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)

February 2020

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, Linguistics and Information Science
Supervisor

Mark Schmidt, Computer Science
Supervisor

Additional Supervisory Committee Members:

Leonid Sigal, Computer Science
Supervisory Committee Member

Abstract

Natural language processing (NLP) has pervasive applications in everyday life, and has recently witnessed rapid progress. Incorporating latent variables in NLP systems can allow for explicit representations of certain types of information. In neural machine translation systems, for example, latent variables have the potential of enhancing semantic representations. This could help improve general translation quality. Previous work has focused on using variational inference with diagonal covariance 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 enables more flexible posterior distribution estimates by introducing a change of variables with invertible functions. They have previously been successfully used in computer vision to enable more flexible posterior distributions of image data. In this work, we investigate the impact of normalizing flows in autoregressive neural machine translation systems. We do so in the context of two currently successful approaches, attention mechanisms, and language models. Our results suggest that normalizing flows can improve translation quality in some scenarios, and require certain modelling assumptions to achieve such improvements.

Lay Summary

In this work, we consider the question of how to better encode complex information between languages (such as the hidden 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 distributions 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 suggest normalizing flows can benefit existing machine translation systems, but require particular modelling design to be better utilized.

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. Aaron Mischkin provided code snippets to help visualize the latent variables. 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, and 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

Acknowledgements

Thank you to my supervisors Muhammad Abdul-Mageed and Mark Schmidt for their support throughout this whole project. Thank you to Frank Wood for several informative discussions on focusing on latent variable models for machine translation. Thank you to William Harvey who helped with understanding the *Pyro* probabilistic programming language in which all models were implemented. 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

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 [28, 31, 42], demonstrating robustness to domain mis-match between training and test data [6], as well as enabling word level imputation for noisy sentences [28].

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 [25, 29]. Generating semantically meaningful sentences is a useful property, because research has shown that synthetically generated *bi-text* can improve translation system quality [5, 26]. 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, 24]. In the context of translation, this involves maximizing the log-likelihood of the conditional distribution $p(y|x,z)$ where y is the target language sentence, x is the source language sentence, and z is the introduced latent variable. Authors have assumed the variational posterior distribution is a Gaussian with diagonal covariance and learn a variational distribution $q_\phi(z \mid \cdot)$ 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 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 [44]. These results suggest that assuming the latent factors follow an isotropic Gaussian distribution is not representative 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 [33, 34]. This change of variables is achieved by invertible functions to transform samples from a chosen base distribution [23]. This variational approach has the added benefit of empirical findings showing more accurate approximations of target posterior distributions when such distributions are known [23].

In the literature, normalizing flows work has seen a number of successes in computer vision, and more recently in natural language processing. Particularly with the task of image generation, normalizing flows have been successful in producing high resolution images [13, 15, 35, 38]. Schulz et al. [25] 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, 36]. Continuous normalizing flows have been used for nonautoregressive language modelling as one means to produce meaningful sentences with faster decoding

[1] Some condition on both the target and source sentence [6, 28, 42], just the target sentences [25], or even just the source [6].

[44]. Most closely related to our work is the work of Ma et al. [19] who created a non-autoregressive normalizing flow machine translation system. Our work differs from Ma et al. [19] 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.

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 find that the direct contribution of the latent variable to final translation performance to be marginal overall for systems trained to include such variables.

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 about the particular NMT architecture considered in this work are discussed in Chapter 3.

Neural Machine Translation

Statistical machine translation (SMT) is a field which applies statistical methods to train computer systems that perform language translation. Historically these methods have typically involved learning word-level mappings, 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 SMT literature, NMT refers specifically to training SMT systems which incorporate neural networks as the primary model in the system.

NMT systems have achieved state of the art (SOTA) results across a variety of language pairs over alternative approaches in the SMT literature [1, 17, 39]. For human understanding, the goal is to correctly translate from source language x (for example, English) to target language y (for example, German) such that the sentence is coherent and captures the meaning of the source sentence.[1] Capturing

[1]There are other criteria for defining quality in translations including concepts such as adequacy, fidelity, and fluency [16, 21].

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 $p(y \mid x)$:

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

Here, θ represents the parameters of the NMT model, $y_{<i}$ refers to conditioning on all previous words excluding i, and T is the sequence length.

There are a variety of hyperparameter choices when building an NMT system. One core component of many SOTA systems are 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. The hidden representation can be viewed as the network's *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

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

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

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

In these equations, 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 [40]. 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 its own hidden state, to help inform the decoding process for target sentence 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*, or *label bias*, problem in which case the most likely words at the current time step lead to a series of unlikely words later in translation [17, 18]. 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 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.

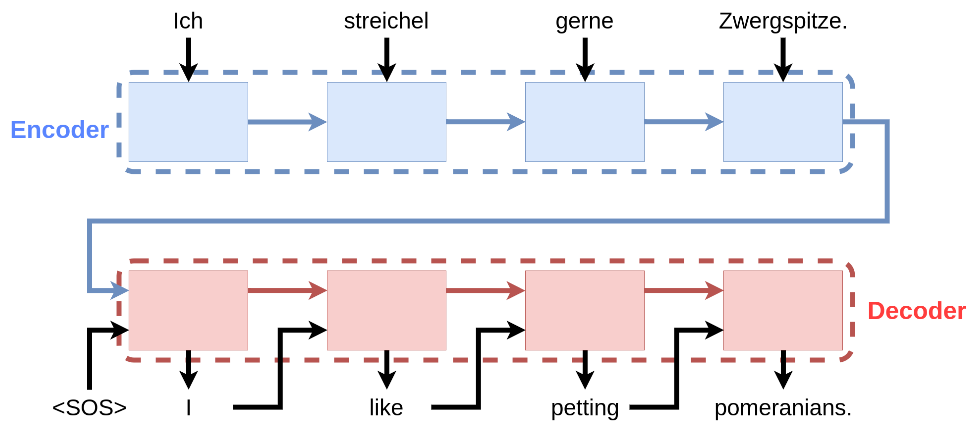

Figure 2.1: Simplest formulation of a SEQ2SEQ model in NMT [32].

An important aspect of VAEs is amortized inference, which involves introducing functions $f(x)$ that produce distribution parameters for each sentence pair in our case.[3] The motivation for employing amortized inference is eliminating the need to learn separate distribution parameters for each sentence pair. Otherwise, each sentence pair 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) represents 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.

[3]More generally, the amortization function learn distribution per datum, which could be of any modality or appropriate representation depending on the application.

Instead, researchers often solve this problem with optimization. This involves 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 $q(z \mid X)$:

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

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 (in other words, they 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 *re-parametrization trick* in which the variational distribution is rewritten as a function and sampling is done from some surrogate distribution [14, 24] . Here, we show this approach for the Gaussian with mean μ and diagonal covariance σ conditioned on x

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

Here, $\circ$ is an element-wise operation between each dimension $\sigma_\phi(x)$ and ε. 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 $p(z_k)$:

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

Here, $\circ$ 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

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

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 flow models are optimized in the literature.

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

$$\mathbb{E}_{q_\phi(z_0 \mid X)}[log(p_\theta(X \mid z)] - KL(q_\phi(z_0 \mid X)||p(z_k)) + E_{q_\phi(z_0 \mid X)}\left[\sum_{i=1}^{k} log\left|\frac{\delta f_i}{\delta z_{i-1}}\right|\right]. \tag{2.11}$$

Previous works typically employ hyper-networks [10] to amortize the parameters of the normalizing flows instead of directly optimizing them [23, 35, 37]. In the VI setting this enables the variational distribution q to capture possible posterior representations beyond those achievable with the base distribution.

Alternatively, normalizing flows enable direct optimization of the log probability of the 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, 20]; flows which by construction have analytic Jacobians [23, 37], or volume preserving flows which have the Jacobian equal to one [12, 35, 36]. Note that these groupings do not 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 Figure 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 [39], 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 and 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. 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

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

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 (in other words, 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 $s_0 = \tanh(\text{affine}(h_T))$. *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 j in the target sentence, the decoder includes a MLP which uses the maxout activation over the hidden state values [8]:

$$\text{maxout}(h_j) = [\max(h_j^{2k-1}, h_j^{2k})]_{k=1}^{K/2}. \tag{3.4}$$

For clarity, the maxout operation takes the maximum value between h_j^{2k-1} and h_j^{2k} which are positions in the input vector h_j of length K. These maxout values are then fed to a final layer which represents the conditional distribution:

$$t_j = \text{maxout}(\text{affine}([y_{j-1}; c_j; s_j])), \tag{3.5}$$

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

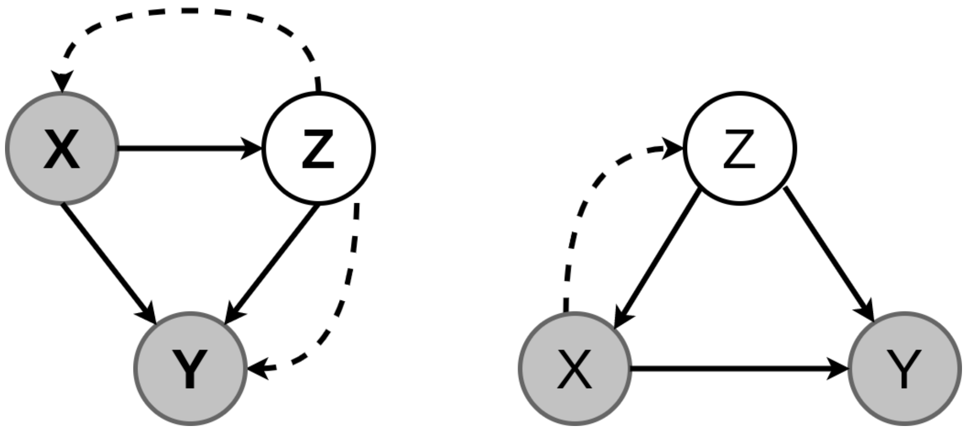

Figure 3.1: Graphical representation of LVNMT systems we consider. Left, is the discriminative model [42]. Right, is the joint or generative model [6]. Dashed lines represent the variational distribution. Solid lines are the model.

Representation of Latent Variables

Both models we consider follow the same process to encode 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 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, 42]

Discriminative Translation Model

The discriminative LVNMT model we considered is a variation of the work by Zhang et al. [42]. It models the conditional distribution $p(y \mid x)$ which is the typical 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 $q_\phi(z \mid x, z)$ where as [42] samples from the prior $p_\theta(z \mid x)$:

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

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 sentence pair's latent distributions apart arbitrarily 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 [42].

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

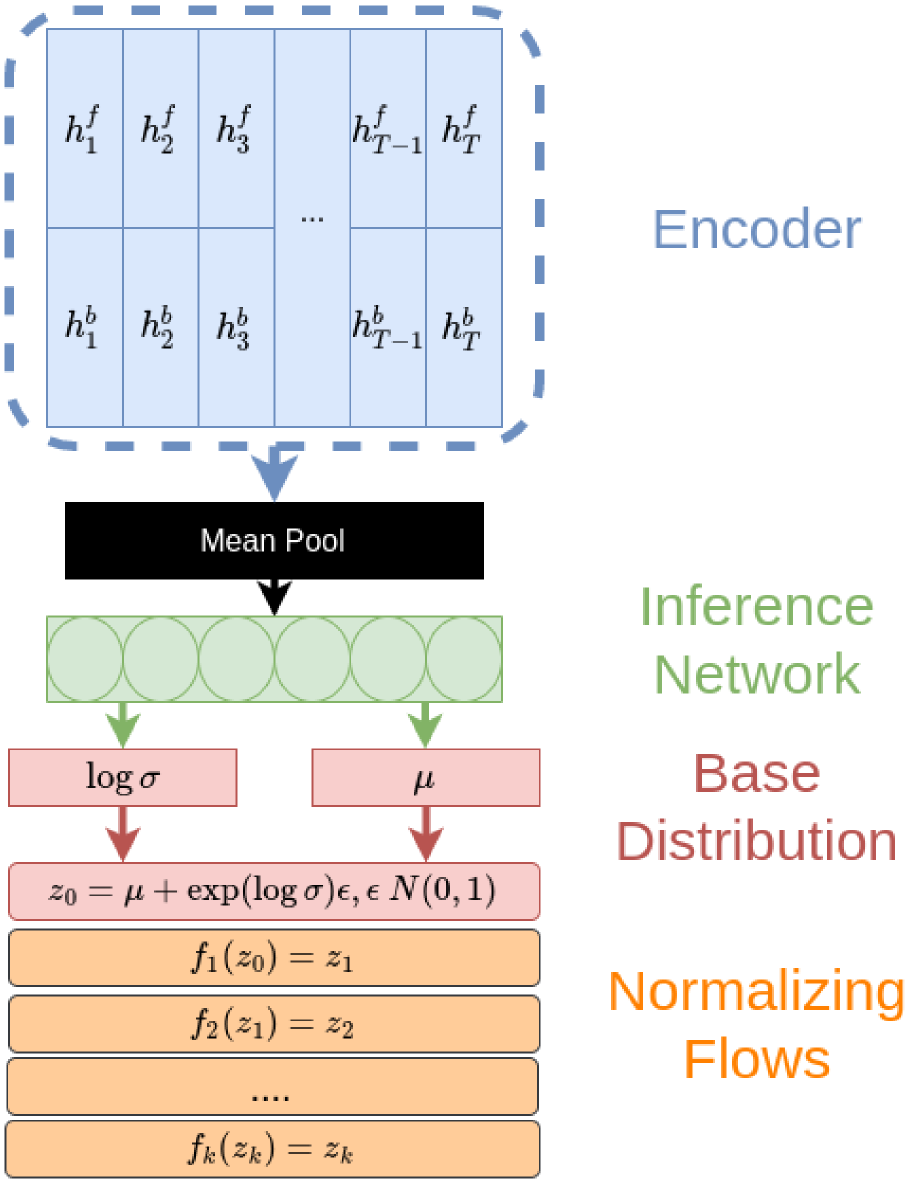
h_1^f h_2^f h_3^f ... h_{T-1}^f h_T^f
h_1^b h_2^b h_3^b h_{T-1}^b h_T^b
Encoder
Mean Pool
Inference Network
$\log\sigma$
μ
Base Distribution
$z_0 = \mu + \exp(\log\sigma)\epsilon, \epsilon\ N(0,1)$
$f_1(z_0) = z_1$
$f_2(z_1) = z_2$
....
$f_k(z_k) = z_k$
Normalizing Flows

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.

inputs to the decoder at each time-step in the decoding process:

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

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 [28]. 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:

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

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 most 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. The motivation for optimizing this component is to help improve translation quality by adding additional inductive bias for our latent space to encode share information between source and target sentence.

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. We visualize this process in Figure 3.2 in orange.

We follow previous research which makes flows data dependent [15, 23, 35, 38]. 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 case for the discriminative model, and is based on the derivation from Rezende and Mohamed [23], Section 4.2:

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

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 decoding time, we replace the variational distribution with the prior which has 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. Note, however, 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 [23]. 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 an auto-regressive network which was proposed in the work of Germain et al. [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 auto-regressive network are shared between sentence pairs. 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 Gaussian as the prior. Part of this behaviour has been accredited 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. [43] 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, 30] and word dropout [2].

KL-annealing typically involves annealing the weight β of the divergence term in the ELBO:

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

In normalizing flows research, it is often cited that KL-annealing helps improve performance even without strong decoders [15, 23, 35, 38, 44]. 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 its 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, 28].

[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 consider settings to empirically evaluate the impact of latent variables on LVNMT systems with attention and language models. We include hyperparameter values in the supplement material Table A.1, 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. As part of this work, we release our code.[1]

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 helpful in the discriminative NMT case [28].

We conducted our experiments with the IWSLT 2016 data sets available in the *torchtext* library.[2] We evaluated our models with the German–English (De–En) language pair. We chose this language direction because we could more naturally evaluate the translations.[3] This dataset consists of 233,213 training, 2052 validation, and 9773 test sentences. We measure performance with the raw BLEU score implementation available in *sacreBLEU* [22]. For all experiments, we keep the

[1]Experiment code: https://github.com/gamerDecathlete/NormalizingFlowsNMT
[2]https://github.com/pytorch/text
[3] We did not do any formal human evaluation of translation quality.

random seed fixed to a single value due to stochastic variables.

We represented our vocabulary for each language with byte-pair encoding (BPE) [27]. We used vocabularies of 10,000 BPEs per language. We found that larger vocabularies resulted in sub-word units that occurred infrequently enough to be uninformative from a practical learning perspective. We performed BPE using the *SentencePiece* library.[4] We trained our NMT models on sequences of maximum length 50. We used beam search with a beam width of 10, 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 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.

Our baselines include the LVNMT models we described in Chapter 3 with just the diagonal Gaussian for the variational distribution. Our baselines optimize the ELBO with an equal 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, 24]. 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 training. Each model was trained for 47 epochs.[5] The boldfaced results represent the best performing version of a model for the given latent dimensions and flow type.

Regardless of flow type, we found our generative model to perform better for

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

[5]An epoch here refers to training on all mini-batches before reshuffling the sentence pairs.

Table 5.1: BLEU score for our models with normalizing flows for German-English (De–En) translation. The best performances are in bold as compared to differing numbers of flows and the baseline. Yellow rows represent our VNMT model results, and red our GNMT results for differing number and type of flows.

Latent Dimension: 128

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	18.84	18.84	19.14	19.02	**19.18**	18.768	VNMT
IAF	18.89	18.96	**19.29**	18.81	18.92		
Planar	20.59	20.60	20.48	20.55	20.66	**20.73**	GNMT
IAF	20.64	20.64	20.51	20.65	20.50		

Latent Dimension: 256

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	18.93	**19.26**	19.02	18.80	18.82	18.76	VNMT
IAF	18.95	**19.17**	19.02	18.99	18.77		
Planar	20.55	**20.67**	20.54	20.51	**20.67**	20.66	GNMT
IAF	20.85	**20.86**	20.64	20.62	20.66		

any number of flows or latent space size. Even our worst performing GNMT model (z=128, with 4 Planar flows) provides at least 1.20 BLEU score above the best performing discriminative model (z=128, with 4 inverse autoregressive flow (IAF)). This result seems congruent with previous research suggesting joint modelling can be more effective than the simpler discriminative representation [6].

We found overall that VNMT benefited more from the inclusion of normalizing flows than GNMT. For VNMT, even a single normalizing flow results in improvements. Between flows, we do not note any clear distinctions for the number of flows or type of flow in the VNMT case although best performance included more than just one flow. 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.

Overall these results suggest that flows can indeed be added to existing models and provide benefit to the final translation quality. In several instances our flows models outperform baseline results. Here, it seems the input feeding VNMT may benefit more than the initialization approach in our GNMT model.

Importance of Attention

In this section, to investigate the utility of including latent variables, we train simplified versions of our LVNMT systems which do not include the attention mechanism. Our motivation for this experiment is to tease apart 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 due to their success in SOTA models [1, 39]. However, we hypothesize that if the latent variable can encode useful information in translation, then NMT models will still benefit from this global latent information. As an extension, normalizing flows will then enable these variables to be more beneficial by making the latent variable distribution more flexible.

We include the latent variable in the generator network as a substitute for attention, which is the same design choice as Bahdanau et al. [1] for including 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 compare to 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. In our tables these two modified models are the No Latent (NL) baselines we compare our latent variable models against. Table 5.2 show's results for our modified models and the baselines when evaluated with BLEU score. The best performances are bold. They were chosen by comparing models with differing numbers of the same flow, and the baselines.

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 achieving close to our best GNMT baseline. Unfortunately, the IAF VNMT models did not improve as much. There could be several possible reasons. One interpretation we suspect is related to the data conditioning approach

Table 5.2: Results for translation systems without attention mechanism. Baseline includes models with normalizing flows and deterministic version of model excluding latent variables. VNMT results are in yellow, and GNMT results are in red with best models. Best models across number and type of flow are in bold. No Latent (NL) are models trained without z included.

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**NL (Baseline)**	**Model**
Planar	6.61	6.64	6.63	6.93	**6.78**	6.25	6.38	VNMT
IAF	**6.42**	6.32	6.28	5.98	5.92			
Planar	7.2	7.22	7.19	7.11	7.25	**7.37**	7.16	GNMT
IAF	7.31	7.37	7.33	7.18	**7.41**	7.37		

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**NL (Baseline)**	**Model**
Planar	6.31	6.81	6.91	7.37	**7.38**	6.43	6.38	VNMT
IAF	6.45	6.47	6.4	6.23	**6.71**			
Planar	7.4	7.22	7.33	7.26	7.07	**7.51**	7.16	GNMT
IAF	7.35	7.34	7.28	7.04	7.36			

of planar flows compared to IAF. As our planar flows provide per-sentence flow parameters, this enables them to provide more unique distributions per sentence. In comparison, the IAF is unable to capture this more nuanced information by simply conditioning on a context vector. This argument has been pointed out in previous research for other types of flow [38].

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 initializes the translation system beginning in the encoder network as compared to being passed as input during decoding. Based on our results, it seems this design is less effective compared to the input feeding approach of VNMT to utilize latent variables. We conjecture this is likely without explicitly passing z to the decoder, GNMT is relying on the GRUs to implicitly keep this information from beginning to end.

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 by optimizing the model 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 Table 5.4 shows the difference in BLEU score when z is the 0 vector for our models with attention.

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-zero, but even in several instances a lower KL divergence still leads to loss of performance with z (see VNMT, z=256 results for 8 flows vs. 16 flows). These substitution approaches have also previously been shown to provide minimal or mixed results in terms of comparative performance [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.[6] 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 affected by initialization for final performance [41].

[6]In a similar setting, one bug we found in our implementation was excluding the projection layer and found similar performance drops to those without IAF flows results.

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

Latent Dimension: 128 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	2.16	1.27	1.18	1.04	1.63	1.04	VNMT
IAF	0.68	1.35	1.32	1.6	1.06		
Planar	3.23	3.15	2.36	2.82	3.64	4.39	GNMT
IAF	4.31	4.36	4.19	4.43	4.14		

Latent Dimension: 256 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.92	2.35	2.37	1.11	1.27	2.0	VNMT
IAF	2.41	1.94	1.14	1.1	1.1		
Planar	4.34	3.93	3.36	2.75	3.62	3.84	GNMT
IAF	3.87	3.74	4.0	3.99	3.98		

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 to more utility in translation performance.

When we perform the same experiment with our LVNMT systems without attention, we see more dependence on the latent variable for all models. Our results are in Table 5.5 and 5.6. In most cases with planar flows, we see our models depend more on the latent variable over the baseline without planar flows. Our IAF flow 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.

Language Modelling Performance

For most of our previous experiments, we have found that GNMT largely shows only slight gains from normalizing flows. As previously discussed in chapter 3,

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: 128 (BLEU Difference)

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	-0.29	-0.37	-0.06	-0.14	0.1	0.04	VNMT
IAF	18.64	12.17	15.52	17.4	17.56		
Planar	0.14	0.08	0.02	0.06	0.1	0.08	GNMT
IAF	0.12	0.08	-0.01	0.1	-0.01		

Latent Dimension: 256 (BLEU Difference)

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	0.02	-0.07	-0.03	-0.24	0.16	-0.11	VNMT
IAF	12.22	17.27	18.5	10.84	17.42		
Planar	0.1	0	0.11	0.15	0.04	0.09	GNMT
IAF	0.04	0.09	0.04	0.06	0.19		

Table 5.5: Average KL divergence for test set for models without attention. VNMT should generally be lower, where as GNMT shoulder generally have non-zero KL terms.

Latent Dimension: 128 (KL Divergence)

Flows	1	2	4	8	16	0 (Baseline)	NL (Baseline)	Model
Planar	0.22	0.56	0.76	0.39	0.05	0.6	N/A	VNMT
IAF	0.59	0.39	0.2	0.18	0.23			
Planar	4.28	4.48	3.84	4.12	4.58	2.23	N/A	GNMT
IAF	3.38	2.97	3.89	4.57	4.53			

Latent Dimension: 256 (KL Divergence)

Flows	1	2	4	8	16	0 (Baseline)	NL (Baseline)	Model
Planar	0.35	0.04	0.3	0.16	0.07	0.72	N/A	VNMT
IAF	0.74	0.08	0.7	0.25	0.22			
Planar	4.1	4.26	3.57	3.54	3.65	4.19	N/A	GNMT
IAF	3.99	3.19	3.99	3.12	3.45			

Table 5.6: Change in BLEU score without attention models. Positive values indicate latent variable z is important for translation system. All models seem to depend on z when attention is unavailable.

Latent Dimension: 128 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**NL (Baseline)**	**Model**
Planar	3.02	3.08	2.73	3.51	3.5	0.6	N/A	VNMT
IAF	2.33	1.75	1.82	1.59	0.88			
Planar	0.91	0.93	0.7	0.49	0.72	0.46	N/A	GNMT
IAF	0.88	0.59	0.88	0.79	1.1			

Latent Dimension: 256 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**NL (Baseline)**	**Model**
Planar	3.15	4.09	4.51	4.88	4.98	1.79	N/A	VNMT
IAF	3.03	3.79	3.98	2.29	3.09			
Planar	0.76	0.98	0.29	0.48	0.51	0.68	N/A	GNMT
IAF	0.66	0.5	0.63	0.41	0.49			

GNMT trains a language model as part of the system. In this section, we train GNMT models that do not optimize this language model as part of the training procedure. The goal of this experiment is to test the impact of normalizing flows in the absence of language model for GNMT.

Table 5.7 shows our results with attention but no language model (in blue) compared against our GNMT results from Table 5.1. Our results show similar performance to training our GNMT model with the language model. At a 128 dimensional latent space our baseline still outperforms all flow models, as well as the training of GNMT with a language model. At a 256 latent dimensional space, we see results close to our language model training, but with slight decreases in performance. Given how close these numbers are, these results suggest that z is not the key factor for performance gains in our translation system for GNMT. In addition, when we measure the KL divergence (see Table 5.9) we find that the latent variable has largely collapsed to the prior in many cases. Measurements of change of BLEU and without BLEU are in supplementary material Tables A.4, A.5,and A.6 as these show largely the same behaviour.

Table 5.8 shows results of GNMT with no language model training along with

Table 5.7: BLEU scores for GNMT without language model training. Bold entries are the best performing models. We include previous GNMT results in red to compare with GNMT without language model training (blue).

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	20.46	20.42	20.5	20.65	20.41	**20.77**	GNMT (No LM)
IAF	20.48	20.45	20.71	20.45	20.38		
Planar	20.59	20.60	20.48	20.55	20.66	**20.73**	GNMT
IAF	20.64	20.64	20.51	20.65	20.50		

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	20.32	20.36	20.47	20.21	**20.65**	20.59	GNMT (No LM)
IAF	20.56	**20.64**	20.51	20.54	20.41		
Planar	20.55	**20.67**	20.54	20.51	**20.67**	20.66	GNMT
IAF	20.85	**20.86**	20.64	20.62	20.66		

attention removed. Here, we see that our flow models show some improvement over baselines unlike the GNMT case. The language modelling seems to be more important as GNMT does outperform our GNMT without language modelling in several cases. Overall though, our best performances in this table is GNMT without language modelling and a single planar flow. We did observe *posterior collapse* (included in supplementary material) which might suggest these gains are not directly because of latent variable z. This could be simply a by-product of the stochastic behavior due to the training procedure. It has been previously suggested that the stochastic behaviour injected by latent variables does help with training performance [42].

Table 5.8: GNMT training results without attention or language model training. In several instances, GNMT without language model training outperform models which optimize the language model.

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	7.23	7.03	7.25	**7.32**	7.27	7.31	GNMT (No LM)
IAF	**7.5**	7.42	7.37	7.31	7.16		
Planar	7.2	7.22	7.19	7.11	7.25	**7.36**	GNMT
IAF	7.31	7.37	7.33	7.18	**7.41**	7.36	

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	**7.58**	7.45	7.28	7.44	7.33	7.21	GNMT (No LM)
IAF	7.25	7.35	7.32	7.08	7.47		
Planar	7.4	7.22	7.33	7.26	7.07	**7.51**	GNMT
IAF	7.35	7.34	7.28	7.04	7.36		

Table 5.9: KL divergence of GNMT models without language model training. Typically a near 0 KL divergence with stationary priors indicates *posterior collapse* has occurred.

Latent Dimension: 128 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.01	0.0	0.0	0.01	0.01	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.01	0.01		
Planar	3.23	3.15	2.36	2.82	3.64	4.39	GNMT
IAF	4.31	4.36	4.19	4.43	4.14		

Latent Dimension: 256 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.01	0.01	0.01	0.01	0.01	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.0	0.01		
Planar	4.34	3.93	3.36	2.75	3.62	3.84	GNMT
IAF	3.87	3.74	4.0	3.99	3.98		

Chapter 6

Conclusion

In this work we considered the inclusion of normalizing flows in existing LVNMT models. Whether with attention or without attention, our VNMT model seemed to benefit from the inclusion of normalizing flows. Particularly when attention was removed, our probing of the latent z found that models would depend more so on the latent variable z for translation quality. In contrast, GNMT did not benefit as much in any setting we included normalizing flows. Although we saw more dependence on the latent z when removing attention for GNMT, our baseline models often had better final translation performance. One explanation from this came to light from removing the optimization of the language model, in which case our GNMT model completely ignored the latent z for translation. This might suggest the initialization approach to including z in our GNMT implementation is less effective than the input-feeding approach of VNMT for enabling normalizing flows to improve translation performance.

We caution interpretations of our results given the breadth of hyper-parameters to consider, such as the amount of word-dropout, KL-annealing, or even MC samples used for approximating the ELBO. Given these other factors, it can be difficult to fully give credit to normalizing flows themselves for performance gains. Understanding how different annealing schedules might impact normalizing flows is a promising future research direction itself, particularly given understanding KL annealing is an active area of research [11, 41].

Despite these considerations, we believe normalizing flows do have much future potential benefits in machine translation systems, particularly given previous success in non-autoregressive translation [19]. However, normalizing flows are not necessarily something that can be added for a guaranteed performance benefit. To properly see benefit from normalizing flows, additional considerations must be taken into account. One potential future work is extending our findings to sequential latent variable NMT systems. These have previously been introduced to help with longer sentence translation as well as diversifying translation. Another direction could be considering joint modelling of flow based models to generate synthetic sentence paired data. Synthetically generated data has been studied in machine translation as one approach to improve existing systems performances [26]. Most recently, discrete flows have been proposed showing improvement on existing flow based language modelling results [36]. Their the authors cite the vocabulary size as a limiting factor, but from our results with a small BPE vocabulary it may be plausible to still utilize these flows effectively for translation.

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473. → pages 4, 11, 12, 13, 27

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi:10.18653/v1/K16-1002. URL https://www.aclweb.org/anthology/K16-1002. → page 22

[3] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=BysvGP5ee. → page 22

[4] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 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, Oct. 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179. → page 5

[5] S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1045. URL https://www.aclweb.org/anthology/D18-1045. → page 1

[6] B. Eikema and W. Aziz. Auto-encoding variational neural machine translation. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 124–141, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4315. URL https://www.aclweb.org/anthology/W19-4315. → pages xi, 1, 2, 14, 15, 17, 23, 24, 26, 27, 29

[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 13

[9] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. 2011. → page 5

[10] D. Ha, A. Dai, and Q. Le. Hypernetworks. 2016. → pages 9, 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. → pages 22, 35

[12] E. Hoogeboom, J. W. T. Peters, R. van den Berg, and M. Welling. Integer discrete flows and lossless compression. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 12134–12144, 2019. URL http://papers.nips.cc/paper/9383-integer-discrete-flows-and-lossless-compression. → pages 2, 10

[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 2

[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 1, 8, 25

[15] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS16, page 47434751, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819. → pages 2, 10, 19, 22, 23

[16] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. ISBN 0521874157, 9780521874151. → pages 2, 4

[17] P. Koehn. Neural machine translation. *CoRR*, abs/1709.07809, 2017. → pages 4, 6, 13

[18] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML 01, page 282289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781. → page 6

[19] X. Ma, C. Zhou, X. Li, G. Neubig, and E. Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow, 09 2019. → pages 3, 36

[20] 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 10

[21] 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 4

[22] 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

[23] 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 2, 9, 10, 19, 20, 21, 23

[24] 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 1, 8, 25

[25] 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 1, 2, 44

[26] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1009. URL https://www.aclweb.org/anthology/P16-1009. → pages 1, 36

[27] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162. → page 25

[28] 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 1, 2, 17, 23, 24

[29] 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 1

[30] C. K. Sø nderby, T. Raiko, L. Maalø e, 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

[31] J. Su, S. Wu, D. Xiong, Y. Lu, X. Han, and B. Zhang. Variational recurrent neural machine translation. In S. A. McIlraith and K. Q. Weinberger, editors, *AAAI*, pages 5488–5495. AAAI Press, 2018. URL http://dblp.uni-trier.de/db/conf/aaai/aaai2018.html#SuWXLHZ18. → page 1

[32] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf. → pages xi, 7

[33] 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 2

[34] 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 2

[35] J. M. Tomczak and M. Welling. Improving variational auto-encoders using householder flow. *CoRR*, abs/1611.09630, 2016. URL http://arxiv.org/abs/1611.09630. → pages 2, 9, 10, 19, 23

[36] D. Tran, K. Vafa, K. K. Agrawal, L. Dinh, and B. Poole. Discrete flows: Invertible generative models of discrete data. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 14692–14701, 2019. URL http://papers.nips.cc/paper/9612-discrete-flows-invertible-generative-models-of-discrete-data. → pages 2, 10, 36

[37] 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 9, 10

[38] R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference. In *UAI*, 2018. → pages 2, 19, 23, 28

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. pages 5998–6008, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf. → pages 4, 11, 27

[40] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS15, page 27732781, Cambridge, MA, USA, 2015. MIT Press. → page 6

[41] J. Xu and G. Durrett. Spherical latent spaces for stable variational autoencoders. 2018. → pages 29, 35

[42] B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530, Austin, Texas, Nov. 2016. Association for Computational Linguistics. doi:10.18653/v1/D16-1050. URL https://www.aclweb.org/anthology/D16-1050. → pages xi, 1, 2, 14, 15, 33

[43] S. Zhao, J. Song, and S. Ermon. Infovae: Balancing learning and inference in variational autoencoders. pages 5885–5892, 2019. doi:10.1609/aaai.v33i01.33015885. URL https://doi.org/10.1609/aaai.v33i01.33015885. → page 22

[44] Z. Ziegler and A. Rush. Latent normalizing flows for discrete sequences. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7673–7682, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/ziegler19a.html. → pages 2, 3, 23

Appendix A

Supporting Materials

Discussion on other latent dimensions

In this section we include our results when the latent space was set to 2. These results are available in Table A.3. 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 [25].[1] 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. We chose to exclude this result from the main thesis because to our knowledge there is no precedence for such performance gains, and further future work is necessary.

We also considered the same drop in performance scenario discussed in the main thesis. The KL divergence results are in Table A.4. The BLEU difference are in Table A.5. Here, we see similar trends as in the original models where performance drops are quite small with the latent variable z. For most VNMT models we again see that the latent variable seems to negatively impact performance with $z = 2$.

Additionally, we visualized the latent space of several sentence pairs as the latent dimensions were quite small. Our VNMT results are available in Figures A.3 and A.4. Our GNMT results are available in Figures A.1 and A.2. These plots

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

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 Norm	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, 128, 256
Global Attention Mechanism	
Key Size	512
Query Size	256
IAF Details	
Autoregressive NN hidden sizes	320, 320
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 use in our analysis when limiting max sentence length to 50. Values in parentheses represent full sentence counts of each dataset.

De–EN	
Train	182,999 (233,213)
Dev	1,873 (2,052)
Test	9,101 (9,773)

Table A.3: 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. This table is similar to Table 5.1 in main thesis, but includes result for latent dimensions set to 2.

Latent Dimension: 2

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	**19.06**	18.95	18.94	19.01	19.05	18.92	VNMT
IAF	18.89	18.48	18.803	**18.96**	18.67		
Planar	20.77	20.70	20.81	20.86	20.37	**21.02**	GNMT
IAF	20.61	20.52	20.58	20.55	20.46		

Latent Dimension: 128

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	18.84	18.84	19.14	19.02	**19.18**	18.768	VNMT
IAF	18.89	18.96	**19.29**	18.81	18.92		
Planar	20.59	20.60	20.48	20.55	20.66	**20.73**	GNMT
IAF	20.64	20.64	20.51	20.65	20.50		

Latent Dimension: 256

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	18.93	**19.26**	19.02	18.80	18.82	18.76	VNMT
IAF	18.95	**19.17**	19.02	18.99	18.77		
Planar	20.55	**20.67**	20.54	20.51	**20.67**	20.66	GNMT
IAF	20.85	**20.86**	20.64	20.62	20.66		

were generated by first encoding each sentence pair to produce the variational posterior parameters. With these parameters, we generated 10,000 samples from the distributions. Each plot is then a Gaussian kernel density estimation of the samples after being transformed by K normalizing flows with the base distribution on the far left, and $k = 8$ flows on the far right. These plots largely show us the expected behaviours of the transformations. In the planar flows cases we see mostly the distribution being shifted around in the latent space but otherwise remaining uni-modal. In the IAF cases we see interestingly that the normalizing flows are collapsing our 2D latent spaces into 1D. This may be an indication that the stochastic nature of the latent variable is actually negatively impacting the training as the IAF flows attempt to flatten the distribution.

Table A.4: Average KL divergence for the test set. For VNMT the KL term should be smaller meaning the distributions encode 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**	**0 (Baseline)**	**Model**
Planar	0.94	1.45	0.68	0.0	0.0	1.15	VNMT
IAF	1.15	1.56	1.4	0.85	1.56		
Planar	2.48	2.32	2.01	0.10	3.64	2.99	GNMT
IAF	0.0	2.36	3.91	2.09	2.63		

Latent Dimension: 128 (KL Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	2.16	1.27	1.18	1.04	1.63	1.04	VNMT
IAF	0.68	1.35	1.32	1.6	1.06		
Planar	3.23	3.15	2.36	2.82	3.64	4.39	GNMT
IAF	4.31	4.36	4.19	4.43	4.14		

Latent Dimension: 256 (KL Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.92	2.35	2.37	1.11	1.27	2.0	VNMT
IAF	2.41	1.94	1.14	1.1	1.1		
Planar	4.34	3.93	3.36	2.75	3.62	3.84	GNMT
IAF	3.87	3.74	4.0	3.99	3.98		

Table A.5: 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	0 (Baseline)	Model
Planar	0	-0.09	-0.06	0.04	0	-0.1	VNMT
IAF	-0.01	0.06	-0.1	-0.1	-0.11		
Planar	0.25	-0.03	0.11	0.04	0.30	0.24	GNMT
IAF	0	0.11	0.01	0.1	-0.02		

Latent Dimension: 128 (BLEU Difference)

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	-0.29	-0.37	-0.06	-0.14	0.1	0.04	VNMT
IAF	18.64	12.17	15.52	17.4	17.56		
Planar	0.14	0.08	0.02	0.06	0.1	0.08	GNMT
IAF	0.12	0.08	-0.01	0.1	-0.01		

Latent Dimension: 256 (BLEU Difference)

Flows	1	2	4	8	16	0 (Baseline)	Model
Planar	0.02	-0.07	-0.03	-0.24	0.16	-0.11	VNMT
IAF	12.22	17.27	18.5	10.84	17.42		
Planar	0.1	0	0.11	0.15	0.04	0.09	GNMT
IAF	0.04	0.09	0.04	0.06	0.19		

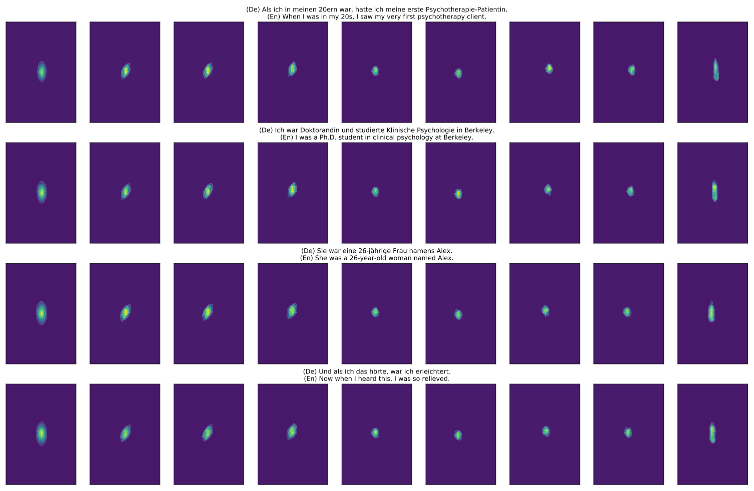

Figure A.1: One of our GNMT models with 2 dimensional latent space, and 8 planar flows. Each row is a sentence pair with an increasing number of flows applied from 0 through 8. The ordering is 0 flows on the far left, and 8 flows on the far right.

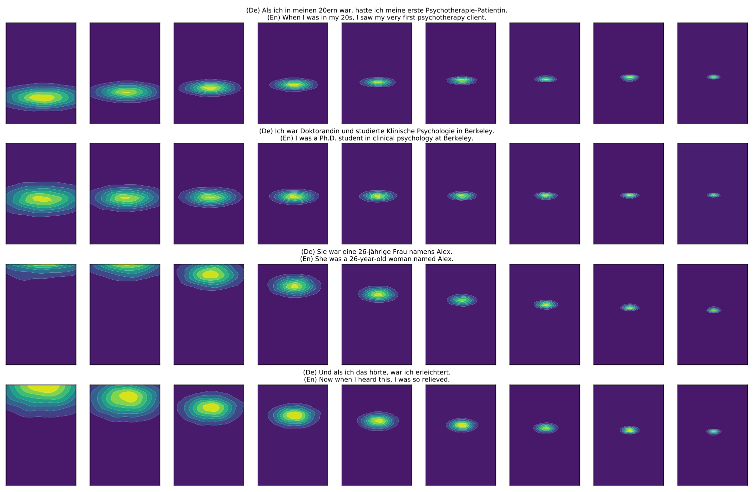

Figure A.2: One of our 2 dimensional latent variable GNMT models trained with 8 IAF flows. Each row is one sentence pair with an increasing number of flows applied to samples from the base distribution. Each row is a sentence pair with an increasing number of flows applied from 0 through 8. The ordering is 0 flows on the far left, and 8 flows on the far right.

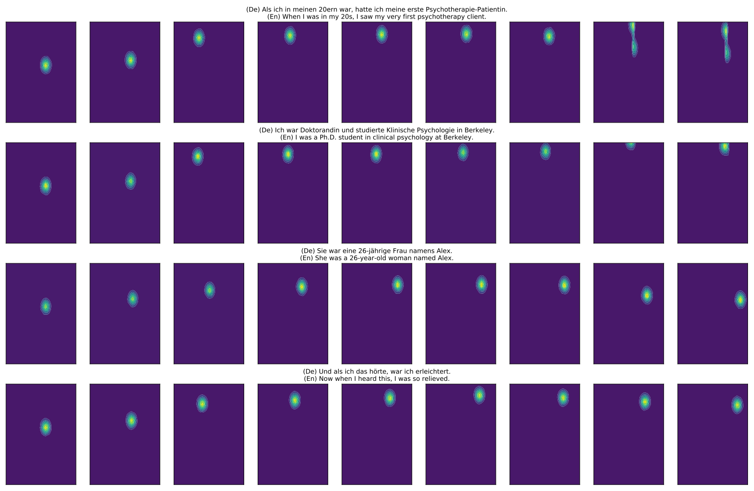

Figure A.3: Two dimensional latent variable VNMT with 8 planar flows. Captions are the sentence pairs. Each row is a sentence pair with an increasing number of flows applied from 0 through 8. The ordering is 0 flows on the far left, and 8 flows on the far right.

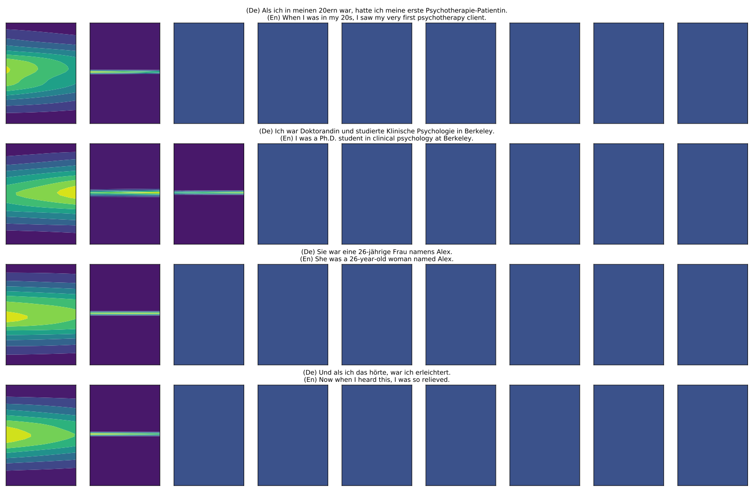

Figure A.4: Visualization of several sentence pairs latent spaces with VNMT model and 8 IAF flows. Solid blue images mean the distribution has been flattened into a line. Each row is a sentence pair with 0 flows on the far left and 8 flows applied on the right.

Table A.6: BLEU score difference of GNMT with attention when language model is not optimized during training. We include GNMT with language model training for reference. The comparison just shows that without the language model, GNMT does not incorporate information at all from the latent variable z.

Latent Dimension: 128 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.0	0.0	0.0	0.0	0.0	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.0	0.0		
Planar	0.14	0.08	0.02	0.06	0.1	0.08	GNMT
IAF	0.12	0.08	-0.01	0.1	-0.01		

Latent Dimension: 256 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.0	0.0	0.0	0.0	0.0	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.0	0.0		
Planar	0.1	0	0.11	0.15	0.04	0.09	GNMT
IAF	0.04	0.09	0.04	0.06	0.19		

Table A.7: KL divergence for GNMT models trained without attention or language model optimization. Results suggest posterior collapse has occurred as the values are almost all near 0.

Latent Dimension: 128 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.0	0.0	0.0	0.01	0.01	0.0	GNMT (No LM)
IAF	0.0	0.0	0.01	0.0	0.0		
Planar	4.28	4.48	3.84	4.12	4.58	2.23	GNMT
IAF	3.38	2.97	3.89	4.57	4.53		

Latent Dimension: 256 (KL Divergence)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.01	0.01	0.01	0.01	0.02	0.0	GNMT (No LM)
IAF	0.01	0.01	0.01	0.01	0.01		
Planar	4.1	4.26	3.57	3.54	3.65	4.19	GNMT
IAF	3.99	3.19	3.99	3.12	3.45		

Table A.8: Measure of performance drop when z is removed from trained system for GNMT without attention or language model optimization.

Latent Dimension: 128 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.0	0.0	0.0	0.0	0.0	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.0	0.0		
Planar	0.91	0.93	0.7	0.49	0.72	0.46	GNMT
IAF	0.88	0.59	0.88	0.79	1.1		

Latent Dimension: 256 (BLEU Difference)

Flows	**1**	**2**	**4**	**8**	**16**	**0 (Baseline)**	**Model**
Planar	0.0	0.0	0.0	0.0	0.0	0.0	GNMT (No LM)
IAF	0.0	0.0	0.0	0.0	0.0		
Planar	0.76	0.98	0.29	0.48	0.51	0.68	GNMT
IAF	0.66	0.5	0.63	0.41	0.49		