Investigating the impact of Normalizing Flows on Latent Variable Machine Translation

by

Michael Przystupa

Bachelors of Science, University of British Columbia, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master's Thesis

in

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2019

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Investigating the impact of Normalizing Flows on Latent Variable Machine Translation

submitted by **Michael Przystupa** in partial fulfillment of the requirements for the degree of **Master's Thesis** in **Computer Science**.

Examining Committee:

Muhammad Abdul-Mageed, Information Sciences
Supervisor

Mark Schmidt, Computer Science
Supervisor

Additional Supervisory Committee Members:

Leonid Sigal, Computer Science
Supervisory Committee Member

Abstract

Incorporating latent variables in neural machine translation systems allows explicit representations for lexical and semantic information. These representations help improve general translation quality, as well as provide more robust longer sentence and out-of-domain translations. Previous work has focused on using variational inference with isotropic Gaussian distributions, which we hypothesize cannot sufficiently encode latent factors of language which could exhibit multi-modal distributive behavior. Normalizing flows are an approach that enable more flexible posterior distribution estimates by introduce a change of variables with invertible functions. They have previously been applied successfully in computer vision to enable more flexible posterior distributions of image data. In this work, we add normalizing flows on top of two previously proposed methods for latent variable machine translation and compare model performances with and without normalizing flows with changes to the model configuration. Our results suggest that normalizing flows can improve translation quality, but require certain modelling assumptions.

Lay Summary

In this work, we consider the question of how to better encode complex information between languages (such as the latent semantic meaning of sentences) in order to improve machine translation systems. In previous work this is accomplished by introducing continuous random variables which are assumed to have simple probability distributions. We extend these works by enabling these distributions to be more flexible beyond these simple distribution by adding normalizing flows. These are invertible functions that can help transform simple distributions into complex ones. Normalizing flows have previously been quite helpful in other areas of artificial intelligence including computer vision. Our results would suggest insert what happens after experiments

Preface

All of the work presented henceforth was conducted jointly by the Machine Learning lab in the Department of Computer Science and the Deep Learning and Natural Language Processing lab in the School Information both located at the University of British Columbia, Point Grey Campus. All work was performed under the supervision of Dr. Muhammad Abdul-Mageed and Dr. Mark Schmidt. Guidance was provided by Frank Wood on direction towards latent variable models for machine translation. Aaron Mischkin provided code snippets to help visualize the latent variables. William Harvey helped with understanding the *Pyro* probabilistic programming language in which all models were implemented. Preliminary results were presented at the Invertible and Normalizing Flows workshop as part of the 2019 International Conference of Machine Learning. The remainder of this thesis, including writing, figures, experiments, implementation were conducted by the author of this thesis.

Table of Contents

List of Tables

List of Figures

Glossary

VAE Variational autoencoder

NMT neural machine translation

LVNMT latent variable neural machine translation

GNMT generative neural machine translation

VNMT variational neural machine translation

ELBO Evidence Lower Bound

RNN recurrent neural network

SOTA state of the art

SEQ2SEQ sequence to sequence

MLP multi-layer perceptron

NLM neural language model

SMT statistical machine translation

GRU gated recurrent unit

VI variational inference

IAF inverse autoregressive flow

BPE byte-pair encoding

MC Monte Carlo

Acknowledgments

Thank you to my supervisors Muhammad Abdul-Mageed and Mark Schmidt for their support throughout this whole project. Thank you to my parents who have loved and supported me through the years, and to my sister who helped improve my writing over the years. Thank you to the University of British Columbia for all the memories.

Chapter 1

Introduction

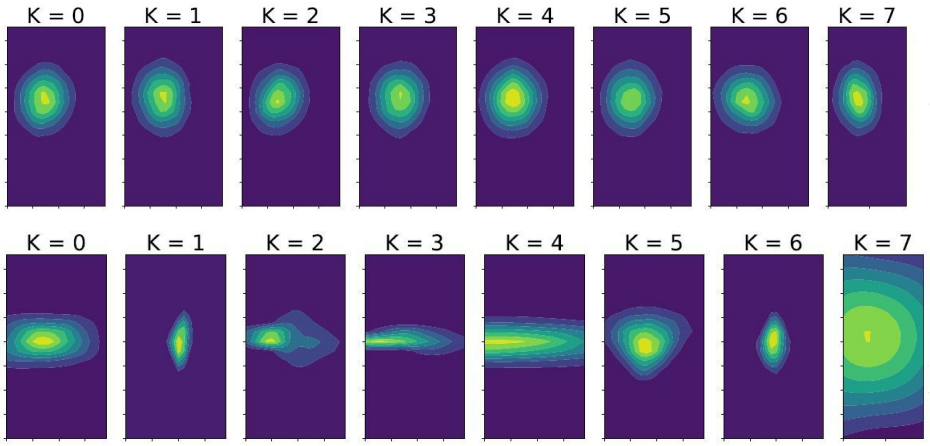


Figure 1.1: Kernel density estimate contour plots of 10,000 samples from $q(z \mid x, y)$ for each intermittent normalizing flow transformation of the distribution using planar (top) and IAF (bottom) flows. The sentence pair is "Als ich in meinen 20ern war, hatte ich meine erste Psychotherapie-Patientin." (De), translated to "When I was in my 20s, I saw my very first psychotherapy client." (En)

Incorporating latent variables to explicitly capture aspects of language, such as semantics, has previously been shown to improve neural machine translation (NMT) quality. This includes difficult scenarios in machine translation, such as translating

longer sentences better [27, 30, 39], demonstrating robustness to domain mis-match between training and test data [6], as well as enabling word level imputation for noisy sentences [27].

Another utility of latent variable neural machine translation (LVNMT) systems is encoding lexical variation. This is achieved by sampling from the latent variables and using beam search to find semantically similar sentences [24, 28]. Generating semantically meaningful sentences is a useful property, because research has shown that synthetically generated *bi-text* can improve translation system quality [5, 25]. In machine translation literature, *bi-text* generally refers to paired sentences from a source language and its translation into a target language. Depending on the model formulation, LVNMT systems can likely help build even better machine translation systems by generating synthetic bi-text of sufficiently good quality.

To our knowledge, much of the research in LVNMT applies amortised variational inference to learn the posterior distribution of paired language data. Authors generally have focused on creating variational auto-encoder type models which optimize the evidence lower bound (ELBO) [14, 23]. In the context of translation, this involves maximizing the log-likelihood of the conditional distribution $p(\mathbf{y}|\mathbf{x},\mathbf{z})$ where $\mathbf{y}$ is the target language, $\mathbf{x}$ is the source language, and $\mathbf{z}$ is the introduced latent variable. Authors have assumed the variational posterior distribution is an isotropic Gaussian and learn a variational distribution $q_\phi(\mathbf{z} \mid \cdot)$ conditioned on different combinations of available paired sentences.[1]

The primary focus of this work, is to investigate the choice of variational distribution to encode information about translation data. A criticism of variational inference is the limited guarantees on approximating, even asymptotically, the true posterior distribution. There are several empirical findings which suggest that choosing the isotropic Gaussian as the variational distribution family may not truly represent latent aspects of language. One simple example is the power-law distribution behaviour that words exhibit in large corpora of text [16]. Previous work in language models showed experimental results demonstrating multi-modal distributive behavior even at the character level [41]. These results suggest that assuming the latent factors follow an isotropic Gaussian distribution is not repre-

[1] Some condition on both the target and source sentence [6, 27, 39], just the target sentences [24], or even just the source [6].

sentative of the true distributive behavior of languages. If latent variables are to be more effectively utilized for machine translation, one needs to consider more flexible variational distributions.

Normalizing flows represent one variational inference approach towards producing more accurate posterior distribution estimates. They accomplish this by transforming a base distribution into a more complex, possibly multi-modal, distribution [31, 32]. This change of variables is achieved by choosing a class invertible functions to transform samples from a chosen base distribution [22]. This variational approach has the added benefit of empirical findings showing more accurate approximations of target posterior distributions when such distributions are known [22].

In the literature, normalizing flows work has seen a number of successes in computer vision, and more recently in natural language processing as well. Particularly with the task of image generation, normalizing flows have been successful in producing high resolution images [13, 15, 33, 36]. Schulz et al. [24] proposed normalizing flows as a potential improvement to their work in LVNMT systems, but to our knowledge never actually expanded to this direction. Recent works have also considered flows on discrete distributions with modulus operations [12, 34].The work of Ziegler and Rush [41] where the authors focused on faster decoding for language models. Most closely related to our work is the work of Ma et al. [18] who created a non-autoregressive normalizing flow machine translation system[18]. Our work differs from there's as we consider incorporating normalizing flows with variations of existing autoregressive LVNMT systems.

We conjecture that normalizing flows are capable of helping achieve better posterior approximations of language factors, and that these improved estimates can help the expressiveness of latent codes in machine translation. Figure 1. shows kernel density estimate contour plots of samples after applying normalizing flow transformation of our distribution, as we further explain in Chapter 6 of this thesis.

Overall, we make the following contributions:

1. We investigate the use of normalizing flows in LVNMT and discuss related considerations and challenges

2. Our experiments seem to suggest that performance improvements due to the

introduction of normalizing flows are minute compared to baseline models.

3. We visualize the learned posterior distribution of paired sentences using a 2D latent space to show how the normalizing flows transform the base distribution.

Chapter 2

Background

In this section, we provide background information on several subjects related to the work in this thesis. These include discussions on variational auto-encoders, normalizing flows, and a general description of NMT. Further details into the particular NMT architecture considered in this work are discussed in Chapter 3.

Neural Machine Translation

Statistical machine translation is a field which applies statistical methods to train computer systems that perform language translation. Historically these methods have typically involved learning phrase tables and language models to perform translation [16]. These components of the system are learned with statistical methods like expectation maximization by analyzing the *bi-text* to extract relevant phrases and derive their likelihood under the provided bi-text. In statistical machine translation (SMT) literature, NMT refers specifically to training SMT systems which incorporate neural networks as the primary model in the system, although machine learning researchers also are actively studying NMT systems.

NMT systems have achieve state of the art (SOTA) results across a variety of language pairs over alternative approaches in SMT literature [1, 17, 37]. For human understanding, the goal is to correctly translate from source language x (e.g. English) to target language y (e.g. German) such that the sentence is coherent and captures the

meaning of the source sentence. [1] Capturing the subjective quality of a translation makes defining an optimization objective a difficult problem. Instead, similar to historical SMT , NMT systems learn to maximize the log-likelihood of the conditional distribution $p(y \mid x)$:

$$max_\theta \log p_\theta(y \mid x) = \sum_{i=1}^{T} \log p(y_i \mid x, y_{<i}). \tag{2.1}$$

Here, θ represents the parameters of the NMT model.

There are a variety of hyperparameter choices when building an NMT system. One core component of many SOTA systems include auto-regressive neural networks which condition on the previous output of the network for data which have sequential relations. One category of such networks is the recurrent neural network (RNN) in which an internal hidden representation is maintained which can be viewed as the networks *memory* of a sequence [9]. In the above objective, this hidden state generally is interpreted as allowing the system to condition on the whole source sentence x and all previous target words $y_{<i}$. Unfortunately, RNNs can suffer from long term dependencies problems. This can lead to gradients either vanishing or exploding during the training process. Researchers have developed a number of architectures to address this problem, such as the long short term memory cells [9].

As there are several types of RNNs, we only describe the gated recurrent unit (GRU) because it is the one used in our work [4]. The GRU can be viewed as a simplification of a long short term memory cell which helps mitigate the effect of gradient exploding or vanishing problem due to long sequence dependencies [9]. The network can be described with the following equations

$$z_t = \text{sigmoid}(W_z[x_t; h_{t-1}]) \tag{2.2}$$

$$r_t = \text{sigmoid}(W_r[x_t; h_{t-1}]) \circ h_{t-1} \tag{2.3}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ (W_h[x_t; r_t]) \tag{2.4}$$

[1] There are other criteria for defining quality in translations including things like adequacy, fidelity, and fluency [20].

Here W_z, W_r, W_h represent the learned weight matrices which can include a bias term. $[a;b]$ refers to a concatenation operation, and $\circ$ is the element-wise product. Intuitively, the GRU works as a soft logic gate where the *update gate* z controls the 'relevance' of previous and current state in the memory of the network. The *reset gate* r decides the importance of previous information in conjunction with the new input.

Whether choosing the GRU, or alternatives, all NMT models generally follow the sequence to sequence (SEQ2SEQ) framework. In SEQ2SEQ models, the source sentence x is *encoded* as a series of latent representations capturing words-in-context information. A *decoder* utilizes these hidden states, such as for initializing it's own hidden state, to help inform the decoding process for target sentences y. The decoder can be interpreted as a conditional *neural language model* [17].[2]

The one other major consideration for NMT is picking the best translation. A naive solution is to take the most likely word at each step of decoding. This can lead to the *garden-path problem* in which case the most likely words at the current time step lead to a series of unlikely words later in translation [17]. Instead, typically *beam search* is employed to maintain n possible beams (translation hypotheses) [17]. At each step, after committing to the top n beams in the previous time step, a score function is calculated and the top n new beams are selected. The score function is the partial probability of up to the current time step i, $\prod_{i=1}^{T} p(x_i|x_{<i})$. When a beam encounters an end-of-sentence token, or designated maximum length, it is held out and the number of active beams is reduced. Once all beams are in-active, the best translation is then chosen from the remaining beams based on their probability normalized by length $\frac{p(y_1,...,y_n)}{n}$.

Variational Autoencoders

The Variational autoencoder (VAE) is a class of generative model which represent the joint distribution $P(X,z)$, where X is the data set and z is an introduced random variable. The joint distribution is factorized typically as follows: $P(X,z) = P(X \mid z)P(z)$. This is interpreted as assuming the dataset X was generated by latent process z.

[2]This simply means a neural network is chosen to model the joint probability of a sentence $p(y_1, y_2, ..., y_T)$ with a continuous representations.

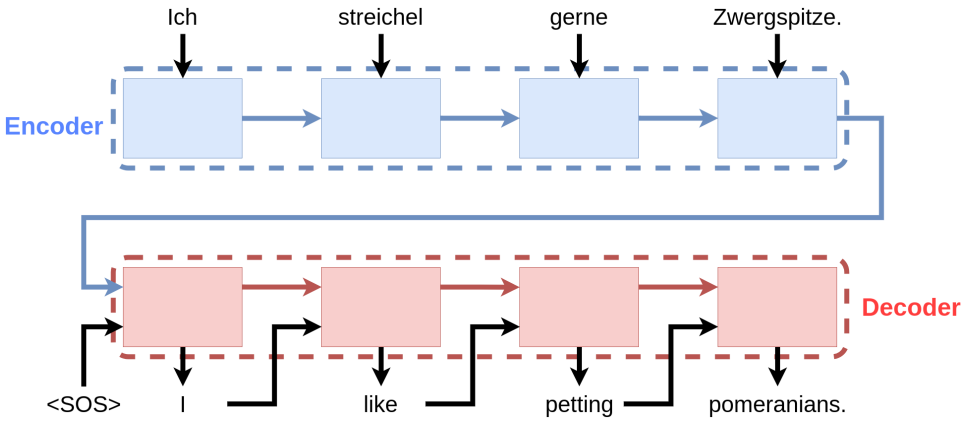


Figure 2.1: Simplest formulation of a SEQ2SEQ model in NMT

An important aspect of VAEs is amortized inference, which involves introducing functions $f(x)$ that produce distribution parameters per datum. The motivation for employing amortized inference is eliminating the need to learn per datum distribution parameters. Otherwise, each datum would require their own parameters, requiring much more computation. In VAEs this function is typically represented with a neural network. Throughout this paper when we describe any distribution as $p_\theta(\cdot)$, the chosen greek symbol (as it will vary beyond θ) represent the parameters of neural networks which handle amortization of $p(\cdot)$.

In order to learn a good representation of $p(X,z)$ the objective is to maximize the log-likelihood of the marginal distribution p(X). To calculate $p(X)$ one then would need to integrate over the random variable z:

$$\log P(X) = \log \int P(X \mid z)P(z)dz. \tag{2.5}$$

Unfortunately, this integral is generally considered intractable. This is often due to choosing a neural network to handle the amortization of distribution parameters. This could be addressed by Monte Carlo sampling from $p(z)$, but the latent variable is unknown and the posterior $p(z \mid X)$ is also difficult to calculate.

Instead, researchers often solve this problem with optimization. This involved applying variational inference (VI) to learn an approximation of the true posterior $p(z \mid X)$. This introduces a variational distribution $q_\phi(z \mid X)$, parametrized by some

neural network ϕ, which will be optimized along with our model. This can be interpreted as introducing an inference network which performs the amortization of the prior parameters $p(z)$ as discussed previously. The objective in VI then involves maximizing the Evidence Lower Bound (ELBO), or variational free energy, on the joint distribution $p(X,z)$ and $q(z \mid X)$.

$$log p_\theta(x) \geq \mathbb{E}_{q(z \mid X)}[log(P_\theta(X \mid z)] - KL(q_\phi(z \mid X)||p(z)) \tag{2.6}$$

An important thing to note in the above equation is that the prior $p(z)$ is typically chosen to be stationary. This means the ELBO can be interpreted as optimizing two conflicting objectives. The expectation term seeks to maximize the reconstruction of data from z whereas the second term bounds the variational distribution to stay within some latent space.

Despite this conflict we have gained some important properties. As the KL divergence is non-negative, and only 0 when the distributions match (i.e. are identical), it is theoretically possible to recover the true log likelihood of the data. We can also sample from $q_\phi(z \mid X)$ to approximate the expectation term which was not possible before.

Unfortunately, directly sampling $q_\phi(z \mid X)$ introduces a discrete operation which means we cannot calculate gradients end-to-end in the model. One approach to mitigate this is the r*e-parametrization trick* in which the variational distribution is rewritten as a function and sampling is done from some surrogate distribution [14, 23] . Here, we show this approach for the isotropic Gaussian with mean μ and variance σ conditioned on x

$$f_\phi(x,\varepsilon) = \mu_\phi(x) + \varepsilon \circ \sigma_\phi(x), \varepsilon \sim N(0,1). \tag{2.7}$$

With this, the expectation in the ELBO can be rewritten with respect to $p(\varepsilon)$ and enables end-to-end optimization

$$log p_\theta(x) \geq \mathbb{E}_{p(\varepsilon)}[log(P_\theta(X \mid f_\phi(x,\varepsilon))] - KL(q_\phi(z \mid X)||p(z)). \tag{2.8}$$

Normalizing Flows

Normalizing flows are an application of the change of variables theorem in machine learning. They introduce a series of k invertible functions $f_{1:k}$ which transform a base distribution $p(z_0)$ into another distribution $p(z_k)$.

$$p(z_k) = f_k \circ f_{k-1} \circ ... \circ f_0(p(z_0)) \tag{2.9}$$

where $\circ$ is shorthand for the nested calls for functions f_i. This allows one to transform samples from the base distribution z_0 into samples z_k

$$z_K = q_0(z_0) \prod_{k=1}^{K} \left| det(\frac{\delta f_i}{\delta z_{i-1}}) \right| \tag{2.10}$$

Alternatively, it becomes possible to do density estimation of samples by calling the inverse f_i^{-1} on samples z_k. To our knowledge, there are 2 main ways normalizing flows models are optimized in the literature.

The first approach incorporates normalizing flows into the ELBO which introduce an entropy term of the log absolute Jacobian with respect to the variational distribution, as follows:

$$\mathbb{E}_{q_\phi(z \mid X)}[log(p_\theta(X \mid z)] - KL(q_\phi(z \mid X)||p(z)) + E_{q_\phi(z \mid X)}\left[\sum_{i=1}^{k} log\left|\frac{\delta f_i}{\delta z_{i-1}}\right|\right] \tag{2.11}$$

Previous works typically employ hyper-networks [10] to amortize the parameters of the normalizing flows instead of directly optimizing them [22, 33, 35]. In the VI setting this enables the variational distribution q to capture possible posterior representations beyond those achievable with the base distribution per datum.

Alternatively, normalizing flows enable direct optimization of the log probability of data distribution:

$$\log P(X) = \log P(z_0) - \sum_{i=1}^{k} \log \left| \frac{\delta f_i}{\delta z_{i-1}} \right| \tag{2.12}$$

This approach is quite useful as it eliminates the need for the variational distribution in VI. The choice of flow becomes particularly important though because of the

dependence on the absolute log Jacobian.

The predominant focus of normalizing flows research has been on defining invertible functions which have computationally efficient Jacobians. This has lead to a variety of flows with different Jacobian formulations such as autoregressive flows which lead to triangular Jacobians [15, 19]; flows which by construction have analytic Jacobians [22, 35], or volume preserving flows which have the Jacobian equal to one [12, 33, 34]. Note that these groupings necessarily reflect the full range of approaches to normalizing flows.

Chapter 3

Latent Variable Neural Machine Translation

In this chapter we describe the LVNMT models we considered as part of our analysis. We begin by describing the underlying NMT architecture common to both approaches. We then describe our discriminative model $p(y \mid x)$, and our generative model $p(x,y)$. These models abstractly are represented as graphical models in 3.1. The final section discusses the practical details of representing the latent variables and defining the inputs from the appropriate components in each architecture. Throughout this chapter, we will often reference vectors as $\mathbb{R}^n$ but the actual dimensions for these vectors does not mean all the vectors are of the same dimensions and will otherwise specify the actual dimension whenever appropriate.

Neural Architecture

In this section we explain the base neural architecture which our LVNMT build from. Inherently, there is nothing necessarily unique to this architecture for incorporating latent variables. The ideas in each LVNMT model considered are applicable to alternative neural architectures, such as the Transformer [37], which may benefit from introduced latent variables as well.

The NMT architecture we consider is a SEQ2SEQ model proposed by the work of Bahdanau et al. [1]. The core components include source & target word embeddings,

an encoder, attention mechanism, and decoder. We describe all the layers except the word embeddings which are projections of the source and target vocabularies into a continuous space $\mathbb{R}^n$. When we refer words x_i or y_i, these actually correspond to each word's associated word embedding as inputs to the model.

Encoder

The encoder is a bi-directional RNN which generates hidden states from reading the source sequence x both forwards and backwards. Formally, the RNN produces forward hidden states $h_i^f \in \mathbb{R}^n$ for each input word x_i. Each h_i^f is conditioned on all previous $x_{<i}$ words through h_{i-1}^f, and previous word embedding. The sequence is then read backwards by the RNN to produce hidden states $h_i^b \in \mathbb{R}^n$ for each word x_i. Each h_i^b is conditioned on all subsequent words in the sentence $x_{>i}$. The final hidden states are a concatenation of these complementary embeddings $h_i = [h_i^f; h_i^b]$ $\forall i \in [0, ..., T]$ where T is the sentence length. Intuitively, each h_i can be viewed as a contextual embedding of each source word in the sentence. We next describe how these embeddings are utilized for decoding via the *global attention* mechanism.

Global Attention

In the context of SEQ2SEQ models, global attention mechanisms combine the encoder hidden states to inform the decoding process. This is achieved by a function of the current decoder state s_j and encoder states $H \in \mathbb{R}^{T \times n}$ to output an energy vector $e \in \mathbb{R}^T$. In the work of Bahdanau et al. [1] the authors propose a multi-layer perceptron (MLP) attention function

$$e_i = MLP(h_i, s_j), \forall i \in [1...T]. \tag{3.1}$$

These energy values are usually normalized to provide weights α_i per hidden state. Bahdanau et al. [1] choose the softmax function for this operation to produce a context vector c_j

$$\alpha_i = \frac{exp(e_i)}{\sum_{t=1}^{T} exp(e_t')} \tag{3.2}$$

$$c_j = \sum_{1}^{T} \alpha_i h_i \tag{3.3}$$

Note that α_i is a scalar while h_i is a vector. The intuition for c_j is that it captures alignment information between the source sequence and the current position in the translation (i.e. target) sequence [1].

Decoder

The decoder is a feed-forward RNN which generates the translated target sentence. It reads the sequence forward producing hidden states $s_j \in R^n$ for each target word y_j in the sequence of length K. In the literature, it can be viewed as a conditional language model [17] whose hidden state is initialized as $s_0 = tanh(affine(h_T))$. *Affine* refers to a linear layer learned weight matrix and bias term. The decoder has three inputs which include the previous word y_{j-1}, the previous decoder hidden state s_{j-1} and the context vector c_j as mentioned in our discussion on the global attention mechanism.

To generate the probabilities for each word in the target sentence, the decoder includes a MLP which uses the maxout activation over the hidden state values [8]. These maxout values are then fed to a final layer which represents the conditional distribution:

$$t_j = maxout(affine([y_{j-1}; c_j; s_j])) \tag{3.4}$$

$$maxout(t) = [max(t_{2j-1}, t_{2j})]_{j=1}^{K/2} \tag{3.5}$$

$$p(y_j \mid x, y_{<j}) = affine(t_j). \tag{3.6}$$

Representation of Latent Variables

Agnostic to either model we consider they follow the same process to encoder the latent variable z which is visually shown in 3.2. There are slight variations in how z is incorporated during decoding, or the exact inputs to our inference networks between the two models. Those details are described in each respective model

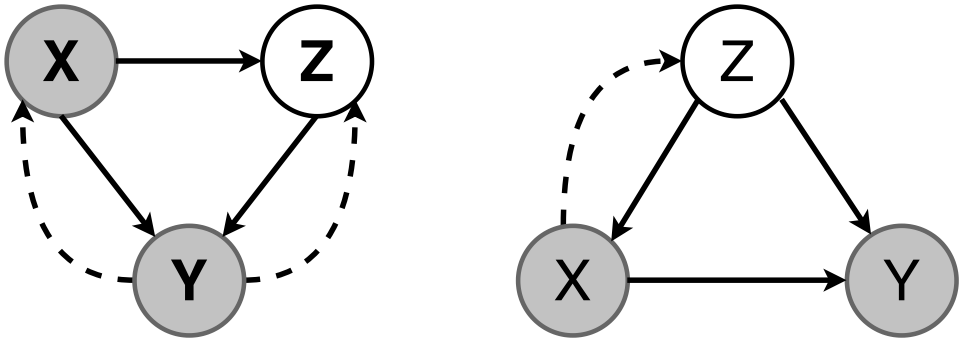


Figure 3.1: Graphical representation of LVNMT systems we consider. Left, is the discriminative model. Right, is the joint or generative model. Dashed lines represent the variational distribution, solid lines are the model.

section that follow.

In either case, the inputs of our models will be the hidden representations produced by an encoder as described in the previous section. These hidden states are then averaged over to produce a single vector representation of the sentence:

$$h_{mean} = \frac{1}{T}\sum_{i=1}^{T} h_i. \tag{3.7}$$

These h_{mean} vectors are passed through a single hidden layer MLP that produces distribution parameters μ and $\log\sigma$ for an isotropic Gaussian distribution. We use a *softplus* operation on the $\log\sigma$ when sampling from the distribution.

Another agnostic decision is setting the value of z at decode time. During training, we sample the latent variable as usual in VAE style models. However, at decode time this sampling proceedure means different translations could be produced by our LVNMT. Instead, we set $z = \mu$ at evaluation time, which has been previously considered in the literature [6, 39]

Discriminative Translation Model

The discriminative LVNMT model we considered is a variation of the work by Zhang et al. [39]. It models the conditional distribution $p(y \mid x)$ which is the typical

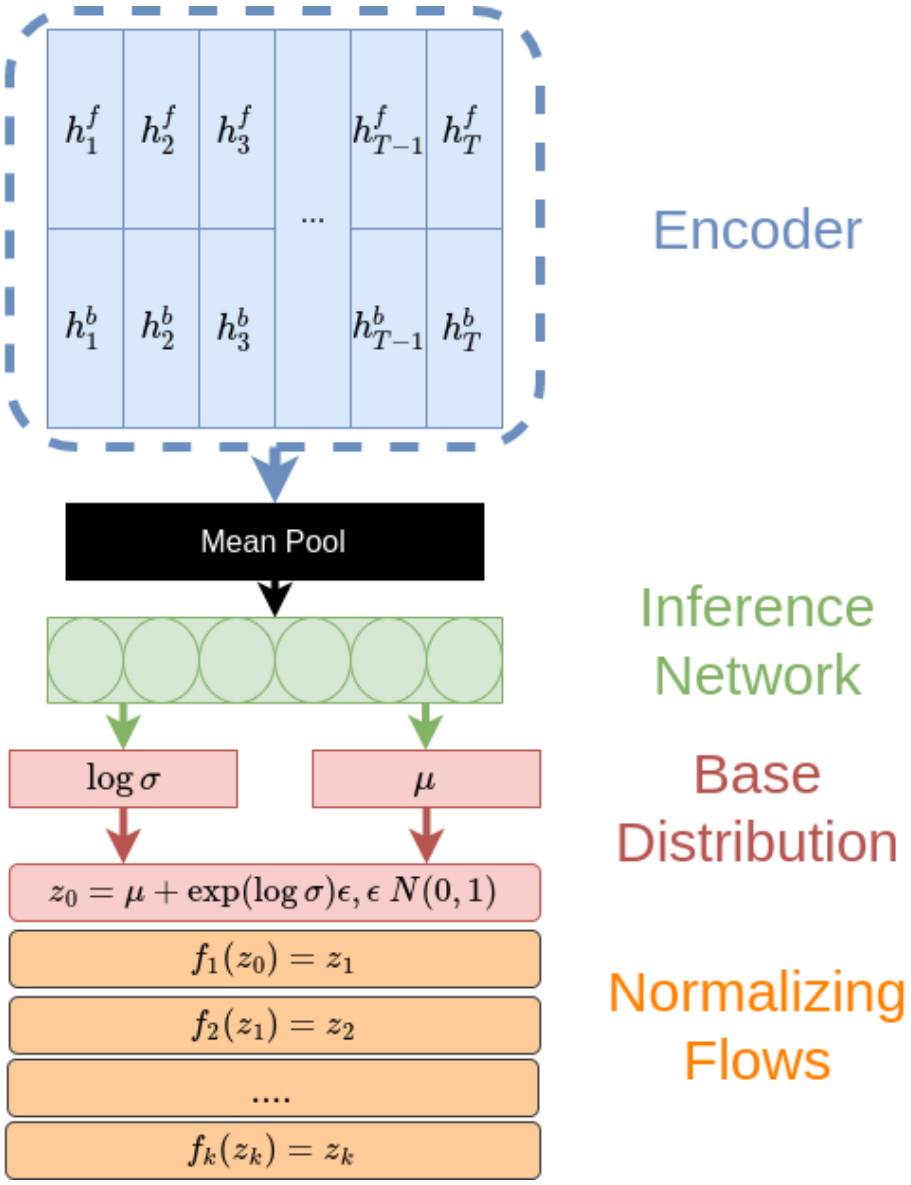


Figure 3.2: General approach to encoding the parameters of the latent variable Z in both models considered. Encoder is part of inference network in generative case.

distribution considered in NMT:

$$p(y \mid x) = \int p(y \mid x, z) p(z \mid x) dz. \tag{3.8}$$

As previously discussed, the integral over latent variable z is often intractable. This requires optimizing the ELBO and introducing a variational distribution $q(z \mid x, z)$. In our version of the model, we optimize the ELBO by generating samples from $q(z \mid x, z)$ where as [39] samples from the prior $p_\theta(z \mid x)$:

$$E_{q_\phi(z \mid x,y)}[log p(y \mid x, z)] - KL(q_\phi(z \mid x, y) || p_\theta(z \mid x)). \tag{3.9}$$

In either formulation, our objectives deviate from the typical VAE models, as the prior is a parametrized distribution $p_\theta(z \mid x)$. Often, the prior is chosen to be a stationary distribution such as $N(0, I)$. This could potentially cause degeneracy in the latent space, because the distributions are not anchored to a fixed target. The model can push each datum's latent distributions apart in order to maximize the translation objective term.

Despite this drawback, the parametrized prior can be beneficial in the translation setting. To translate novel sentences with our variational distribution, we would require having target sentence y, which is not available at decode time. As we minimize KL divergence between q_ϕ and p_θ, they should encode similar information. This means p_θ can replace the variational distribution when translating sentences [39].

As the posterior distribution conditions on both source and target sentences, we need to encode the target sentence y as well during training. We accomplish this by simply passing our target sentence through our encoder as well and average over these target encoder states. The posterior's input is then the concatenation of h^x_{mean} and h^y_{mean}. The samples from these distributions are then included as additional inputs to the decoder at each time-step in the decoding process:

$$y_j = decoder(s_j, y_{j-1}, c_j, z) \tag{3.10}$$

Generative Translation Model

In the generative model, we learn the joint distribution $p(x,y)$ which has otherwise been considered by the work of Eikema and Aziz [6] and Shah and Barber [27]. In their work, the latent variable is included in the joint distribution which is marginalized out during translation

$$p(x,y) = \int p(y \mid x,z)p(x \mid z)p(z)dz. \tag{3.11}$$

In this framework, the latent variable z represents shared aspects of language between the source and target language. We introduce a variational distribution $q(z \mid x)$ which was shown to be as effective as conditioning on both target and source [6]. We expand the ELBO of this objective to explicitly show each model optimized:

$$E_{q_\phi(z \mid x,y)}[logp_\theta(y \mid x,z)] + E_{q_\phi(z \mid x,y)}[logp_\theta(x \mid z)] - KL(q_\phi(z \mid x)||p(z)). \tag{3.12}$$

Our primary goal is learning a translation system through the distribution $p(y \mid x,z)$, but as an artifact train a neural language model (NLM) on the distribution $p(x \mid z)$.

Latent Variable in Language and Translation Model

Our system largely follows the architecture described in Eikema and Aziz [6], with the exception of sharing a projection layer from the latent variable to the translation and language model components. For both the systems, z initializes the hidden state of an RNN. Particularly for the translation system z initializes the hidden state of the encoder to provide a global semantic context during translation. Otherwise, this model behaves the same way as the baseline NMT system. In the language model this corresponds to initializing a language model with the latent variable z.

The only other notable difference in our generative model is the inclusion of optimizing $p(x \mid z)$. The language model behaves similarly to the decoder in our base translation system with the exclusion of (1) the attention mechanism, and (2) initialization by the latent variable z instead of an encoder network. One might note that our incorporation of the latent variable varies between the discriminative and generative scenario.

Chapter 4

Normalizing Flows in Machine Translation

In this chapter we discuss our approach to incorporating normalizing flows into the LVNMT systems discussed in Chapter 3. This includes descriptions of the normalizing flows we considered, and regularization techniques to improve the utilization of the latent variable with auto-regressive models.

Applying Flows to Latent Variables

As a reminder, normalizing flows transform samples from a base distribution $p(z_0)$ to samples of more complex distribution by applying k invertible functions f_i sequentially:

$$z_k = f_k \circ f_{k-1} \ldots \circ f_2 \circ f_1(z_0), z_0 \sim p(z_0) \tag{4.1}$$

In our LVNMT models, $p(z_0)$ refers to our variational posterior distributions. Each f_i can be viewed as adding additional network layer between the base Gaussian distribution and the designated part of the translation model that the latent variable Z is included as input (see figure).

We follow previous research which make flows data dependent [15, 22, 33, 36]. Generally speaking, this means each input sentence x will have unique transforms f_i enabling more flexible latent distributions per sentence pair. We leave further

details on this to the next section as each flow handles this differently. Agnostic to the flow choice, we condition on the source sentence via h^X_{mean}. [1] This choice was because in either model we can only condition on x at decode time.

During training, we optimize the ELBO, which we present again as formulated more specifically to machine translation and is based on the derivation from Rezende and Mohamed [22], Section 4.2:

$$\begin{aligned} & E_{q(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y})}\left[\sum_{j=1}^{p} \log p_\theta(y_j \mid \mathbf{z}_k, \mathbf{x}, y_{<j})\right] \\ & - KL(q_\phi(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y}) || p(\mathbf{z}_k \mid \mathbf{x})) \\ & + E_{q_\phi(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y})}\left[\sum_{k=1}^{K} \log \left|\frac{\delta f_k}{\delta \mathbf{z}_{k-1}}\right|\right]. \end{aligned} \tag{4.2}$$

This simply introduces maximizing the sum of log absolute Jacobian from our flows. [2]

Unfortunately, in normalizing flows we cannot analytically derive the KL divergence and instead we must perform Monte Carlo sampling to optimize this objective. When evaluating translation quality, we set the latent variable to the expected value of the Gaussian distribution , $z_0 = \mu_\theta(x)$, and apply our flows to this value.

Specific to our discriminative LVNMT system, we choose to share the flow parameters on both the prior and variational distribution. At decode time, we replace the variational distribution with the prior which as been optimized to generate distributions similar to the variational posterior. Learning separate flows for each distribution would otherwise be unnecessary computation overhead as the two base distributions should ideally match each other. This also changes the above equation to be quite similar to the original ELBO without normalizing flows, with the exception that we use z_k instead of z_0.

[1] As a reminder, this is the averaged hidden representation of the source sentence produced by our inference network.

[2] In the joint distribution case, there would be the inclusion of optimizing a language model as well.

Considered Flows for Analysis

For our analysis we consider two types of flows from the literature, but note that there are a variety of choices as research, at the time of writing, is quite active. The Jacobian is different for each flow we consider, offering alternative influences on the training procedure for our LVNMT architectures.

Planar Flows

Planar flows were proposed in the work of Rezende and Mohamed [22]. They can be viewed as a scale-shift operation of the following form:

$$f_i(z) = z + u_i h(w_i^T z + b_i). \tag{4.3}$$

Here, $u_i, w_i \in R^d$ and $b \in R$ are the parameters of planar flow i, and h is a non-linear activation. For our experiments we use *tanh* but the authors note that alternative activations are permissible. A convenient aspect of these flows is they provide an analytical term of the Jacobian:

$$\left| \det \left(\frac{\delta f}{\delta z} \right) \right| = \left| 1 + u^t h'(w_i^T z + b_i) w \right| \tag{4.4}$$

Here, h' is the derivative of the nonlinear activation. For clarity, the second term in the right hand side of the equation is a dot product. Intuitively, this transformation can be seen as contracting or expanding the samples z along a single plane in space. This is a simple transformation and requires many planar flows to represent complicated distributions.

As previously mentioned, the parameters of flows are generally made to be data dependent. In the case of planar flows this is achieved by utilizing a *hyper-network* [10] which outputs the parameters of the flow:

$$MLP(h_x^{mean}) = (u, w, b). \tag{4.5}$$

In our experiments each of our flows has a separate hyper-network with a single hidden layer with *tanh* activations to enable flows to have sufficient flexibility to transform distributions.

Inverse Autoregressive Flows

Inverse autoregessive flows were proposed by Kingma et al. [15] to enable parallel sampling by defining an invertible function as a sequentially dependent inverse scale and shift operation in a sequence of random variables

$$z_{k+1}^{i} = \frac{z_k^i - \sigma(z_k^{1:i-1}, h)}{\mu(z_k^{1:i-1}, h)}. \tag{4.6}$$

Here, z^i is the dimension i of the vector of the latent variable z, $\sigma(\cdot)$ and $\mu(\cdot)$ are the outputs of a *MADE* network [7]. Here, h is referred to as the context input which we use $h = h_x^{mean}$. It represents the data conditioning of the normalizing flow, and otherwise the parameters of the MADE layers are shared between data points. Defining a normalizing flow in this way provides a lower triangular Jacobian which means the absolute Jacobian is a product of the diagonal terms

$$\left| \det \frac{\delta f}{\delta z} \right| = \prod_{t=0}^{T} \sigma(z_k^{1:t-1}, h_x^{mean}) \tag{4.7}$$

Regularization Tricks

An often cited challenge including latent variables in auto-regressive models is *posterior collapse* [11]. In order to maximize the ELBO, the variational distribution parameters, for all the training data, are pushed to more closely match the prior. This leads to uninformative latent variables typically when choosing the isotropic Gaussian as the prior. Part of this behaviour has been accredit to strong decoders, like auto-regressive models, which are flexible enough to model the output even by ignoring z. We recommend Chen et al. [3] or Zhao et al. [40] which provide more thorough discussions on the subject. For this work, we address this potential problem with previously proposed approaches referred to as KL-annealing [2, 29] and word dropout [2].

KL-annealing typically involves annealing the weight β of the divergence term

in the ELBO:

$$
\begin{aligned}
& E_{q(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y})}\left[\sum_{j=1}^{p} \log p_\theta(y_j \mid \mathbf{z}_k, \mathbf{x}, y_{<j})\right] \\
& -\beta KL(q_\phi(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y}) || p_\theta(\mathbf{z}_k \mid \mathbf{x})) \\
& +\beta E_{q_\phi(\mathbf{z}_0 \mid \mathbf{x},\mathbf{y})}\left[\sum_{k=1}^{K} \log \left|\frac{\delta f_k}{\delta \mathbf{z}_{k-1}}\right|\right].
\end{aligned} \tag{4.8}
$$

In normalizing flows research, it is often cited that KL-annealing helps improve performance even without strong decoders [15, 22, 33, 36, 41]. In our experiments, we use a linear schedule which increases the importance of our regularization terms after each mini-batch update.

Word dropout is a procedure during training time where with some probability ρ the current word embedding for x_i is replaced with the word embedding for the unknown token.[3] The intuition for it's effectiveness is it encourages the model to depend more on the latent variable for information at decoding. This approach has particularly been important for improving performance of generative LVNMT models [6, 27].

[3]The unknown token is used to handle words not included in the vocabulary during training.

Chapter 5

Experiments

In this chapter we discuss our experiments to evaluate performance of LVNMT models with and without normalizing flows. We also consider settings to empirically evaluate the impact of latent variables on NMT system. We include hyperparameter values in the supplement material, basing them mostly from Eikema and Aziz [6]. These hyperparameters were picked primarily with the generative LVNMT system in mind, not the discriminative model. However, our main interest is not in the exact gains between models, and empirically these hyperparameters were sufficient.

To help prevent *posterior collapse*, we perform KL-annealing linearly over the first 80,000 mini-batch updates. We choose a word-dropout rate of 0.1 for both models. We note, previous work suggested that word dropout is not necessarily help in the discriminative NMT case [27].

We conducted our experiments with the IWSLT 2016 data sets available in the *torchtext* library.[1] We evaluated our models with the De–En language pair. This language pair was one we could more naturally evaluate subjectively.[2] We list total sentence counts in the supplement material. We evaluated performance with raw BLEU score implementation form sacreBLEU [21]. For all experiments, we keep the random seed fixed to a single value for fair evaluation of results.

We represented our vocabulary for each language with byte-pair encoding (BPE) [26]. We used vocabularies of 10,000 BPEs per language. We found that larger

[1] https://github.com/pytorch/text

[2] We did not do any formal human evaluation of translation quality.

vocabularies resulted in sub-word units that occurred infrequently enough to be uninformative from a practical learning perspective. We performed BPE using the Sentence Piece library.[3] We trained our NMT models on sequences of maximum length 50. We used beam search to translate sentences with considering 10 beams, and length normalization set to 1.0. Throughout our experiments we will refer to our discriminative models as variational neural machine translation (VNMT) and our joint system as generative neural machine translation (GNMT).

General Translation

in table 1 dimension of Z = 2, with planar flows set to 2 we get so and so results. (point to cells in table) In this section we report our results when including normalizing flows on top of LVNMT systems. Our hypothesis was that, given normalizing flows success in computer vision, similar gains can be achieved by including normalizing flows in previously considered LVNMT systems. with latent dim = 2, flows do not really offer any improvement, but if you go with a higher dim space, results suggest adding flows can improve performance. Smaller latent space, GNMT is better without flows, but VNMT benefits from normalizing flows. If we go to higher dimension we start to see some gains with GNMT. Pattern: Planar flows usually do well with a higher number of flows seem to help where as IAF helps with smaller flow

Our baselines include the LVNMT models we described in Chapter 3 with just the isotropic Gaussian distribution for the variational distribution. Our baselines optimize the ELBO with the same number of Monte Carlo (MC) samples as our normalizing flows models to provide a fair comparison. This corresponds to better approximations of the negative log-likelihood of sequence predictions. The KL divergence is analytic in the Gaussian case which does not require sampling [14, 23]. Although we do not report numbers, we did find just increasing the number of MC samples improved translation quality on the validation set.

Table 5.1 shows the BLEU score of our results on the test set. The best model was picked based on validation BLEU score from checkpoints after every epoch of

[3] https://github.com/google/sentencepiece

training. Each model was trained for 47 epochs.[4] The boldfaced results represent the best performing version of a model for the given latent dimensions.

Overall, we found our generative model to perform better for any number of flows or latent space size. Even our worst performing generative model provide at least 1.0 BLEU score above the best performing discriminative model. This result is congruent with previous research suggesting joint modelling can be more effective than the simpler discriminative representation [6].

When considering latent dimensions of 128 and 256, we do find that normalizing flows can improve performance slightly based on BLEU score. These gains were more prominent in the discriminative modelling case as compared to the generative case. For VNMT, even a single normalizing flow provided minute benefits. In contrast, GNMT only saw performance gains with a latent space of 256, and even then only a few of the flows models outperformed the baseline.

Our results seem to confirm previous findings on transform complexity, when comparing the flow type for a given number of flows for latent spaces 128 and 256. Generally the IAF flows gave higher BLEU score with fewer flows compared to the same number of planar flows. This seems to affirm previous research showing that more complex flows require fewer layers to be beneficial [15, 36]. However, as we begin using higher numbers of flows we find that planar flows begin to perform marginally better. This would seem to highlight research suggesting better amortization schemes of flow parameters can generally be beneficial [36], as well the limitations of planar flows transformation capabilities [22, 36].

The most surprising results was with our best overall model which had a latent space of 2. We originally considered such a small latent space for plotting purposes, but found it to be quite effective. Our findings would seem congruent with previous results on the value of smaller latent spaces [24].[5] Interestingly, our conclusions with flow complexity are reverse for this latent space where mostly planar flows perform better than IAF. This would suggest for especially small latent spaces, that flexible transformations are more of a hindrance than beneficial to translation performance.

[4] When we say epoch we mean having updated on all sentence pairs before reshuffling mini-batches.

[5] We do not know if other researchers have also considered such small latent spaces.

Table 5.1: BLEU score for our models with normalizing flows for De-En translation. The best performing models are in bold for each type and number of flows. 0 flows IS my baseline, explain what your basleines are in. Bold baseline 1 (column explainin it), baseline 2 (column explainin what it is?)

Latent Dimension: 2

Flows	1	2	4	8	16	Baseline	Model
Planar	**19.062**	18.949	18.935	19.009	19.051	18.924	VNMT
IAF	18.894	18.477	18.803	**18.964**	18.67		
Planar	20.773	20.696	20.808	20.863	20.365	**21.024**	GNMT
IAF	20.607	20.515	20.582	20.552	20.464		

Latent Dimension: 128

Flows	1	2	4	8	16	Baseline	Model
Planar	18.843	18.838	19.139	19.019	**19.176**	18.768	VNMT
IAF	18.891	18.957	**19.29**	18.813	18.92		
Planar	20.585	20.595	20.476	20.553	20.655	**20.731**	GNMT
IAF	20.639	20.64	20.513	20.646	20.503		

Latent Dimension: 256

Flows	1	2	4	8	16	Baseline	Model
Planar	18.934	**19.263**	19.022	18.79	18.82	18.761	VNMT
IAF	18.953	**19.173**	19.023	18.986	18.773		
Planar	20.551	20.666	20.537	20.51	**20.671**	20.655	GNMT
IAF	20.85	**20.864**	20.636	20.619	20.664		

Importance of Attention

In this section, to see the utility of including latent variables, we train simplified version of our NMT system which do not include the attention mechanism. Our motivation for this experiment is to see the impact of our latent variable as the only additional information to the decoding process. We do not expect this system to outperform the models with attention, particularly given the success of Transformers[37]. However, we hypothesize that if the latent variable can encode useful information in translation, then latent variables will still be better over systems without the latent variable. As an extension, normalizing flows will then enable these variables to be more beneficial by making the latent variable distribution more flexible.

Our modified models follow Bahdanau et al. [1] which means we include the latent variable in the generator network as a substitute for attention. In our previous experiment we did not consider this, as the focus was on incorporating normalizing flows in variations of existing models. We compare our modified discriminative model against a version of Bahdanau et al. [1] where attention is simply removed. In the joint modelling case, we use a baseline similar to the joint model of Eikema and Aziz [6] except without attention. Their model optimize the language model and translation systems separately except for sharing the source language word embeddings. All other hyperparameters are the same. Table 5.2 show's results for our modified models and the baselines when evaluated with BLEU score. The best performing models are bold.

Considering only the latent dimension size, we find an increase in performance simply doubling the dimensions of the latent variable. This would suggest that bigger latent spaces become more important when the model depends more on the latent variable. The combination of planar flows and VNMT seem to benefit most from the latent variable, where as IAF flows seem less effective. There could be several reasons, the most immediate conclusion being a bi-product of the random seed. One interpretation we suspect is related to the IAFs design compared to planar flows. As our GRUs are fairly small compared to other models, and sensitive to small perturbations, they cannot as well utilize these more complex distributions. Another might be that as our planar flows provide per-sentence flow parameters, making them more flexible. In comparison, the IAF is unable to capture this more nuance information by simply conditioning on a context vector.

Unfortunately, our GNMT model did not benefit from the inclusion of normalizing flows and was actually hindered in performance for most choices of flows. We reason this is due to the formulation of GNMT. The latent variable initialize the translation system beginning in the encoder network as compared to being passed as input during decoding. This means, our decoder never actually see the latent variable directly. Without this explicit information, the model is relying on the GRUs to implicitly keep this information from beginning to end which, based on our results is less effective than the input feeding approach of VNMT to benefit from normalizing flows.

Table 5.2: Results for translation systems without attention mechanism. Baseline are deterministic version of models excluding latent variables. to baselines: no flows, no latent variable. Baseline = (2 parentheses, model without attention lean towards to have explanatory. Look at just table and caption people can understand the table) when we remove this etc. etc. Move 0 flows to right column, so baseline 1: just add latent variable, baseline 2: model just with out attention, makes consistent with first table

Latent Dimension: 128

Flows	0	1	2	4	8	16	Baseline	Model
Planar	6.249	6.61	6.637	6.626	**6.931**	6.782	6.383	VNMT
IAF		**6.417**	6.321	6.278	5.983	5.918		
Planar	**7.365**	7.2	7.217	7.19	7.105	7.246	7.163	GNMT
IAF	7.365	7.314	7.371	7.325	7.182	**7.413**		

Latent Dimension: 256

Flows	0	1	2	4	8	16	Baseline	Model
Planar	6.432	6.307	6.813	6.911	7.371	**7.383**	6.383	*VNMT*
IAF		6.452	6.474	6.4	6.226	**6.708**		
Planar	**7.505**	7.398	7.217	7.332	7.262	7.074	7.163	*GNMT*
IAF		7.349	7.344	7.278	7.04	7.359		

Understanding Latent Variable

In previous works, the utility of latent variables is typically justified by training a similar system without the latent variable included, or being optimized differently. In cases where the prior is stationary, authors then typically report the KL divergence to justify the latent variables usage. This metric is inapplicable to VNMT where the prior is learned. In the ideal VNMT scenario, the KL divergence being 0 suggests both distributions encode the same information.

As an alternative approach to measure the value of our latent variable during translation, we set Z to the 0 vector. We measure the difference in BLEU score with and without Z during the decoding process. This can give us direct insight into the importance of Z when translating sentences, whether the prior is learned or not. Table 5.3 provides the KL divergence and 5.4 show the difference in BLEU score when Z is the 0 vector.

Interestingly, for many of our VNMT models with planar flows we see including Z often negatively impacts performance of the translation system. This could

suggest that the latent variable itself is not helping the translation, but improves representations in the encoder or source word embeddings. Another explanation is that our substitution of p_θ for q_ϕ at decode time provide information that is too dissimilar to the variational distribution. This is plausible given that in many cases the KL divergence is non 0, but even in several instances a lower KL divergence still lead to loss of performance with Z. These substitution approaches have also previously been shown to provide minimal or mixed results in terms of comparative performances [6].

The exception to our above observations of course is our VNMT with IAF models in higher dimensions which seem to heavily depend on the latent variable. One possible reason for this has to do with the amortization of IAF flows. As these flows are composed of neural networks shared across data points, they can more easily be viewed as additional layers in the network. In a similar setting, one bug we found in our implementation was excluding the projection layer and found similar performance drops to those with out IAF flows results. This could also be a more volatile result which occurs due to different choices of initialization. It has been noted KL annealing schemes can be effected by initialization for final performance [38].

In comparison, GNMT seems to generally show at least minute dependence on the latent variable regardless of number of flows. There do not seem to be any clear patterns between choices about flows and the information encoded by z. Likewise, a higher KL divergence does not necessarily correspond with more utility in translation performance.

In comparison, when we perform the same experiment with our LVNMT systems without attention, we see more dependence on the latent variable. Our results are in table 5.6 and 5.5. In most cases with planar flows, we see our models depend more on the latent variable as opposed to without the system. Our IAF flows models are more mixed in terms of performance changes, where in GNMT our models depend more on the flows with a smaller latent dimension space as opposed to larger latent space.

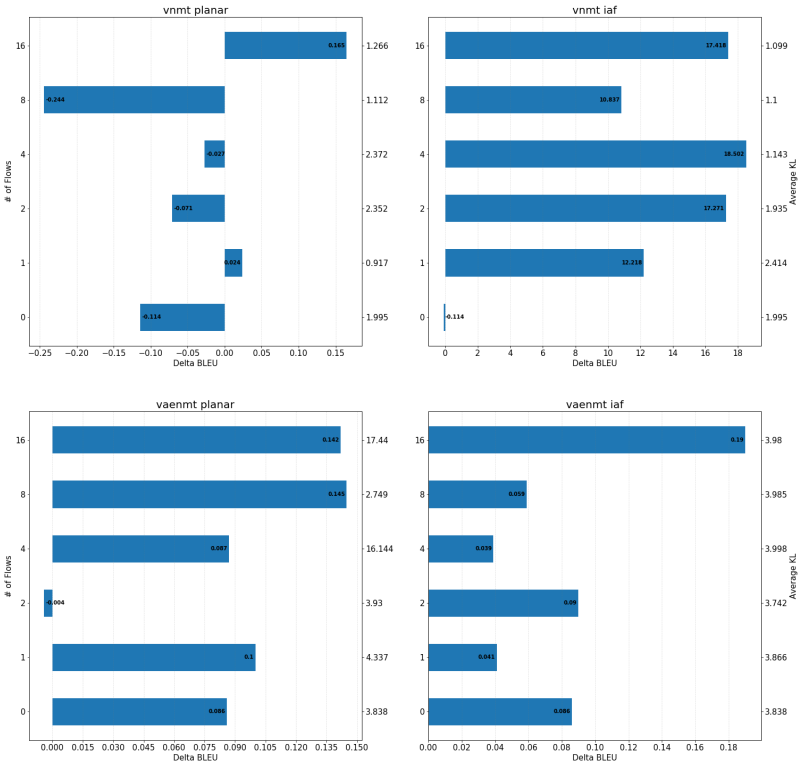


Figure 5.1: The drop in performance when we set the value of the latent variable of Z to 0 during evaluation for latent dimensions set to 256. Bars are $BLEU_{Z=\mu} - BLEU_{Z=0}$. KL divergence of model is included on right axis, number of flows on left axis.

Table 5.3: Average KL divergence for the test set. For VNMT the KL term should be smaller meaning the distributions encoder similar information. for GNMT they should be higher as these suggest more informative latent spaces.

Latent Dimension: 2 (KL Difference)

Flows	1	2	4	8	16	Baseline	Model
Planar	0.94	1.446	0.679	0	0	1.154	VNMT
IAF	1.15	1.564	1.395	0.846	1.559		
Planar	2.481	2.319	2.005	0.104	3.642	2.992	GNMT
IAF	0	2.362	3.913	2.086	2.625		

Latent Dimension: 128

Flows	1	2	4	8	16	Baseline	Model
Planar	2.157	1.269	1.181	1.037	1.625	1.039	VNMT
IAF	0.682	1.346	1.318	1.595	1.059		
Planar	3.226	3.154	2.356	2.824	3.644	4.394	GNMT
IAF	4.308	4.356	4.185	4.432	4.141		

Latent Dimension: 256

Flows	1	2	4	8	16	Baseline	Model
Planar	0.917	2.352	2.372	1.112	1.266	1.995	*VNMT*
IAF	2.414	1.935	1.143	1.1	1.099		
Planar	4.337	3.93	3.355	2.749	3.624	3.838	*GNMT*
IAF	3.866	3.742	3.998	3.985	3.98		

Language Modelling Performance

Hypothesis: Including normalizing flows improve the performance of the language model learned as part of the generative machine translation system. [This is related to a SPECIFIC model and is not applicable to the discriminative version of this stuff...just want to make sure that's clear] As previously discussed in chapter 3, the generative machine translation system is trained with a language model component as part of the optimization process. In this section, we look at how the performance of this component is affected by the inclusion of normalizing flows by evaluating the performance of the learned language model during the training procedure. We measure the BLEU score and perplexity [are there other metrics I should consider?] for the generated source side sentences in our GNMT model. Note that the -ELBO in [Fig of ELBO] is already reported as this module is learned along with the

Table 5.4: Change in BLEU score when Z is set to 0 vector at decode time. Negative numbers indicate our models do better without Z included during translation.

Latent Dimension: 2 (BLEU difference)

Flows	1	2	4	8	16	Baseline	Model
Planar	0.003	-0.094	-0.06	0.04	-0.003	-0.102	VNMT
IAF	-0.008	0.058	-0.102	-0.102	-0.114		
Planar	0.246	-0.027	0.109	0.036	0.303	0.239	GNMT
IAF	0.003	0.109	0.005	0.1	-0.023		

Latent Dimension: 128

Flows	1	2	4	8	16	Baseline	Model
Planar	-0.294	-0.367	-0.059	-0.136	0.1	0.044	VNMT
IAF	18.643	12.169	15.517	17.396	17.558		
Planar	0.143	0.076	0.025	0.064	0.103	0.08	GNMT
IAF	0.115	0.078	-0.01	0.104	-0.009		

Latent Dimension: 256

Flows	1	2	4	8	16	Baseline	Model
Planar	0.024	-0.071	-0.027	-0.244	0.165	-0.114	*VNMT*
IAF	12.218	17.271	18.502	10.837	17.418		
Planar	0.1	-0.004	0.113	0.145	0.049	0.086	*GNMT*
IAF	0.041	0.09	0.039	0.059	0.19		

Table 5.5: KL divergence for models without attention

Latent Dimension: 128

Flows	1	2	4	8	16	Baseline	Model
Planar	2.157	1.269	1.181	1.037	1.625	0.599	VNMT
IAF	0.682	1.346	1.318	1.595	1.059		
Planar	3.226	3.154	2.356	2.824	3.644	2.231	GNMT
IAF	4.308	4.356	4.185	4.432	4.141		

Latent Dimension: 256

Flows	1	2	4	8	16	Baseline	Model
Planar	0.345	0.04	0.3	0.161	0.066	0.719	*VNMT*
IAF	0.739	0.082	0.701	0.255	0.216		
Planar	4.102	4.262	3.567	3.54	3.652	4.186	*GNMT*
IAF	3.989	3.192	3.994	3.12	3.446		

Table 5.6: Change in BLEU score without attention. Positive values indicate latent variable Z is more important for translation

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**Baseline**	**Model**
Planar	2.55	2.324	2.149	2.682	3.024	0.257	VNMT
IAF	1.758	1.019	1.07	1.076	0.402		
Planar	0.752	0.916	0.805	0.739	0.862	0.709	GNMT
IAF	0.89	0.89	0.744	0.752	0.987		

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**Baseline**	**Model**
Planar	2.725	2.714	3.486	3.592	3.918	1.105	*VNMT*
IAF	2.016	2.677	3.142	1.404	2.184		
Planar	1.071	1.247	0.351	0.415	0.513	0.597	*GNMT*
IAF	0.557	0.595	0.492	0.6	0.533		

translation system.

for Language model: Intrinsic metric: generate sequence that is expected: Lower the better (evaluating itself against...itself? you can evaluate. Use perplexity =¿ method of identifying how close the model will predict a given sequence. How close to the language model it was trained on.

perplexity is sitll for the language model performance

Extrinsic evaluation: Put your language model into something else. e.g. plug into MT system, does it help machine translation? How do we know if it helped or not? did bleu score go up or not? does the effect adding the language model vs adding flows 4 settings: LM + flows, LM + w/o flows, w/o LM + flows, w/o Lm model + w/o flows

Table 5.7: BLEU scores for vaenmt without language model training. In bold are best performances.

Latent Dimension: 2

Flows	1	2	4	8	16	Baseline	Model
Planar	20.546	20.404	**20.662**	20.315	20.62	20.399	GNMT
IAF	**20.505**	20.266	20.471	20.297	20.085		

Latent Dimension: 128

Flows	1	2	4	8	16	Baseline	Model
Planar	20.461	20.422	20.497	20.653	20.413	**20.774**	GNMT
IAF	20.478	20.449	20.708	20.455	20.383		

Latent Dimension: 256

Flows	1	2	4	8	16	Baseline	Model
Planar	20.325	20.36	20.468	20.213	**20.655**	20.586	*GNMT*
IAF	20.557	**20.637**	20.509	20.544	20.412		

Table 5.8: BLEU score difference of GNMT when language model is not included during training.

Latent Dimension: 2

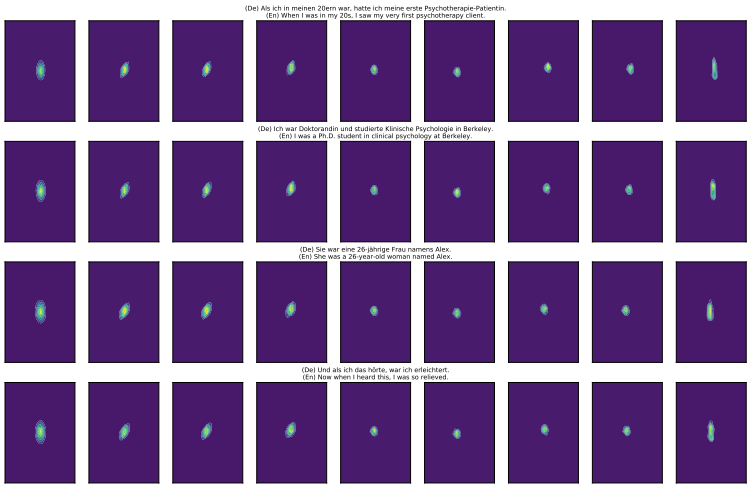
Flows	1	2	4	8	16	Baseline	Model
Planar	0	0	0.003	0	0.001	0	GNMT
IAF	**0**	-0.001	-0.004	0	0		

Latent Dimension: 128

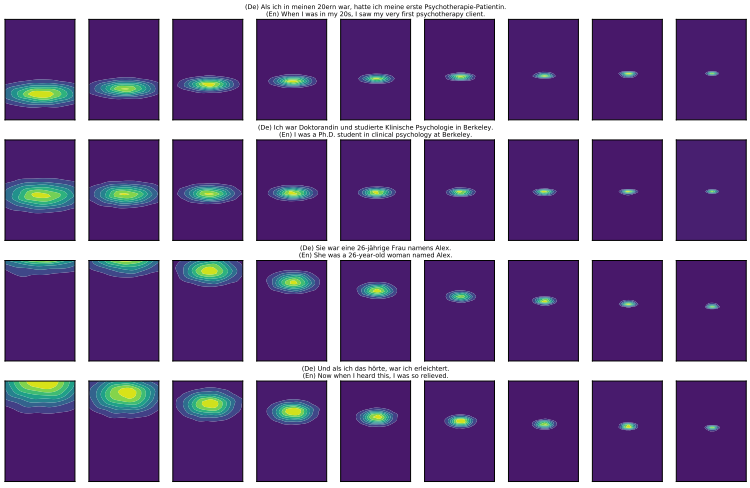
Flows	1	2	4	8	16	Baseline	Model
Planar	0	-0.002	0	-0.001	0	0	GNMT
IAF	-0.001	-0.001	0.002	-0.002	0.002		

Latent Dimension: 256

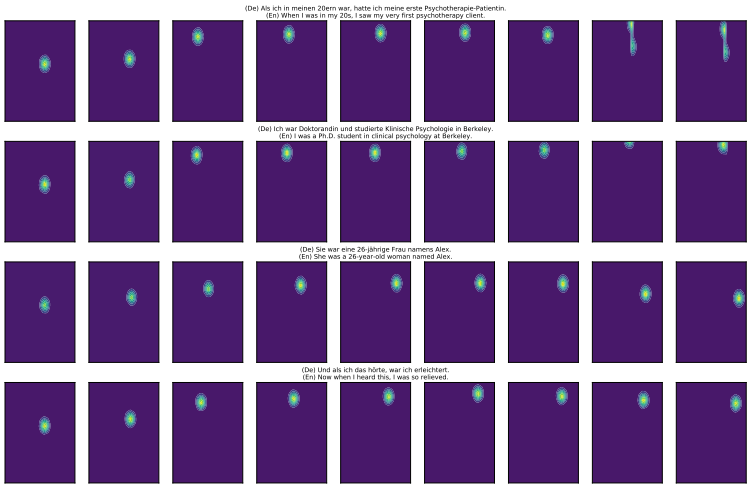
Flows	1	2	4	8	16	Baseline	Model
Planar	-0.003	0.003	0	0.001	-0.001	0	*GNMT*
IAF	0	-0.001	0	-0.003	0		



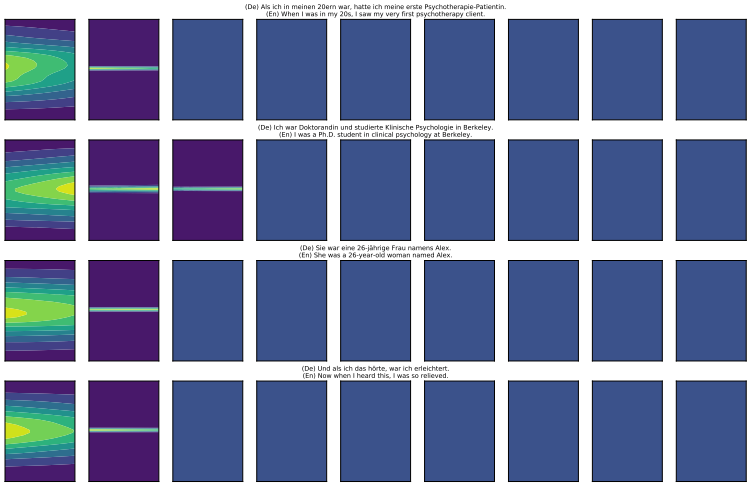
(a) GNMT with planar flows



(b) GNMT IAF flows



(a) VNMT with planar flows



(b) VNMT IAF flows

Table 5.9: Kl divergence of GNMT models without language model training.

Latent Dimension: 2

Flows	**1**	**2**	**4**	**8**	**16**	**Baseline**	**Model**
Planar	0	0	0.001	0.002	0	0	GNMT
IAF	0	0	0	0	0		

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**Baseline**	**Model**
Planar	0.006	0.003	0.004	0.006	0.011	0	GNMT
IAF	0.002	0.002	0.002	0.006	0.007		

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**Baseline**	**Model**
Planar	0.006	0.007	0.007	0.012	0.013	0	*GNMT*
IAF	0.005	0.004	0.005	0.005	0.013		

Chapter 6

Conclusion

everything here right now is complete crap.... In this work, we have considered normalizing flows in several settings and their impact on translation performance. From our experience, and based on our results, we would not recommend normalizing flows as a component to add on existing LVNMT systems. They introduce a number of additional hyper-parameters and added computation costs that for their performance gains are trivial compared to other hyperparameters. We did observe minute improvements, but visualizing a 2D latent space of our variable suggests that our flows largely still find uni-modal representations.

We believe that repeated executions of our experiments would verify our hypothesis that the performance gains with including flows is statistically insignificant.

This conclusion does not necessarily mean that normalizing flows are necessarily useless in translation. It simply indicates more wholesome considerations must be taken, such as the end-to-end flow model considered in non-autoregressive translation system [18].

For instance, one future work direction could be expanding our work without attention. Perhaps a normalizing flows based attention method could provide robust performance.

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv e-prints*, abs/1409.0473, Sept. 2014. URL https://arxiv.org/abs/1409.0473. → pages 5, 12, 13, 14, 28

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015. URL http://arxiv.org/abs/1511.06349. → page 22

[3] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *CoRR*, abs/1611.02731, 2016. → page 22

[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078. → page 6

[5] S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. *CoRR*, abs/1808.09381, 2018. URL http://arxiv.org/abs/1808.09381. → page 2

[6] B. Eikema and W. Aziz. Auto-encoding variational neural machine translation. *CoRR*, abs/1807.10564, 2018. URL http://arxiv.org/abs/1807.10564. → pages 2, 15, 18, 23, 24, 26, 28, 30

[7] M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/germain15.html. → page 22

[8] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1319–1327, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL http://proceedings.mlr.press/v28/goodfellow13.html. → page 14

[9] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. 2011. → page 6

[10] D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. *CoRR*, abs/1609.09106, 2016. URL http://arxiv.org/abs/1609.09106. → pages 10, 21

[11] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylDfnCqF7. → page 22

[12] E. Hoogeboom, J. W. T. Peters, R. van den Berg, and M. Welling. Integer discrete flows and lossless compression. *CoRR*, abs/1905.07376, 2019. URL http://arxiv.org/abs/1905.07376. → pages 3, 11

[13] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10215–10224. Curran Associates, Inc., 2018. → page 3

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114. → pages 2, 9, 25

[15] D. P. Kingma, T. Salimans, and M. Welling. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934, 2016. → pages 3, 11, 19, 22, 23, 26

[16] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. ISBN 0521874157, 9780521874151. → pages 2, 5

[17] P. Koehn. Neural machine translation. *CoRR*, abs/1709.07809, 2017. → pages 5, 7, 14

[18] X. Ma, C. Zhou, X. Li, G. Neubig, and E. Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow, 09 2019. → pages 3, 39

[19] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 2338–2347. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6828-masked-autoregressive-flow-for-density-estimation.pdf. → page 11

[20] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi:10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135. → page 6

[21] M. Post. 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, 2018. URL http://aclweb.org/anthology/W18-6319. → page 24

[22] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/rezende15.html. → pages 3, 10, 11, 19, 20, 21, 23, 26

[23] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL http://proceedings.mlr.press/v32/rezende14.html. → pages 2, 9, 25

[24] P. Schulz, W. Aziz, and T. Cohn. A stochastic decoder for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1243–1252, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1115. → pages 2, 3, 26

[25] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. *CoRR*, abs/1511.06709, 2015. URL http://arxiv.org/abs/1511.06709. → page 2

[26] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL http://arxiv.org/abs/1508.07909. → page 24

[27] H. Shah and D. Barber. Generative neural machine translation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1346–1355. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7409-generative-neural-machine-translation.pdf. → pages 2, 18, 23, 24

[28] T. Shen, M. Ott, M. Auli, and M. Ranzato. Diverse machine translation with a single multinomial latent variable, 2019. URL https://openreview.net/forum?id=BJgnmhA5KQ. → page 2

[29] C. K. Sø nderby, T. Raiko, L. Maaløe, S. r. K. Sø nderby, and O. Winther. Ladder variational autoencoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3738–3746. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6275-ladder-variational-autoencoders.pdf. → page 22

[30] J. Su, S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang. Variational recurrent neural machine translation. *CoRR*, abs/1801.05119, 2018. URL http://arxiv.org/abs/1801.05119. → page 2

[31] E. Tabak and E. Vanden Eijnden. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010. ISSN 1539-6746. → page 3

[32] E. G. Tabak and C. V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2): 145–164, 2013. doi:10.1002/cpa.21423. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21423. → page 3

[33] J. M. Tomczak and M. Welling. Improving variational auto-encoders using householder flow. *CoRR*, abs/1611.09630, 2016. → pages 3, 10, 11, 19, 23

[34] D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole. Discrete flows: Invertible generative models of discrete data. *CoRR*, abs/1905.10347, 2019. URL http://arxiv.org/abs/1905.10347. → pages 3, 11

[35] R. van den Berg, L. Hasenclever, J. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018. → pages 10, 11

[36] R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *UAI*, 2018. → pages 3, 19, 23, 26

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762. → pages 5, 12, 27

[38] J. Xu and G. Durrett. Spherical latent spaces for stable variational autoencoders. *CoRR*, abs/1808.10805, 2018. URL http://arxiv.org/abs/1808.10805. → page 30

[39] B. Zhang, D. Xiong, and J. Su. Variational neural machine translation. *CoRR*, abs/1605.07869, 2016. URL http://arxiv.org/abs/1605.07869. → pages 2, 15, 17

[40] S. Zhao, J. Song, and S. Ermon. Infovae: Information maximizing variational autoencoders. *CoRR*, abs/1706.02262, 2017. URL http://arxiv.org/abs/1706.02262. → page 22

[41] Z. M. Ziegler and A. M. Rush. Latent normalizing flows for discrete sequences. *CoRR*, abs/1901.10548, 2019. → pages 2, 3, 23

Appendix A

Supporting Materials

This would be any supporting material not central to the dissertation. For example:

- additional details of methodology and/or data;
- diagrams of specialized equipment developed.;
- copies of questionnaires and survey instruments.

Table A.1: List of hyperparameters used for experiments

Optimization Parameters	
Optimizer	Adam
Learning Rate	0.0003
KL Annealing Schedule	80,000 steps
Word Dropout Rate	0.1
Clip nNorm	1.0
Mini Batch Size	64
Number of Samples (ELBO)	10
Model Parameters	
Source Embedding Size	256
Target Embedding Size	256
Encoder Hidden Dimensions	256
Number of Encoder Layers	1
Decoder Hidden Dimensions	256
Number of Decoder Layers	1
Dropout	0.5
Z dim (latent variable)	2 , 218, 256
Global Attention Mechanism	
Key Size	512
Query Size	256
I.A.F Details	
Autoregressive NN	320 Units
Planar Flows Details	
Hidden layer dimensions	150

Table A.2: IWSLT sentence counts for De–En language pair. Counts represent actual number of sentences we used in our analysis. Values in parentheses represent full counts in dataset.

De–EN	
Train	182999 (233213)
Dev	1873 (2052)
Test	9101 (9773)