URL https://arxiv.org/abs/1910.13437.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, 2019. URL https://arxiv.org/abs/810.04805.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. Convolutional sequence to sequence learning. In *Proc. of ICML*, 2017. URL https://arxiv.org/abs/1705.03122.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. S. Mask-predict: Parallel decoding of conditional masked language models. In *Proc. of EMNLP*, 2019. URL https://arxiv.org/abs/1904.09324.

Ghazvininejad, M., Karpukhin, V., Zettlemoyer, L., and Levy, O. Aligned cross entropy for non-autoregressive machine translation, 2020a. URL https://arxiv.org/abs/2004.01655.

Ghazvininejad, M., Levy, O., and Zettlemoyer, L. Semi-autoregressive training improves mask-predict decoding, 2020b. URL https://arxiv.org/abs/2001.08785.

Graves, A. Adaptive computation time for recurrent neural networks, 2016. URL https://arxiv.org/abs/1603.08983.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. Non-autoregressive neural machine translation. In *Proc. of ICLR*, 2018. URL https://arxiv.org/abs/1711.02281.

Gu, J., Liu, Q., and Cho, K. Insertion-based decoding with automatically inferred generation order. *TACL*, 2019a. URL https://arxiv.org/abs/1902.01370.

Gu, J., Wang, C., and Zhao, J. Levenshtein transformer. In *Proc. of NeurIPS*, 2019b. URL https://arxiv.org/abs/1905.11006.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., Liu, S., Liu, T.-Y., Luo, R., Menezes, A., Qin, T., Seide, F., Tan, X., Tian, F., Wu, L., Wu, S., Xia, Y., Zhang, D., Zhang, Z., and Zhou, M. Achieving human parity on automatic Chinese to English news translation, 2018. URL https://arxiv.org/abs/1803.05567.

Kaiser, L., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. Fast decoding in sequence models using discrete latent variables. In *Proc. of ICML*, 2018. URL https://arxiv.org/abs/1803.03382.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In *Proc. of EMNLP*, 2016. URL https://arxiv.org/abs/1606.07947.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proc. of ICLR*, 2015. URL https://arxiv.org/abs/1412.6980.

Lee, J. D., Mansimov, E., and Cho, K. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proc. of EMNLP*, 2018. URL https://arxiv.org/abs/1802.06901.

Li, X., Meng, Y., Yuan, A., Wu, F., and Li, J. LAVA NAT: A non-autoregressive translation model with look-around decoding and vocabulary attention, 2020. URL https://arxiv.org/abs/2002.03084.

Li, Z., Lin, Z., He, D., Tian, F., Qin, T., Wang, L., and Liu, T.-Y. Hint-based training for non-autoregressive machine translation. In *Proc. of EMNLP*, 2019. URL https://arxiv.org/abs/1909.06708.

Libovický, J. and Helcl, J. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proc. of EMNLP*, 2018. URL https://arxiv.org/abs/1811.04719.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. S., and Stoyanov, V. RoBERTa: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*, September 2015. URL https://www.aclweb.org/anthology/D15-1166.

Ma, X., Zhou, C., Li, X., Neubig, G., and Hovy, E. H. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proc. of EMNLP*, 2019. URL https://arxiv.org/abs/1909.02480.

Mansimov, E., Wang, A., and Cho, K. A generalized framework of sequence generation with application to undirected sequence models, 2019. URL https://arxiv.org/abs/1905.12790.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., and Wu, H. Mixed precision training. In *Proc. of ICLR*, 2018. URL https://arxiv.org/abs/1710.03740.

Nakayama, S., Kano, T., Tjandra, A., Sakti, S., and Nakamura, S. Recognition and translation of code-switching speech utterances. In *Proc. of Oriental COCOSDA*, 2019. URL https://ahcweb01.naist.jp/papers/conference/2019/201910_OCOCOSDA_sahoko-n/201910_OCOCOSDA_sahoko-n.paper.pdf.

Ott, M., Edunov, S., Grangier, D., and Auli, M. Scaling neural machine translation. In *Proc. of WMT*, 2018. URL https://arxiv.org/abs/1806.00187.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL Demonstrations*, 2019. URL https://arxiv.org/abs/1904.01038.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, 2002. URL https://www.aclweb.org/anthology/P02-1040/.

Post, M. A call for clarity in reporting BLEU scores. In *Proc. of WMT*, 2018. URL https://www.aclweb.org/anthology/W18-6319.

Ran, Q., Lin, Y., Li, P., and Zhou, J. Guiding non-autoregressive neural machine translation decoding with reordering information, 2019. URL https://arxiv.org/abs/1911.02215.

Saharia, C., Chan, W., Saxena, S., and Norouzi, M. Non-autoregressive machine translation with latent alignments, 2020. URL https://arxiv.org/abs/2004.07437.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proc. of ACL*, 2016. URL https://www.aclweb.org/anthology/P16-1162.

Shao, C., Zhang, J., Feng, Y., Meng, F., and Zhou, J. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *Proc. of AAAI*, 2020. URL https://arxiv.org/abs/1911.09320.

Shu, R., Lee, J., Nakayama, H., and Cho, K. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proc. of AAAI*, 2020. URL https://arxiv.org/abs/1908.07181.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. In *Proc. of NeurIPS*, 2018. URL https://arxiv.org/abs/1811.03115.

Stern, M., Chan, W., Kiros, J. R., and Uszkoreit, J. Insertion transformer: Flexible sequence generation via insertion operations. In *Proc. of ICML*, 2019. URL https://arxiv.org/abs/1902.03249.

Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., and Deng, Z. Fast structured decoding for sequence models. In *Proc. of NeurIPS*, 2019. URL https://arxiv.org/abs/1910.11555.

Tu, L., Pang, R. Y., Wiseman, S., and Gimpel, K. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proc. of ACL*, 2020. URL https://arxiv.org/abs/2005.00850.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proc. of NeurIPS*, 2017. URL https://arxiv.org/abs/1706.03762.

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y. Non-autoregressive machine translation with auxiliary regularization. In *Proc. of AAAI*, 2019. URL https://arxiv.org/abs/1902.10245.

Wu, F., Fan, A., Baevski, A., Dauphin, Y., and Auli, M. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*, 2019. URL https://arxiv.org/abs/1901.10430.

Yang, B., Liu, F., and Zou, Y. Non-autoregressive video captioning with iterative refinement, 2019a. URL https://arxiv.org/abs/1911.12018.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *Proc. of NeurIPS*, 2019b. URL https://arxiv.org/abs/1906.08237.

Zhang, Y., Wang, G., Li, C., Gan, Z., Brockett, C., and Dolan, B. POINTER: Constrained text generation via insertion-based generative pre-training, 2020. URL https://arxiv.org/abs/2005.00558.

Zhou, C., Neubig, G., and Gu, J. Understanding knowledge distillation in non-autoregressive machine translation. In *Proc. of ICLR*, 2020. URL https://arxiv.org/abs/1911.02727.

Zhou, J. and Keung, P. Improving non-autoregressive neural machine translation with monolingual data. In *Proc. of ACL*, 2020. URL https://arxiv.org/abs/2005.00932.