Latent Variable Models and Iterative Refinement for Non-Autoregressive Neural Machine Translation

by

Jason Lee

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May 2021

Professor Kyunghyun Cho

© Jason Lee

All Rights Reserved, 2021

Dedication

To my wife and best friend Bo Jung.

Acknowledgements

When I started my Ph.D 6 years ago, I hardly knew that the next few years would be an incredibly exciting and intellectually challenging journey. A Ph.D teaches (and forces) you to be independent and offers you a unique opportunity to completely immerse yourself into a subject of your interest. But it is not something you do alone. Without the help, guidance and collaboration that I was fortunate to have from many people, this Ph.D would not have been possible.

My deepest gratitude goes to Kyunghyun Cho, who responded to my email asking to work with him in the summer of 2016, and took me as his student the following year. Throughout our works together over the next several years, he taught me not only machine learning but what it means to do good science by example — with rigor, vision and optimism — and helped me become a better researcher and also a human being. I am also indebted to my committee members He He, Andrew Wilson, Graham Neubig and Orhan Firat for providing me valuable feedback and guidance during different parts of my Ph.D.

I would like to thank both faculty members and students I interacted with at NYU and CILVR (in alphabetical order): Aishwarya Kamath, Alex Wang, Andrew Wilson, Cheolhyoung Lee, David Brandfonbrener, Dmitry Storcheus, Elman Mansimov, Ethan Perez, He He, Iacer Calixto, Ilia Kulikov, Ilya Kostrikov, Jake Zhao, Jason Phang, Jiakai Zhang, Jiwoong Daniel Im, Joan Bruna, Katharina Kann, Mikael Henaff, Min Jae Song, Nikita Nangia, Nishant Subramani, Phu Mon Htut, Roberta Raileanu, Rodrigo Nogueira, Sainbayar Sukhbaatar, Sean Welleck, Sebastien Jean, Stanislas Lauly, Stanislaw Jastrzebski, William Falcon, Xiang Zhang, and Yuanzhe (Richard) Pang. I am also grateful to Hong Tam and Santiago Pizzini for their excellent administrative support and to Shenglong Wang and Shirley Palma for managing our GPUs (and for occasional bump in priority during deadline week).

Before coming to NYU, I spent two years at the Data Analytics Lab at ETH Zürich. I am grateful for the connections I made from those years: Octavian Ganea, Piyush Bansal, Andreas Marfurt, Yannic Kilcher and Hadi Daneshmand. I am particularly indebted to Thomas Hofmann for being so supportive in my move to NYU and also over the years as we stayed in touch. I would also like to thank my external collaborators and friends Jiatao Gu and Raphael Shu, with whom I brainstormed and discussed ideas frequently.

During my Ph.D, I was fortunate to spend every summer interning at an industry research lab. I would like to thank Douwe Kiela, Kyunghyun Cho and Jason Weston for hosting me at Facebook for our two projects on multi-agent communication and grounding. I am grateful to my mentors at Google Research Orhan Firat and Dustin Tran, with whom I discussed project ideas and progress daily and learned a lot from during my year at Google. I am also grateful to Macduff Hughes, and the rest of the Translate and Brain team that I interacted with.

Outside of research, I would like to thank my friends Steven Ghoussain, David Yenicelik, Jungkeun Kim, Doo Re Song and Sung Hoon Choi for being a good sport and reminding me to have a life outside research every now and then. Last but not least, I want to extend my deepest gratitude to my sister and parents for their unconditional support, and to my wife Bo for believing in me every step of the way.

Abstract

Deep neural networks have fundamentally transformed the field of machine translation, and replaced statistical phrase-based approaches to serve translations to millions of users in production systems every day [Bahdanau et al., 2015; Fan et al., 2020; Hassan et al., 2018; Wu et al., 2016]. Despite impressive progress in translation accuracy, improving decoding speed remains a key challenge as most systems are *autoregressive* and generate a sentence word-by-word. As neural machine translation (NMT) models are becoming increasingly deep and complex, there is a growing need for more efficient translation systems with sub-linear or constant inference latency, with respect to the sentence length.

The main challenge in *non-autoregressive* machine translation is capturing the dependencies between tokens in a target sentence without autogression. Motivated by a rich history of probabilistic graphical models in sequence generation, this thesis proposes to use *latent variables* to model intra-sentence dependencies, such that the output distribution can be factorized given the latent variables. We also present several inference algorithms for non-autoregressive machine translation based on *iterative refinement*, which revises a sentence over multiple iterations. Our non-autoregressive models based on latent variables and iterative refinement can deliver significant decoding speedup with comparable translation accuracy relative to a strong autoregressive baseline [1]. Finally, we investigate the correlation between training (log-likelihood) and test objective (BLEU) of several model families. We observe the two metrics are not correlated when comparing models from different families (e.g. between autoregressive and latent variable models).

1. 6.2$\times$ decoding speedup with 0.9 BLEU score difference on WMT'14 En$\rightarrow$De [Lee et al., 2020a].

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

Most perceptual data arise in a sequence of individual elements. Indeed, there often exists a temporal ordering or internal structure to many data sources — e.g. text, speech, music, image and video — that enables the whole to be organized as a sequence of parts. Humans excel at understanding and making sense of these objects in terms of their structural and temporal relationship, that we perceive and generate these objects in an effortless manner. Building an intelligent machine that can perform human-level perception and generation of sequential data, however, has remained elusive for decades.

Following the success in training deep neural networks in the 2000s [Hinton et al., 2006; Salakhutdinov and Hinton, 2009] and the breakthrough in image recognition in 2012 [Krizhevsky et al., 2012], deep neural networks fundamentally transformed sequence generation across many domains and perceptual modalities. Fueled by large amounts of training data and fast computation, deep neural networks brought impressive improvements on generation tasks and made statistical models obsolete. These include but are not limited to: image generation [Chen et al., 2018b; Menick and Kalchbrenner, 2019; van den Oord et al., 2016b,c], speech recognition [Chan et al., 2016; Graves and Jaitly, 2014; Graves et al., 2013; Hinton et al., 2012], speech synthesis [Kalchbrenner et al., 2018; van den Oord et al., 2016a; Wang et al., 2017], machine translation [Bahdanau et al., 2015; Cho et al., 2014b; Firat et al., 2016; Sutskever et al., 2014] and music generation [Dhariwal et al., 2020; Engel et al., 2017; Huang et al., 2019a].

Many successful models highlighted above are *autoregressive* and generate a sequence of data one element at a time, mostly in a left-to-right order. While autoregressive models can be straightforwardly trained on exact data log-likelihood, sampling from them is inherently sequential and cannot be parallelized, despite wide availability of parallel hardware accelerators. As their inference latency grows linearly with respect to the length of the generated sequence, this limits their practical applicability to generating long sequences such as images (1M samples for images of 1024×1024 resolution), speech (16,000 samples per second for 16kHz audio) and text documents (an average article in Wikipedia contains 320 words).

As neural sequence models become increasingly larger and more complex, the problem of slow inference intensifies. To improve the generation throughput of autoregressive neural sequence models, several works proposed speed optimizations. In speech synthesis, Kalchbrenner et al. [2018] achieved real-time generation of 24kHz audio without loss of perceptual quality by replacing the deep convolutional WaveNet decoder [van den Oord et al., 2016a] with a single-layer recurrent neural network (RNN). In machine translation, Kim et al. [2019] also saw speech gains by using a light-weight RNN decoder, among other optimizations such as half-precision floating-point and knowledge distillation [Hinton et al., 2015]. While these solutions are practically useful, they do not improve the asymptotic decoding complexity of sequence generation.

Motivated by recent success in non-autoregressive neural machine translation [Gu et al., 2018a] and speech synthesis [van den Oord et al., 2018], this thesis proposes several training and inference algorithms for non-autoregressive neural machine translation. The main challenges in non-autoregressive machine translation are twofold: (1) capturing the dependencies between tokens in a target sentence without directly doing so using autoregression, and (2) given a source sentence at test time, finding the most likely target sentence in a constant or sub-linear number of steps with respect to the length of the sentence. The former is a problem of learning and the latter is that of inference. Motivated by a long and rich history of graphical models for sequential data, we explore using latent variables to capture the dependencies between the target tokens, such that the distribution of the target tokens can be factorized given the latent variables. We use iterative refinement for efficient inference, where the model first generates a rough output which is then successively refined until some convergence criterion is satisfied.

This thesis is organized as follows. Chapter 2 reviews autoregressive models and probabilistic graphical models for sequence generation. Then, we present a non-autoregressive model for machine translation that performs refinement in a discrete space (Chapter 3). Next, we present a continuous latent variable model and a novel inference algorithm that performs refinement in a hybrid space consisting of both discrete and continuous variables (Chapter 4). In Chapter 5, we present an inference algorithm where the refinement is purely performed in a continuous space. In Chapter 6, we investigate the correlation between the training objective (log-likelihood) and evaluation metric (BLEU) between several density estimators on five machine translation tasks. Finally, we present our conclusion and future avenues for research in Chapter 7.

1.1 List of Contributions

The Chapters 3, 4, 5 and 6 of this thesis appeared in the following publications respectively:

- **Jason Lee***, Elman Mansimov* and Kyunghyun Cho.
  Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. Empirical Methods in Natural Language Processing (EMNLP), 2018.
  Project source code and pretrained models can be found at `https://github.com/nyu-dl/dl4mt-nonauto`.

- Raphael Shu, **Jason Lee**, Hideki Nakayama and Kyunghyun Cho.
  Latent-Variable Non-Autoregressive Neural Machine Translation with Deterministic Inference using a Delta Posterior. AAAI Conference on Artificial Intelligence (AAAI), 2020.
  Project source code can be found at `https://github.com/zomux/lanmt`.

- **Jason Lee**, Raphael Shu and Kyunghyun Cho.
  Iterative Refinement in the Continuous Space for Non-Autoregressive Neural Machine Translation. Empirical Methods in Natural Language Processing (EMNLP), 2020.
  Project source code can be found at `https://github.com/zomux/lanmt-ebm`.

- **Jason Lee**, Dustin Tran, Orhan Firat and Kyunghyun Cho.

  On the Discrepancy between Density Estimation and Sequence Generation. Empirical Methods in Natural Language Processing (EMNLP), Workshop on Structured Prediction for NLP, 2020.

  Project source code can be found at

  `https://github.com/tensorflow/tensor2tensor`.

Chapter 2

Background

We provide a brief history of neural machine translation (NMT) and its slow inference speed that motivates this thesis (section 2.1). We then describe autoregressive sequence models (section 2.2) and review prior work on speed optimizations for autoregressive NMT systems (section 2.3). Probabilistic graphical models have a long and successful history of modeling sequential data (section 2.4). Indeed, several prior works on non-autoregressive neural sequence models that inspired this thesis (which we review in section 2.5) obviate the need for autoregression by employing latent variables to capture the structure underlying the data. Finally, we outline relevant work on generating text using iterative refinement, another key element in this thesis (section 2.6).

2.1 Brief History of Neural Machine Translation

For decades, Statistical Machine Translation (SMT) has been the dominant paradigm for machine translation [Brown et al., 1990, 1988, 1993]. One of the most successful statistical methods was phrase-based SMT models that first construct a phrase table from a corpus of sentence pairs, which is later used to generate translations by finding the most likely sequence of phrase pairs scored by an external language model [Koehn et al., 2003; Och and Ney, 2002, 2004; Zens et al., 2002].

Even before the advent of fully neural machine translation models, several work proposed to use neural networks as components in an SMT system, e.g. by using their conditional probabilities as a feature in an SMT decoder [Cho et al., 2014b; Devlin et al., 2014]. Encouraged by these results, neural networks were then deployed to directly model the full conditional distribution of the target sentence given the source sentence in an end-to-end fashion [Cho et al., 2014a; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014]. Thanks to increasing availability of parallel computation and advances in model parameterization [e.g. attention, Bahdanau et al., 2015; Vaswani et al., 2017], NMT systems have achieved the new state-of-the-art in machine translation [Wu et al., 2016] and even outperformed human-level translation quality in some language pairs [Hassan et al., 2018].

Despite their remarkable success, NMT systems come with much higher computational overhead compared to their SMT counterparts, as computing the conditional distribution for each target token involves several layers of computations in a deep neural network. Junczys-Dowmunt et al. [2016] showed that decoding with a recurrent NMT system [Nematus, Sennrich et al., 2017] is at least an order of magnitude slower than a state-of-the-art SMT system [Moses, Koehn et al., 2007] on a CPU with 16 threads. As slow inference speed adversely affects the applicability of NMT systems to real-life applications, researchers and practitioners seeking to deploy NMT systems proposed various speed optimizations (which we review in section 2.3). This dissertation specifically seeks to improve the asymptotic inference complexity of NMT systems by proposing training and inference algorithm for non-autoregressive machine translation, where the target sentence is not generated word-by-word, but in parallel.

2.2 Autoregressive Models for Sequences

Given a sequence of variables $\mathbf{y}_{1:t}$, autoregressive models directly factorize their joint distribution into a product of conditional distributions.

$$p(\mathbf{y}_{1:t}|\mathbf{x}) = \prod_{i=1}^{t} p(\mathbf{y}_i|\mathbf{y}_{<i}, \mathbf{x}), \tag{2.1}$$

where $\mathbf{x}$ is any contextual information the prediction might depend on (e.g. source sentence in machine translation, or possibly an empty set in the case of unconditional generation such as language modeling).

In fully visible sigmoid belief networks [Frey et al., 1995; Neal, 1992], the joint distribution of all the variables is similarly decomposed into a product of conditional distribution of each variable given its parent.

2.2.1 n-gram Models

One popular class of autoregressive models known as n-gram models assumes an n-th order Markov property, where the conditional distribution in Eq. 2.1 is truncated such that each prediction only depends on the $n - 1$ preceding variables. This assumption results in the likelihood in the form of:

$$\begin{aligned} p(\mathbf{y}_{1:t}) &= \prod_{i=1}^{t} p(\mathbf{y}_i | \mathbf{y}_{i-n+1:i-1}) \\ &= \prod_{i=1}^{t} \frac{C(\mathbf{y}_{i-n+1:i})}{C(\mathbf{y}_{i-n+1:i-1})}, \end{aligned} \tag{2.2}$$

where $C(\cdot)$ denotes the number of times a given n-gram occurs in the training data. Therefore, Eq 2.2 can be estimated by dividing the number of occurrences of every n-gram by that of the preceding $(n-1)$-gram in the training data. From this definition, an n-gram model assigns a 0 probability to an n-gram that is never seen during training. This is a major shortcoming due to the *curse of dimensionality*: the model can encounter novel n-grams at test time, even when a large training data was used. To combat this, various *smoothing* and *back-off* techniques were proposed. Smoothing methods allocate extra probability mass to unseen n-grams by discounting the probabilities of words encountered in the training data [Chen and Goodman, 1999; Jelinek and Mercer, 1980; Ney et al., 1994]. On the other hand, back-off models estimate the probability of an unseen n-gram by progressively considering shorter n-grams until such n-gram is found in the training data [Kneser and Ney, 1995]. Despite these empirical solutions, n-gram models suffer from a fundamental limitation as a word sequence on which the model will be tested is likely to be different from all the word sequences seen during training. Therefore, an approach that represents the meaning of a word by its co-occurrences is bound to not generalize well. While their simplicity and effectiveness made n-gram models a popular choice for language modeling and speech recognition tasks, this limitation motivated research into learning distributed representations of words with neural networks, such that semantically similar words will be closer in vector space than others [Bengio et al., 2003].

2.2.2 Neural autoregressive models

One of the first successful neural parameterization of an autoregressive sequence model is the neural probabilistic language model [Bengio et al., 2003]. While it is still an n-gram model, the model first embeds every $(n-1)$ preceding word to a distributed representation, which is fed into a 2-layer multi-layer perceptron (MLP) with a tanh nonlinearity. The final softmax layer produces a categorical distribution over the next word. This work is notable because (1) discrete symbols are embedded into a continuous representation, such that similar words are mapped to similar vector representations, and (2) vector representations (embeddings) are jointly trained with the MLP that produces the predictive distribution. Bengio et al. [2003] report significantly better perplexities when using neural networks and distributed word representations, compared to the best reported n-gram models. While this model achieves great generalization and avoids curse of dimensionality, it is still limited by the Markov assumption.

Recurrent neural language models [Mikolov et al., 2010, 2011] do not suffer from this as they condition the prediction of every variable on all preceding variables. The hidden state representation $\mathbf{h}_t$ of a recurrent neural network [RNN, Elman, 1990] at time t summarizes all the input variables up to that timestep $(\mathbf{x}_1, \ldots, \mathbf{x}_t)$. However, training recurrent neural networks with gradient descent has remained challenging for years. Bengio et al. [1994] discovered that the norm of the gradient (specifically its long term components) either tends to a very large quantity (also known as *exploding* gradients) or 0 (known as *vanishing* gradients) during training, making it impossible for the RNN to learn long range dependencies. Pascanu et al. [2013] demonstrated that gradient clipping, or rescaling the gradient norm whenever it exceeds a certain threshold, can mitigate exploding gradients. Introducing gated skip connections to the RNN effectively solved the vanishing gradient problem, as the output gate regulates the long term dependencies. Gated variants of recurrent neural networks, such as long short-term memory [LSTM, Hochreiter and Schmidhuber, 1997] and gated recurrent units [GRUs, Chung et al., 2014], could be successfully trained for language modeling and machine translation tasks.

While LSTMs and GRUs showed promising results on machine translation tasks [Cho et al., 2014a; Sutskever et al., 2014], these models were still outperformed by statistical phrase-based machine translation systems. Cho et al. [2014a] showed that translation accuracy drops significantly for longer sentences, and hypothesized that the main weakness of neural models stems from encoding the entire source sentence into a single fixed-size vector. To combat this, Bahdanau et al. [2015] proposed to take a weighted average of the source sentence hidden states across all timesteps, instead of simply taking the last hidden state. The multiplicative weights are produced by an *attention* module that computes the relevance of every target word to every source word. With the addition of the attention mechanism, recurrent NMT models outperformed the phrase-based machine translation models [Jean et al., 2015; Wu et al., 2016].

As recurrent neural networks perform an inherently sequential computation, where the amount of computation scales linearly with the length of the input data, training RNNs cannot be accelerated with parallel hardware accelerators such as GPUs. To overcome this, several works proposed non-recurrent parameterizations of autoregressive sequence models, e.g. using convolutions [Gehring et al., 2017; van den Oord et al., 2016a] and self-attention [Vaswani et al., 2017], augmented with causal masking such that the temporal dependencies are preserved. As these models are fully feed-forward, both forward and backward passes in backpropagation can be performed in parallel for all timesteps, unlike RNNs where these computations are sequential. With the increasing availability of parallel computation such as GPUs and tensor processing units [TPUs, Jouppi et al., 2017], it became possible to scale up the size of Transformer models [Vaswani et al., 2017] to billions of parameters [Brown et al., 2020; Chen et al., 2018a; Huang et al., 2019b; Lepikhin et al., 2020; Ott et al., 2018b; Shazeer et al., 2018], achieving impressive performance across several text generation tasks.

2.2.3 Inference

Given an autoregressive model $p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{x})$, performing maximum-a-posteriori (MAP) decoding is a discrete combinatorial search problem.

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}_1,\ldots,\mathbf{y}_t}\, p_\theta(\mathbf{y}_1, \ldots, \mathbf{y}_t|\mathbf{x}).$$

As the size of the search space grows exponentially with respect to the length t, an exact search is intractable. Therefore, approximate search algorithms are used instead.

Greedy search Greedy search recursively selects the most likely variable at each timestep, until the first <EOS> token.

$$\hat{\mathbf{y}}_t = \text{argmax}_{v \in V} \log p_\theta(\mathbf{y}_t = v|\hat{\mathbf{y}}_{<t}, \mathbf{x}),$$

where V is the target token vocabulary.

Beam search Beam search maintains a set of K partially-decoded sequences, called hypotheses. At each time step, beam search extends each hypothesis by one element by appending every token in the vocabulary, forming $K \cdot |V|$ candidate hypotheses. It then scores these and only keeps the highest K hypotheses. As the de-facto decoding algorithm for machine translation, it gives superior performance to greedy search in practice, albeit at significant computational overhead.

Inference complexity Due to the autoregressive dependencies, inference complexity scales linearly with respect to the length of the sequence being generated. For modeling long sequences such as documents or waveform, autoregressive models can be prohibitively slow to use in practice.

2.3 Fast Decoding for Neural Machine Translation

Due to the high inference complexity discussed above, several works proposed speed optimizations for autoregressive NMT systems. Devlin [2017] achieved equivalent decoding throughput to a phrase-based decoder on a CPU by using half-precision floating-point matrix multiplications. Zhang et al. [2018] employed cube pruning to coarsen the search space by clustering similar target hidden states, leading to less softmax operations on the large vocabulary. Hoang et al. [2018] and Iglesias et al. [2018] proposed to batch several input sentences and refine k-best extraction with specialized GPU kernel functions. Argueta and Chiang [2019] achieved considerable decoding speedup by accelerating sparse matrix operations (for both the input token embeddings and the pre-softmax output linear layer). Senellart et al. [2018] trained a smaller model with knowledge distillation [Hinton et al., 2015] for faster decoding speed.

Kim et al. [2019] introduced a C++ NMT system with a suite of speed optimizations [Marian, Junczys-Dowmunt et al., 2018], including pre-packed 8-bit matrix products, improved batched decoding, cache-friendly student architectures with parameter sharing and light-weight RNN-based decoder architectures, that significantly improved the Pareto curve of translation quality versus speed. Bogoychev et al. [2020] extensively used quantization for the weights and activations of the neural network to reduce model size and improve decoding speed.

2.4 Probabilistic Graphical Models for Sequences

2.4.1 Dynamic Bayesian Networks

Graphical models (or Bayesian networks) can provide a probabilistic framework for learning temporal dependencies in sequential or time-varying data by enforcing a causal structure based on temporal order. Indeed, many successful time series models such as hidden Markov models (HMMs) [Rabiner and Juang, 1986] and Kalman filters [Kalman, 1960] can be viewed as special cases of dynamic Bayesian networks. Given a sequence of observations $(\mathbf{y}_1, \ldots, \mathbf{y}_t)$, a dynamic Bayesian network posits that each observation $\mathbf{y}_i$ is dependent on a latent state $\mathbf{z}_i$, and that the sequence of states forms a Markov process.

$$p(\mathbf{y}_{1:t}, \mathbf{z}_{1:t}) = p(\mathbf{z}_1) p(\mathbf{y}_1 | \mathbf{z}_1) \prod_{i=2}^{t} p(\mathbf{y}_i | \mathbf{z}_i)\, p(\mathbf{z}_i | \mathbf{z}_{i-1}), \tag{2.3}$$

where $p(\mathbf{y}_i|\mathbf{z}_i)$ and $p(\mathbf{z}_i|\mathbf{z}_{i-1})$ are respectively referred to as observation and state transition probabilities.

Hidden Markov Models In HMMs, the hidden state is assumed to be a Categorical random variable that can take one of K distinct values. Therefore, both the initial $p(\mathbf{z}_1)$ and state transition probabilities $p(\mathbf{z}_i|\mathbf{z}_{i-1})$ are Categorical distributions. The joint probability of the observed and hidden variables can be exactly specified as Eq. 2.3.

Kalman filters In Kalman filters (or state-space models), the hidden state is assumed to be real-valued: $\mathbf{z}_i \in \mathbb{R}^K$. In linear-Gaussian state-space models (classical Kalman filters), the state transition and the observation probabilities are modeled as linear functions perturbed by Gaussian noise.

Inference and Learning Parameters of a dynamic Bayesian network can be estimated by maximizing the log-likelihood of the training data $\log p_\theta(\mathbf{y})$. Omitting the time-index suffixes for notational simplicity, we have the following lowerbound on the log-likelihood:

$$
\begin{aligned}
\log p_\theta(\mathbf{y}) &= \log \sum_z p_\theta(\mathbf{z}, \mathbf{y}) \\
&= \log \sum_z q(\mathbf{z}) \frac{p_\theta(\mathbf{z}, \mathbf{y})}{q(\mathbf{z})} && (2.4) \\
&= \log \mathbb{E}_{\mathbf{z}\sim q}\left[\frac{p_\theta(\mathbf{z}, \mathbf{y})}{q(\mathbf{z})}\right] \\
&\geq \mathbb{E}_{\mathbf{z}\sim q}\left[\log \frac{p_\theta(\mathbf{z}, \mathbf{y})}{q(\mathbf{z})}\right] && (2.5) \\
&= \mathbb{E}_{\mathbf{z}\sim q}\left[\log p_\theta(\mathbf{y}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q(\mathbf{z})\right] \\
&= \mathcal{F}(q, \theta),
\end{aligned}
$$

where Eq. 2.4 holds for any proposal distribution $q(\mathbf{z})$ over the hidden variables, and Eq. 2.5 follows from Jensen's inequality (and the concavity of the log function). When the hidden variable $\mathbf{z}_i$ is continuous, one can replace each summation with an integral.

The lowerbound can be maximized with the Expectation-Maximization (EM) algorithm [Baum et al., 1970; Dempster et al., 1977], which alternates between maximizing $\mathcal{F}$ with respect to $q(\mathbf{z})$ and θ, while holding the other fixed. It is an iterative algorithm that given an initial parameter θ_0 and the proposal distribution $q_0(\mathbf{z})$, alternates the following:

$$\textbf{E step: } q_{i+1} \leftarrow \text{argmax}_q \mathcal{F}(q, \theta_i)$$

$$\textbf{M step: } \theta_{i+1} \leftarrow \text{argmax}_\theta \mathcal{F}(q_{i+1}, \theta)$$

The EM algorithm in the context of HMMs is known as the Baum-Welch algorithm, whose E step uses the forward-backward procedure. The forward and backward probabilities (which respectively account for past and future timesteps) can be efficiently computed using dynamic programming, and are used to revise the posterior. Finding the most likely sequence in an HMM uses Viterbi algorithm, a similar dynamic programming approach where expectation is replaced by maximization [Viterbi, 1967].

While HMMs and Kalman filters enjoyed widespread adoption in a variety of disciplines, including speech recognition, filtering and navigation applications, they suffer from fundamental limitations. As the true state dynamics and observation distributions can be nonlinear for most realistic data, Kalman filters cannot model these time series accurately. While HMMs do not make these assumptions, they can easily overfit the data as the state space is large (K^T for T objects being modeled).

2.4.2 Variational Autoencoders

Variational Autoencoders (VAEs) [Kingma and Welling, 2014a; Rezende et al., 2014] are a family of graphical models where a shared non-linear function $q_\phi(\mathbf{z}|\mathbf{y})$ approximates the variational parameters from each datapoint (*local* latent variables), and the evidence lowerbound is maximized end-to-end. It is also referred to as amortized variational inference, as the cost of performing posterior inference is amortized across the entire dataset via the shared inference network.

$$\log p_\theta(\mathbf{y}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\cdot|\mathbf{y})}\big[\log p_\theta(\mathbf{y}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{y})\big] \tag{2.6}$$

VAEs have been one of the most successful classes of generative models across several domains.

2.4.3 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) [Graves et al., 2006] is an algorithm for labeling a sequential data with a sequence of discrete labels. Proposed in the context of speech recognition where the input and output sequence lengths differ by orders of magnitude, it can learn the alignment between input and output sequences with varying length. Given an input and output sequence $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{t'})$ and $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_t)$, it assumes there exists a monotonic alignment $a = (a_1, \ldots, a_{t'})$ between $\mathbf{x}$ and $\mathbf{y}$, where a_i indicates the element in $\mathbf{y}$ that $\mathbf{x}_i$ is aligned to. CTC makes a conditional independence assumption between individual predictions:

$$p_\theta(a|\mathbf{x}) = \prod_{i=1}^{T} p_\theta(a_i|\mathbf{x}) \tag{2.7}$$

Then, CTC is trained to maximize the likelihood of all possible alignments of **y** to **x**:

$$p_\theta(\mathbf{y}|\mathbf{x}) = \sum_{a \in \beta(\mathbf{y},\mathbf{x})} p_\theta(a|\mathbf{x}) \tag{2.8}$$

where $\beta(\mathbf{y}, \mathbf{x})$ marginalizes over all possible alignments between **y** and **x**. Due to the conditional independence assumption, Eq. 2.8 can be exactly evaluated via dynamic programming, and prediction for all positions can be produced in a single step in a non-autoregressive fashion.

2.4.4 Conditional Random Fields

Conditional Random Fields (CRFs) [Lafferty et al., 2001; Sutton and McCallum, 2012] are a class of discriminative model to segment and label sequential data that assumes first order Markov property and uses feature functions of observations and states (labels).

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\big(\sum_{i=1}^{n} s(\mathbf{y}_i, \mathbf{x}, i) + \sum_{i=2}^{n} t(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}, i)\big), \tag{2.9}$$

Similarly to HMMs, learning and inference in a linear-chain CRF can be done efficiently with the forward-backward algorithm with $\mathcal{O}(t \cdot n^2)$ complexity, where t is sequence length and n is the total number of labels.

2.5 Non-Autoregressive Neural Sequence Models

2.5.1 Definition

Given a sequence of variables $\mathbf{y}_{1:t}$, assuming each variable is independent of each other allows one to factorize the joint distribution into a product of individual distributions:

$$p(\mathbf{y}_{1:t}|\mathbf{x}) = \prod_{i=1}^{t} p(\mathbf{y}_i|\mathbf{x}). \tag{2.10}$$

However, this assumption is unrealistic for most natural data. For example, neighboring pixel values are highly correlated, and rules of grammar govern the possible combination of words. In order to capture the dependencies between individual variables, many non-autoregressive sequence models turn to latent variables, such that the decoding distribution may be factorized.

$$p(\mathbf{y}_{1:t}, \mathbf{z}|\mathbf{x}) = \left(\prod_{i=1}^{t} p(\mathbf{y}_i|\mathbf{z}, \mathbf{x}) \right) p(\mathbf{z}|\mathbf{x}). \tag{2.11}$$

2.5.2 Prior Work

Generative models for images have largely been non-autoregressive, from Restricted Boltzmann Machines [Hinton, 2007; Salakhutdinov and Hinton, 2009] and Deep Belief Networks [Hinton et al., 2006] trained to generate MNIST digits and small faces to the state-of-the-art Generative Adversarial Networks [Brock et al., 2019] and Variational Autoencoders [Child, 2020; Vahdat and Kautz, 2020] producing hyperrealistic images that are indistinguishable from real images.

Generative models for text that do not use autoregression, however, have relatively had little success to date. One of the main difficulties of non-autoregressive text generation is fully capturing the dependencies between words without autoregression. In phrase-based machine translation, Schwenk [2012] directly learned the conditional distribution of a target sentence given a source sentence, while dropping the dependence between the target words. Their success was however limited as the maximum sentence length was 7 words. In an investigation of neural GPU [Kaiser and Sutskever, 2016] for machine translation, Kaiser and Bengio [2016] observed that a fully non-autoregressive model yields very poor results.

The first successful non-autoregressive machine translation system was proposed by Gu et al. [2018a]. They introduced a sequence of discrete latent variables to summarize the alignment between each source and target word (known as *fertility*). However, they used an external word alignment tool [Dyer et al., 2013] to supervise the posterior distribution (in latent variable inference). Libovický and Helcl [2018] proposed a non-autoregressive NMT system based on the CTC framework [Graves et al., 2006], using dynamic programming to sum the log-likelihood of the output sequence over all possible combinations. While their models yielded worse translation accuracy than [Gu et al., 2018a], Saharia et al. [2020] later showed that using CTC loss with sequence-level distillation [Kim and Rush, 2016] leads to very strong results rivalling autoregressive baselines. Recently, Ghazvininejad et al. [2019] proposed an iterative refinement-based system trained on masked language modeling objective similar to BERT [Devlin et al., 2019]. By controlling the number of masked tokens in each refinement step, their model can generate translations in a constant number of steps.

Meanwhile, Kaiser et al. [2018] proposed a semi-autoregressive NMT system that learns a shorter sequence of discrete latent variables that summarizes the target sentence, and autoregressively predicts the latent sequence from the source sentence. Similar approach has been used for image generation [Reed et al., 2017] Sun et al. [2019] proposed a non-autoregressive NMT system with a light-weight CRF [Lafferty et al., 2001] before the softmax layer to introduce additional dependencies without incurring huge computational overhead.

This thesis was also inspired by work on non-autoregressive sequence model for efficient speech synthesis [van den Oord et al., 2018]. By distilling an autoregressive WaveNet [van den Oord et al., 2016a] teacher into a non-autoregressive inverse autoregressive flow [Kingma et al., 2016], they achieved 1000$\times$ speedup compared to the original WaveNet model, capable of serving in production text-to-speech systems.

2.6 Iterative Refinement for Sequence Generation

Refinement has a long history in text generation. Several works proposed to retrieve a (input, output) tuple from the training data and perform edit operations on the output [Gu et al., 2018c; Hashimoto et al., 2018; Song et al., 2016; Sumita and Iida, 1991; Weston et al., 2018]. The idea of refinement has also been applied in automatic post-editing [Grangier and Auli, 2017; Novak et al., 2016] and style transfer Li et al. [2018].

Energy-based models [LeCun et al., 2006] and gradient-based inference [Kingma and Welling, 2014b; Lee et al., 2019] can also be viewed as systems that generate by iterative refinement. These models can generate images [Bordes et al., 2017], generate structured outputs [Belanger et al., 2017] and segment images [Gygli et al., 2017].

Chapter 3

Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement

We propose a conditional non-autoregressive neural sequence model based on iterative refinement. The proposed model is designed based on the principles of latent variable models and denoising autoencoders, and is generally applicable to any sequence generation task. We extensively evaluate the proposed model on machine translation (IWSLT'16 De$\leftrightarrow$En, WMT'16 En$\leftrightarrow$Ro and WMT'14 En$\leftrightarrow$De) and image caption generation, and observe that it significantly speeds up decoding while maintaining the generation quality comparable to the autoregressive counterpart.

3.1 Introduction

Conditional neural sequence modeling has become a *de facto* standard in a variety of tasks [see, e.g., Cho et al., 2015, and references therein]. Much of this recent success is built on top of autoregressive sequence models in which the probability of a target sequence is factorized as a product of conditional probabilities of next symbols given all the preceding ones. Despite its success, neural autoregressive modeling has its weakness in decoding, i.e., finding the most likely sequence. Because of intractability, we must resort to suboptimal approximate decoding, and due to its sequential nature, decoding cannot be easily parallelized and results in a large latency [see, e.g., Cho, 2016]. This has motivated the recent investigation into non-autoregressive neural sequence modeling by Gu et al. [2018a] in the context of machine translation and van den Oord et al. [2018] in the context of speech synthesis.

In this chapter, we propose a non-autoregressive neural sequence model based on iterative refinement, which is generally applicable to any sequence generation task beyond machine translation. The proposed model can be viewed as both a latent variable model and a conditional denoising autoencoder. We thus propose a learning algorithm that is hybrid of lowerbound maximization and reconstruction error minimization. We further design an iterative inference strategy with an adaptive number of steps to minimize the generation latency without sacrificing the generation quality.

We extensively evaluate the proposed conditional non-autoregressive sequence model and compare it against the autoregressive counterpart, using the state-of-the-art Transformer [Vaswani et al., 2017], on machine translation and image caption generation. In the case of machine translation, the proposed deterministic non-autoregressive models are able to decode approximately $2-3\times$ faster than beam search from the autoregressive counterparts on both GPU and CPU, while maintaining 90-95% of translation quality on IWSLT'16 En$\leftrightarrow$De, WMT'16 En$\leftrightarrow$Ro and WMT'14 En$\leftrightarrow$De. On image caption generation, we observe approximately $3\times$ and $5\times$ faster decoding on GPU and CPU, respectively, while maintaining 85% of caption quality.

3.2 Non-Autoregressive Sequence Models

Sequence modeling in deep learning has largely focused on autoregressive modeling. That is, given a sequence $Y = (y_1, \ldots, y_T)$, we use some form of a neural network to parametrize the conditional distribution over each variable y_t given all the preceding variables, i.e.,

$$\log p(y_t|y_{<t}) = f_\theta(y_{<t}),$$

where f_θ is for instance a recurrent neural network. This approach has become a *de facto* standard in language modeling [Mikolov et al., 2010]. When this is conditioned on an extra variable X, it becomes a conditional sequence model $\log p(Y|X)$ which serves as a basis on which many recent advances in, e.g., machine translation [Bahdanau et al., 2015; Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014] and speech recognition [Chiu et al., 2017; Chorowski et al., 2015] have been made.

Despite the recent success, autoregressive sequence modeling has a weakness due to its nature of sequential processing. This weakness shows itself especially when we try to decode the most likely sequence from a trained model, i.e.,

$$\hat{Y} = \arg\max_Y \log p(Y|X).$$

There is no known polynomial algorithm for solving it exactly, and practitioners have relied on approximate decoding algorithms [see, e.g., Cho, 2016; Hoang et al., 2017a]. Among these, beam search has become the method of choice, due to its superior performance over greedy decoding, which however comes with a substantial computational overhead [Cho, 2016].

As a solution to this issue of slow decoding, two recent works have attempted non-autoregressive sequence modeling. Gu et al. [2018a] have modified the Transformer [Vaswani et al., 2017] for non-autoregressive machine translation, and van den Oord et al. [2018] a convolutional network [van den Oord et al., 2016a] for non-autoregressive modeling of waveform. Non-autoregressive modeling factorizes the distribution over a target sequence given a source into a product of conditionally independent per-step distributions:

$$p(Y|X) = \prod_{t=1}^{T} p(y_t|X),$$

breaking the dependency among the target variables across time. This allows us to trivially find the most likely target sequence by taking $\arg\max_{y_t} p(y_t|X)$ for each t, effectively bypassing the computational overhead and sub-optimality of decoding from an autoregressive sequence model.

This desirable property of exact and parallel decoding however comes at the expense of potential performance degradation [Kaiser and Bengio, 2016]. The potential modeling gap, which is the gap between the underlying, true model and the neural sequence model, could be larger with the non-autogressive model compared to the autoregressive one due to challenge of modeling the factorized conditional distribution above.

3.3 Iterative Refinement for Deterministic Non-Autoregressive Sequence Models

3.3.1 Latent variable model

Similarly to two recent works [Gu et al., 2018a; van den Oord et al., 2018], we introduce latent variables to implicitly capture the dependencies among target variables. We however remove any stochastic behavior by interpreting this latent variable model, introduced immediately below, as a process of iterative refinement.

Our goal is to capture the dependencies among target symbols given a source sentence without auto-regression by introducing L intermediate random variables and marginalizing them out:

$$p(Y|X) = \sum_{Y^0,\ldots,Y^L} \left(\prod_{t=1}^{T} p(y_t|Y^L, X)\right) \left(\prod_{t=1}^{T} p(y_t^L|Y^{L-1}, X)\right) \cdots \left(\prod_{t=1}^{T} p(y_t^0|X)\right). \tag{3.1}$$

Each product term inside the summation is modelled by a deep neural network that takes as input a source sentence and outputs the conditional distribution over the target vocabulary V for each t.

Deterministic Approximation The marginalization in Eq. (3.1) is intractable. In order to avoid this issue, we consider two approximation strategies; deterministic and stochastic approximation. Without loss of generality, let us consider the case of single intermediate latent variable, that is $L = 1$. In the *deterministic* case, we set $\hat{y}_t^0$ to the most likely value according to its distribution $p(y_t^0|X)$, that is $\hat{y}_t^0 = \arg\max_{y_t^0} p(y_t^0|X)$. The entire lower bound can then be written as:

$$\log p(Y|X) \geq \left(\sum_{t=1}^{T} \log p(y_t|\hat{Y}^L, X) \right) + \cdots \\ + \left(\sum_{t=1}^{T} \log p(y_t^1|\hat{Y}^0, X) \right) + \left(\sum_{t=1}^{T} \log p(\hat{y}_t^0|X) \right).$$

Stochastic Approximation In the case of *stochastic* approximation, we instead sample $\hat{y}_t^0$ from the distribution $p(y_t^0|X)$. This results in the unbiased estimate of the marginal log-probability $\log p(Y|X)$. Other than the difference in whether most likely values or samples are used, the remaining steps are identical.

Latent Variables Although the intermediate random variables could be anonymous, we constrain them to be of the same type as the output Y is, in order to share an underlying neural network. This constraint allows us to view each conditional $p(Y^l|\hat{Y}^{l-1}, X)$ as a single-step of refinement of a rough target sequence $\hat{Y}^{l-1}$. The entire chain of L conditionals is then the L-step iterative refinement. Furthermore, sharing the parameters across these refinement steps enables us to dynamically adapt the number of iterations per input X. This is important as it substantially reduces the amount of time required for decoding, as we see later in the experiments.

Training For each training pair (X, Y^*), we first approximate the marginal log-probability. We then minimize

$$J_{\text{LVM}}(\theta) = -\sum_{l=0}^{L+1} \left(\sum_{t=1}^{T} \log p_\theta(y_t^*|\hat{Y}^{l-1}, X) \right), \tag{3.2}$$

where $\hat{Y}^{l-1} = (\hat{y}_1^{l-1}, \ldots, \hat{y}_T^{l-1})$, and θ is a set of parameters. We initialize $\hat{y}_t^0$ (t-th target word in the first iteration) as $x_{t'}$, where $t' = (T'/T) \cdot t$. T' and T are the lengths of the source X and target Y^*, respectively.

3.3.2 Denoising Autoencoder

The proposed approach could instead be viewed as learning a conditional denoising autoencoder which is known to capture the gradient of the log-density. That is, we implicitly learn to find a direction Δ_Y in the output space that maximizes the underlying true, data-generating distribution $\log P(Y|X)$. Because the output space is discrete, much of the theoretical analysis by Alain and Bengio [2014] are not strictly applicable. We however find this view attractive as it serves as an alternative foundation for designing a learning algorithm.

Training We start with a corruption process $C(Y|Y^*)$, which introduces noise to the correct output Y^*. Given the reference translation Y^*, we sample $\tilde{Y} \sim C(Y|Y^*)$ which becomes as an input to each conditional in Eq. (3.1). Then, the goal of learning is to maximize the log-probability of the original reference Y^* given the corrupted version. That is, to minimize

$$J_{\text{DAE}}(\theta) = -\sum_{t=1}^{T} \log p_\theta(y_t^*|\tilde{Y}, X). \tag{3.3}$$

Once this cost J_{DAE} is minimized, we can recursively perform the maximum-a-posterior inference, i.e., $\hat{Y} = \arg\max_Y \log p_\theta(Y|X)$, to find $\hat{Y}$ that (approximately) maximizes $\log p(Y|X)$.

Corruption Process C There is little consensus on the best corruption process for a sequence, especially of discrete tokens. In this work, we use a corruption process proposed by Hill et al. [2016], which has recently become more widely adopted [see, e.g., Artetxe et al., 2017; Lample et al., 2017]. Each y_t^* in a reference target $Y^* = (y_1^*, \ldots, y_T^*)$ is corrupted with a probability $\beta \in [0, 1]$. If decided to corrupt, we either (1) replace y_{t+1}^* with this token y_t^*, (2) replace y_t^* with a token uniformly selected from a vocabulary of all unique tokens at random, or (3) swap y_t^* and y_{t+1}^*. This is done sequentially from y_1^* until y_T^*.

3.3.3 Learning

Cost function Although it is possible to train the proposed non-autoregressive sequence model using either of the cost functions above (J_{LVM} or J_{DAE},) we propose to stochastically mix these two cost functions. We do so by randomly replacing each term $\hat{Y}^{l-1}$ in Eq. (3.2) with $\tilde{Y}$ in Eq. (3.3):

$$J(\theta) = -\sum_{l=0}^{L+1} \left(\alpha_l \sum_{t=1}^{T} \log p_\theta(y_t^* | \hat{Y}^{l-1}, X) + (1 - \alpha_l) \sum_{t=1}^{T} \log p_\theta(y_t^* | \tilde{Y}, X) \right), \quad (3.4)$$

where $\tilde{Y} \sim C(Y|Y^*)$, and α_l is a sample from a Bernoulli distribution with the probability p_{DAE}. p_{DAE} is a hyperparameter. As the first conditional $p(Y^0|X)$ in Eq. (3.1) does not take as input any target Y, we set $\alpha_0 = 1$ always.

Distillation Gu et al. [2018a], in the context of machine translation, and van den Oord et al. [2018], in the context of speech generation, have recently discovered that it is important to use knowledge distillation [Hinton et al., 2015; Kim and Rush, 2016] to successfully train a non-autoregressive sequence model. Following Gu et al. [2018a], we also use knowledge distillation by replacing the reference target Y^* of each training example (X, Y^*) with a target Y^{AR} generated from a well-trained autoregressive counterpart. Other than this replacement, the cost function in Eq 3.4 and the model architecture remain unchanged.

Target Length Prediction One difference between the autoregressive and non-autoregressive models is that the former naturally models the length of a target sequence without any arbitrary upper-bound, while the latter does not. It is hence necessary to separately model $p(T|X)$, where T is the length of a target sequence, although during training, we simply use the length of each reference target sequence.

3.3.4 Inference: Decoding

Inference in the proposed approach is entirely deterministic. We start from the input X and first predict the length of the target sequence $\hat{T} = \arg\max_T \log p(T|X)$. Then, given X and $\hat{T}$ we generate the initial target sequence by $\hat{y}_t^0 = \arg\max_{y_t} \log p(y_t^0|X)$, for $t = 1, \ldots, T$ We continue refining the target sequence by $\hat{y}_t^l = \arg\max_{y_t} \log p(y_t^l|\hat{Y}^{l-1}, X)$, for $t = 1, \ldots, T$.

Because these conditionals, except for the initial one, are modeled by a single, shared neural network, this refinement can be performed as many iterations as necessary until a predefined stopping criterion is met. A criterion can be based either on the amount of change in a target sequence after each iteration (i.e., $D(\hat{Y}^{l-1}, \hat{Y}^l) \leq \epsilon$), or on the amount of change in the conditional log-probabilities (i.e., $|\log p(\hat{Y}^{l-1}|X) - \log p(\hat{Y}^{l-1}|X)| \leq \epsilon$) or on the computational budget. In our experiments, we use the first criterion and use

Jaccard distance as our distance function D.

3.4 Related Work

Non-Autoregressive Neural Machine Translation Schwenk [2012] proposed a continuous-space translation model to estimate the conditional distribution over a target phrase given a source phrase, while dropping the conditional dependencies among target tokens. The evaluation was however limited to reranking and to short phrase pairs (up to 7 words on each side) only. Kaiser and Bengio [2016] investigated neural GPU [Kaiser and Sutskever, 2016], for machine translation. They evaluated both non-autoregressive and autoregressive approaches, and found that the non-autoregressive approach significantly lags behind the autoregressive variants. It however differs from our approach that each iteration does not output a refined version from the previous iteration. The recent paper by Gu et al. [2018a] is most relevant to the proposed work. They similarly introduced a sequence of discrete latent variables. They however use supervised learning for inference, using the word alignment tool [Dyer et al., 2013]. To achieve the best result, Gu et al. [2018a] stochastically sample the latent variables and rerank the corresponding target sequences with an external, autoregressive model. This is in contrast to the proposed approach which is fully deterministic during decoding and does not rely on any extra reranking mechanism.

Parallel WaveNet Simultaneously with Gu et al. [2018a], van den Oord et al. [2018] presented a non-autoregressive sequence model for speech generation. They use inverse autoregressive flow [IAF, Kingma et al., 2016] to map a sequence of independent random variables to a target sequence. They apply the IAF multiple times, similarly to our iterative refinement strategy. Their approach is however restricted to continuous target variables, while the proposed approach in principle could be applied to both discrete and continuous variables.

Post-Editing for Machine Translation Novak et al. [2016] proposed a convolutional neural network that iteratively predicts and applies token substitutions given a translation from a phase-based translation system. Unlike their system, our approach can edit an intermediate translation with a higher degree of freedom. QuickEdit [Grangier and Auli, 2017] and deliberation network [Xia et al., 2017] incorporate the idea of refinement into neural machine translation. Both systems consist of two autoregressive decoders. The second decoder takes into account the translation generated by the first decoder. We extend these earlier efforts by incorporating more than one refinement steps without necessitating extra annotations.

Infusion Training Bordes et al. [2017] proposed an unconditional generative model for images based on iterative refinement. At each step l of iterative refinement, the model is trained to maximize the log-likelihood of target Y given the weighted mixture of generated samples from the previous iteration $\hat{Y}^{l-1}$ and a corrupted target $\tilde{Y}$. That is, the corrupted version of target is "infused" into generated samples during training. In the domain of text, however, computing a weighted mixture of two sequences of discrete tokens is not well defined, and we propose to stochastically mix denoising and lowerbound maximization objectives.

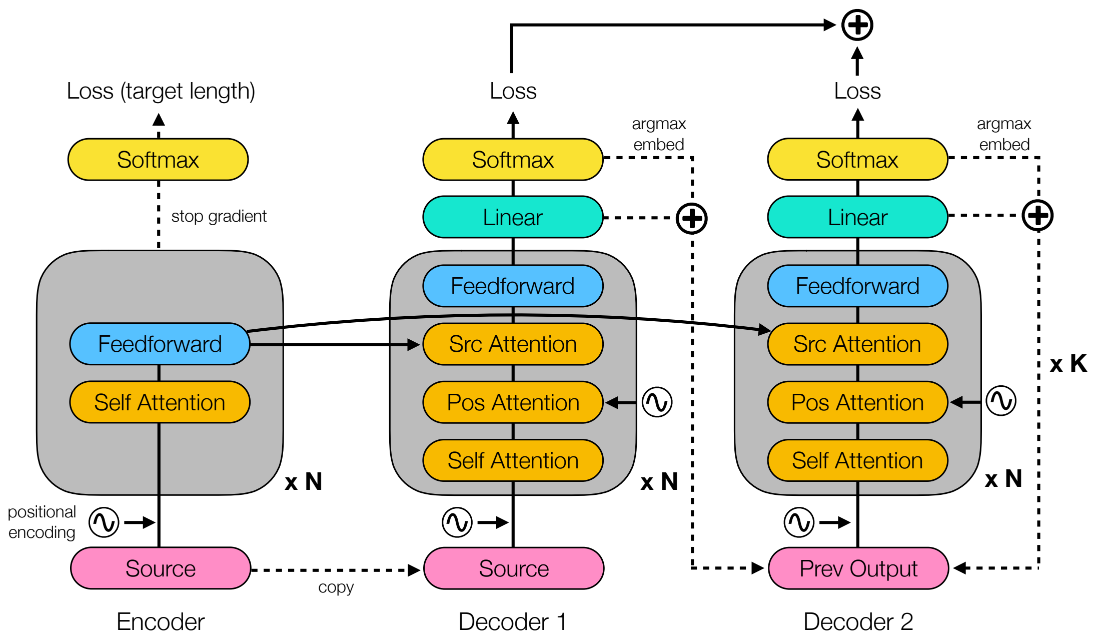


Figure 3.1: We compose three transformer blocks ("Encoder", "Decoder 1" and "Decoder 2") to implement the proposed non-autoregressive sequence model.

3.5 Parameterization

We use three transformer-based network blocks to implement our model. The first block ("Encoder") encodes the input X, the second block ("Decoder 1") models the first conditional $\log p(Y^0|X)$, and the final block ("Decoder 2") is shared across iterative refinement steps, modeling $\log p(Y^l|\hat{Y}^{l-1}, X)$. These blocks are depicted side-by-side in Fig. 3.1. The encoder is identical to that from the original Transformer [Vaswani et al., 2017]. We however use the decoders from Gu et al. [2018a] with additional positional attention and use the highway layer [Srivastava et al., 2015] instead of the residual layer [He et al., 2016].

The original input X is padded or shortned to fit the length of the reference target sequence before being fed to Decoder 1. At each refinement step l, Decoder 2 takes as input the predicted target sequence $\hat{Y}^{l-1}$ and the sequence of final activation vectors from the previous step.

3.6 Experimental Setting

We evaluate the proposed approach on two sequence modeling tasks: machine translation and image caption generation. We compare the proposed non-autoregressive model against the autoregressive counterpart both in terms of generation quality, measured in terms of BLEU [Papineni et al., 2002], and generation efficiency, measured in terms of (source) tokens and images per second for translation and image captioning, respectively.

Machine Translation We choose three tasks of different sizes: IWSLT'16 En$\leftrightarrow$De (196k pairs), WMT'16 En$\leftrightarrow$Ro (610k pairs) and WMT'14 En$\leftrightarrow$De (4.5M pairs). We tokenize each sentence using a script from Moses [Koehn et al., 2007] and segment each word into subword units using BPE [Sennrich et al., 2016b]. We use 40k tokens from both source and target for all the tasks. For WMT'14 En-De, we use newstest-2013 and newstest-2014 as development and test sets. For WMT'16 En-Ro, we use newsdev-2016 and newstest-2016 as development and test sets. For IWSLT'16 En-De, we use test2013 for validation.

We closely follow the setting by Gu et al. [2018a]. In the case of IWSLT'16 En-De, we use the small model ($d_{\text{model}} = 278, d_{\text{hidden}} = 507, p_{\text{dropout}} = 0.1, n_{\text{layer}} = 5$ and $n_{\text{head}} = 2$).[1] For WMT'14 En-De and WMT'16 En-Ro, we use the base transformer

1. Due to the space constraint, we refer readers to [Gu et al., 2018a; Vaswani et al., 2017] for more details.

by Vaswani et al. [2017] ($d_{\text{model}} = 512, d_{\text{hidden}} = 512, p_{\text{dropout}} = 0.1, n_{\text{layer}} = 6$ and $n_{\text{head}} = 8$). We use the warm-up learning rate scheduling [Vaswani et al., 2017] for the WMT tasks, while using linear annealing (from 3×10^{-4} to 10^{-5}) for the IWSLT task. We do not use label smoothing nor average multiple check-pointed models. These decisions were made based on the preliminary experiments. We train each model either on a single P40 (WMT'14 En-De and WMT'16 En-Ro) or on a single P100 (IWSLT'16 En-De) with each minibatch consisting of approximately 2k tokens. We use four P100's to train non-autoregressive models on WMT'14 En-De.

Image Caption Generation: MS COCO We use MS COCO [Lin et al., 2014]. We use the publicly available splits [Karpathy and Li, 2015], consisting of 113,287 training images, 5k validation images and 5k test images. We extract 49 512-dimensional feature vectors for each image, using a ResNet-18 [He et al., 2016] pretrained on ImageNet [Deng et al., 2009]. The average of these vectors is copied as many times to match the length of the target sentence (reference during training and predicted during evaluation) to form the initial input to Decoder 1. We use the base transformer [Vaswani et al., 2017] except that n_{layer} is set to 4. We train each model on a single 1080ti with each minibatch consisting of approximately 1,024 tokens.

Target Length Prediction We formulate the target length prediction as classification, predicting the difference between the target and source lengths for translation and the target length for image captioning. All the hidden vectors from the n_{layer} layers of the encoder are summed and fed to a softmax classifier after affine transformation. We however do not tune the encoder's parameters for target length prediction. We use this

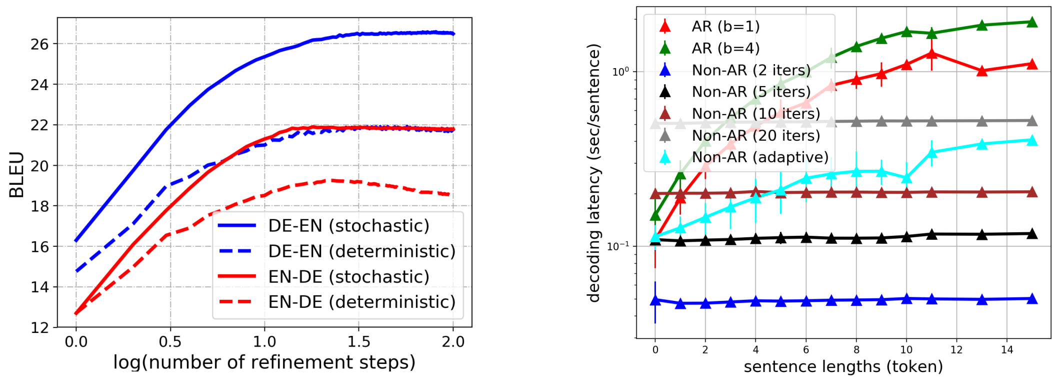


Figure 3.2: (a) BLEU scores on WMT'14 En-De w.r.t. the number of refinement steps (up to 10^2). The x-axis is in the logarithmic scale. (b) the decoding latencies (sec/sentence) of different approaches on IWSLT'16 En→De. The y-axis is in the logarithmic scale.

length predictor only during test time. We find it important to accurately predict the target length for good overall performance. See Section 3.7 for an analysis on our length prediction model.

Training and Inference We use Adam [Kingma and Ba, 2015] and use $L = 3$ in Eq. (3.1) during training ($i_{\text{train}} = 4$ from hereon.) We use $p_{\text{DAE}} = 0.5$. We use the deterministic strategy for IWSLT'16 En-De, WMT'16 En-Ro and MS COCO, while the stochastic strategy is used for WMT'14 En-De. These decisions were made based on the validation set performance. After both the non-autogressive sequence model and target length predictor are trained, we decode by first predicting the target length and then running iterative refinement steps until the outputs of consecutive iterations are the same (or Jaccard distance between consecutive decoded sequences is 1). To assess the effectiveness of this adaptive scheme, we also test a fixed number of steps (i_{dec}).

		IWSLT'16 En-De				WMT'16 En-Ro			
		En→	De→	GPU	CPU	En→	Ro→	GPU	CPU
AR	b = 1	28.64	34.11	70.3	32.2	31.93	31.55	55.6	15.7
	b = 4	28.98	34.81	63.8	14.6	32.40	32.06	43.3	7.3
NAT	FT	26.52	–	–	–	27.29	29.06	–	–
	FT+NPD	28.16	–	–	–	29.79	31.44	–	–
Our Model	i_{dec} = 1	22.20	27.68	573.0	213.2	24.45	25.73	694.2	98.6
	i_{dec} = 2	24.82	30.23	423.8	110.9	27.10	28.15	332.7	62.8
	i_{dec} = 5	26.58	31.85	189.7	52.8	28.86	29.72	194.4	29.0
	i_{dec} = 10	27.11	32.31	98.8	24.1	29.32	30.19	93.1	14.8
	Adaptive	27.01	32.43	125.9	29.3	29.66	30.30	118.3	16.5

Table 3.1: Generation quality (BLEU↑) and decoding efficiency (tokens/sec↑ for translation, images/sec↑ for image captioning). Decoding efficiency is measured sentence-by-sentence. AR: autoregressive models. b: beam width. i_{dec}: the number of refinement steps taken during decoding. Adaptive: the adaptive number of refinement steps. NAT: non-autoregressive transformer models [Gu et al., 2018a]. FT: fertility. NPD reranking using 100 samples.

In machine translation, we remove any repetition by collapsing multiple consecutive occurrences of a token.

3.7 Results and Analysis

We make some important observations in Table 3.1. First, the generation quality improves across all the tasks as we run more refinement steps i_{dec} even beyond that used in training ($i_{\text{train}} = 4$), which supports our interpretation as a conditional denoising autoencoder in Sec. 3.3.2. To further verify this, we run decoding on WMT'14 (both directions) up to 100 iterations. As shown in Fig. 3.2 (a), the quality improves well beyond the number of refinement steps used during training.

Second, the generation efficiency decreases as more refinements are made. We plot

		WMT'14 En-De				MS COCO		
		En→	De→	GPU	CPU	BLEU	GPU	CPU
AR	b = 1	23.77	28.15	54.0	15.8	23.47	4.3	2.1
	b = 4	24.57	28.47	44.9	7.0	24.78	3.6	1.0
NAT	FT	17.69	21.47	–	–	–	–	–
	FT+NPD	19.17	23.30	–	–	–	–	–
Our Model	i_{dec} = 1	13.91	16.77	511.4	83.3	20.12	17.1	8.9
	i_{dec} = 2	16.95	20.39	393.6	49.6	20.88	12.0	5.7
	i_{dec} = 5	20.26	23.86	139.7	23.1	21.12	6.2	2.8
	i_{dec} = 10	21.61	25.48	90.4	12.3	21.24	2.0	1.2
	Adaptive	21.54	25.43	107.2	20.3	21.12	10.8	4.8

Table 3.2: Generation quality (BLEU↑) and decoding efficiency (tokens/sec↑ for translation, images/sec↑ for image captioning). Decoding efficiency is measured sentence-by-sentence. AR: autoregressive models. b: beam width. i_{dec}: the number of refinement steps taken during decoding. Adaptive: the adaptive number of refinement steps. NAT: non-autoregressive transformer models [Gu et al., 2018a]. FT: fertility. NPD reranking using 100 samples.

the average seconds per sentence in Fig. 3.2 (b), measured on GPU while sequentially decoding one sentence at a time. As expected, decoding from the autoregressive model linearly slows down as the sentence length grows, while decoding from the non-autoregressive model with a fixed number of iterations has the constant complexity. However, the generation efficiency of non-autoregressive model decreases as more refinements are made. To make a smooth trade-off between the quality and speed, the adaptive decoding scheme allows us to achieve near-best generation quality with a significantly lower computational overhead. Moreover, the adaptive decoding scheme automatically increases the number of refinement steps as the sentence length increases, suggesting that this scheme captures the amount of information in the input well. The increase in latency is however less severe than that of the autoregressive model.

We also observe that the speedup in decoding is much clearer on GPU than on CPU.

	i_{train}	p_{DAE}	distill	En→De rep	En→De no rep	De→En rep	De→En no rep
AR		$b = 1$		28.64		34.11	
		$b = 4$		28.98		34.81	
Our Models	1	0		14.62	18.03	16.70	21.18
	2	0		17.42	21.08	19.84	24.25
	4	0		19.22	22.65	22.15	25.24
	4	1		19.83	22.29	24.00	26.57
	4	0.5		20.91	23.65	24.05	28.18
	4	0.5	√	26.17	27.11	31.92	32.59

Table 3.3: Ablation study on the dev set of IWSLT'16.

This is a consequence of highly parallel computation of the proposed non-autoregressive model, which is better suited to GPUs, showcasing the potential of using the non-autoregressive model with a specialized hardware for parallel computation, such as Google's TPUs [Jouppi et al., 2017]. The results of our model decoded with adaptive decoding scheme are comparable to the results from [Gu et al., 2018a], without relying on any external tool. On WMT'14 En-De, the proposed model outperforms the best model from [Gu et al., 2018a] by two points.

Lastly, it is encouraging to observe that the proposed non-autoregressive model works well on image caption generation. This result confirms the generality of our approach beyond machine translation, unlike that by Gu et al. [2018a] which was for machine translation or by van den Oord et al. [2018] which was for speech synthesis.

Ablation Study We use IWSLT'16 En-De to investigate the impact of different number of refinement steps during training (denoted as i_{train}) as well as probability of using denoising autoencoder objective during training (denoted as p_{DAE}). The results are pre-

stochastic	distill	IWSLT'16 (En→)	WMT'14 (En→)
		23.65	7.56
√		22.80	16.56
	√	27.11	18.91
√	√	25.39	21.22

Table 3.4: Deterministic and stochastic approximation

sented in Table 3.3.

First, we observe that it is beneficial to use multiple iterations of refinement during training. By using four iterations (one step of decoder 1, followed by three steps of decoder 2), the BLEU score improved by approximately 1.5 points in both directions. We also notice that it is necessary to use the proposed hybrid learning strategy to maximize the improvement from more iterations during training ($i_{\text{train}} = 4$ vs. $i_{\text{train}} = 4, p_{\text{DAE}} = 1.0$ vs. $i_{\text{train}} = 4, p_{\text{DAE}} = 0.5$.) Knowledge distillation was crucial to close the gap between the proposed deterministic non-autoregressive sequence model and its autoregressive counterpart, echoing the observations by Gu et al. [2018a] and van den Oord et al. [2018]. Finally, we see that removing repeating consecutive symbols improves the quality of the best trained models ($i_{\text{train}} = 4, p_{\text{DAE}} = 0.5$) by approximately +1 BLEU. This suggests that the proposed iterative refinement is not enough to remove repetitions on its own. Further investigation is necessary to properly tackle this issue, which we leave as a future work.

We then compare the deterministic and stochastic approximation strategies on IWSLT'16 En→De and WMT'14 En→De. According to the results in Table 3.4, the stochastic strategy is crucial with a large corpus (WMT'14), while the deterministic strategy works as well or better with a small corpus (IWSLT'16). Both of the strategies benefit from

	IWSLT'16		WMT'16		WMT'14	
	En→	→En	En→	→En	En→	→En
pred	27.01	32.43	29.66	30.30	21.54	25.43
ref	28.15	33.11	30.42	31.26	22.10	26.40

Table 3.5: BLEU scores on each dataset when using reference length (ref) and predicted target length (pred).

knowledge distillation, but the gap between the two strategies when the dataset is large is much more apparent without knowledge distillation.

Impact of Length Prediction The quality of length prediction has an impact on the overall translation/captioning performance. When using the reference target length (during inference), we consistently observed approximately 1 BLEU score improvement over reported results in the tables and figures across different datasets in the chapter (see Table 3.5 for more detailed comparison).

We additionally compared our length prediction model with a simple baseline that uses length statistics of the corresponding training dataset (a non-parametric approach). To predict the target length for a source sentence with length L_s, we take the average length of all the target sentences coupled with the sources sentences of length L_s in the training set. Compared to this approach, our length prediction model predicts target length correctly twice as often (16% vs. 8%), and gives higher prediction accuracy within five tokens (83% vs. 69%)

3.7.1 Qualitative Analysis

Machine Translation In Table 3.6, we present three sample translations and their iterative refinement steps from the development set of IWSLT'16 (De$\rightarrow$En). As expected, the sequence generated from the first iteration is a rough version of translation and is iteratively refined over multiple steps. By inspecting the underlined sub-sequences, we see that each iteration does not monotonically improve the translation, but overall modifies the translation towards the reference sentence. Missing words are added, while unnecessary words are dropped. For instance, see the second example. The second iteration removes the unnecessary "were", and the fourth iteration inserts a new word "mostly". The phrase "at the time" is gradually added one word at a time.

Image Caption Generation Tables 3.7 and 3.8 show two examples of image caption generation. We observe that each iteration captures more and more details of the input image. In Table 3.7, the bus was described only as a "yellow bus" in the first iteration, but the subsequent iterations refine it into "yellow and black bus". Similarly, "road" is refined into "lot". We notice this behavior in the second example (Table 3.8) as well. The first iteration does not specify the place in which "a woman" is "standing on", which is fixed immediately in the second iteration: "standing on a tennis court". In the final and fourth iteration, the proposed model captures the fact that the "woman" is "holding" a racquet.

3.8 Conclusion

Following on the exciting, recent success of non-autoregressive neural sequence modeling by Gu et al. [2018a] and van den Oord et al. [2018], we proposed a deterministic non-autoregressive neural sequence model based on the idea of iterative refinement. We designed a learning algorithm specialized to the proposed approach by interpreting the entire model as a latent variable model and each refinement step as denoising.

We implemented our approach using the Transformer and evaluated it on two tasks: machine translation and image caption generation. On both tasks, we were able to show that the proposed non-autoregressive model performs closely to the autoregressive counterpart with significant speedup in decoding. Qualitative analysis revealed that the iterative refinement indeed refines a target sequence gradually over multiple steps.

Despite these promising results, we observed that proposed non-autoregressive neural sequence model is outperformed by its autoregressive counterpart in terms of the generation quality. The following directions should be pursued in the future to narrow this gap. First, we should investigate better approximation to the marginal log-probability. Second, the impact of the corruption process on the generation quality must be studied. Lastly, further work on sequence-to-sequence model architectures could yield better results in non-autoregressive sequence modeling.

3.9 Developments Since Publication

Since the time of this chapter's publication in 2018, several works proposed non-autoregressive neural machine translation systems that perform iterative refinement in the token space.

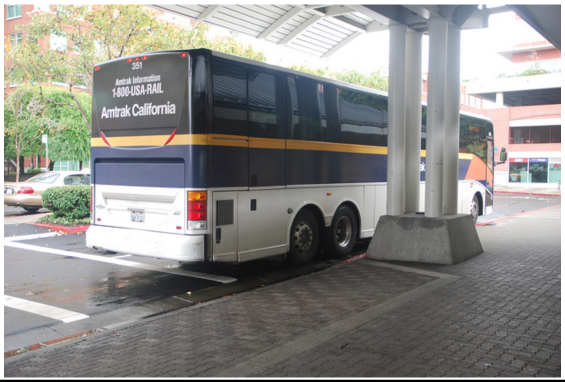
Among these, models trained on masked language modeling loss [Devlin et al., 2019] have been particularly successful [Chan et al., 2019; Ghazvininejad et al., 2019, 2020; Kasai et al., 2020; Kreutzer et al., 2020; Mansimov et al., 2019b]. On WMT'14 En$\rightarrow$De, a system proposed by Ghazvininejad et al. [2019] delivered comparable translation quality to autoregressive baseline in 10 refinement steps. Another successful class of models learn to generate text in non-monotonic orders [Gu et al., 2019a,b; Stern et al., 2019]. By specifying multiple token positions as insertion points, these models can generate a sentence in a fixed number of steps. Finally, Stern et al. [2018]; Wang et al. [2018] proposed semi-autoregressive approaches that autoregressively generate multiple tokens at a time, thereby reducing inference latency. Overall, text generation by refining the output remains an active avenue or research, and we believe that more exciting works will follow.

Src	seitdem habe ich sieben Häuser in der Nachbarschaft mit den Lichtern versorgt und sie funktionierenen wirklich gut .
Iter 1	and I 've been seven homes since in neighborhood with the lights and they 're really functional .
Iter 2	and I 've been seven homes in the neighborhood with the lights , and they 're a really functional .
Iter 4	and I 've been seven homes in neighborhood with the lights , and they 're a really functional .
Iter 8	and I 've been providing seven homes in the neighborhood with the lights and they 're a really functional .
Iter 20	and I 've been providing seven homes in the neighborhood with the lights , and they 're a very good functional .
Ref	since now , I 've set up seven homes around my community , and they 're really working .

Src	er sah sehr glücklich aus , was damals ziemlich ungewöhnlich war , da ihn die Nachrichten meistens deprimierten .
Iter 1	he looked very happy , which was pretty unusual the , because the news was were usually depressing .
Iter 2	he looked very happy , which was pretty unusual at the , because the news was s depressing .
Iter 4	he looked very happy , which was pretty unusual at the , because news was mostly depressing .
Iter 8	he looked very happy , which was pretty unusual at the time because the news was mostly depressing .
Iter 20	he looked very happy , which was pretty unusual at the time , because the news was mostly depressing .
Ref	there was a big smile on his face which was unusual then , because the news mostly depressed him .

Src	furchtlos zu sein heißt für mich , heute ehrlich zu sein .
Iter 1	to be , for me , to be honest today .
Iter 2	to be fearless , me , is to be honest today .
Iter 4	to be fearless for me , is to be honest today .
Iter 8	to be fearless for me , me to be honest today .
Iter 20	to be fearless for me , is to be honest today .
Ref	so today , for me , being fearless means being honest .

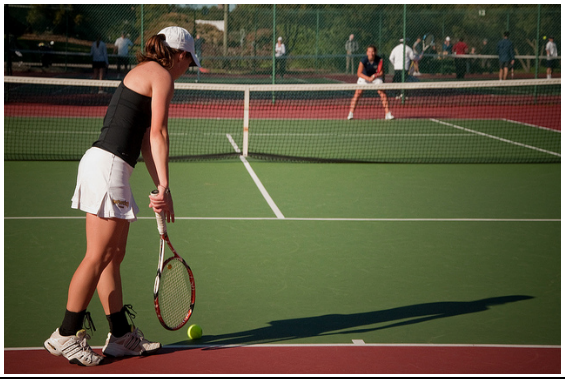
Table 3.6: Three sample De→En translations from the non-autoregressive sequence model. Source sentences are from the dev set of IWSLT'16. The first iteration corresponds to Decoder 1, and from thereon, Decoder 2 is repeatedly applied. Sub-sequences with changes across the refinement steps are underlined.



Generated Caption	
Iter 1	a yellow bus parked on parked in of parking road .
Iter 2	a yellow and black on parked in a parking lot .
Iter 3	a yellow and black bus parked in a parking lot .
Iter 4	a yellow and black bus parked in a parking lot .

Reference Captions
a tour bus is parked on the curb waiting
city bus parked on side of hotel in the rain .
bus parked under an awning next to brick sidewalk
a bus is parked on the curb in front of a building .
a double decked bus sits parked under an awning

Table 3.7: A sample image caption from the proposed non-autoregressive sequence model. The images are from the development set of MS COCO. The first iteration is from decoder 1, while the subsequent ones are from decoder 2. Subsequences with changes across the refinement steps are underlined.



Generated Caption	
Iter 1	a woman standing on playing tennis on a tennis racquet .
Iter 2	a woman standing on a tennis court a tennis racquet .
Iter 3	a woman standing on a tennis court a a racquet .
Iter 4	a woman standing on a tennis court holding a racquet .
Reference Captions	
a female tennis player in a black top playing tennis	
a woman standing on a tennis court holding a racquet .	
a female tennis player preparing to serve the ball .	
a woman is holding a tennis racket on a court	
a woman getting ready to reach for a tennis ball on the ground	

Table 3.8: Another sample image caption from the proposed non-autoregressive sequence model. The images are from the development set of MS COCO. The first iteration is from decoder 1, while the subsequent ones are from decoder 2. Subsequences with changes across the refinement steps are underlined.

Chapter 4

Latent-Variable Non-Autoregressive Neural Machine Translation with Deterministic Inference using a Delta Posterior

Although neural machine translation models reached high translation quality, the autoregressive nature makes inference difficult to parallelize and leads to high translation latency. Inspired by recent refinement-based approaches, we propose LaNMT, a latent-variable non-autoregressive model with continuous latent variables and deterministic inference procedure. In contrast to existing approaches, we use a deterministic inference algorithm to find the target sequence that maximizes the lowerbound to the log-probability. During inference, the length of translation automatically adapts itself. Our experiments show that the lowerbound can be greatly increased by running the inference algorithm, resulting in significantly improved translation quality. Our proposed model closes the performance gap between non-autoregressive and autoregressive approaches on ASPEC Ja-En dataset with 8.6x faster decoding. On WMT'14 En-De dataset, our model narrows the gap with autoregressive baseline to 2.0 BLEU points with 12.5x speedup. By decoding multiple initial latent variables in parallel and rescore using a teacher model, the proposed model further brings the gap down to 1.0 BLEU point on WMT'14 En-De task with 6.8x speedup.

4.1 Introduction

The field of Neural Machine Translation (NMT) has seen significant improvements in recent years Bahdanau et al. [2015]; Gehring et al. [2017]; Vaswani et al. [2017]; Wu et al. [2016]. Despite impressive improvements in translation accuracy, the autoregressive nature of NMT models have made it difficult to speed up decoding by utilizing parallel model architecture and hardware accelerators. This has sparked interest in *non-autoregressive* NMT models, which predict every target tokens in parallel. In addition to the obvious decoding efficiency, non-autoregressive text generation is appealing as it does not suffer from exposure bias and suboptimal inference.

Inspired by recent work in non-autoregressive NMT using discrete latent variables [Kaiser et al., 2018] and iterative refinement [Lee et al., 2018], we introduce a sequence of continuous latent variables to capture the uncertainty in the target sentence. We motivate such a latent variable model by conjecturing that it is easier to refine lower-dimensional continuous variables[1] than to refine high-dimensional discrete variables, as done in Lee et al. [2018]. Unlike Kaiser et al. [2018], the posterior and the prior can be jointly trained to maximize the evidence lowerbound of the log-likelihood $\log p(y|x)$.

In this work, we propose a deterministic iterative algorithm to refine the approximate posterior over the latent variables and obtain better target predictions. During inference, we first obtain the initial posterior from a prior distribution $p(z|x)$ and the initial guess of the target sentence from the conditional distribution $p(y|x, z)$. We then alternate between updating the approximate posterior and target tokens with the help of an approximate posterior $q(z|x, y)$. We avoid stochasticity at inference time by introducing a *delta posterior* over the latent variables. We empirically find that this iterative algorithm significantly improves the lowerbound and results in better BLEU scores. By refining the latent variables instead of tokens, the length of translation can dynamically adapt throughout this procedure, unlike previous approaches where the target length was fixed throughout the refinement process. In other words, even if the initial length prediction is incorrect, it can be corrected simultaneously with the target tokens.

Our models[2] outperform the autoregressive baseline on ASPEC Ja-En dataset with 8.6x decoding speedup and bring the performance gap down to 2.0 BLEU points on WMT'14 En-De with 12.5x decoding speedup. By decoding multiple latent variables sampled from the prior and rescore using a autoregressive teacher model, the proposed model is able to further narrow the performance gap on WMT'14 En-De task down to 1.0 BLEU point with 6.8x speedup. The contributions of this work can be summarize as follows:

1. We use 8-dimensional latent variables in our experiments.
2. Our code can be found in https://github.com/zomux/lanmt .

1. We propose a continuous latent-variable non-autoregressive NMT model for faster inference. The model learns identical number of latent vectors as the input tokens. A length transformation mechanism is designed to adapt the number of latent vectors to match the target length.

2. We demonstrate a principle inference method for this kind of model by introducing a deterministic inference algorithm. We show the algorithm converges rapidly in practice and is capable of improving the translation quality by around 2.0 BLEU points.

4.2 Background

4.2.1 Autoregressive NMT

In order to model the joint probability of the target tokens $y_1, \cdots, y_{|y|}$ given the source sentence x, most NMT models use an autoregressive factorization of the joint probability which has the following form:

$$\log p(y|x) = \sum_{i=1}^{|y|} \log p(y_i|y_{<i}, x),$$

where $y_{<i}$ denotes the target tokens preceding y_i. Here, the probability of emitting each token $p(y_i|y_{<i}, x)$ is parameterized with a neural network.

To obtain a translation from this model, one could predict target tokens sequentially by greedily taking *argmax* of the token prediction probabilities. The decoding progress ends when a "`</s>`" token, which indicates the end of a sequence, is selected. In practice, however, this greedy approach yields suboptimal sentences, and *beam search* is often used to decode better translations by maintaining multiple hypotheses. However, decoding with a large beam size significantly decreases translation speed.

4.2.2 Non-Autoregressive NMT

Although autoregressive models achieve high translation quality through recent advances in NMT, the main drawback is that autoregressive modeling forbids the decoding algorithm to select tokens in multiple positions simultaneously. This results in inefficient use of computational resource and increased translation latency.

In contrast, non-autoregressive NMT models predict target tokens without depending on preceding tokens, depicted by the following objective:

$$\log p(y|x) = \sum_{i=i}^{|y|} \log p(y_i|x). \tag{4.1}$$

As the prediction of each target token y_i now depends only on the source sentence x and its location i in the sequence, the translation process can be easily parallelized. We obtain a target sequence by applying *argmax* to all token probabilities.

The main challenge of non-autoregressive NMT is on capturing dependencies among target tokens. As the probability of each target token does not depend on the surrounding tokens, applying *argmax* at each position i may easily result in an inconsistent sequence, that includes duplicated or missing words. It is thus important for non-autoregressive models to apply techniques to ensure the consistency of generated words.

4.3 Latent-Variable Non-Autoregressive NMT

In this work, we propose LaNMT, a latent-variable non-autoregressive NMT model by introducing a sequence of continuous latent variables to model the uncertainty about the target sentence. These latent variables z are constrained to have the same length as the source sequence, that is, $|z| = |x|$. Instead of directly maximizing the objective function in Eq. (4.1), we maximize a lowerbound to the marginal log-probability $\log p(y|x) = \log \int p(y|z, x)p(z|x)dz$:

$$\mathcal{L}(\omega, \phi, \theta) = \mathbb{E}_{z \sim q_\phi} \big[\log p_\theta(y|x, z) \big] - \mathrm{KL}\big[q_\phi(z|x, y) || p_\omega(z|x) \big], \tag{4.2}$$

where $p_\omega(z|x)$ is the prior, $q_\phi(z|x, y)$ is an approximate posterior and $p_\theta(y|x, z)$ is the decoder. The objective function in Eq. (4.2) is referred to as the evidence lowerbound (ELBO). As shown in the equation, the lowerbound is parameterized by three sets of parameters: ω, ϕ and θ.

Both the prior p_ω and the approximate posterior q_ϕ are modeled as spherical Gaussian distributions. The model can be trained end-to-end with the reparameterization trick Kingma and Welling [2014a].

4.3.1 A Modified Objective Function with Length Prediction

During training, we want the model to maximize the lowerbound in Eq. (4.2). However, to generate a translation, the target length l_y has to be predicted first. We let the latent variables model the target length by parameterizing the decoder as:

$$\begin{aligned} p_\theta(y|x,z) &= \sum_l p_\theta(y,l|x,z) \\ &= p_\theta(y,l_y|x,z) \\ &= p_\theta(y|x,z,l_y)p_\theta(l_y|z). \end{aligned} \tag{4.3}$$

Here l_y denotes the length of y. The second step is valid as the probability $p_\theta(y, l \neq l_y|x,z)$ is always zero. Plugging in Eq. 4.3, with the independent assumption on both latent variables and target tokens, the objective has the following form:

$$\mathbb{E}_{z\sim q_\phi}\left[\sum_{i=1}^{|y|}\log p_\theta(y_i|x,z,l_y) + \log p_\theta(l_y|z)\right] - \sum_{k=1}^{|x|}\mathrm{KL}\left[q_\phi(z_k|x,y)||p_\omega(z_k|x)\right]. \tag{4.4}$$

4.3.2 Parameterization

As evident from in Eq. (4.4), there are four parameterized components in our model: the prior $p_\omega(z|x)$, approximate posterior $q_\phi(z|x,y)$, decoder $p_\theta(y|x,z,l_y)$ and length predictor $p_\theta(l_y|z)$. The architecture of the proposed non-autoregressive model is depicted in Fig. 4.1, which reuses modules in Transformer [Vaswani et al., 2017] to compute the aforementioned distributions.

Main Components To compute the prior $p_\omega(z|x)$, we use a multi-layer self-attention encoder which has the same structure as the Transformer encoder. In each layer, a feed-forward computation is applied after the self-attention. To obtain the probability, we apply a linear transformation to reduce the dimensionality and compute the mean and variance vectors.

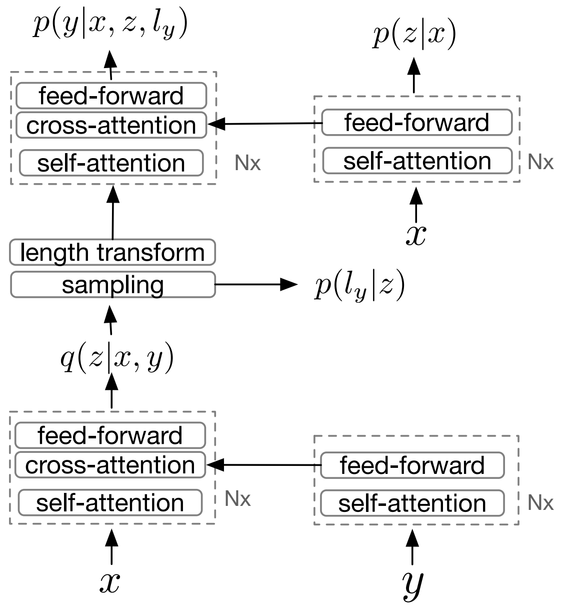
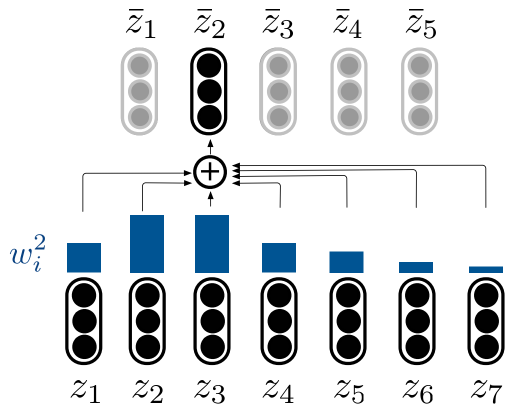


Figure 4.1: Architecture of the proposed non-autogressive model. The model is composed of four components: prior $p(z|x)$, approximate posterior $q(z|x,y)$, length predictor $p(l_y|z)$ and decoder $p(y|x,z)$. These components are trained end-to-end to maximize the evidence lower-bound.

For the approximate posterior $q_\phi(z|x,y)$, as it is a function of the source x and the target y, we first encode y with a self-attention encoder. Then, the resulting vectors are fed into an attention-based decoder initialized by x embeddings. Its architecture is similar to the Transformer decoder except that no causal mask is used. Similar to the prior, we apply a linear layer to obtain the mean and variance vectors.

To backpropagate the loss signal of the decoder to q_ϕ, we apply the reparameterization trick to sample z from q_ϕ with $g(\epsilon, q) = \mu_q + \sigma_q * \epsilon$. Here, $\epsilon \sim \mathcal{N}(0,1)$ is Gaussian noise.

The decoder computes the probability $p_\theta(y|x,z,l_y)$ of outputting target tokens y given the latent variables sampled from $q_\phi(z|x,y)$. The computational graph of the



$\bar{z}_1$ $\bar{z}_2$ $\bar{z}_3$ $\bar{z}_4$ $\bar{z}_5$
w_i^2
z_1 z_2 z_3 z_4 z_5 z_6 z_7

Figure 4.2: Illustration of the length transformation mechanism.

decoder is also similar to the Transformer decoder without using causal mask. To combine the information from the source tokens, we reuse the encoder vector representation created when computing the prior.

Length Prediction and Transformation Given a latent variable z sampled from the approximate posterior q_ϕ, we train a length prediction model $p_\theta(l_y|z)$. We train the model to predict the length difference between $|y|$ and $|x|$. In our implementation, $p_\theta(l_y|z)$ is modeled as a categorical distribution that covers the length difference in the range $[-50, 50]$. The prediction is produced by applying softmax after a linear transformation.

As the latent variable $z \sim q_\phi(z|x, y)$ has the length $|x|$, we need to transform the latent variables into l_y vectors for the decoder to predict target tokens. We use a monotonic location-based attention for this purpose, which is illustrated in Fig. 4.2. Let the resulting vectors of length transformation be $\bar{z}_1, ..., \bar{z}_{l_y}$. we produce each vector with

$$\begin{aligned}\bar{z}_j =& \sum_{k=1}^{|x|} w_k^j z_k,\\ w_k^j =& \frac{\exp(a_k^j)}{\sum_{k'=1}^{|x|}\exp(a_{k'}^j)},\\ a_k^j =& -\frac{1}{2\sigma^2}(k - \frac{|x|}{l_y} j)^2,\end{aligned}$$

where each transformed vector is a weighted sum of the latent variables. The weight is computed with a softmax over distance-based logits. We give higher weights to the latent variables close to the location $\frac{|x|}{l_y} j$. The scale σ is the only trainable parameter in this monotonic attention mechanism.

4.3.3 Training

If we train a model with the objective function in Eq. (4.4), the KL divergence often drops to zero from the beginning. This yields a degenerate model that does not use the latent variables at all. This is a well-known issue in variational inference called posterior collapse [Bowman et al., 2016; Dieng et al., 2019; Razavi et al., 2019]. We use two techniques to address this issue. Similarly to Kingma et al. [2016], we give a budget to the KL term as

$$\sum_{k=1}^{|x|} \max(b, \mathrm{KL}\big[q_\phi(z_k|x, y)||p_\omega(z_k|x)\big]),$$

where b is the budget of KL divergence for each latent variable. Once the KL value drops below b, it will not be minimized anymore, thereby letting the optimizer focus on the reconstruction term in the original objective function. As b is a critical hyperparameter, it is time-consuming to search for a good budget value. Here, we use the following annealing schedule to gradually lower the budget:

$$b = \begin{cases} 1, & \text{if } s < M/2 \\ \frac{(M-s)}{M/2}, & \text{otherwise} \end{cases}$$

s is the current step in training, and M is the maximum step. In the first half of the training, the budget b remains 1. In the second half of the training, we anneal b until it reaches 0.

Similarly to previous work on non-autoregressive NMT, we apply sequence-level knowledge distillation [Kim and Rush, 2016] where we use the output from an autoregressive model as target for our non-autoregressive model.

4.4 Inference with a Delta Posterior

Once the training has converged, we use an inference algorithm to find a translation y that maximizes the lowerbound in Eq. (4.2):

$$\text{argmax}_y \mathbb{E}_{z \sim q_\phi} \big[\log p_\theta(y|x,z) \big] - \text{KL}\big[q_\phi(z|x,y) || p_\omega(z|x) \big]$$

It is intractable to solve this problem exactly due to the intractability of computing the first expectation. We avoid this issue in the training time by reparametrization-based Monte Carlo approximation. However, it is desirable to avoid stochasticity at inference time where our goal is to present a single most likely target sentence given a source sentence.

We tackle this problem by introducing a proxy distribution $r(z)$ defined as

$$r(z) = \begin{cases} 1, & \text{if } z = \mu \\ 0, & \text{otherwise} \end{cases}$$

This is a Dirac measure, and we call it a *delta posterior* in our work. We set this delta posterior to minimize the KL divergence against the approximate posterior q_ϕ, which is equivalent to

$$\nabla_\mu \log q_\phi(\mu|x, y) = 0 \Leftrightarrow \mu = \mathbb{E}_{q_\phi}\left[z\right]. \tag{4.5}$$

We then use this proxy instead of the original approximate posterior to obtain a *deterministic lowerbound*:

$$\hat{\mathcal{L}}(\omega, \theta, \mu) = \log p_\theta(y|x, z = \mu) - \log p_\omega(\mu|x).$$

As the second term is constant with respect to y, maximizing this lowerbound with respect to y reduces to

$$\text{argmax}_y \log p_\theta(y|x, z = \mu), \tag{4.6}$$

which can be approximately solved by beam search when p_θ is an autoregressive sequence model. If p_θ factorizes over the sequence y, as in our non-autoregressive model, we can solve it exactly by

$$\hat{y}_i = \text{argmax}_{y_i} \log p_\theta(y_i|x, z = \mu).$$

With every estimation of y, the approximate posterior q changes. We thus alternate between fitting the delta posterior in Eq. (4.5) and finding the most likely sequence y in Eq. (4.6).

Algorithm 1 Deterministic Iterative Inference

```
Inputs:
    $x$ : source sentence
    $T$ : maximum step
$\mu_0 = \mathbb{E}_{p_\omega(z|x)}[z]$
$y_0 = \text{argmax}_y \log p_\theta(y|x, z = \mu_0)$
for $t \leftarrow 1$ to $T$ do
    $\mu_t = \mathbb{E}_{q_\phi(z|x,y_{t-1})}[z]$
    $y_t = \text{argmax}_y \log p_\theta(y|x, z = \mu_t)$
    if $y_t = y_{t-1}$ then
        break
output $y_t$
```

We initialize the delta posterior r using the prior distribution:

$$\mu = \mathbb{E}_{p_\omega(z|x)}[z] .$$

With this initialization, the proposed inference algorithm is fully deterministic. The complete inference algorithm for obtaining the final translation is shown in Algorithm 1.

4.5 Related Work

This work is inspired by a recent line of work in non-autoregressive NMT. Gu et al. [2018a] first proposed a non-autoregressive framework by modeling word alignment as a latent variable, which has since then been improved by Wang et al. [2019]. Lee et al. [2018] proposed a deterministic iterative refinement algorithm where a decoder is trained to refine the hypotheses. Our approach is most related to Kaiser et al. [2018]; Roy et al. [2018]. In both works, a discrete autoencoder is first trained on the target sentence, then an autoregressive prior is trained to predict the discrete latent variables given the source sentence. Our work is different from them in three ways: (1) we use continuous latent variables and train the approximate posterior $q(z|x,y)$ and the prior $p(z|x)$ jointly; (2) we use a non-autoregressive prior; and (3) the refinement is performed in the latent space, as opposed to discrete output space (as done in most previous works using refinement for non-autoregressive machine translation).

Concurrently to our work, Ghazvininejad et al. [2019] proposed to translate with a masked-prediction language model by iterative replacing tokens with low confidence. Gu et al. [2019a]; Stern et al. [2019]; Welleck et al. [2019] proposed insertion-based NMT models that insert words to the translations with a specific strategy. Unlike these works, our approach performs refinements in the low-dimensional latent space, rather than in the high-dimensional discrete space.

Similarly to our latent-variable model, Zhang et al. [2016] proposed a variational NMT, and Shah and Barber [2018] and Eikema and Aziz [2018] models the joint distribution of source and target. Both of them use autoregressive models. Shah and Barber [2018] designed an EM-like algorithm similar to Markov sampling Arulkumaran et al. [2017]. In contrast, we propose a deterministic algorithm to remove any non-determinism during inference.

4.6 Experimental Settings

Data and preprocessing We evaluate our model on two machine translation datasets: ASPEC Ja-En [Nakazawa et al., 2016] and WMT'14 En-De [Bojar et al., 2014]. The ASPEC dataset contains 3M sentence pairs, and the WMT'14 dataset contains 4.5M senence pairs.

To preprocess the ASPEC dataset, we use Moses toolkit [Koehn et al., 2007] to

	ASPEC Ja-En		
	BLEU(%)	speedup	wall-clock (std)
Base Transformer, beam size=3	27.1	1x	415ms (159)
Base Transformer, beam size=1	24.6	1.1x	375ms (150)
Latent-Variable NAR Model	13.3	17.0x	24ms (2)
+ knowledge distillation	25.2	17.0x	24ms (2)
+ deterministic inference	27.5	8.6x	48ms (2)
+ latent search	28.3	4.8x	86ms (2)

Table 4.1: Comparison of the proposed non-autoregressive (NAR) models with the autoregressive baselines. Our implementation of the Base Transformer is 1.0 BLEU point lower than the original paper Vaswani et al. [2017] on WMT'14 dataset.

	WMT'14 En-De		
	BLEU(%)	speedup	wall-clock (std)
Base Transformer, beam size=3	26.1	1x	602ms (274)
Base Transformer, beam size=1	25.6	1.3x	461ms (219)
Latent-Variable NAR Model	11.8	22.2x	27ms (1)
+ knowledge distillation	22.2	22.2x	27ms (1)
+ deterministic inference	24.1	12.5x	48ms (8)
+ latent search	25.1	6.8x	88ms (8)

Table 4.2: Comparison of the proposed non-autoregressive (NAR) models with the autoregressive baselines. Our implementation of the Base Transformer is 1.0 BLEU point lower than the original paper Vaswani et al. [2017] on WMT'14 dataset.

tokenize the English sentences, and Kytea [Neubig et al., 2011] for Japanese sentences. We further apply byte-pair encoding [Sennrich et al., 2016b] to segment the training sentences into subwords. The resulting vocabulary has 40K unique tokens on each side of the language pair. To preprocess the WMT'14 dataset, we apply sentencepiece Kudo and Richardson [2018a] to both languages to segment the corpus into subwords and build a joint vocabulary. The final vocabulary size is 32K for each language.

Learning To train the proposed non-autoregressive models, we adapt the same learning rate annealing schedule as the Base Transformer. Model hyperparameters are selected based on the validation ELBO value.

The only new hyperparameter in the proposed model is the dimension of each latent variable. If each latent is a high-dimension vector, although it has a higher capacity, the KL divergence in Eq. (4.2) becomes difficult to minimize. In practice, we found that latent dimensionality values between 4 and 32 result in similar performance. However, when the dimensionality is significantly higher or lower, we observed a performance drop. In all experiments, we set the latent dimensionionality to 8. We use a hidden size of 512 and feedforward filter size of 2048 for all models in our experiments. We use 6 transformer layers for the prior and the decoder, and 3 transformer layers for the approximate posterior.

Evaluation We evaluate the *tokenized BLEU* for ASPEC Ja-En datset. For WMT'14 En-De datset, we use SacreBLEU Post [2018] to evaluate the translation results. We follow Lee et al. [2018] to remove repetitions from the translation results before evaluating BLEU scores.

Latent Search To further exploit the parallelizability of GPUs, we sample multiple initial latent variables from the prior $p_\omega(z|x)$. Then we perform the deterministic inference on each latent variable to obtain a list of candidate translations. However, we can not afford to evaluate each candidate using Eq. (4.4), which requires importance sampling on q_ϕ. Instead, we use the autoregressive baseline model to score all the candidates, and pick the candidate with the highest log probability. Following Parmar et al. [2018], we reduce the temperature by a factor of 0.5 when sampling latent variables, resulting in better translation quality. To avoid stochasticity, we fix the random seed during sampling.

4.7 Result and Analysis

4.7.1 Quantitative Analysis

Our quantitative results on both datasets are presented in Table 4.1, 4.2. The baseline model in our experiments is a base Transformer. Our implementation of the autoregressive baseline is 1.0 BLEU points lower than the original paper Vaswani et al. [2017] on WMT'14 En-De datase. We measure the latency of decoding each sentence on a single NVIDIA V100 GPU for all models, which is averaged over all test samples.

As shown in Table 4.1, 4.2, without knowledge distillation, we observe a significant gap in translation quality compared to the autoregressive baseline. This observation is in line with previous works on non-autoregressive NMT Gu et al. [2018a]; Lee et al. [2018]; Wang et al. [2019]. The gap is significantly reduced by using knowledge distillation, as translation targets provided by the autoregressive model are easier to predict.

With the proposed deterministic inference algorithm, we significantly improve translation quality by 2.3 BLEU points on ASPEC Ja-En dataset and 1.9 BLEU points on WMT'14 En-De dataset. Here, we only run the algorithm for one step. We observe gain on ELBO by running more iterative steps, which is however not reflected by the BLEU scores. As a result, we outperform the autoregressive baseline on ASPEC dataset with a speedup of 8.6x. For WMT'14 dataset, although the proposed model reaches a speedup of 12.5x, the gap with the autoregressive baseline still remains, at 2.0 BLEU points. We conjecture that WMT'14 En-De is more difficult for our non-autoregressive model as it contains a high degree of noise [Ott et al., 2018a].

By searching over multiple initial latent variables and rescoring with the teacher Transformer model, we observe an increase in performance by $0.7 \sim 1.0$ BLEU score at the cost of lower translation speed. In our experiments, we sample 50 candidate latent variables and decode them in parallel. The slowdown is mainly caused by rescoring. With the help of rescoring, our final model further narrows the performance gap with the autoregressive baseline to 1.0 BLEU with 6.8x speedup on WMT'14 En-De task.

4.7.2 Non-autoregressive NMT Models

In Table 4.3, we list the results on WMT'14 En-De by existing non-autoregressive NMT approaches. All the models use Transformer as their autoregressive baselines. In comparison, our proposed model suffers a drop of 1.0 BLEU points over the baseline, which is a relatively small gap among the existing models. Thanks to the rapid convergence of the proposed deterministic inference algorithm, our model achieves a higher speed-up compared to other refinement-based models and provides a better speed-accuracy tradeoff.

Concurrently to our work, the mask-prediction language model Ghazvininejad et al. [2019] was found to reduce the performance gap down to 0.9 BLEU on WMT'14 En-De while still maintaining a reasonable speed-up. The main difference is that we update a delta posterior over latent variables instead of target tokens. Both Ghazvininejad et al. [2019] and Wang et al. [2019] with autoregressive rescoring decode multiple candidates in batch and choose one final translation from them. FlowSeq Ma et al. [2019] is an

	BLEU(%)	speed-up
Transformer Vaswani et al. [2017]	27.1	-
Baseline Gu et al. [2018a]	23.4	1x
NAT (+FT +NPD S=100)	19.1 (-4.3)	2.3x
Baseline Lee et al. [2018]	24.5	1x
Adaptive NAR Model	21.5 (-3.0)	1.9x
Baseline Kaiser et al. [2018]	23.5	1x
LT, Improved Semhash	19.8 (-3.7)	3.8x
Baseline Wang et al. [2019]	27.3	1x
NAT-REG, no rescoring	20.6 (-6.7)	27.6x*
NAT-REG, autoregressive rescoring	24.6 (-2.7)	15.1x*
Baseline Ghazvininejad et al. [2019]	27.8	1x
CMLM with 4 iterations	26.0 (-1.8)	-
CMLM with 10 iterations	26.9 (-0.9)	2$\sim$3x
Baseline Ma et al. [2019]	27.1	-
FlowSeq-large (NPD n = 30)	25.3 (-1.8)	-
Baseline (Ours)	26.1	1x
NAR with deterministic Inference	24.1 (-2.0)	12.5x
+ latent search	25.1 (-1.0)	6.8x

Table 4.3: A comparison of non-autoregressive NMT models on WMT'14 En-De dataset in BLEU(%) and decoding speed-up. $\star$ measured on IWSLT'14 DE-EN dataset.

recent interesting work on flow-based prior. With noisy parallel decoding, FlowSeq can be fairly compared to the latent search setting of our model. In Table 4.3, we can see that our model is equivalently or more effective without a flow-based prior. It is intriguing to see a combination with the flow approach.

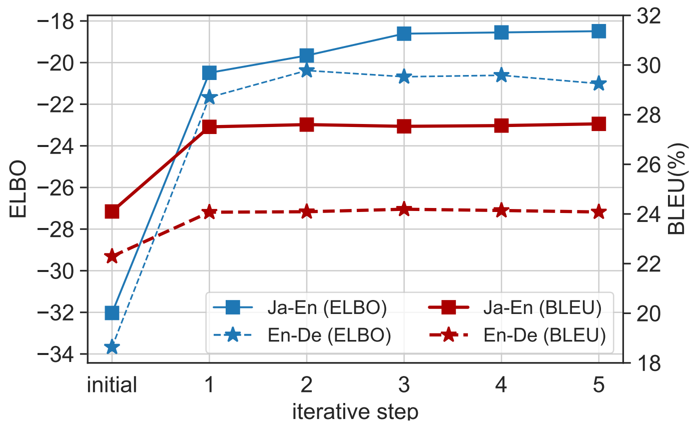


Figure 4.3: ELBO and BLEU scores measured with the target predictions obtained at each inference step for ASPEC Ja-En and WMT'14 En-De datasets.

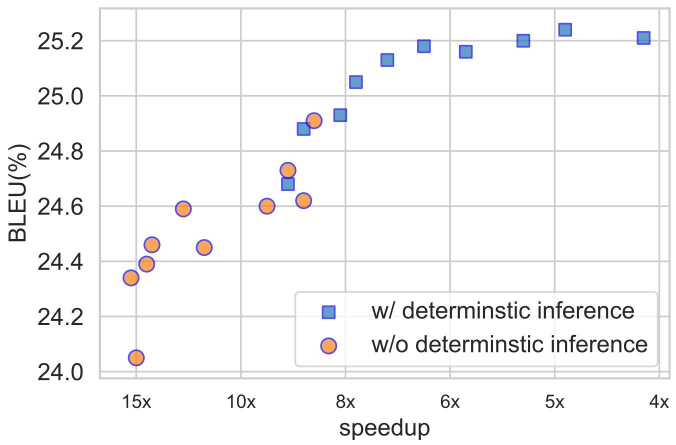


Figure 4.4: Trade-off between BLEU scores and speedup on WMT'14 En-De task by varying the number of candidates computed in parallel from 10 to 100.

4.7.3 Analysis of Deterministic Inference

Convergences of ELBO and BLEU In this section, we empirically show that the proposed deterministic iterative inference improves the ELBO in Eq. (4.2). As the ELBO is a function of x and y, we measure the ELBO value with the new target prediction after each iteration during inference. For each instance, we sample 20 latent variables to compute the expectation in Eq. (4.2). The ELBO value is further averaged over data samples.

In Fig. 4.3, we show the ELBO value and the resulting BLEU scores for both datasets. In the initial step, the delta posterior is initialized with the prior distribution $p_\omega(z|x)$. We see that the ELBO value increases rapidly with each refinement step, which means a higher lowerbound to $\log p(y|x)$. The improvement is highly correlated with increasing BLEU scores. For around 80% of the data samples, the algorithm converges within three steps. We observe the BLEU scores peaked after only one refinement step.

Trade-off between Quality and Speed In Fig. 4.4, we show the trade-off between translation quality and the speed gain on WMT'14 En-De task when considering multiple candidates latent variables in parallel. We vary the number of candidates from 10 to 100, and report BLEU scores and relative speed gains in the scatter plot. The results are divided into two groups. The first group of experiments search over multiple latent variables and rescore with the teacher Transformer. The second group applies the proposed deterministic inference before rescoring.

We observe that the proposed deterministic inference consistently improves translation quality in all settings. The BLEU score peaks at 25.2. As GPUs excel at processing massive computations in parallel, we can see that the translation speed only degrades by a small magnitude.

Example 1: Sequence modified without changing length

Source	hyouki gensuiryou hyoujun no kakuritsu wo kokoromita. (JA)
Reference	the establishment of an optical fiber attenuation standard ...
Initial	an attempt was made establish establish damping attenuation ...
Refined	an attempt was to establish the damping attenuation standard ...

Example 2: One word removed from the sequence

Source	...``sen bouchou keisu no toriatsukai'' nitsuite nobeta. (JA)
Reference	... handling of linear expansion coefficient .
Initial	... `` handling of of linear expansion coefficient '' are ...
Refined	... `` handling of linear expansion coefficient '' are ...

Example 3: Four words added to the sequence

Source	... maikuro manipyureshon heto hatten shite kite ori ...(JA)
Reference	... with wide application fields so that it has been ...
Initial	... micro micro manipulation and ...
Refined	... and micro manipulation , and it has been developed , ...

Table 4.4: Ja-En sample translation with the proposed iterative inference algorithm. In the first example, the initial guess is refined without a change in length. In the last two examples, the iterative inference algorithm changes the target length along with its content. This is more pronounced in the last example, where a whole clause is inserted during refinement.

4.7.4 Qualitative Analysis

We present some translation examples to demonstrate the effect of the proposed iterative inference in Table 4.4. In Example 1, the length of the target sequence does not change but only the tokens are replaced over the refinement iterations. The second and third examples show that the algorithm removes or inserts words during the iterative inference by adaptively changing the target length. Such a significant modification to the predicted sequence mostly happens when translating long sentences.

For some test examples, however, we still find duplicated words in the final translation after applying the proposed deterministic inference. For them, we notice that the quality of the initial guess of translation is considerably worse than average, which typically contains multiple duplicated words. Thus, a high-quality initial guess is crucial for obtaining good translations.

4.8 Conclusion

Our work presents the first approach to use continuous latent variables for non-autoregressive Neural Machine Translation. The key idea is to introduce a sequence of latent variables to capture the uncertainly in the target sentence. The number of latent vectors is always identical to the number of input tokens. A length transformation mechanism is then applied to adapt the latent vectors to match the target length. We train the proposed model by maximizing the lowerbound of the log-probability $\log p(y|x)$.

We then introduce a deterministic inference algorithm that uses a *delta posterior* over the latent variables. The algorithm alternates between updating the delta posterior and the target tokens. Our experiments show that the algorithm is able to improve the evidence lowerbound of predicted target sequence rapidly. In our experiments, the BLEU scores converge in one refinement step.

Our non-autoregressive NMT model closes the performance gap with autoregressive baseline on ASPEC Ja-En task with a 8.6x speedup. By decoding multiple latent variables sampled from the prior, our model brings down the gap on En-De task down to 1.0 BLEU with a speedup of 6.8x.

4.9 Developments Since Publication

Following our publication, Ma et al. [2019] proposed a continuous latent variable model for non-autoregressive NMT that employs a normalizing flow prior. Across several published non-autoregressive NMT models including the latent variable models, Zhou et al. [2019] investigated the cause of the importance of knowledge distillation [Kim and Rush, 2016] and discovered a strong correlation between model capacity and the complexity of the distilled data (measured by, e.g. conditional entropy). Outside of machine translation, conditional latent variable models have been proposed for dialogue generation [Han et al., 2020], controllable story generation [Fang et al., 2021] and non-autoregressive speech synthesis [Elias et al., 2020].

Chapter 5

Iterative Refinement in the Continuous Space for Non-Autoregressive Neural Machine Translation

We propose an efficient inference procedure for non-autoregressive machine translation that iteratively refines translation purely in the continuous space. Given a continuous latent variable model for machine translation [Shu et al., 2020], we train an inference network to approximate the gradient of the marginal log probability of the target sentence, using only the latent variable as input. This allows us to use gradient-based optimization to find the target sentence at inference time that approximately maximizes its marginal probability. As each refinement step only involves computation in the latent space of low dimensionality (we use 8 in our experiments), we avoid computational overhead incurred by existing non-autoregressive inference procedures that often refine in token space. We compare our approach to a recently proposed EM-like inference procedure [Shu et al., 2020] that optimizes in a hybrid space, consisting of both discrete and continuous variables. We evaluate our approach on WMT'14 En$\rightarrow$De, WMT'16 Ro$\rightarrow$En and IWSLT'16 De$\rightarrow$En, and observe two advantages over the EM-like inference: (1) it is computationally efficient, i.e. each refinement step is twice as fast, and (2) it is more effective, resulting in higher marginal probabilities and BLEU scores with the same number of refinement steps. On WMT'14 En$\rightarrow$De, for instance, our approach is able to decode 6.2 times faster than the autoregressive model with minimal degradation to translation quality (0.9 BLEU).

5.1 Introduction

Most neural machine translation systems are autoregressive, hence decoding latency grows linearly with respect to the length of the target sentence. For faster generation, several work proposed non-autoregressive models with sub-linear decoding latency given sufficient parallel computation [Gu et al., 2018a; Kaiser et al., 2018; Lee et al., 2018].

As it is challenging to precisely model the dependencies among the tokens without autoregression, many existing non-autoregressive models first generate an initial translation which is then iteratively refined to yield better output [Ghazvininejad et al., 2019; Gu et al., 2019a; Lee et al., 2018]. While various training objectives are used to admit refinement (e.g. denoising, evidence lowerbound maximization and mask language modeling), the generation process of these models is similar in that the refinement process happens in the *discrete* space of sentences.

Meanwhile, another line of work proposed to use *continuous* latent variables for non-autoregressive translation, such that the distribution of the target sentences can be factorized over time given the latent variables [Ma et al., 2019; Shu et al., 2020]. Unlike the models discussed above, finding the most likely target sentence under these models requires searching over continuous latent variables. To this end, Shu et al. [2020] proposed an EM-like inference procedure that optimizes over a hybrid space consisting of both continuous and discrete variables. By introducing a deterministic delta posterior, it maximizes a proxy lowerbound by alternating between matching the delta posterior to the original approximate posterior (continuous optimization), and finding a target sentence that maximizes the proxy lowerbound (discrete search).

In this work, we propose an iterative inference procedure for latent variable non-autoregressive models that purely operates in the continuous space.[1] Given a latent variable model, we train an inference network to estimate the gradient of the marginal log probability of the target sentence, using only the latent variable as input. At inference time, we find the target sentence that approximately maximizes the log probability by (1) initializing the latent variable e.g. as the mean of the prior, and (2) following the gradients estimated by the inference network.

We compare the proposed approach with the EM-like inference [Shu et al., 2020] on three machine translation datasets: WMT'14 En$\rightarrow$De, WMT'16 Ro$\rightarrow$En and IWSLT'16 De$\rightarrow$En. The advantages of our approach are twofold: (1) each refinement step is twice as fast, as it avoids discrete search over a large vocabulary, and (2) it is more effective, giving higher marginal probabilities and BLEU scores with the same number of refinement steps. Our procedure results in significantly faster inference, for instance giving $6.2\times$ speedup over the autoregressive baseline on WMT'14 En$\rightarrow$De at the expense of

1. We open source our code at `https://github.com/zomux/lanmt-ebm`

0.9 BLEU score.

5.2 Background: Iterative Refinement for Non-Autoregressive Translation

We motivate our approach by reviewing existing refinement-based non-autoregressive models for machine translation in terms of their inference procedure. Let us use V, D, T and L to denote vocabulary size, latent dimensionality, target sentence length and the number of refinement steps, respectively.

Most machine translation models are trained to maximize the conditional log probability $\log p(\mathbf{y}|\mathbf{x})$ of the target sentence $\mathbf{y}$ given the source sentence $\mathbf{x}$, averaged over the training data consisting of sentence pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. To find the most likely target sentence at test time, one performs maximum-a-posteriori inference by solving a search problem $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}} \log p(\mathbf{y}|\mathbf{x})$.

5.2.1 Refinement in a Discrete Space

As the lack of autoregression makes it challenging to model the dependencies among the target tokens, most of the existing non-autoregressive translation models use iterative refinement to impose dependencies in the generation process. Various training objectives are used to incorporate refinement, e.g. denoising [Lee et al., 2018], mask language modeling [Ghazvininejad et al., 2019] and evidence lowerbound maximization [Chan et al., 2019; Gu et al., 2019a]. However, inference procedures employed by these models are similar in that an initial hypothesis is generated and then successively refined. We refer the readers to [Mansimov et al., 2019b] for a formal definition of a sequence generation framework that unifies these models, and briefly discuss the inference procedure below.

By viewing each refinement step as introducing a discrete random variable $\mathbf{z}_i$ (a $T \times V$-dimensional matrix, where each row is one-hot), inference with L refinement steps requires finding $\mathbf{y}$ that maximizes the log probability $\log p(\mathbf{y}|\mathbf{x})$.

$$\begin{aligned}
\log p_\theta(\mathbf{y}|\mathbf{x}) &= \log \sum_{\mathbf{z}_{1:L}} p_\theta(\mathbf{y}, \mathbf{z}_{1:L}|\mathbf{x}) \\
&= \log \sum_{\mathbf{z}_{1:L}} \Big(p_\theta(\mathbf{y}|\mathbf{z}_{1:L}, \mathbf{x}) \cdot \prod_{i=1}^{L} p_\theta(\mathbf{z}_i|\mathbf{z}_{<i}, \mathbf{x}) \Big) \\
&\geq \sum_{\mathbf{z}_{1:L}} \Big(\log p_\theta(\mathbf{y}|\mathbf{z}_{1:L}, \mathbf{x}) + \sum_{i=1}^{L} \log p_\theta(\mathbf{z}_i|\mathbf{z}_{<i}, \mathbf{x}) \Big).
\end{aligned}$$

As the marginalization over $\mathbf{z}_{1:L}$ is intractable, inference for these models instead maximize the log joint probability with respect to $\hat{\mathbf{z}}_{1:L}$ and $\mathbf{y}$:

$$\log p_\theta(\mathbf{y}|\hat{\mathbf{z}}_{1:L}, \mathbf{x}) + \sum_{i=1}^{L} \log p_\theta(\hat{\mathbf{z}}_i|\hat{\mathbf{z}}_{<i}, \mathbf{x}).$$

Approximate search methods are used to find $\hat{\mathbf{z}}_{1:L}$ as $\hat{\mathbf{z}}_i = \operatorname{argmax}_{\mathbf{z}_i} \log p_\theta(\mathbf{z}_i|\hat{\mathbf{z}}_{<i}, \mathbf{x})$.

5.2.2 Refinement in a Hybrid Space

Learning On the other hand, Ma et al. [2019]; Shu et al. [2020] proposed to use *continuous* latent variables for non-autoregressive translation. By letting the latent variables $\mathbf{z}$ (of dimensionality $T \times D$) capture the dependencies between the target tokens, the decoder $p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ can be factorized over time. As exact posterior inference and learning is intractable for most deep parameterized prior and decoder distributions, these models are trained to maximize the evidence lowerbound (ELBO) [Kingma and Welling, 2014a; Wainwright and Jordan, 2008].

$$\log p_\theta(\mathbf{y}|\mathbf{x}) \geq \mathop{\mathbb{E}}_{\mathbf{z}\sim q_\phi} \left[\log \frac{p_\theta(\mathbf{y}, \mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})} \right]$$

Inference Exact maximization of ELBO with respect to $\mathbf{y}$ is challenging due to the expectation over $\mathbf{z} \sim q_\phi$. To approximately maximize the ELBO, Shu et al. [2020] proposed to optimize a deterministic proxy lowerbound using a Dirac delta posterior:

$$\delta(\mathbf{z}|\boldsymbol{\mu}) = \mathbb{1}_{\boldsymbol{\mu}}(\mathbf{z})$$

Then, the ELBO reduces to the following proxy lowerbound:

$$\begin{aligned} &\mathop{\mathbb{E}}_{\mathbf{z}\sim\delta(\mathbf{z}|\boldsymbol{\mu})} \Big[p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x}) + p_\theta(\mathbf{z}|\mathbf{x}) \Big] + \overbrace{\mathcal{H}(\delta)}^{=0}, \\ &= \log p_\theta(\mathbf{y}|\boldsymbol{\mu}, \mathbf{x}) + \log p_\theta(\boldsymbol{\mu}|\mathbf{x}). \end{aligned}$$

Shu et al. [2020] proposed to approximately maximize the ELBO with an EM-like inference procedure, to which we refer as *delta inference*. It alternates between continuous and discrete optimization: (1) E-step matches the delta posterior with the approximate posterior by minimizing their KL divergence: $\boldsymbol{\mu}_i = \text{argmin}_{\boldsymbol{\mu}} \text{KL}\big[\, \delta(\mathbf{z}|\boldsymbol{\mu}) \,\big\|\, q_\phi(\mathbf{z}|\hat{\mathbf{y}}_{i-1}, \mathbf{x}) \,\big]$, and (2) M-step maximizes the proxy lowerbound with respect to $\mathbf{y}$: $\hat{\mathbf{y}}_i = \text{argmax}_{\mathbf{y}} \log p_\theta(\mathbf{y}|\boldsymbol{\mu}_i, \mathbf{x})$. Overall, delta inference finds $\mathbf{y}$ and $\boldsymbol{\mu}$ that maximizes $\log p_\theta(\mathbf{y}|\boldsymbol{\mu}, \mathbf{x}) + \log q_\phi(\boldsymbol{\mu}|\mathbf{y}, \mathbf{x})$. This iterative inference procedure in hybrid space was empirically shown to result in improved BLEU scores and ELBO on each refinement step [Shu et al., 2020].

5.3 Iterative Refinement in a Continuous Space

While the delta inference procedure is an effective inference algorithm for machine translation models with continuous latent variables, it is unsatisfactory as the M-step requires searching over V tokens T times for each refinement step. As V is large for most machine translation models, this is an expensive operation, even when the T searches can be parallelized. We thus propose to replace the delta inference with continuous optimization in the latent space only, given the underlying latent variable model.

5.3.1 Learning

Let us define $\tau_\theta(\mathbf{z};\mathbf{x})$ as the marginal log probability of the most likely target sentence under the latent variable model given $\mathbf{z}$.

$$\tau_\theta(\mathbf{z};\mathbf{x}) = \log p_\theta(\hat{\mathbf{y}}|\mathbf{x}), \tag{5.1}$$

where $\hat{\mathbf{y}} = \text{argmax}_\mathbf{y} \log p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})$. Our goal is to find a function $-E_\psi(\mathbf{z};\mathbf{x})$ that approximates $\tau_\theta(\mathbf{z};\mathbf{x})$ up to an additive constant and a positive multiplicative factor, such that

$$\text{argmin}_\mathbf{z}\big(E_\psi(\mathbf{z};\mathbf{x})\big) \approx \text{argmax}_\mathbf{z}\big(\tau_\theta(\mathbf{z};\mathbf{x})\big).$$

In this work, instead of directly approximating τ_θ, we train $-E_\psi$ to learn the difference of τ_θ between a pair of configurations of latent variables. Omitting the source sentence $\mathbf{x}$ and the model parameters θ for notational simplicity, we solve the following problem for $\mathbf{z} \neq \bar{\mathbf{z}}$:

$$
\begin{aligned}
&\min_{\psi} \left\| \Big(- E_{\psi}(\bar{\mathbf{z}}) + E_{\psi}(\mathbf{z}) \Big) - \Big(\tau(\bar{\mathbf{z}}) - \tau(\mathbf{z}) \Big) \right\|^2 && (5.2) \\
\approx &\min_{\psi} \left\| \Big((\bar{\mathbf{z}} - \mathbf{z})^\intercal \cdot \nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) \Big) \right. \\
&\qquad \left. - \Big((\bar{\mathbf{z}} - \mathbf{z})^\intercal \cdot \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big) \right\|^2 && (5.3) \\
\approx &\min_{\psi} \left\| (\bar{\mathbf{z}} - \mathbf{z})^\intercal \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big) \right\|^2 \\
\approx &\min_{\psi} \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big)^\intercal (\bar{\mathbf{z}} - \mathbf{z}) \\
&\qquad (\bar{\mathbf{z}} - \mathbf{z})^\intercal \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big) \\
\approx &\min_{\psi} \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big)^\intercal \|\bar{\mathbf{z}} - \mathbf{z}\|^2 \\
&\qquad \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big) && (5.4) \\
\approx &\min_{\psi} \left\| \nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) - \nabla_{\mathbf{z}} \tau(\mathbf{z}) \right\|^2 \\
\approx &\min_{\psi} \left\| \nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) \right\|^2 + \left\| \nabla_{\mathbf{z}} \tau(\mathbf{z}) \right\|^2 \\
&\qquad - 2 \Big(\nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z}))^\intercal \cdot \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big) && (5.5) \\
\approx &\min_{\psi} \left\| \nabla_{\mathbf{z}} E_{\psi}(\mathbf{z}) \right\|^2 + 2 \Big(\nabla_{\mathbf{z}} E_{\psi}(\mathbf{z})^\intercal \cdot \nabla_{\mathbf{z}} \tau(\mathbf{z}) \Big)
\end{aligned}
$$

Eq. 5.3 follows from linear approximation, as

$$
\begin{aligned}
-\big(E_{\psi}(\bar{\mathbf{z}}) - E_{\psi}(\mathbf{z})\big) &\approx (\bar{\mathbf{z}} - \mathbf{z})^\intercal \cdot \nabla_{\mathbf{z}} (-E_{\psi}(\mathbf{z})) \\
\tau(\bar{\mathbf{z}}) - \tau(\mathbf{z}) &\approx (\bar{\mathbf{z}} - \mathbf{z})^\intercal \cdot \nabla_{\mathbf{z}} \tau(\mathbf{z})
\end{aligned}
$$

$\|\bar{\mathbf{z}} - \mathbf{z}\|^2$ in Eq. 5.4 can be eliminated, as dividing the objective with a positive constant does not change the solution. The second term in Eq. 5.5 is also a constant with respect to ψ, hence can be ignored. Intuitively, $\nabla_{\mathbf{z}}\big(- E_{\psi}(\mathbf{z}; \mathbf{x})\big)$ is trained to approximate $\nabla_{\mathbf{z}}\, \tau_{\theta}(\mathbf{z}; \mathbf{x})$, as Eq. 5.2 maximizes their dot product while minimizing its squared norm.

As $\tau_\theta(\mathbf{z};\mathbf{x})$ is not differentiable with respect to $\mathbf{z}$ due to the argmax operation in Eq. 5.1, $\nabla_{\mathbf{z}}\,\tau_\theta(\mathbf{z};\mathbf{x})$ is not defined. We thus use a proxy gradient from delta inference. Furthermore, we weigh the latent configuration $\mathbf{z}$ according to the prior. Our final training objective for E_ψ is then as follows:

$$\mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})}\left[\left\|\nabla_{\mathbf{z}}E_\psi(\mathbf{z};\mathbf{x})\right\|^2 + 2\left(\left(\nabla_{\mathbf{z}}E_\psi(\mathbf{z};\mathbf{x})\right)^\intercal \cdot (\widetilde{\mathbf{z}}-\mathbf{z})\right)\right], \tag{5.6}$$

where $\widetilde{\mathbf{z}}$ is the output of applying k steps of delta inference on $\mathbf{z}$. If delta inference improves the log probability at each iteration, we hypothesize that $(\widetilde{\mathbf{z}}-\mathbf{z})$ is a reasonable approximation to the true gradient $\nabla_{\mathbf{z}}\,\tau_\theta(\mathbf{z};\mathbf{x})$. We empirically show that this is indeed the case in Sec. 5.5.2.

5.3.2 Parameterization

We have two options for parameterizing $\nabla_{\mathbf{z}}E_\psi(\mathbf{z};\mathbf{x})$ when minimizing Eq. 5.6. First, we can parameterize it as the gradient of a scalar-valued function E, to which earlier work have referred as an *energy* function [LeCun et al., 2006; Teh et al., 2003]. Second, we can parameterize it as a function $S_\psi(\mathbf{z};\mathbf{x})$ that directly outputs the gradient of the log probability with respect to $\mathbf{z}$ (which is often referred to as a *score* function [Hyvärinen, 2005]), without estimating the energy directly.

While previous work found direct score estimation that bypasses energy estimation unstable [Alain and Bengio, 2014; Saremi et al., 2018], it leads to faster inference by avoiding backpropagation in each refinement step. We compare the two approaches in our experiments.

Algorithm 2 Inference for Latent Variable Models using Learned Gradients

```
Inputs:
    $\mathbf{x}$ : source sentence
    $\alpha$ : step size
    $\theta$ : latent variable model
    $\psi$ : inference module
$\mathbf{z} = \mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{x})}[\mathbf{z}]$
while termination condition not met, do
    $\mathbf{z} = \mathbf{z} - \alpha \cdot (\nabla_{\mathbf{z}} E_\psi(\mathbf{z}; \mathbf{x}))$
$\hat{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}} \log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$
output $\hat{\mathbf{y}}$
```

5.3.3 Inference

At inference time, we initialize the latent variable (e.g. using either a sample from the prior or its mean) and iteratively update the latent variable using the estimated gradients (see Alg. 2). As our inference procedure only involves optimization in the continuous space each step, we avoid having to search over a large vocabulary. We can either perform iterative refinement for a fixed number of steps, or until some convergence condition is satisfied.

5.4 Experimental Setup

5.4.1 Datasets and Preprocessing

We evaluate our approach on three widely used machine translation datasets: IWSLT'16 De$\rightarrow$En[2] (containing 197K training, 2K development and 2K test sentence pairs), WMT'16

2. `https://wit3.fbk.eu/`

Ro$\rightarrow$En[3] (612K, 2K, 2K pairs) and WMT'14 En$\rightarrow$De[4] (4.5M, 3K, 3K pairs).

We use sentencepiece tokenization [Kudo and Richardson, 2018b] with 32K sentencepieces on all datasets. For WMT'16 Ro$\rightarrow$En, we follow Sennrich et al. [2016a] and normalize Romanian and remove diacritics before applying tokenization. For training, we discard sentence pairs if either the source or the target length exceeds 64 tokens.

Following Lee et al. [2018], we remove repetitions from the translations with a simple postprocessing step before computing BLEU scores. We use detokenized BLEU with Sacrebleu [Post, 2018].

Distillation Following previous work on non-autoregressive translation, we train non-autoregressive models on the target sentences generated by an autoregressive model [Gu et al., 2018a; Kim and Rush, 2016] trained using the FairSeq framework [Ott et al., 2019].

5.4.2 Models and Baselines

Autoregressive baselines We use Transformers [Vaswani et al., 2017] with the following hyperparameters. For WMT'16 Ro$\rightarrow$En and WMT'14 En$\rightarrow$De, we use Transformer-base. For IWSLT'16 De$\rightarrow$En, we use a smaller model with $(d_{\text{model}}, d_{\text{filter}}, n_{\text{layers}}, n_{\text{heads}}) = (256, 1024, 5, 2)$.

Non-autoregressive latent variable models We closely follow the implementation details from [Shu et al., 2020]. The prior and the approximate posterior distributions are

3. `www.statmt.org/wmt16/translation-task.html`
4. `www.statmt.org/wmt14/translation-task.html`

spherical Gaussian distributions with learned mean and variance, and the decoder is factorized over time. The only difference is at inference time, the target sentence length is predicted once and fixed throughout the refinement procedure. Therefore, the latent variable dimensionality $\mathbb{R}^{T \times D}$ does not change.

The decoder, prior and approximate posterior distributions are all parameterized using n_{layers} Transformer decoder layers (the last two also have a final linear layer that outputs mean and variance). For IWSLT'16 De→En, we use $(d_{\text{model}}, d_{\text{filter}}, n_{\text{layers}}, n_{\text{heads}}) = (256, 1024, 3, 4)$. For WMT'14 En→De and WMT'16 Ro→En, we use $(512, 2048, 6, 8)$. The latent dimensionality d_{latent} is set to 8 across all datasets. The source sentence encoder is implemented with a standard Transformer encoder. Given the hidden states of the source sentence, the length predictor (a 2-layer MLP) predicts the length difference between the source and target sentences as a categorical distribution in $[-50, 50]$.

Energy function $E_\psi(\mathbf{z}; \mathbf{x})$ is parameterized with n_{layers} Transformer decoder layers and a final linear layer with the output dimensionality of 1. We average the last Transformer hidden states across time and feed it to a linear layer to yield a scalar energy value.

Score function When directly estimating the gradient of the log probability with respect to **z**, $S_\psi(\mathbf{z}; \mathbf{x})$ is parameterized with n_{layers} Transformer decoder layers and a final linear layer with the output dimensionality of d_{latent}.

5.4.3 Training and Optimization

We use the Adam optimizer [Kingma and Ba, 2015] with batch size of 8192 tokens and the learning rate schedule used by Vaswani et al. [2017] with warmup of 8K steps.

When training our inference networks, we fix the underlying latent variable model. Our inference networks are trained for 1M steps to minimize Eq. 5.6, where $\widetilde{\mathbf{z}}$ is obtained by applying $k(=4)$ iterations of delta inference on $\mathbf{z}$ sampled from the prior. We also find that stochastically applying one gradient update (using the estimated gradients) to $\mathbf{z}$ before computing $\widetilde{\mathbf{z}}$ leads to better performance.

5.4.4 Inference

Step size For the proposed inference procedure, we use the step size $\alpha = 1.0$ as it performed well on the development set.

Length prediction Given a distribution of target sentence length, we can either (1) take the argmax, or (2) select the top l candidates and decode them in parallel [Ghazvininejad et al., 2019]. In the second case, we select the output candidate with the highest log probability under an autoregressive model, normalized by its length.

Latent search In Alg. 2, we can either initialize the latent variable with a sample from the prior, or its mean. We use n_w samples from the prior and perform iterative refinement (e.g. delta inference or the proposed inference procedures) in parallel. Similarly to length prediction, we select the output with the highest log probability. To avoid stochasticity, we fix the random seed during sampling.

		WMT'14 En→De			WMT'16 Ro→En			IWSLT'16 De→En		
		Bleu	Speed	Time	Bleu	Speed	Time	Bleu	Speed	Time
AR	$b = 1$	27.5	1.1×	251 ±175	30.9	1.1×	511 ±560	31.1	1.1×	178 ±139
	$b = 4$	28.3	1×	291 ±194	31.5	1×	610 ±630	31.5	1×	210 ±161
NAR LVM Delta	$L = 0$	25.7	15×	19 ±1	28.4	34×	18 ±5	27.0	19×	11 ±5
	$L = 1$	26.1	6.3×	46 ±5	29.0	19×	32 ±5	28.3	11×	18 ±6
	$L = 2$	26.2	4.0×	72 ±3	29.1	14×	45 ±7	28.5	8.0×	26 ±7
	$L = 4$	26.1	2.8×	103 ±5	29.1	8.5×	72 ±5	28.6	5.2×	40 ±9
	Search	26.9	5.5×	63 ±8	30.3	13×	48 ±7	29.7	6.0×	35 ±6
NAR LVM Energy	$L = 0$	25.7	15×	19 ±1	28.4	34×	18 ±5	27.0	19×	11 ±5
	$L = 1$	26.1	5.8×	50 ±3	28.8	17×	36 ±5	28.6	9.5×	22 ±7
	$L = 2$	26.1	4.2×	69 ±4	28.9	11×	55 ±9	28.7	7.0×	30 ±9
	$L = 4$	26.0	2.5×	117 ±6	28.8	7.1×	85 ±5	28.8	4.5×	46 ±9
	Search	27.1	4.4×	66 ±9	30.4	12×	53 ±7	29.9	5.0×	42 ±7
NAR LVM Score	$L = 0$	25.7	15×	19 ±1	28.4	34×	18 ±5	27.0	19×	11 ±5
	$L = 1$	26.3	10×	29 ±2	29.1	24×	25 ±5	28.8	13×	16 ±6
	$L = 2$	26.3	7.6×	38 ±2	29.1	19×	32 ±6	29.0	10×	20 ±5
	$L = 4$	26.3	5.7×	51 ±4	29.1	14×	44 ±5	29.1	7.5×	28 ±5
	Search	**27.4**	**6.2**×	47 ±8	**30.4**	**15**×	41 ±6	**30.2**	**6.3**×	33 ±4

Table 5.1: Translation quality and inference speed of autoregressive baseline (AR) and several inference procedures for non-autoregressive latent variable model (NAR LVM): Delta inference (*Delta*) [Shu et al., 2020], the proposed inference procedure with estimated energy (*Energy*) or score (*Score*). *Speed*: inference speedup compared to the autoregressive model with beam width 4. *Time*: Average wall clock time per example in milliseconds on a Tesla V100 GPU (with standard deviations). b: beam width, L: the number of refinement steps. *Search*: parallel decoding with 5 length candidates and 5 samples from the prior, with 1 refinement step. Results above *Search* are obtained by initializing the latent variable as the mean of the prior. We boldface the highest BLEU among the latent variable models.

5.5 Quantitative Results

5.5.1 Translation Quality and Speed

Table 5.1 presents translation performance and inference speed of several inference procedures for the non-autoregressive latent variable models, along with the autoregressive baselines. We emphasize that the same underlying latent variable model is used across three different inference procedures (Delta, Energy, Score), to compare their efficiency and effectiveness.

Translation quality We observe that both of the proposed inference procedures result in improvements in translation quality with more refinement steps. For instance, 4 refinement steps using the learned score function improves BLEU by 2.1 on IWSLT'16 De→En. Among the proposed inference procedures, we find it more effective to use a learned score function, as it gives comparable or better performance to delta inference on all datasets. A learned energy function results in comparable performance to delta inference. Parallel decoding over multiple target length candidates and sampled latent variables leads to significant improvements in BLEU, resulting in 1 BLEU increase or more on all datasets. Similarly to delta inference, we find that the proposed iterative inference procedures converge quite quickly, and often 1 refinement step gives comparable translation quality to running 4 refinement steps.

Inference speed We observe that using a learned score function is significantly faster than delta inference: twice as fast on IWSLT'16 De→En and WMT'16 Ro→En and almost four times as fast on WMT'14 En→De. On WMT'14 En→De, the decoding

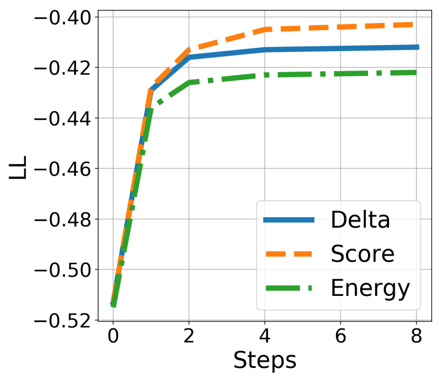


Figure 5.1: Marginal log probability $\log p_\theta(\hat{\mathbf{y}}|\mathbf{x})$ of output $\hat{\mathbf{y}}$ from each refinement step.

latency for 4 steps using the score is close to (within one standard deviation of) running 1 refinement step of delta inference. On the other hand, we find that using the learned energy function is slower, presumably due to the overhead from backpropagation. We find its wall clock time to be similar to delta inference. As the entire inference process can be parallelized, we find that parallel decoding with multiple length candidates and latent variable samples only incurs minimal overhead. Finally, we confirm that decoding latency for non-autoregressive models is indeed constant with respect to the sequence length (given parallel computation), as the standard deviation is small (< 10 ms) across test examples.

Overall result Overall, we find the proposed inference procedure using the learned score function highly effective and efficient. On WMT'14 En$\rightarrow$De, using 1 refinement step and parallel search leads to $6.2\times$ speedup over the autoregressive baseline with minimal degradation to translation quality (0.9 BLEU score).

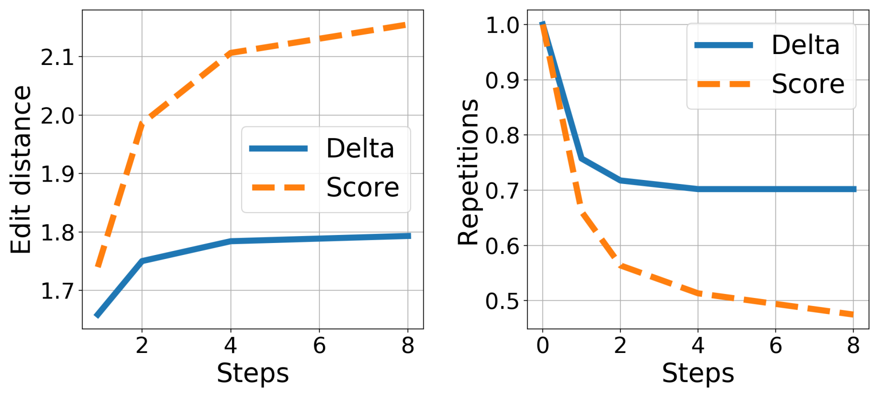


Figure 5.2: Edit distance from the first output (left) and the number of repetitions in the output (right) for $L = \{1, 2, 4, 8\}$ refinement steps for delta inference and inference using a learned score function.

5.5.2 Log Probability Comparison

In Fig 5.1, we report the marginal log probability $\log p_\theta(\hat{\mathbf{y}}|\mathbf{x})$ of $\hat{\mathbf{y}}$ found after L steps of each iterative inference procedure on IWSLT'16 De→En. We estimate the marginal log probability by importance sampling with 500 samples from the approximate posterior. We observe that the log probability improves with more refinement steps for all inference procedures (delta inference and the proposed procedures). We draw two conclusions from this. First, delta inference indeed increases log probability at each iteration. Second, the proposed optimization scheme increases the target objective function it was trained on (log probability).

5.5.3 Token Statistics

We compare delta inference and the proposed inference with a learned score function in terms of token statistics in the output translations on IWSLT'16 De→En. In Figure 5.2 (left), we compute the average edit distance (in sentencepieces) per test example from the

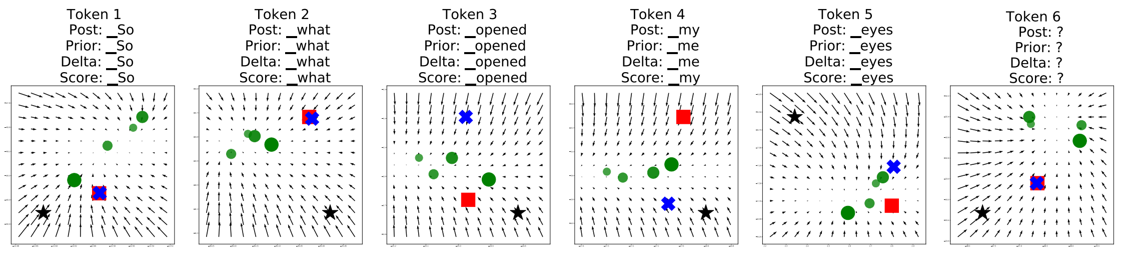


Figure 5.3: Visualization of estimated gradients and optimization trajectory. Above each plot are tokens predicted from the following latent variables: (1) approximate posterior mean, (2) prior mean, (3) delta inference and (4) inference with the learned score. **Black star**: latent variable before refinement (prior mean). **Blue cross**: latent variables after $L = \{1, 2, 3, 4\}$ steps of delta inference (collapsed into a single point). **Green circle**: latent variables after L steps of inference with a learned score function. Marker size decreases with successive refinement steps. **Red square**: approximate posterior mean.

initial output (mean of the prior). It is clear that each refinement step using a learned score function results in more changes in terms of edit distance than delta inference. In Figure 5.2 (right), we compute the number of token repetitions in the output translations (before removing them in a post-processing step), relative to the initial output. We observe that refining with a learned score function results in less repetitive output compared to delta inference.

5.6 Qualitative Results

5.6.1 Visualization of learned gradients

We visualize the learned gradients and the optimization trajectory in Figure 5.3, from a score inference network trained on a two-dimensional latent variable model on IWSLT'16 De→En. The example used to generate the visualization is shown below.

Source	Was öffnete mir also die Augen?
Reference	So what opened my eyes ?
Posterior	So what opened my eyes ?
Prior	So what opened me eyes ?
Delta	So what opened me eyes ?
Score	So what opened my eyes ?

Example 1

Source	There aren 't many doctors in the west African country ; just one for every 5,000 people
Reference	In dem westafrikanischen Land gibt es nicht viele Ärzte, nur einen für 5.000 Menschen
Original	Es gibt nicht viele Ärzte im westafrikanischen Land, nur eine für 5.000 Menschen.
Refined	Im westafrikanischen Land gibt es nicht viele Ärzte, nur eine für 5.000 Menschen.

Example 2

Source	Costumes are expected to account for $ 1.2 billion dollars out of the $ 6.9 billion spent , according to the NRF .
Reference	Die Kostüme werden etwa 1,2 Milliarden der 6,9 Milliarden ausgegebenen US-Dollar ausmachen, so der NRF.
Original	Es wird von, Kostüme, dass sie die dem NRF ausgegebenen 6,9 Milliarden Dollar 1,2 Milliarden Dollar ausmachen.
Refined	Es wird erwartet, dass die Kostüme nach Angaben des NRF 1,2 Milliarden Dollar aus den 6,9 Milliarden Dollar ausmachen.

Example 3

Source	It was with this piece of Bedouin wisdom that the first ever chairman Wolfgang Henne described the history and fascination behind the "Helping Hands" society .
Reference	Mit dieser Beduinenweisheit beschrieb der erste Vorsitzende Wolfgang Henne die Geschichte und Faszination des Vereins "Helfende Hände".
Original	Der erste Vorsitzende Wolfgang Henne beschrieb mit dieser erste Weisheit in Bedouin" die Geschichte und Faszination hinter der "Helenden Hands" Gesellschaft
Refined	Mit diesem Stück Bedouin-Weisheit beschrieb der erste Vorsitzende Wolfgang Henne jemals die Geschichte und Faszination hinter der "Heling Hands" Gesellschaft

Table 5.2: Sample translations on WMT'14 En→De. We show the translation from a latent variable sampled from the prior (*Original*) and the translation after one refinement step in the continuous space with the learned score function (*Refined*). We emphasize phrases whose positions are swapped in the refinement process in red and blue.

We observe that for tokens 1, 2 and 6, delta inference converges quickly to the approximate posterior mean. We also find that the local optima estimated by the score function do not necessarily coincide with the approximate posterior mean. For Token 4, while the local optima estimated by the score function (green circle) is far from the posterior mean (red square), they both map to the reference translation (“my”), indicating that there exist multiple latent variables that map to the reference output.

5.6.2 Sample translations

We demonstrate that refining in the continuous space results in non-local, non-trivial revisions to the original sentence. For each example in Table 5.2, we show the English source sentence, German reference sentence, original translation decoded from a sample from the prior, and the revised translation with one gradient update using the estimated score function.

In Example 1, the positions of the main clause (“Es gibt nicht viele Ärzte”) and the prepositional phrase (“im westafrikanischen Land”) are reversed in the continuous refinement process. Inside the main clause, “es gibt” is revised to “gibt es”, a correct grammatical form in German when the prepositional phrase comes before the main clause.

In Example 2, the two numbers are exchanged (“ 1,2 Milliarden Dollar” and “ 6,9 Milliarden Dollar”) in the revised translation. Also, the phrase “aus den” (out of the) is correctly inserted between the two.

In Example 3, the noun phrase “Weisheit in Bedouin” is combined into a single German compound noun “Bedouin-Weisheit”. Also, the phrases “Der erste ...” and “mit dieser ...” are swapped in the refinement process, to better resemble the reference sentence.

5.7 Related Work

Learning Our training objective is closely related to the score matching objective Hyvärinen [2005], with the following differences. First, we approximate the gradient of the data log density using a proxy gradient, whereas this term is replaced by the Hessian of the energy in the original score matching objective. Second, we only consider samples from the prior. Saremi et al. [2018] proposed a denoising interpretation of the Parzen score objective [Vincent, 2011] that avoids estimating the Hessian. Although score function estimation that bypasses energy estimation was found to be unstable [Alain and Bengio, 2014; Saremi et al., 2018], it has been successfully applied to generative modeling of images [Song and Ermon, 2019].

Inference While we categorize inference methods for machine translation as (1) discrete search, (2) hybrid optimization [Shu et al., 2020] and (3) continuous optimization (this work) in Section 6.2, another line of work relaxes discrete search into continuous optimization [Gu et al., 2018b; Hoang et al., 2017b; Tu et al., 2020]. By using Gumbel-softmax relaxation [Jang et al., 2017; Maddison et al., 2017], they train an inference network to generate target tokens that maximize the log probability under a pretrained model.

Gradient-based Inference Performing gradient descent over structured outputs was mentioned in LeCun et al. [2006], and has been successfully applied to many structured prediction tasks [Belanger and McCallum, 2016; Belanger et al., 2017; Wang et al., 2016]. Other work performed gradient descent over the latent variables to optimize objectives for a wide variety of tasks, including chemical design [Gómez-Bombarelli et al., 2018] and text generation [Mueller et al., 2017]

Generation by Refinement Refinement has a long history in text generation. The retrieve-and-refine framework retrieves an (input, output) pair from the training set that is similar to the test example, and performs edit operations on the corresponding output [Gu et al., 2018c; Hashimoto et al., 2018; Song et al., 2016; Sumita and Iida, 1991; Weston et al., 2018]. The idea of refinement has also been applied in automatic post-editing [Grangier and Auli, 2017; Novak et al., 2016].

5.8 Conclusion

We propose an efficient inference procedure for non-autoregressive machine translation that refines translations purely in the continuous space. Given a latent variable model for machine translation, we train an inference network to approximate the gradient of the marginal log probability with respect to the target sentence, using only the latent variable. This allows us to use gradient based optimization to find a target sentence at inference time that approximately maximizes the marginal log probability. As we avoid discrete search over a large vocabulary, our inference procedure is more efficient than previous inference procedures that refine in the token space.

We compare our approach with a recently proposed delta inference procedure that optimizes jointly in discrete and continuous space on three machine translation datasets: WMT'14 En$\rightarrow$De, WMT'16 Ro$\rightarrow$En and IWSLT'16 De$\rightarrow$En. With the same underlying latent variable model, the proposed inference procedure using a learned score function has following advantages: (1) it is twice as fast as delta inference, and (2) it is able to find target sentences resulting in higher marginal probabilities and BLEU scores.

While we showed that iterative inference with a learned score function is effective for spherical Gaussian priors, more work is required to investigate if such an approach will also be successful for more sophisticated priors, such as Gaussian mixtures or normalizing flows. This will be particularly interesting, as recent study showed latent variable models with a flexible prior give high test log-likelihoods, but suffer from poor generation quality as inference is challenging [Lee et al., 2020b].

5.9 Developments Since Publication

Concurrent to our work, Saharia et al. [2020] proposed a non-autoregressive NMT system based on the CTC loss [Graves et al., 2006]. While an earlier work by Libovický and Helcl [2018] explored using CTC for non-autoregressive MT, Saharia et al. [2020] observed that a CTC model trained with sequence-level distillation [Kim and Rush, 2016] can achieve very strong performance. Furthermore, their proposed model gives comparable performance to the autoregressive baseline on WMT'14 En$\rightarrow$Dein only 2 iterations. Most recently, a system proposed by Gu and Kong [2020] gave 27.49 BLEU on WMT'14 En$\rightarrow$Dewith 1 iteration, effectively achieving performance parity with autoregressive models on this particular benchmark with 16.5$\times$ decoding speedup. Their success was enabled by combining two successful approaches: latent variable and CTC loss, and is currently the state-of-the-art in non-autoregressive NMT as of writing this thesis.

Outside of machine translation, several generative models that learn the gradient of the data log density (score) were proposed across different domains, including images [Durkan and Song, 2021; Ho et al., 2020; Saharia et al., 2021; Song et al., 2020] and speech [Chen et al., 2020]. Overall, we hope that our work on text refinement using gradient-based inference inspires new avenues in non-autoregressive text generation and encourage more researchers to explore this area.

Chapter 6

On the Discrepancy between Density Estimation and Sequence Generation

Many sequence-to-sequence generation tasks, including machine translation and text-to-speech, can be posed as estimating the density of the output y given the input x: $p(y|x)$. Given this interpretation, it is natural to evaluate sequence-to-sequence models using conditional log-likelihood on a test set. However, the goal of sequence-to-sequence generation (or structured prediction) is to find the best output $\hat{y}$ given an input x, and each task has its own downstream metric R that scores a model output by comparing against a set of references y^*: $R(\hat{y}, y^*|x)$. While we hope that a model that excels in density estimation also performs well on the downstream metric, the exact correlation has not been studied for sequence generation tasks. In this chapter, by comparing several density estimators on five machine translation tasks, we find that the correlation between rankings of models based on log-likelihood and BLEU varies significantly depending on the range of the model families being compared. First, log-likelihood is highly correlated with BLEU when we consider models within the same family (e.g. autoregressive models, or latent variable models with the same parameterization of the prior). However, we observe no correlation between rankings of models across different families: (1) among non-autoregressive latent variable models, a flexible prior distribution is better at density estimation but gives worse generation quality than a simple prior, and (2) autoregressive models offer the best translation performance overall, while latent variable models with a normalizing flow prior give the highest held-out log-likelihood across all datasets.

6.1 Introduction

Sequence-to-sequence generation tasks can be cast as conditional density estimation $p(y|x)$ where x and y are input and output sequences. In this framework, density estimators are trained to maximize the conditional log-likelihood, and also evaluated using log-likelihood on a test set. However, many sequence generation tasks require finding the best output $\hat{y}$ given an input x at test time, and the output is evaluated against a set of references y^* on a task-specific metric: $R(\hat{y}, y^*|x)$. For example, machine translation systems are evaluated using BLEU scores [Papineni et al., 2002], image captioning systems use METEOR [Banerjee and Lavie, 2005] and text-to-speech systems use MOS (mean opinion scores). As density estimators are optimized on log-likelihood, we want models with higher held-out log-likelihoods to give better generation quality, but the correlation has not been well studied for sequence generation tasks. In this work, we investigate the correlation between rankings of density estimators based on (1) test log-likelihood and (2) the downstream metric for machine translation.

On five language pairs from three machine translation datasets (WMT'14 En$\leftrightarrow$De, WMT'16 En$\leftrightarrow$Ro, IWSLT'16 De$\rightarrow$En), we compare the held-out log-likelihood and BLEU scores of several density estimators: (1) autoregressive models [Vaswani et al., 2017], (2) latent variable models with a non-autoregressive decoder and a simple (diagonal Gaussian) prior [Shu et al., 2020], and (3) latent variable models with a non-autoregressive decoder and a flexible (normalizing flow) prior [Ma et al., 2019].

We present two key observations. First, among models within the same family, we find that log-likelihood is strongly correlated with BLEU. The correlation is almost perfect for autoregressive models and high for latent variable models with the same prior. Between models of different families, however, log-likelihood and BLEU are not correlated. Latent variable models with a flow prior are in fact the best density estimators (even better than autoregressive models), but they give the worst generation quality. Gaussian prior models offer comparable or better BLEU scores, while autoregressive models give the best BLEU scores overall. From these findings, we conclude that the correlation between log-likelihood and BLEU scores varies significantly depending on the range of model families considered.

Second, we find that knowledge distillation drastically hurts density estimation performance across different models and datasets, but consistently improves translation quality of non-autoregressive models. For autoregressive models, distillation slightly hurts translation quality. Among latent-variable models, iterative inference with a delta posterior [Shu et al., 2020] significantly improves the translation quality of latent variable models with a Gaussian prior, whereas the improvement is relatively small for the flow prior. Overall, for fast generation, we recommend a latent variable non-autoregressive model using a simple prior (rather than a flexible one), knowledge distillation, and iterative inference. This is 5–7x faster than the autoregressive model at the expense of 2 BLEU scores on average, and it improves upon latent variable models with a flexible prior across generation speed, BLEU, and parameter count.

6.2 Background

Sequence-to-sequence generation is a supervised learning problem of generating an output sequence given an input sequence. For many such tasks, conditional density estimators have been very successful [Bahdanau et al., 2015; Sutskever et al., 2014; Vinyals and Le, 2015; Vinyals et al., 2015].

To learn the distribution of an output sequence, it is crucial to give enough capacity to the model to be able to capture the dependencies among the output variables. We explore two ways to achieve this: (1) directly modeling the dependencies with an autoregressive factorization of the variables, and (2) letting latent variables capture the dependencies, so the distribution of the output sequence can be factorized given the latent variables and therefore more quickly be generated. We discuss both classes of density estimators in depth below. We denote the training set as a set of tuples $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ and each input and output example as sequences of random variables $\mathbf{x} = \{x_1, \dots, x_{T'}\}$ and $\mathbf{y} = \{y_1, \dots, y_T\}$ (where we drop the subscript n for notational simplicity). We use θ to denote the model parameters.

6.2.1 Autoregressive Models

Learning Autoregressive models factorize the joint distribution of the sequence of output variables $\mathbf{y} = \{y_1, \ldots, y_T\}$ as a product of conditional distributions:

$$\log p_{\text{AR}}(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(y_t | y_{<t}, \mathbf{x}).$$

They are trained to maximize the log-likelihood of the training data:

$$L_{\text{AR}}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log p_{\text{AR}}(\mathbf{y}_n | \mathbf{x}_n).$$

Parameterization Recurrent neural networks and their gated variants are natural parameterizations of autoregressive models [Chung et al., 2014; Elman, 1990; Hochreiter and Schmidhuber, 1997]. By ensuring that no future information $y_{\geq t}$ is used in predicting the current timestep y_t, non-recurrent architectures can also parameterize autoregressive models, such as convolutions [Gehring et al., 2017; van den Oord et al., 2016a] and Transformers [Vaswani et al., 2017], which are feedforward networks with self-attention.

Inference Finding the most likely output sequence given an input sequence under an autoregressive model amounts to solving a search problem: $\text{argmax}_{y_{1:T}} \sum_{t=1}^{T} \log p_\theta(y_t | y_{<t}, \mathbf{x})$. As the size of the search space grows exponentially with the length of the output sequence T, solving this exactly is intractable. Therefore, approximate search algorithms are often used such as greedy search or beam search.

6.2.2 Latent Variable Models

Learning Latent variable models posit a joint distribution of observed variables ($\mathbf{y}$) and unobserved variables ($\mathbf{z}$). They are trained to maximize the marginal log-likelihood of the training data:

$$\log p_{\mathrm{LVM}}(\mathbf{y}|\mathbf{x}) = \log \int_{\mathbf{z}} p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})\, p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z}. \tag{6.1}$$

As the marginalization over **z** makes computing the marginal log-likelihood and posterior inference intractable, variational inference proposes to use a parameterized family of distributions $q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})$ to approximate the true posterior $p(\mathbf{z}|\mathbf{y},\mathbf{x})$. Then, we have the evidence lowerbound (ELBO) [Kingma and Welling, 2014a; Wainwright and Jordan, 2008]:

$$\begin{aligned}\log\, p_{\mathrm{LVM}}(\mathbf{y}|\mathbf{x}) &\geq \mathrm{ELBO}(\mathbf{y},\mathbf{x};\theta,\phi) \qquad (6.2)\\ &= \mathop{\mathbb{E}}_{\mathbf{z}\sim q_\phi}\big[\log p_\theta(\mathbf{y},\mathbf{z}|\mathbf{x}) - \log q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})\big],\end{aligned}$$

where $p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x})$ is the decoder, $q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})$ is the variational posterior and $p_\theta(\mathbf{z}|\mathbf{x})$ is the prior. Both the model and variational parameters θ,ϕ are estimated to maximize ELBO over the training set: $L_{\mathrm{LVM}}(\theta,\phi) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{ELBO}(\mathbf{y}_n,\mathbf{x}_n;\theta,\phi)$.

Parameterization As latent variables can capture the dependencies between the output variables, the decoding distribution can be factorized: $p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x}) = \prod_{t=1}^{T} p_\theta(y_t|\mathbf{z},\mathbf{x})$. The approximate posterior distribution is also often factorized, which can be parameterized by any neural network that outputs mean and standard deviation for each output position: $q_\phi(z_{1:T}|\mathbf{y},\mathbf{x}) = \prod_{t=1}^{T}\mathcal{N}\Big(z_t\Big|\mu_{\phi,t}(\mathbf{y},\mathbf{x}),\sigma_{\phi,t}(\mathbf{y},\mathbf{x})\Big)$. We discuss prior distributions in section 6.2.3.

Inference Generating the most likely output given an input with a latent variable model requires optimizing ELBO with respect to the output: $\mathrm{argmax}_{\mathbf{y}}\mathrm{ELBO}(\mathbf{y},\mathbf{x};\theta,\phi)$. As computing the expectation in Eq. 6.2 is intractable, we instead optimize a proxy lowerbound using a delta posterior [Shu et al., 2020]:

$$\delta(\mathbf{z}|\boldsymbol{\mu}) = \begin{cases} 1, & \text{if } \mathbf{z} = \boldsymbol{\mu} \\ 0, & \text{otherwise} \end{cases}$$

Then, the ELBO reduces to:

$$\begin{aligned} &\mathop{\mathbb{E}}_{\mathbf{z}\sim\delta(\mathbf{z}|\boldsymbol{\mu})} \Big[p_\theta(\mathbf{y}|\mathbf{z},\mathbf{x}) + p_\theta(\mathbf{z}|\mathbf{x})\Big] + \overbrace{\mathcal{H}(\delta)}^{=0}, \\ &= \log p_\theta(\mathbf{y}|\boldsymbol{\mu},\mathbf{x}) + \log p_\theta(\boldsymbol{\mu}|\mathbf{x}). \end{aligned} \tag{6.3}$$

We maximize Eq. 6.3 with iterative refinement: the EM algorithm alternates between (1) matching the proxy to the original lowerbound by setting $\boldsymbol{\mu} = \mathbb{E}_{q_\phi}[\mathbf{z}]$, and (2) maximizing the proxy lowerbound with respect to $\mathbf{y}$ by: $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}}(\log p_\theta(\mathbf{y}|\boldsymbol{\mu},\mathbf{x}))$. The delta posterior is initialized using the prior (e.g. $\boldsymbol{\mu} = \mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})}[\mathbf{z}]$ in case of a Gaussian prior) so that the inference algorithm is fully deterministic, a desirable property for sequence generation tasks. We study the effect of iterative refinement on BLEU score in detail.

6.2.3 Prior for Latent Variable Models

Several work have discovered that the prior distribution plays a critical role in balancing the variational posterior and the decoder, and a standard normal distribution may be too rigid for the aggregate posterior to match [Hoffman and Johnson, 2016; Rosca et al., 2018]. Indeed, follow-up work found that more flexible prior distributions outperform simple priors on several density estimation tasks [Bauer and Mnih, 2019; Tomczak and

Welling, 2018]. Therefore, we explore two choices for the prior distribution: a factorized Gaussian and a normalizing flow.

Diagonal Gaussian A simple model of the conditional prior is a factorized Gaussian distribution:

$$\log p_\theta(z_{1:T}|\mathbf{x}) = \sum_{t=1}^{T} \log \mathcal{N}\Big(z_t \Big| \mu_{\theta,t}(\mathbf{x}), \sigma_{\theta,t}(\mathbf{x})\Big),$$

where each latent variable z_t is modeled as a diagonal Gaussian with mean and standard deviation computed from a learned function.

Normalizing Flow Normalizing flows [Papamakarios et al., 2019; Rezende and Mohamed, 2015; Tabak and Turner, 2013] offer a general method to construct complex probability distributions over continuous random variables. It consists of (1) a base distribution $p_b(\epsilon)$ (often chosen as a standard Gaussian distribution) and an invertible transformation f and its inverse f^{-1}, such that $f(\mathbf{z}) = \epsilon,\ \ f^{-1}(\epsilon) = \mathbf{z}$. As our prior is conditioned on $\mathbf{x}$, so are the transformations: $f(\mathbf{z}; \mathbf{x}) = \epsilon,\ \ f^{-1}(\epsilon; \mathbf{x}) = \mathbf{z}$. Then, by change-of-variables, we can evaluate the exact density of the latent variable $\mathbf{z}$ under the flow prior:

$$\log p_\theta(\mathbf{z}|\mathbf{x}) = \log p_b\Big(f(\mathbf{z}; \mathbf{x})\Big) + \log \left|\det \frac{\partial f(\mathbf{z}; \mathbf{x})}{\partial \mathbf{z}}\right|.$$

Affine coupling flows [Dinh et al., 2017] enable efficient generation and computation of the Jacobian determinant by constructing each transformation such that only a subset of the random variables undergoes affine transformation, using parameters computed from the remaining variables:

$$\begin{aligned}
&\mathbf{z}_{\text{id}}, \mathbf{z}_{\text{tr}} = \text{split}(\mathbf{z}) \\
&\mathbf{s}, \mathbf{b} = g_{\text{param}}(\mathbf{z}_{\text{id}}) \\
&f(\mathbf{z}) = \text{concat}(\mathbf{z}_{\text{id}};\ \ \mathbf{s} \cdot \mathbf{z}_{\text{tr}} + \mathbf{b}),
\end{aligned} \tag{6.4}$$

where g_{param} can be arbitrarily complex as it needs not be invertible. As invertibility is closed under function composition and the Jacobian determinant is multiplicative, increasingly flexible coupling flows can be constructed by stacking multiple flow layers and reordering such that all the variables are transformed.

6.2.4 Knowledge Distillation

While most density estimators for sequence generation tasks are trained to maximize the log-likelihood of the training data, recent work have shown that it is possible to improve the performance of non-autoregressive models significantly by training them on the predictions of a pre-trained autoregressive model [Gu et al., 2018a; van den Oord et al., 2018]. While Zhou et al. [2019] recently found that distillation reduces complexity of the training data, its effect on density estimation performance has not been studied.

6.3 Problem Definition

On a sequence generation task, a conditional density estimator $F \in \mathcal{H}$ (where $\mathcal{H}$ is a hypothesis set of density estimators in section 6.2) is trained to maximize the log-likelihood (or its approximation) of the training set $\{(x_n, y_n)\}_{n=1}^{N}$:

$$L(F) = \frac{1}{N} \sum_{n=1}^{N} \log p_F(y_n | x_n).$$

Once training converges, the model F is evaluated on the test set $\{(x_m, y_m)\}_{m=1}^{M}$ using a downstream metric R:

$$R(F) = R\big(\{(x_m, y_m, \hat{y}_m)\}_{m=1}^{M}\big),$$

where $\hat{y}_m = \text{argmax}_y \log p_F(y|x_m)$.

To perform model selection, we can rank a set of density estimators $\{F_1, \ldots, F_K\}$ based on either the held-out log-likelihood or the downstream metric. We measure the correlation between the rankings given by the log-likelihood $L(F)$ and the downstream metric $R(F)$.

6.4 Experimental Setup

On machine translation, we train several autoregressive models and latent variable models and analyze the correlation between their rankings based on log-likelihood and BLEU.

6.4.1 Datasets and Preprocessing

We use five language pairs from three translation datasets: IWSLT'16 De→En[1] (containing 197K training, 2K development and 2K test sentence pairs), WMT'16 En↔Ro[2]

1. `https://wit3.fbk.eu/`
2. `www.statmt.org/wmt16/translation-task.html`

(612K, 2K, 2K pairs) and WMT'14 En$\leftrightarrow$De[3] (4.5M, 3K, 3K pairs). For WMT'14 En$\leftrightarrow$De and WMT'16 En$\leftrightarrow$Ro, both directions are used.

We use the preprocessing scripts with default hyperparameters from the `tensor2tensor` framework.[4] Namely, we use wordpiece tokenization [Schuster and Nakajima, 2012] with 32K wordpieces on all datasets. For WMT'16 En$\leftrightarrow$Ro, we follow Sennrich et al. [2016a] and normalize Romanian and remove diacritics before applying wordpiece tokenization. For training, we discard sentence pairs if either the source or the target length exceeds 64 tokens. As splitting along the time dimension [Ma et al., 2019] in the coupling flow layer requires that the length of the output sequence is a multiple of 2 at each level, `<EOS>` tokens are appended to the target sentence until its length is a multiple of 4.

6.4.2 Autoregressive Models

We use three Transformer [Vaswani et al., 2017] models of different sizes: Transformer-big (Tr-L), Transformer-base (Tr-B) and Transformer-small (Tr-S). The first two models have the same hyperparameters as in Vaswani et al. [2017]. Transformer-small has 2 attention heads, 5 encoder and decoder layers, $d_{\text{model}} = 256$ and $d_{\text{filter}} = 1024$.

6.4.3 Latent Variable Models

The latent variable models in our experiments are composed of the source sentence encoder, length predictor, prior, decoder and posterior. The source sentence encoder is implemented with a standard Transformer encoder. Given the hidden states of the source

3. `www.statmt.org/wmt14/translation-task.html`
4. `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/bin/t2t-datagen`

sentence, the length predictor (a 2-layer MLP) predicts the length difference between the source and target sentences as a categorical distribution in $[-30, 30]$. We implement the decoder $p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$ with a standard Transformer decoder that outputs the logits of all target tokens in parallel. The approximate posterior $q_\phi(\mathbf{z}|\mathbf{y}, \mathbf{x})$ is implemented as a Transformer decoder with a final Linear layer with weight normalization [Salimans and Kingma, 2016] to output the mean and standard deviation (having dimensionality d_{latent}). Both the decoder and the approximate posterior attend to the source hidden states.

Diagonal Gaussian Prior The diagonal Gaussian prior is implemented with a Transformer decoder which receives a sequence of positional encodings of length T as input, and outputs the mean and standard deviation of each target token (of dimensionality d_{latent}). We train two models of different sizes: Gauss-base (Ga-B) and Gauss-large (Ga-L). Gauss-base has 4 attention heads, 3 posterior layers, 3 decoder layers and 6 encoder layers, whereas Gauss-large has 8 attention heads, 4 posterior layers, 6 decoder layers, 6 encoder layers. $(d_{\text{model}}, d_{\text{latent}}, d_{\text{filter}})$ is (512, 512, 2048) for WMT experiments and (256, 256, 1024) for IWSLT experiments.

Normalizing Flow Prior The flow prior is implemented with Glow [Kingma and Dhariwal, 2018]. We use a single Transformer decoder layer with a final Linear layer with weight normalization to parameterize g_{param} in Eq. 6.4. This produces the shift and scale parameters for the affine transformation. Our flow prior has the multi-scale architecture with three levels [Dinh et al., 2017]: at the end of each level, half of the latent variables are modeled with a standard Gaussian distribution. We use three split patterns and multi-headed 1x1 convolution from Ma et al. [2019]. We experiment with the following hyperparameter settings: Flow-small (Fl-S) with 12/12/8 flow layers in each level and Flow-base (Fl-B) with 12/24/16 flow layers in each level. The first level corresponds to the latent distribution and the last level corresponds to the base distribution. $(d_{\text{model}}, d_{\text{latent}}, d_{\text{filter}})$ is (320, 320, 640) for all experiments. For the Transformer decoder in g_{param}, we use 4 attention heads for Flow-small and 8 attention heads for Flow-base.

6.4.4 Training and Optimization

We use the Adam optimizer [Kingma and Ba, 2015] with the learning rate schedule used by Vaswani et al. [2017]. The norm of the gradients is clipped at 1.0. We perform early stopping and choose the learning rate warmup steps and dropout rate based on the BLEU score on the development set. To train non-autoregressive models, the loss from the length predictor is minimized jointly with negative ELBO loss.

Knowledge Distillation Following previous work [Gu et al., 2018a; Kim and Rush, 2016; Lee et al., 2018], we construct a distilled dataset by decoding the training set using Transformer-base with beam width 4. For IWSLT'16 De$\rightarrow$En, we use Transformer-small.

Latent Variable Models To ease optimization of latent variable models [Bowman et al., 2016; Higgins et al., 2017], we set the weight of the KL term to 0 for the first 5,000 SGD steps and linearly increase it to 1 over the next 20,000 steps. Similarly with Mansimov et al. [2019a], we find it helpful to add a small regularization term to the training objective that matches the approximate posterior with a standard Gaussian distribution:

$\alpha \cdot \mathrm{KL}\big[q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})\,||\,\mathcal{N}(0,\mathbf{I})\big]$, as the original KL term $\mathrm{KL}\big[q_\phi(\mathbf{z}|\mathbf{y},\mathbf{x})\,\big|\big|\,p_\theta(\mathbf{z}|\mathbf{x})\big]$ does not have a local point minimum but a valley of minima. We find $\alpha = 10^{-4}$ to work best.

Flow Prior Models We perform data-dependent initialization of actnorm parameters for the flow prior [Kingma and Dhariwal, 2018] at the 5,000-th step, which is at the beginning of KL scheduling.

6.4.5 Evaluation Metrics

Log-likelihood is the main metric for measuring density estimation (data modeling) performance. We compute exact log-likelihood for autoregressive models. For latent variable models, we estimate the marginal log-likelihood by importance sampling with 1K samples from the approximate posterior and using the ground truth target length.

BLEU measures the similarity (in terms of n-gram overlap) between a generated output and a set of references, regardless of the model. It is a standard metric for generation quality of machine translation systems.

Generation Speed In addition to the quality-driven metrics, we measure the generation speed of each model in the number of sentences generated per second on a single V100 GPU.

		BLEU (↑)		LL (↑)	
		Raw	Dist.	Raw	Dist.
WMT'14 En→De	Tr-S	24.54	24.94	-1.77	-2.36
	Tr-B	28.18	27.86	-1.44	-2.19
	Tr-L	<u>29.39</u>	28.29	-1.35	-2.23
	Ga-B	15.74	24.54	-1.51	-2.44
	Ga-L	17.33	**25.53**	-1.47	-2.24
	Fl-S	18.17	21.98	-1.41	-2.13
	Fl-B	18.57	21.82	**-1.23**	-2.05
	Fl-B$^{(*)}$	18.55	21.45		
	Fl-L$^{(*)}$	20.85	23.72		
WMT'14 De→En	Tr-S	29.15	28.40	-1.66	-2.24
	Tr-B	32.21	32.24	-1.42	-2.12
	Tr-L	<u>33.16</u>	32.24	-1.35	-2.05
	Ga-B	21.64	29.29	-1.41	-2.17
	Ga-L	23.03	**30.30**	-1.31	-2.04
	Fl-S	23.17	27.14	-1.28	-1.73
	Fl-B	23.12	26.72	**-1.20**	-1.71
	Fl-B$^{(*)}$	23.36	26.16		
	Fl-L$^{(*)}$	25.40	28.39		

Table 6.1: Test BLEU score and log-likelihood of each model. Raw: models trained on raw data. Dist.: models trained on distilled data. Tr-S: Transformer-small. Tr-B: Transformer-base. Tr-L: Transformer-big. Ga-B: Gauss-base. Ga-L: Gauss-large. Fl-S: Flow-small. Fl-B: Flow-base. Fl-L: Flow-large. We use beam search with width 4 for inference with autoregressive models, and one step of iterative inference [Shu et al., 2020] for latent variable models. On most datasets, our Flow-base model gives comparable results to those from Ma et al. [2019], which are denoted with $(*)$. We boldface the best log-likelihood overall and the best BLEU score among the latent variable models. We underscore best BLEU score among the autoregressive models.

		BLEU (↑)		LL (↑)	
		Raw	Dist.	Raw	Dist.
WMT'16 En→Ro	Tr-S	30.12	29.57	-1.72	-1.95
	Tr-B	<u>33.46</u>	33.28	-1.63	-2.52
	Ga-B	28.03	29.71	-2.38	-3.48
	Ga-L	28.16	**30.91**	-2.44	-3.54
	Fl-S	26.85	28.63	-1.53	-2.42
	Fl-B	27.49	29.09	**-1.50**	-2.31
	Fl-B$^{(*)}$	29.26	29.34		
	Fl-L$^{(*)}$	29.86	29.73		
WMT'16 Ro→En	Tr-S	29.33	28.87	-1.84	-1.93
	Tr-B	<u>32.19</u>	31.15	-1.79	-2.28
	Ga-B	26.48	27.81	-2.41	-2.92
	Ga-L	27.35	**28.02**	-2.32	-3.01
	Fl-S	26.03	26.12	-1.65	-2.05
	Fl-B	27.14	27.33	**-1.64**	-2.01
	Fl-B$^{(*)}$	30.16	30.44		
	Fl-L$^{(*)}$	30.69	30.72		
IWSLT	Tr-S	31.54	<u>31.72</u>	-1.84	-2.56
	Ga-B	24.36	26.80	-1.98	-2.70
	Fl-S	23.64	26.69	-1.66	-2.28
	Fl-B	24.89	**27.00**	**-1.57**	-2.46
	Fl-B$^{(*)}$	24.75	27.75		

Table 6.2: Test BLEU score and log-likelihood of each model. Raw: models trained on raw data. Dist.: models trained on distilled data. Tr-S: Transformer-small. Tr-B: Transformer-base. Tr-L: Transformer-big. Ga-B: Gauss-base. Ga-L: Gauss-large. Fl-S: Flow-small. Fl-B: Flow-base. Fl-L: Flow-large. We use beam search with width 4 for inference with autoregressive models, and one step of iterative inference [Shu et al., 2020] for latent variable models. On most datasets, our Flow-base model gives comparable results to those from Ma et al. [2019], which are denoted with $(*)$. We boldface the best log-likelihood overall and the best BLEU score among the latent variable models. We underscore best BLEU score among the autoregressive models.

	Tr-B	Ga-B	Fl-B
Raw	0.926	0.831	0.678
Dist.	-0.758	-0.897	-0.873

Table 6.3: Pearson's correlation between log-likelihood and BLEU across the training checkpoints of Transformer-base, Gauss-base and Flow-base on WMT'14 En$\rightarrow$De.

6.5 Results

6.5.1 Correlation between rankings of models

Tables 6.1 and 6.2 presents the comparison of three model families (Transformer, Gauss, Flow) on five language pairs in terms of generation quality (BLEU) and log-likelihood (LL). We present two sets of results: one from models trained on raw data (Raw), and another from models trained on distilled data (Dist.) (which we mostly discuss in section 6.5.2). We use the original test set in computing the log-likelihood and BLEU scores of the distilled models, so the results are comparable with the undistilled models. We make two main observations:

1. Log-likelihood is highly correlated with BLEU when considering models within the same family.

a) Among autoregressive models (Tr-S, Tr-B and Tr-L), there is a perfect correlation between log-likelihood and BLEU. On all five language pairs (undistilled), the rankings of autoregressive models based on log-likelihood and BLEU are identical.

b) Among non-autoregressive latent variable models with the same prior distribution, there is a strong but not perfect correlation. Between Gauss-large and Gauss-base, the model with higher held-out log-likelihood also gives higher BLEU on four out of five datasets. Similarly, Flow-base gives higher log-likelihood and BLEU score than Flow-small on all datasets except WMT'14 De$\rightarrow$En.

2. Log-likelihood is not correlated with BLEU when comparing models from different families.

a) Between latent variable models with different prior distributions, we observe no correlation between log-likelihood and BLEU. On four out of five language pairs (undistilled), Flow-base gives much higher log-likelihood but similar or worse BLEU score than Gauss-base. With distillation, Gauss-large considerably outperforms Flow-base in BLEU on all datasets, while Flow-base gives better log-likelihood.

b) Overall, autoregressive models offer the best translation quality but not the best modeling performance. In fact, Flow-base model with a non-autoregressive decoder gives the highest held-out log-likelihood on all datasets.

Correlation between log-likelihood and BLEU across checkpoints Table 6.3 presents the correlation between log-likelihood and BLEU across the training checkpoints of several models. The findings are similar to Table 6.1: for Transformer-base, there is almost perfect correlation (0.926) across the checkpoints. For Gauss-base and Flow-base, we observe strong but not perfect correlation (0.831 and 0.678). Overall, these findings suggest that there is a high correlation between log-likelihood and BLEU when comparing models within the same family. We discuss the correlation for models trained with distillation below in section 6.5.2.

6.5.2 Knowledge Distillation

In Table 6.3, we observe a strong negative correlation between log-likelihood and BLEU across the training checkpoints of several density estimators trained with distillation. Indeed, distillation severely hurts density estimation performance on all datasets (see Table 6.1). In terms of generation quality, it consistently improves non-autoregressive models, yet the amount of improvement varies across models and datasets. On WMT'14 En→De and WMT'14 De→En, distillation gives a significant 7–9 BLEU increase for diagonal Gaussian prior models, but the improvement is relatively smaller on other datasets. Flow prior models benefit less from distillation, only 3–4 BLEU scores on WMT'14 En↔De and less on other datasets. For autoregressive models, distillation results in a slight decrease in generation performance.

6.5.3 Iterative inference on Gaussian vs. flow prior

We analyze the effect of iterative inference on the Gaussian and the flow prior models. Table 6.4 shows that iterative refinement improves BLEU and ELBO for both Gaussian prior and flow prior models, but the gain is relatively smaller for the flow prior model.

		Number of refinement steps			
		0	1	2	4
BLEU	Ga-B	22.88	24.36	24.60	24.69
	Fl-B	24.57	24.89	24.81	24.92
ELBO	Ga-B	-1.11	-0.93	-0.90	-0.89
	Fl-B	-1.22	-1.17	-1.16	-1.15

Table 6.4: Iterative inference with a delta posterior improves BLEU and ELBO for Gauss-base and Flow-base on IWSLT'16 De→En (without distillation).

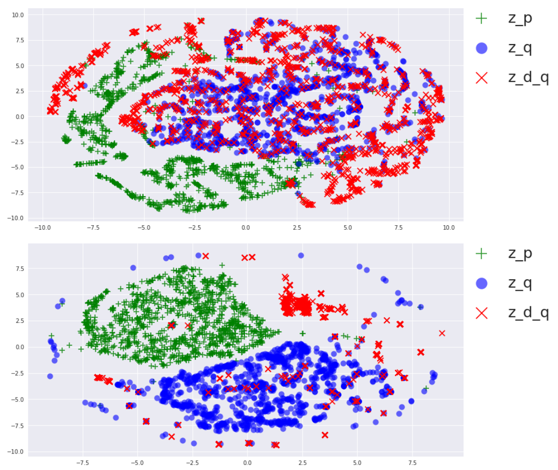


Figure 6.1: Visualization of the latent space with 1K samples from the prior (green plus sign), the approximate posterior (blue circle) and the delta posterior (red cross) of Gauss-base (top) and Flow-small (bottom) on a IWSLT'16 De→En test example.

	BLEU					Speed					Size
$k =$	0	1	2	4	8	0	1	2	4	8	
Tr-S	24.54					2.69					17M
Tr-B	28.18					2.58					60M
Tr-L	29.39					1.93					208M
Ga-B	23.15	24.54	24.87	24.94	24.92	28.77	20.52	16.51	12.00	8.11	75M
Ga-L	24.31	25.53	25.69	25.68	25.68	19.83	14.72	10.25	7.88	4.91	95M
Fl-B	21.57	21.82	21.79	21.81	21.80	5.82	5.60	4.84	3.60	3.37	75M
Fl-L$^{(*)}$	23.72										258M

Table 6.5: BLEU score, generation speed and size of various models on WMT'14 En→De test set. We measure generation speed in sentence/s on a single V100 GPU with batch size 1. We perform inference of autoregressive models using beam search with width 4. For latent variable models, we train perform k steps of iterative inference [Shu et al., 2020] (where $k \in \{0, 1, 2, 4, 8\}$) and report results from models trained with distillation. $(*)$ results are from Ma et al. [2019].

Visualization of latent space In Figure 6.1, we visualize the latent space of the approximate prior, the prior and the delta posterior of the latent variable models using t-SNE [van der Maaten, 2014]. It is clear from the figures that the delta posterior of

Gauss-base has high overlap with the approximate posterior, while the overlap is relatively low for Flow-small. We conjecture that while the loss surface of ELBO contains many local optima that we can reach via iterative refinement, not all of them share the support of the approximate posterior density (hence correspond to data). This is particularly pronounced for the flow prior model.

6.5.4 Generation speed and model size

We compare performance, generation speed and size of various models in Table 6.5. While autoregressive models offer the best translation quality, inference is inherently sequential and slow. Decoding from non-autoregressive latent variable models is much more efficient, and requires constant time with respect to sequence length given parallel computation. Compared to Transformer-base, Gauss-large with 1 step of iterative inference improves generation speed by 6x, at the cost of 2.6 BLEU. On WMT'14 De$\rightarrow$En, the performance degradation is 1.9 BLEU. Flow prior models perform much worse than the Gaussian prior models despite having more parameters and slower generation speed.

6.6 Related Work

For sequence generation, the gap between log-likelihood and downstream metric has long been recognized. To address this discrepancy between density estimation and approximate inference (generation), there has largely been two lines of prior work: (1) structured perceptron training for conditional random fields [Collins, 2002; Lafferty et al., 2001; Liang et al., 2006] and (2) empirical risk minimization with approximate inference [Hopkins and May, 2011; Och, 2003; Povey and Woodland, 2002; Qiang Fu and Biing-Hwang Juang, 2007; Shen et al., 2016; Stoyanov et al., 2011; Valtchev et al., 1997]. More recent work proposed to train neural sequence models directly on task-specific losses using reinforcement learning [Bahdanau et al., 2017; Jaques et al., 2017; Ranzato et al., 2016] or adversarial training [Goyal et al., 2016].

Despite such a plethora of work in bridging the gap between log-likelihood and the downstream task, the exact correlation between the two has not been established well. Our work investigates the correlation for neural sequence models (autoregressive models and latent variable models) in machine translation. Among autoregressive models for open-domain dialogue, a concurrent work [Adiwardana et al., 2020] found a strong correlation between perplexity and a human evaluation metric that awards sensibleness and specificity. This work confirms a part of our finding that log-likelihood is highly correlated with the downstream metric when we consider models within the same family.

Our work is inspired by recent work on latent variable models for non-autoregressive neural machine translation [Gu et al., 2018a; Kaiser et al., 2018; Lee et al., 2018]. Specifically, we compare continuous latent variable models with a diagonal Gaussian prior [Shu et al., 2020] and a normalizing flow prior [Ma et al., 2019]. We find that while having an expressive prior is beneficial for density estimation, a simple prior delivers better generation quality while being smaller and faster.

6.7 Conclusion

In this work, we investigate the correlation between log-likelihood and the downstream evaluation metric for machine translation. We train several autoregressive models and latent variable models on five language pairs from three machine translation datasets (WMT'14 En$\leftrightarrow$De, WMT'16 En$\leftrightarrow$Ro and IWSLT'16 De$\rightarrow$En), and find that the correlation between log-likelihood and BLEU changes drastically depending on the range of model families being compared: Among the models within the same family, log-likelihood is highly correlated with BLEU. Between models of different families, however, we observe no correlation: the flow prior model gives higher held-out log-likelihood but similar or worse BLEU score than the Gaussian prior model. Furthermore, autoregressive models give the highest BLEU scores overall but the latent variable model with a flow prior gives the highest test log-likelihoods on all datasets.

In the future, we will investigate the factors behind this discrepancy. One possibility is the inherent difficulty of inference for latent variable models, which might be resolved by designing better inference algorithms. We will also explore if the discrepancy is mainly caused by the difference in the decoding distribution (autoregressive vs. factorized) or the training objective (maximum likelihood vs. ELBO).

6.8 Developments Since Publication

This chapter was published close to the time of this writing. We hope that our study on the correlation between the training objective and the evaluation metric on machine translation can be a useful reminder to researchers in sequence generation.

Chapter 7

Conclusions and Future Work

Neural sequence generation has seen massive progress in recent years, and deep neural networks trained on large datasets can now successfully generate sequential data of various modalities including text, speech, music, images and video. Many successful generative models proposed thus far are autoregressive, and generate data one element at a time in a given (often temporal) order. While autoregressive models can be straightforwardly trained on exact likelihood, decoding (or sampling) is sequential and cannot be parallelized. Furthermore, inference complexity grows linearly with respect to the length of the sequence being generated. Given the increasing depth and complexity of neural sequence models, generating long sequences such as text documents or waveform with autoregressive models is prohibitively slow to be useful in real applications. While various hardware and software optimizations have been proposed to increase the decoding throughput of neural autoregressive models, improving the asymptotic inference complexity of neural sequence generation is important from a practical standpoint. Motivated by this consideration, this thesis proposes several training and inference algorithms for non-autoregressive neural machine translation.

We explore using latent variables to capture the dependencies between the target tokens, and different ways to incorporate the concept of iterative refinement into the inference procedure. In Chapter 3, we introduced a model that introduces additional dependencies with discrete latent variables, and used denoising autoencoder loss as auxiliary training objective. For inference, we proposed argmax decoding of intermediate discrete latent variables, and an adaptive decoding scheme where the output is repeatedly refined until convergence. In Chapter 4, we introduced a continuous latent variable model trained to maximize the evidence lowerbound (ELBO). We proposed a novel inference algorithm that performs coordinate ascent in both discrete (token) space and continuous (latent) space to maximize ELBO, which we dub delta inference. Delta inference performs refinement in a hybrid space consisting of both discrete and continuous variables. In Chapter 5, we proposed a novel inference algorithm for the continuous latent variable model that performs refinement purely in a continuous space. We train an inference network to approximate the gradient of the log-likelihood with respect to the latent variable, and follow the gradients it approximates to find better latent variable configurations during inference time. In Chapter 6, we investigate the correlation between the training objective (log-likelihood) and the generation quality (BLEU) of several families of density estimators on five machine translation tasks. We discover that the log-likelihood and BLEU are highly correlated when comparing between the same model class, but they are not correlated when comparing between models from different families.

Despite rapid progress in recent years, non-autoregressive neural machine translation is far from solved. We outline a number of avenues for future research in detail below.

Difficulty in learning and MAP inference in VAEs Despite significant progress in variational inference, training variational autoencoders is notoriously difficult and prone to two suboptimal solutions: (1) the generative model ignores the latent variables and the latent codes do not represent the data well [Dai et al., 2020; Lucas et al., 2019], and (2) the aggregate approximate posterior fails to match the prior, hence samples from the prior cannot generate realistic data [Makhzani et al., 2015; Tomczak and Welling, 2018]. While several works studied and proposed solutions to these phenomena for VAEs with an autoregressive decoder [Bowman et al., 2016; Dieng et al., 2019; He et al., 2019; Kim et al., 2018], non-autoregressive VAEs have received relatively little attention to date. As decoder capacity plays a large role in determining the amount of information encoded in the latent codes [Chen et al., 2017], future work should study the extent to which training non-autoregressive VAEs results in such undesirable solutions.

As Lee et al. [2020b] showed, performing efficient maximum-a-posterior (MAP) inference with a latent variable model is challenging, particularly when a flexible prior [Bauer and Mnih, 2019; Ma et al., 2019; Tomczak and Welling, 2018] is used. We observed that while VAEs with a normalizing flow prior achieved superior modeling performance to those with a spherical Gaussian prior, it gave worse generation quality. One potential cause is the tendency of normalizing flows to to assign higher likelihood to out-of-distribution than in-distribution data [Kirichenko et al., 2020; Nalisnick et al., 2019]. While our deterministic delta inference [Shu et al., 2020] had limited success with VAEs with a normalizing flow prior, future work should explore different inference algorithms for VAEs with flexible priors, such as sampling-based approaches [Shah and Barber, 2018] or gradient-based inference [Lee et al., 2020a].

Speed comparison with optimized autoregressive NMT systems Non-autoregressive neural machine translation has achieved tremendous progress in recent years. Most recently, Gu and Kong [2020] achieved 16.5$\times$ speedup on WMT'14 En$\rightarrow$De with comparable performance as the baseline autoregressive Transformer model (27.48 BLEU vs. 27.49 BLEU) on GPU. However, the baseline autoregressive Transformer model used in this work and other work on non-autoregressive machine translation (including this thesis) is not optimized for speed. The autoregressive systems with various speed optimizations (e.g. quantization, light-weight decoder, distillation and improved kernels) [Bogoychev et al., 2020; Kim et al., 2019] report lower latency than non-autoregressive models on standard machine translation tasks in the literature. In order for non-autoregressive machine translation systems to have practical significance, future work should either (1) find use cases where non-autoregressive systems are unequivocally faster, perhaps on document translation, or (2) develop non-autoregressive systems with similar speed optimizations and demonstrate lower latency than autoregressive systems.

Alternative generation paradigm to autoregressive decoding Autoregressive text generation with approximate decoding such as beam search is prone to several well known pathologies. Beam search outputs often contain repetitive text [Holtzman et al., 2020] and can even have infinite length [Welleck et al., 2020]. On the other hand, autoregressive machine translation models often output an empty string when the beam width is sufficiently large [Stahlberg and Byrne, 2019]. In fact, Eikema and Aziz [2020] showed that the most likely translation (mode) under the trained model is often an empty string.

As one possible explanation of such 0-length generation, Shi et al. [2020] recently showed that the probability of observing the <EOS> token is smoothed across different token positions, even allocating probability mass to the first position.

While research on non-autoregressive text generation has mostly focused on improving the decoding speed (including this thesis), it is an interesting future work to investigate whether non-autoregressive decoding with iterative refinement is susceptible to the same pathological generations that suffer autoregressive decoding. Intuitively, non-autoregressive decoding with iterative refinement might potentially find different solutions from autoregressive decoding with beam search. While beam search maintains a fixed number of candidates to choose from, it cannot revise any of the tokens already generated. In contrast, iterative refinement based approaches can refine a word by consulting both its past and future context, several times. We encourage future work to investigate these issues in detail.

Bibliography

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., and Le, Q. V. (2020). Towards a human-like open-domain chatbot. *arXiv preprint arxiv:2001.09977.*

Alain, G. and Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1).

Argueta, A. and Chiang, D. (2019). Accelerating sparse matrix operations in neural networks on graphics processing units. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 6215–6224. Association for Computational Linguistics.

Artetxe, M., Labaka, G., Agirre, E., and Cho, K. (2017). Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041.*

Arulkumaran, K., Creswell, A., and Bharath, A. A. (2017). Improving sampling from generative autoencoders with markov chains. *CoRR*, abs/1610.09296.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A. C., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR.*

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR.*

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments.

Bauer, M. and Mnih, A. (2019). Resampled priors for variational autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 66–75.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Annals of Mathematical Statistics*, 41:164–171.

Belanger, D. and McCallum, A. (2016). Structured prediction energy networks. In *Proceedings of the 33nd International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 983–992.

Belanger, D., Yang, B., and McCallum, A. (2017). End-to-end learning for structured prediction energy networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 429–439. PMLR.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bengio, Y., Simard, P. Y., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166.

Bogoychev, N., Grundkiewicz, R., Aji, A. F., Behnke, M., Heafield, K., Kashyap, S., Farsarakis, E., and Chudyk, M. (2020). Edinburgh's submissions to the 2020 machine translation efficiency task. In *Proceedings of the Fourth Workshop on Neural Generation and Translation, NGT@ACL*, pages 218–224. Association for Computational Linguistics.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A.

(2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Bordes, F., Honari, S., and Vincent, P. (2017). Learning to generate samples from noise through infusion training. In *ICLR*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL*, pages 10–21.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Comput. Linguistics*, 16(2):79–85.

Brown, P. F., Cocke, J., Pietra, S. D., Pietra, V. J. D., Jelinek, F., Mercer, R. L., and Roossin, P. S. (1988). A statistical approach to language translation. In *Proceedings of the 12th International Conference on Computational Linguistics, COLING '88, Budapest, Hungary, August 22-27, 1988*, pages 71–76. John von Neumann Society for Computing Sciences, Budapest.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*.

Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 4960–4964. IEEE.

Chan, W., Kitaev, N., Guu, K., Stern, M., and Uszkoreit, J. (2019). KERMIT: generative insertion-based modeling for sequences. *arXiv preprint arxiv:1906.01604*.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G. F., Jones, L., Schuster, M., Shazeer, N., Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Chen, Z., Wu, Y., and Hughes, M. (2018a). The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 76–86. Association for Computational Linguistics.

Chen, N., Zhang, Y., Zen, H., Weiss, R. J., Norouzi, M., and Chan, W. (2020). Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arxiv:2009.00713*.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Comput. Speech Lang.*, 13(4):359–393.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., and Abbeel, P. (2017). Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR*.

Chen, X., Mishra, N., Rohaninejad, M., and Abbeel, P. (2018b). Pixelsnail: An improved autoregressive generative model. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net.

Child, R. (2020). Very deep vaes generalize autoregressive models and can outperform them on images. *arXiv preprint arxiv:2011.10650*.

Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, K., et al. (2017). State-of-the-art speech recognition with sequence-to-sequence models. *arXiv preprint arXiv:1712.01769*.

Cho, K. (2016). Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

Cho, K., Courville, A., and Bengio, Y. (2015). Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11).

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based models for speech recognition. In *NIPS*.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arxiv:1412.3555*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8. Association for Computational Linguistics.

Dai, B., Wang, Z., and Wipf, D. P. (2020). The usual suspects? reassessing blame for VAE posterior collapse. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2313–2322. PMLR.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*.

Devlin, J. (2017). Sharp models on dull hardware: Fast and accurate neural machine translation decoding on the CPU. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2820–2825. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019)*, pages 4171–4186. Association for Computational Linguistics.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. M., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380. The Association for Computer Linguistics.

Dhariwal, P., Jun, H., Payne, C., Kim, J. W., Radford, A., and Sutskever, I. (2020). Jukebox: A generative model for music. *arXiv preprint arxiv:2005.00341*.

Dieng, A. B., Kim, Y., Rush, A. M., and Blei, D. M. (2019). Avoiding latent variable collapse with generative skip models. In Chaudhuri, K. and Sugiyama, M., editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 2397–2405. PMLR.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. In *International Conference on Learning Representations*.

Durkan, C. and Song, Y. (2021). On maximum likelihood training of score-based generative models. *arXiv preprint arxiv:2101.09258*.

Dyer, C., Chahuneau, V., and Smith, N. A. (2013). A simple, fast, and effective reparameterization of ibm model 2. In *ACL*.

Eikema, B. and Aziz, W. (2018). Auto-encoding variational neural machine translation. In *RepL4NLP@ACL*.

Eikema, B. and Aziz, W. (2020). Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In Scott, D., Bel, N., and Zong, C., editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING*, pages 4506–4520. International Committee on Computational Linguistics.

Elias, I., Zen, H., Shen, J., Zhang, Y., Jia, Y., Weiss, R. J., and Wu, Y. (2020). Parallel tacotron: Non-autoregressive and controllable TTS. *arXiv preprint arxiv:2010.11439*.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Engel, J. H., Resnick, C., Roberts, A., Dieleman, S., Norouzi, M., Eck, D., and Simonyan, K. (2017). Neural audio synthesis of musical notes with wavenet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1068–1077. PMLR.

Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, O., Wenzek, G., Chaudhary, V., Goyal, N., Birch, T., Liptchinsky, V., Edunov, S., Grave, E., Auli, M., and Joulin, A. (2020). Beyond english-centric multilingual machine translation. *arXiv preprint arxiv:2010.11125*.

Fang, L., Zeng, T., Liu, C., Bo, L., Dong, W., and Chen, C. (2021). Transformer-based conditional variational autoencoder for controllable story generation. *arXiv preprint arxiv:2101.00828*.

Firat, O., Cho, K., and Bengio, Y. (2016). Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL HLT 2016, The 2016*

Conference of the North American Chapter of the Association for Computational Linguistics, pages 866–875. The Association for Computational Linguistics.

Frey, B. J., Hinton, G. E., and Dayan, P. (1995). Does the wake-sleep algorithm produce good density estimators? In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, page 661–667, Cambridge, MA, USA. MIT Press.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 1243–1252.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. (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 6111–6120.

Ghazvininejad, M., Levy, O., and Zettlemoyer, L. (2020). Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arxiv:2001.08785*.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276.

Goyal, A., Lamb, A., Zhang, Y., Zhang, S., Courville, A. C., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 4601–4609.

Grangier, D. and Auli, M. (2017). Quickedit: Editing text & translations via simple delete actions. *arXiv preprint arXiv:1711.04805*.

Graves, A., Fernández, S., Gomez, F. J., and Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 369–376.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1764–1772. JMLR.org.

Graves, A., Jaitly, N., and Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278. IEEE.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. (2018a). Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR*.

Gu, J., Im, D. J., and Li, V. O. K. (2018b). Neural machine translation with gumbel-greedy decoding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5125–5132. AAAI Press.

Gu, J. and Kong, X. (2020). Fully non-autoregressive neural machine translation: Tricks of the trade. *arXiv preprint arxiv:2012.15833*.

Gu, J., Liu, Q., and Cho, K. (2019a). Insertion-based decoding with automatically inferred generation order. *Trans. Assoc. Comput. Linguistics*, 7:661–676.

Gu, J., Wang, C., and Zhao, J. (2019b). Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pages 11179–11189.

Gu, J., Wang, Y., Cho, K., and Li, V. O. K. (2018c). Search engine guided neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5133–5140. AAAI Press.

Gygli, M., Norouzi, M., and Angelova, A. (2017). Deep value networks learn to evaluate and iteratively refine structured outputs. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, pages 1341–1351. PMLR.

Han, Q., Meng, Y., Wu, F., and Li, J. (2020). Non-autoregressive neural dialogue generation. *arXiv preprint arxiv:2002.04250*.

Hashimoto, T., Guu, K., Oren, Y., and Liang, P. (2018). A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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., 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. (2018). Achieving human parity on automatic chinese to english news translation. *arXiv preprint arxiv:1803.05667*.

He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. (2019). Lagging inference networks and posterior collapse in variational autoencoders. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR*.

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *NAACL*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hinton, G. E. (2007). To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*.

Hoang, C. D. V., Haffari, G., and Cohn, T. (2017a). Decoding as continuous optimization in neural machine translation. *arXiv preprint arXiv:1701.02854*.

Hoang, C. D. V., Haffari, G., and Cohn, T. (2017b). Towards decoding as continuous optimisation in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156. Association for Computational Linguistics.

Hoang, H., Dwojak, T., Krislauks, R., Torregrosa, D., and Heafield, K. (2018). Fast neural machine translation implementation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 116–121. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hoffman, M. D. and Johnson, M. J. (2016). Elbo surgery: yet another way to carve up the variational evidence lower bound. *Workshop in Advances in Approximate Bayesian Inference, Neurips*.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

Huang, C. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. (2019a). Music transformer: Generating music with long-term structure. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.

Huang, Y., Cheng, Y., Bapna, A., Firat, O., Chen, D., Chen, M. X., Lee, H., Ngiam, J., Le, Q. V., Wu, Y., and Chen, Z. (2019b). Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pages 103–112.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709.

Iglesias, G., Tambellini, W., de Gispert, A., Hasler, E., and Byrne, B. (2018). Accelerating NMT batched beam decoding with LMBR posteriors for deployment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 106–113. Association for Computational Linguistics.

Jang, E., Gu, S., and Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations*. OpenReview.net.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. (2017). Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, pages 1645–1654.

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2015). On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting* *of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397.

Jouppi, N. P., Young, C., Patil, N., Patterson, D. A., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., Gottipati, R., Gulland, W., Hagmann, R., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12. ACM.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? A case study on 30 translation directions. *arXiv preprint arxiv:1610.01108*.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Aji, A. F., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL, System Demonstrations*, pages 116–121. Association for Computational Linguistics.

Kaiser, Ł. and Bengio, S. (2016). Can active memory replace attention? In *NIPS*.

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 2395–2404.

Kaiser, L. and Sutskever, I. (2016). Neural gpus learn algorithms. In *4th International Conference on Learning Representations*.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K. (2018). Efficient neural audio synthesis. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2415–2424. PMLR.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.

Karpathy, A. and Li, F. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Kasai, J., Cross, J., Ghazvininejad, M., and Gu, J. (2020). Parallel machine translation with disentangled context transformer. *arXiv preprint arxiv:2001.05136*.

Kim, Y. and Rush, A. M. (2016). Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D. A., and Rush, A. M. (2018). Semi-amortized variational autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2683–2692. PMLR.

Kim, Y. J., Junczys-Dowmunt, M., Hassan, H., Aji, A. F., Heafield, K., Grundkiewicz, R., and Bogoychev, N. (2019). From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation@EMNLP-IJCNLP*, pages 280–288. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pages 10236–10245.

Kingma, D. P., Salimans, T., Józefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improving variational autoencoders with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29*, pages 4736–4744.

Kingma, D. P. and Welling, M. (2014a). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kingma, D. P. and Welling, M. (2014b). Efficient gradient-based inference through transformations between bayes nets and neural nets. In *Proceedings of the 31th International Conference on Machine Learning,*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1782–1790. JMLR.org.

Kirichenko, P., Izmailov, P., and Wilson, A. G. (2020). Why normalizing flows fail to detect out-of-distribution data. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 181–184. IEEE Computer Society.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (2007). Moses: Open source toolkit for statistical machine translation. In *ACL*.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics.

Kreutzer, J., Foster, G., and Cherry, C. (2020). Inference strategies for machine translation with conditional masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5774–5782. Association for Computational Linguistics.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012*, pages 1106–1114.

Kudo, T. and Richardson, J. (2018a). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP*.

Kudo, T. and Richardson, J. (2018b). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 66–71. Association for Computational Linguistics.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289.

Lample, G., Denoyer, L., and Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

LeCun, Y., Chopra, S., Hadsell, R., Huang, F. J., and et al. (2006). A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press.

Lee, J., Mansimov, E., and Cho, K. (2018). Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182.

Lee, J., Shu, R., and Cho, K. (2020a). Iterative refinement in the continuous space for non-autoregressive neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1006–1015. Association for Computational Linguistics.

Lee, J., Tran, D., Firat, O., and Cho, K. (2020b). On the discrepancy between density estimation and sequence generation. In *Proceedings of the Fourth Workshop on Structured Prediction for NLP, EMNLP*, pages 84–94. Association for Computational Linguistics.

Lee, J. Y., Mehta, S. V., Wick, M. L., Tristan, J., and Carbonell, J. G. (2019). Gradient-based inference for networks with output constraints. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, pages 4147–4154. AAAI Press.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arxiv:2006.16668*.

Li, J., Jia, R., He, H., and Liang, P. (2018). Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 1865–1874. Association for Computational Linguistics.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. (2006). An end-to-end discriminative approach to machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 761–768.

Libovický, J. and Helcl, J. (2018). End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3016–3021. Association for Computational Linguistics.

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *ECCV*.

Lucas, J., Tucker, G., Grosse, R. B., and Norouzi, M. (2019). Understanding posterior collapse in generative latent variable models. In *Deep Generative Models for Highly Structured Data, ICLR 2019 Workshop*. OpenReview.net.

Ma, X., Zhou, C., Li, X., Neubig, G., and Hovy, E. H. (2019). Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 4281–4291.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2017). The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations*.

Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. J. (2015). Adversarial autoencoders. *arXiv preprint arxiv:1511.05644*.

Mansimov, E., Mahmood, O., Kang, S., and Cho, K. (2019a). Molecular geometry prediction using a deep generative graph neural network. *arXiv preprint arxiv:1904.00314*.

Mansimov, E., Wang, A., and Cho, K. (2019b). A generalized framework of sequence generation with application to undirected sequence models. *arXiv preprint arxiv:1905.12790*.

Menick, J. and Kalchbrenner, N. (2019). Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *7th International Conference on Learning Representations, ICLR*. OpenReview.net.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Mikolov, T., Kombrink, S., Burget, L., Cernocký, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, pages 5528–5531. IEEE.

Mueller, J., Gifford, D. K., and Jaakkola, T. S. (2017). Sequence to better sequence: Continuous revision of combinatorial structures. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2536–2544. PMLR.

Nakazawa, T., Yaguchi, M., Uchimoto, K., Utiyama, M., Sumita, E., Kurohashi, S., and Isahara, H. (2016). Aspec: Asian scientific paper excerpt corpus. In *LREC*.

Nalisnick, E. T., Matsukawa, A., Teh, Y. W., Görür, D., and Lakshminarayanan, B. (2019). Do deep generative models know what they don't know? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artif. Intell.*, 56(1):71–113.

Neubig, G., Nakata, Y., and Mori, S. (2011). Pointwise prediction for robust, adaptable japanese morphological analysis. In *ACL*, pages 529–533.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Novak, R., Auli, M., and Grangier, D. (2016). Iterative refinement for machine translation. *arXiv preprint arXiv:1610.06602*.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302. ACL.

Och, F. J. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Comput. Linguistics*, 30(4):417–449.

Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2018a). Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, pages 3953–3962.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (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*, pages 48–53. Association for Computational Linguistics.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018b). Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9. Association for Computational Linguistics.

Papamakarios, G., Nalisnick, E. T., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arxiv:1912.02762*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). 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.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer. In *ICML*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML*, pages 1310–1318.

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics.

Povey, D. and Woodland, P. C. (2002). Minimum phone error and i-smoothing for improved discriminative training. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–105–I–108.

Qiang Fu and Biing-Hwang Juang (2007). Automatic speech recognition based on weighted minimum classification error (w-mce) training method. In *2007 IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*, pages 278–283.

Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR*.

Razavi, A., van den Oord, A., Poole, B., and Vinyals, O. (2019). Preventing posterior collapse with delta-vaes. *CoRR*, abs/1901.03416.

Reed, S. E., van den Oord, A., Kalchbrenner, N., Colmenarejo, S. G., Wang, Z., Chen, Y., Belov, D., and de Freitas, N. (2017). Parallel multiscale autoregressive density estimation. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2912–2921. PMLR.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org.

Rosca, M., Lakshminarayanan, B., and Mohamed, S. (2018). Distribution matching in variational inference. *arXiv preprint arxiv:1802.06847*.

Roy, A., Vaswani, A., Neelakantan, A., and Parmar, N. (2018). Theory and experiments on vector quantized autoencoders. *CoRR*, abs/1805.11063.

Saharia, C., Chan, W., Saxena, S., and Norouzi, M. (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. Association for Computational Linguistics.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2021). Image super-resolution via iterative refinement. *arXiv preprint arxiv:2104.07636*.

Salakhutdinov, R. and Hinton, G. E. (2009). Deep boltzmann machines. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, pages 448–455.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems 29*, page 901.

Saremi, S., Mehrjou, A., Schölkopf, B., and Hyvärinen, A. (2018). Deep energy estimator networks. *arXiv preprint arxiv:1805.08306*.

Schuster, M. and Nakajima, K. (2012). Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pages 5149–5152.

Schwenk, H. (2012). Continuous space translation models for phrase-based statistical machine translation. *Proceedings of COLING 2012: Posters*.

Senellart, J., Zhang, D., Wang, B., Klein, G., Ramatchandirin, J., Crego, J. M., and Rush, A. M. (2018). Opennmt system description for WNMT 2018: 800 words/sec on a single-core CPU. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL*, pages 122–128. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Barone, A. V. M., Mokry, J., and Nadejde, M. (2017). Nematus: a toolkit for neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2016a). Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, WMT*, pages 371–376.

Sennrich, R., Haddow, B., and Birch, A. (2016b). Neural machine translation of rare words with subword units. In *ACL*, pages 1715–1725.

Shah, H. and Barber, D. (2018). Generative neural machine translation. In *NeurIPS*.

Shazeer, N., Cheng, Y., Parmar, N., Tran, D., Vaswani, A., Koanantakool, P., Hawkins, P., Lee, H., Hong, M., Young, C., Sepassi, R., and Hechtman, B. A. (2018). Mesh-tensorflow: Deep learning for supercomputers. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems*, pages 10435–10444.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692.

Shi, X., Xiao, Y., and Knight, K. (2020). Why neural machine translation prefers empty outputs. *arXiv preprint arxiv:2012.13454*.

Shu, R., Lee, J., Nakayama, H., and Cho, K. (2020). Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 11895–11907.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arxiv:2011.13456*.

Song, Y., Yan, R., Li, X., Zhao, D., and Zhang, M. (2016). Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *arXiv preprint arxiv:1610.07149*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

Stahlberg, F. and Byrne, B. (2019). On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3354–3360. Association for Computational Linguistics.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. (2019). Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5976–5985.

Stern, M., Shazeer, N., and Uszkoreit, J. (2018). Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems 31*, pages 10107–10116.

Stoyanov, V., Ropson, A., and Eisner, J. (2011). Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure.

In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 725–733.

Sumita, E. and Iida, H. (1991). Experiments and prospects of example-based machine translation. In *29th Annual Meeting of the Association for Computational Linguistics*, pages 185–192. Association for Computational Linguistics.

Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., and Deng, Z. (2019). Fast structured decoding for sequence models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, pages 3011–3020.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Sutton, C. and McCallum, A. (2012). An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373.

Tabak, E. G. and Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164.

Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. *J. Mach. Learn. Res.*, 4:1235–1260.

Tomczak, J. M. and Welling, M. (2018). VAE with a vampprior. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 1214–1223.

Tu, L., Pang, R. Y., Wiseman, S., and Gimpel, K. (2020). Engine: Energy-based inference networks for non-autoregressive machine translation.

Vahdat, A. and Kautz, J. (2020). NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*.

Valtchev, V., Odell, J. J., Woodland, P. C., and Young, S. J. (1997). Mmie training of large vocabulary recognition systems. *Speech Commun.*, 22(4):303–314.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop*, page 125.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Kavukcuoglu, K., Vinyals, O., and Graves, A. (2016b). Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, pages 4790–4798.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016c). Pixel recurrent neural networks. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org.

van den Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., van den Driessche, G., Lockhart, E., Cobo, L. C., Stimberg, F., Casagrande, N., Grewe, D., Noury, S., Dieleman, S., Elsen, E., Kalchbrenner, N., Zen, H., Graves, A., King, H., Walters, T., Belov, D., and Hassabis, D. (2018). Parallel wavenet: Fast high-fidelity speech synthesis. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pages 3915–3923.

van der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 3156–3164.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory*, 13(2):260–269.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Wang, C., Zhang, J., and Chen, H. (2018). Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 479–488. Association for Computational Linguistics.

Wang, S., Fidler, S., and Urtasun, R. (2016). Proximal deep structured models. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, pages 865–873.

Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q. V., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A. (2017). Tacotron: Towards end-to-end speech synthesis. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 4006–4010.

Wang, Y., Tian, F., He, D., Qin, T., Zhai, C., and Liu, T.-Y. (2019). Non-autoregressive machine translation with auxiliary regularization. *CoRR*, abs/1902.10245.

Welleck, S., Brantley, K., III, H. D., and Cho, K. (2019). Non-monotonic sequential text generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6716–6726.

Welleck, S., Kulikov, I., Kim, J., Pang, R. Y., and Cho, K. (2020). Consistency of a recurrent language model with respect to incomplete decoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 5553–5568. Association for Computational Linguistics.

Weston, J., Dinan, E., and Miller, A. H. (2018). Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2nd International Workshop on Search-Oriented Conversational AI, SCAI@EMNLP 2018, Brussels, Belgium, October 31, 2018*, pages 87–92. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arxiv:1609.08144*.

Xia, Y., Tian, F., Wu, L., Lin, J., Qin, T., Yu, N., and Liu, T.-Y. (2017). Deliberation networks: Sequence generation beyond one-pass decoding. In *NIPS*.

Zens, R., Och, F. J., and Ney, H. (2002). Phrase-based statistical machine translation. In *KI 2002: Advances in Artificial Intelligence, 25th Annual German Conference on AI*, volume 2479 of *Lecture Notes in Computer Science*, pages 18–32. Springer.

Zhang, B., Xiong, D., and Su, J. (2016). Variational neural machine translation. In *EMNLP*.

Zhang, W., Huang, L., Feng, Y., Shen, L., and Liu, Q. (2018). Speeding up neural machine translation decoding by cube pruning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4284–4294. Association for Computational Linguistics.

Zhou, C., Neubig, G., and Gu, J. (2019). Understanding knowledge distillation in non-autoregressive machine translation. *arXiv preprint arxiv:1911.02727*.