# Neural Multi-Language Generation using Variational Autoencoders

Author: *John Isak Texas Falk*
Supervisor: *Prof. David Barber*

A dissertation submitted in partial fulfillment

of the requirements for the degree of

**Master of Science**

of

**Computational Statistics and Machine Learning**.

The Centre for Computational Statistics and Machine Learning

University College London



September 5, 2017

# Abstract

A language model is a statistical model that parametrise a distribution over sentences. Latent variable models have found widespread use throughout machine learning due to its ability to encode information about data in an underlying smaller space. Applying latent variable models to language models enable us to generate novel sentences from a continuous stochastic latent variable. While powerful, latent variable models are notoriously hard to train due to intractability of the log-likelihood. The VAE framework lets us to choose recognition model and a generative model in order to learn a deep latent variable models for text, avoiding having to optimise the log-likelihood directly. We introduce the VAE framework in an NLP setting and apply it to multi-language text generations using neural networks. We extend the theory to the case of two languages sharing a latent space. Implementing the theory and applying it to parallel language data we show how stronger generative models result in better sentences without collapsing the latent distribution to the prior, enabling us to generate coherent sentences in both languages and also perform language reconstruction using the recognition model.

# Contents

# List of Figures

## Chapter 1

# Introduction

Natural language processing (NLP) is an old field with roots in many different disciplines including but not exclusive to electrical engineering, artificial intelligence and linguistics [2, p. 10-15]. Machine learning often view NLP as a statistical problem. A language model is a statistical model that trained on a training corpus assign probabilities to new sentences [3]. Latent variable models enables these language models to generate novel sentences from an underlying stochastic variable. While using probabilistic models enables us to do inference in a principled manner, it is non-trivial to optimise the log-likelihood specified by a latent variable model. Variational Inference bounds the log-likelihood from below using the Evidence Lower Bound (ELBO) by introducing an approximate distribution, $q(\boldsymbol{z})$, of the conditional probability of the latent variable, $p(\boldsymbol{z}|\boldsymbol{x})$ [4].

Although Variational Inference specifies ways of approximating the log-likelihood by a variational distribution $q$ it does not specify the form of this $q$. Variational Autoencoder (VAE) [5], also under the name of stochastic backpropagation [6], is a framework for training deep generative models with latent variables by introducing a recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$. Except for the most basic of models the ELBO function is not tractable. However, it is possibly to approximate the ELBO by sampling the latent variable instead of integrating it out. Using the reparameterization trick to sample this latent variable gives us the differentiable unbi-

ased Stochastic Gradient Variational Bayes (SGVB) estimator of the ELBO, leading to the Auto-Encoding Variational Bayes (AEVB) algorithm. AEVB optimises the introduced recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ jointly with the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})$ with respect to the SGVB objective [5]. As the recognition model maps the input sentences to the latent space, it effectively tries to compress information about the sentences through $\boldsymbol{z}$, resulting in distributed latent representations of sentences [7].

In this thesis we introduce a deep generative model which generates sentences in two different languages, $\mathcal{X}$ (EN) and $\mathcal{Y}$ (FR), from one latent variable, $\boldsymbol{z}$. The generative model is based on the WaveNet neural network [1] and is trained using the AEVB algorithm. We show how different recognition models (MLP, RNN, WaveNet) affect the final performance and training time in the mono-language generative model and based on this performance we use the MLP recognition model for the twin-language generative model. Letting the generative WaveNet model in the twin-language model differ in depth and power, we evaluate how this affects the latent representation.

Learning the recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ theoretically lets us do translation. By omitting the dependency of one of the languages, here $\mathcal{Y}$, we force $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}) \propto q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ giving us a way to perform mappings of $\mathcal{X} \rightarrow \mathcal{Y}$. We use metrics including the SGVB estimate of the ELBO and an upper bound on the perplexity to evaluate the models on language generation, reconstruction and translation. We show that the twin-language model avoids KL-collapse for all models and performs well on reconstruction and generation, but fails to translate sentences.

The thesis follows the following structure. In Chapter 2 we review the necessary background knowledge to understand the theory and experiments conducted. Chapter 3 lays out how variational inference and how the AEVB algorithm works, and also specifies the mono and twin-language latent variable model. Chapter 4 presents the experimental setup and a discussion of results and chapter 5 the con-

clusion and future work. The appendix shows an excerpt of generated sentences for all the different models together with plots.

# Chapter 2

# Background Knowledge

This chapter will introduce the necessary background knowledge to follow the theoretical derivations later.

## 2.1 Probability Theory and Statistics

### 2.1.1 Rules and Theorems

We here lay out definitions and theorems that we will make use of in the thesis. For all parts below, assume that $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ are random variables and $p(x)$ respectively $p(y)$ the probability distributions of these variables. It will be clear from the context if $x$ is to be interpreted as a random variable or an observed quantity. We interpret the integral over $x$ to be a sum over $\mathcal{X}$ if $x$ is discrete and a regular Lebesgue integral over $\mathcal{X}$ if $x$ is continuous. Then:

**Definition 2.1.1** (Unit Volume)**.**

$$\int_{\mathcal{X}} p(x)\,\mathrm{d}x = 1$$

**Definition 2.1.2** (Non-negativity)**.**

$$p(x) \geq 0$$

Most manipulations of random variables may be reduced to the following three rules of probability:

**Theorem 2.1.3** (Sum Rule)**.**

$$p(x) = \int_{\mathcal{Y}} p(x, y)\, \mathrm{d}y$$

**Theorem 2.1.4** (Product Rule)**.**

$$p(x, y) = p(y|x)p(x)$$

**Theorem 2.1.5** (Bayes Rule)**.**

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

Two very important operations involving probabilities of random variables are those of *Expectation* and *Covariance*.

**Definition 2.1.6** (Expectation)**.** *Let* $f : \mathcal{X} \to \mathbb{R}$*, then*

$$\mathbb{E}_x[f] = \int_{\mathcal{X}} f(x)p(x)\, \mathrm{d}x$$

**Definition 2.1.7** (Covariance)**.**

$$\mathrm{Cov}(x, y) = \mathbb{E}_{xy}[(x - \mathbb{E}_x[x])(y - \mathbb{E}_y[y])]$$

The variance operator is straightforwardly defined:

**Definition 2.1.8** (Variance)**.**

$$\mathrm{Var}(x) = \mathrm{Cov}(x, x)$$

The generalisation of expectation from $f : \mathcal{X} \to \mathbb{R}$ to $f : \mathcal{X} \to \mathbb{R}^D$ is straightforward. If $\boldsymbol{f} = f(x)$ then:

$$\mathbb{E}_x \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_D \end{bmatrix} = \begin{bmatrix} \mathbb{E}_x \boldsymbol{f}_1 \\ \vdots \\ \mathbb{E}_x \boldsymbol{f}_D \end{bmatrix}$$

similarly $\mathrm{Cov}(\boldsymbol{f})$ is a $D \times D$-dimensional matrix where $\mathrm{Cov}(\boldsymbol{f})_{i,j} = \mathrm{Cov}(\boldsymbol{f}_i, \boldsymbol{f}_j)$[8, 9].

### 2.1.2 The Gaussian Distribution

For a $D$-dimensional random vector $\boldsymbol{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{2.1}$$

where $\boldsymbol{\mu}$ is a $D$-dimensional mean vector, $\Sigma$ is a $D \times D$ dimensional positive definite covariance matrix and $|\Sigma|$ denotes the determinant of $\Sigma$. It is straightforward to show that these parameters correspond to the mean and covariance as defined in 2.1.6 and 2.1.7 [9].

The Gaussian distribution can be seen as a unit $D$-dimensional cube which is translated, sheared and rotated, giving rise to the fact that we can write any Gaussianly distributed random variable $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ as a linear combination of a unit Gaussian random variable $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{I})$. If we let $\Lambda\Lambda^\top = \Sigma$ be the Cholesky decomposition [10, p. 100-102] of $\Sigma$, then we also have that

$$\boldsymbol{x} = \boldsymbol{\mu} + \Lambda \boldsymbol{z}, \tag{2.2}$$

where the equality is in terms of distribution. If we further assume that $\boldsymbol{x}$ is parametrised by $\boldsymbol{\mu}$ and $\Sigma$ such that $\Sigma$ is diagonal positive definite with diagonal $\boldsymbol{\sigma}^2$, and furthermore $\boldsymbol{\sigma} = \sqrt{\boldsymbol{\sigma}^2}$ where the square root is taken elementwise, then this means that if we want to sample a random variable $\boldsymbol{x}$ with diagonal covariance

structure, we can do this by sampling a unit normal $\boldsymbol{z}$ which we then transform, which may be expressed as

$$\boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2), \tag{2.3}$$

where we define a vector $\boldsymbol{\sigma}^2$ as the covariance matrix $\Sigma$ to mean that $\Sigma$ is diagonal positive definite with diagonal $\boldsymbol{\sigma}^2$. In this case we have used the elementwise product operator $\odot$ which in general takes two matrices $\boldsymbol{A}, \boldsymbol{B}$ such that $(\boldsymbol{A} \odot \boldsymbol{B})_{ij} = \boldsymbol{A}_{ij}\boldsymbol{B}_{ij}$.

As the Gaussian distribution is part of the exponential family [9], the density of the joint distribution of iid[1] Gaussian variables are themselves Gaussian distributed where the natural parameters of this joint distribution is the sum of the natural parameters of each random variable in the joint. In particular for the Gaussian distribution, this means that if we have a collection of iid gaussian random variables $\{\boldsymbol{x}_i\}_i^n$, such that $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_i, \Sigma_i)$, then the joint can be found to be Gaussian distributed as

$$\mathcal{N}(\boldsymbol{\mu}, \Sigma),$$

where

$$\Sigma = \left( \sum_i^n \Sigma_i^{-1} \right)^{-1} \tag{2.4}$$

$$\boldsymbol{\mu} = \Sigma \left( \sum_i^n \Sigma^{-1} \boldsymbol{\mu}_i \right). \tag{2.5}$$

[8, p. 78-84]

For example in the case of two independent random variables distributed according to the form as laid out in (2.3), $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_{\boldsymbol{x}}^2)$ and $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\sigma}_{\boldsymbol{y}}^2)$, we have

[1] Identically, Independently Distributed

that the resulting distribution $p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y})$ is distributed such that

$$p(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}_{x,y}, \boldsymbol{\sigma}^2_{x,y})$$

where

$$\boldsymbol{\sigma}_{x,y} = \frac{1}{(\boldsymbol{\sigma}_x^2)^{-1} + (\boldsymbol{\sigma}_y^2)^{-1}} \tag{2.6}$$

$$\boldsymbol{\mu}_{x,y} = \frac{(\boldsymbol{\sigma}_x^2)^{-1} \odot \boldsymbol{\mu}_x + (\boldsymbol{\sigma}_y^2)^{-1} \odot \boldsymbol{\mu}_y}{(\boldsymbol{\sigma}_x^2)^{-1} + (\boldsymbol{\sigma}_y^2)^{-1}} \tag{2.7}$$

with the inverse and division operators being done elementwise. This can be derived in a straightforward manner by using the precision matrix instead of the covariance matrix.

### 2.1.3 Maximum Likelihood Estimation

Assume we have a model $\mathcal{M}$ parametrised by $\boldsymbol{\theta}$ constrained to live in the parameter space $\boldsymbol{\Theta}$. Given data $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ we want to be able to fit the parameters $\boldsymbol{\theta}$ such that the model generalise to unseen data.

We define the likelihood function using the common assumption of iid datapoints

$$\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\boldsymbol{\theta}) = \prod_i^n p(\boldsymbol{x}_i|\boldsymbol{\theta}). \tag{2.8}$$

The MLE of of the parameters of the model is then defined to be

$$\hat{\boldsymbol{\theta}}_{ML} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}}\, \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}). \tag{2.9}$$

While the original MLE is defined in terms of the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$, it's often more practical to work with the logarithm of this function, the log-likelihood function $\ell(\boldsymbol{\theta}|\mathcal{D})$. Using the log-likelihood we transform this product

into a form involving sums

$$\ell(\boldsymbol{\theta}|\mathcal{D}) = \log \mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \sum_{i}^{n} \log p(\boldsymbol{x}_i). \tag{2.10}$$

[11]

Besides from simplifying notation and calculation, the log-likelihood has the added benefit of reducing the risk of arithmetic underflow due to the small magnitude of individual probabilities. In models with long dependency such as language models, working in the log-space is needed to enable computation efficiently and robustly.

## 2.2 Deep Learning

The field of NLP were originally dominated by older machine learning techniques utilising linear models trained over very high-dimensional and sparse feature vectors. Recently the field has switched over to neural networks over dense inputs using word embeddings [12, p. 1 - 2].

Neural networks may be seen from many angles, but from a mathematical point of view a neural network parametrised by parameters (weights) $\boldsymbol{\theta}$ specifies a non-linear functional relationship between the input to the network $\boldsymbol{x}$ and the output $\boldsymbol{y}$. As such a neural network architecture with unspecified parameters $\boldsymbol{\theta}$ in a parameter space $\Theta$ specifies a set of functions [8]. It has been shown that this set of functions of several architectures are universal approximators and thus are able to approximate continuous functions on compact subsets of $\mathbb{R}^m$ [13, 14], justifying their use theoretically.

### 2.2.1 Multilayer Perceptron

Multilayer Perceptrons (MLP's) are neural networks represented by functional composition, where each function is interpreted as a layer of the network. The original MLP can be defined in terms of a recurrence relation such that if we have

input vectors of the form $\boldsymbol{x} \in \mathbb{R}^{d_{in}}$ and output vectors of the form $\boldsymbol{y} \in \mathbb{R}^{d_{out}}$ and $\sigma_i(\cdot)$ represents an arbitrary activation function (also called nonlinearity), then an MLP with $L$ layers have the functional form

$$f(\boldsymbol{x}|\boldsymbol{\theta}) = \sigma_L(\boldsymbol{W}_L \boldsymbol{z}_{L-1} + \boldsymbol{b}_L) \tag{2.11}$$

where for any $l \in \{2, \ldots, L-1\}$

$$\boldsymbol{z}_l = \sigma_l(\boldsymbol{W}_l \boldsymbol{z}_{l-1} + \boldsymbol{b}_l) \tag{2.12}$$

with base case

$$\boldsymbol{z}_1 = \sigma_1(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1). \tag{2.13}$$

[15]

$\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ may be of any dimension as long as it is dimensionally consistent with the input and output of the previous and next layers and conform to the original input and output dimensions. In this case we have that the parameters of the network are all of the biases and weights for the layers, $\boldsymbol{\theta} = \{(\boldsymbol{W}_l, \boldsymbol{b}_l)\}_{l=1}^{L}$.

Activation functions generally are monotonically increasing function mapping real numbers to some interval. Activation functions which will be useful to us are

**Sigmoid**

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{2.14}$$

The sigmoid activation function maps any real number to a value in $(0, 1)$.

**Hyperbolic tangent**

$$\tanh(x) = \frac{2}{1 + \exp(-2x)} - 1 = 2\sigma(2x) - 1 \tag{2.15}$$

**SELU** Scaled Exponential Linear Units [16] have the form

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha(\exp(x) - 1) & \text{if } x \leq 0 \end{cases} \tag{2.16}$$

**Softmax** The Softmax is a mapping from an input space $\mathbb{R}^{d_{in}}$ to an output space $\mathbb{R}^{d_{out}}$, the form given by

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{k=1}^{d_{out}} \exp(z_k)} \quad \text{where} \quad i \in \{1, \ldots, d_{out}\} \tag{2.17}$$

Since it is normalized it is a natural candidate for defining categorical distributions [9] and we will use it for this purpose in our models.

### 2.2.2 Recurrent Neural Networks

RNN's have traditionally been the neural network architecture of choice for NLP due to it being able to handle long-term dependencies well [17]. We use the LSTM cell in our RNN which handles the vanishing gradient problem in neural networks. For in depth treatment of the architecture with respect to NLP see [18, 19].

### 2.2.3 Convolutional Neural Networks

The CNN we will use will be highly specialised for our case. We use the WaveNet as laid out in the paper by DeepMind [1]. We will use the condition form.

Conditional WaveNet works on data of the form of vectors, $\boldsymbol{x} = (x_1, \ldots, x_L)^T$, conditioning on a vector $\boldsymbol{z}$, were we describe the distribution as

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{l=1}^{L} p(x_l|x_1, \ldots, x_{l-1}, \boldsymbol{z}),$$

the form common in NLP. The output of the model has the same time dimensionality as the input, meaning that the model has the ability to output a categori-

cal distribution over the next value $x_l$ with a softmax layer. In our case this is a parametrisation of the output probability at point $l$.

WaveNet uses dilated convolutions, a type of convolution where the filter is applied to an area larger than its length by only considering input values such that at each step you skip $k$ inputs, where $k$ is the dilation. Stacking layers and using dilations that increase exponentially with the depth of the layer, it is possible to get a receptive field which grows exponentially with the depth of the network [1].

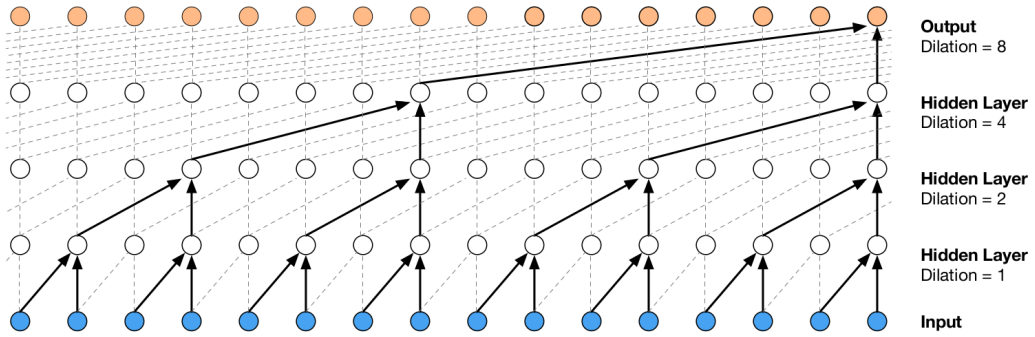



**Figure 2.1:** Dilated Convolutions in WaveNet [1]

We use residual connections in order to speed up training.

## 2.3 Natural Language Processing

Humans use natural language every day to convey concepts and abstractions to each other in an efficient manner. Compared to formal languages found in mathematics and programming, the natural languages we use are often ambiguous systems filled with rules and exceptions [20, 21].

We will here go through the necessary theory for understanding the neural language model we will build.

### 2.3.1 Language model

We define a sentence to be a vector of words $\boldsymbol{x}_{1:L} = (\mathrm{x}_1, \ldots, \mathrm{x}_L)^\top$ such that each word is an atomic element $\mathrm{x}_i \in V$, where $V$ is the dictionary of words in our language. Repeated use of the product rule laid out in (2.1.4) enables us to rewrite

the probability of a sentence in terms of how it is composed of words

$$P(\boldsymbol{x}_{1:L}) = \prod_{l=1}^{L} P(\mathsf{x}_l|\mathsf{x}_1, \ldots, \mathsf{x}_{l-1}) \tag{2.18}$$

where $\mathsf{x}_l$ is the $l$'th word of the sentence $\boldsymbol{x}_{1:L}$ [3].

This form of the probability of a sentence lays bare how the words in a sentence relate to previously occurred words within the same sentence. We will assume that all sentences in the dataset $\mathcal{D} = \{\boldsymbol{x}^n\}_{n=1}^N$ are iid which enables us to express the log-likelihood in the form of equation (2.10).

### 2.3.2 Word embeddings

Breaking down sentences at a word level and processing them into a form that encodes information efficiently is a problem which has gained notorious recognition [22], leading to algorithms such as word2vec [23] and Glove [24]. However, these techniques work less well in a neural network setting where instead the preferred technique is to find the best embeddings jointly with the parameters of the model using backpropagation [12, p. 5-7].

A straightforward way to represent the various words of the dictionary is as one-hot-encoded vectors such that a word $\mathsf{x} \in V$, where the size of $V$ is $|V|$, with an index $i$ given by its place in the dictionary sorted alphabetically in descending order will have the vector representation

$$\text{onehot}(\mathsf{x}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.19}$$

such that $\text{onehot}(\mathsf{x})_j = \delta_{ij}$ [12, p. 6]. While this is a form conceptually easy to understand, it fails to account for the curse of dimensionality as the size of the vocabulary grows large in practice and the fact that the cosine similarity of two words $\mathsf{x}_1, \mathsf{x}_2 \in V$ is zero unless they are the same word,

$$\cos_{similarity}(\text{onehot}(\mathsf{x}_1), \text{onehot}(\mathsf{x}_2)) = \delta_{\mathsf{x}_1\mathsf{x}_2}. \tag{2.20}$$

This means that no meaning is embedded in the vector space except for the location in the sorted dictionary. Instead we would like to associate each word in the vocabulary with a distributed *word feature vector*, that is a dense, real-valued vector in $\mathbb{R}^m$ where $m$ is the dimension of our embedding space. After training this embedding jointly with the parameters of the network, words which share similarities such as `Dog, Puppy` would have a higher similarity score than with an unrelated concepts such as `Dog, Bulwark` [3].

We may represent this in a mathematical form by trying to find a linear map $C$ from any element $\mathsf{x} \in V$ such that $C(\mathsf{x}) \in \mathbb{R}^E$, where $E$ is the embedding dimension. Using the canonical basis of the Euclidean space, we can express this linear map in terms of a matrix $\boldsymbol{C} \in \mathbb{R}^{E \times |V|}$, thus the word feature vector of the learned embedding can be represented by the matrix multiplication $\boldsymbol{C}\text{onehot}(\mathsf{x})$.

### 2.3.3 Neural machine translation

For a long time the dominant paradigm within machine translation was to use phrase based machine translation systems [25, 26], recently however, modelling the word or character level directly with neural networks has overtaken phrase based translation systems [27, 28]. These systems are called Neural Machine Translation.

Most NMT models work in terms of an encoder-decoder architecture where the encoder extracts a fixed length representation $\boldsymbol{c}$, often called a context vector,

from a variable length input sentence $\boldsymbol{x} \in \mathcal{X}$, and the decoder uses this representation to generate a correct translation $\boldsymbol{y} \in \mathcal{Y}$ from this representation [29] as can be seen in Figure 2.2.

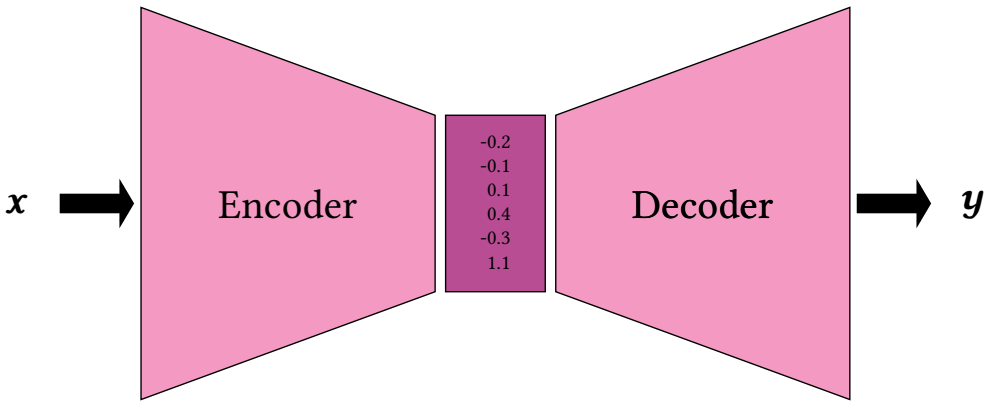



**Figure 2.2:** Encoder decoder schematic

Our model builds on this paradigm but instead of considering context vectors $\boldsymbol{c}$ we instead consider distributions over a latent vector $\boldsymbol{z}$ which in theory should give the model a greater expressive power since the distribution relays more knowledge than a direct point estimate.

## 2.4 Optimization

Most parts of machine learning reduces to optimization of a loss function. While optimisation of convex problems are well-understood there has been considerable interest in ways to optimize non-convex loss functions such as those used traditionally in deep learning.

While theoretical results are lacking, there has been substantial advances in various optimisation techniques fit to attack the highly non-linear, non-convex and high-dimensional optimization problems of learning in deep models, particularly from Stochastic Gradient Descent and its friends. This has led to numerous gradient descent-like algorithms used machine learning [30].

### 2.4.1 Stochastic Gradient Descent

For a normal probabilistic machine learning problem, we have data $\mathcal{D}$ that we try to model with a model $\mathcal{M}$ parametrised by parameters $\boldsymbol{\theta} \in \Theta$. The optimization problem in our case can be recast as an effort to find the parameters $\boldsymbol{\theta}_{ML}$ that maximizes the log-likelihood function (2.10) or equally minimizes the negative log-likelihood.

Gradient descent takes steps in the direction of the gradient, which also is the direction of steepest descent. If we let $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ be our set of sentences and $-\ell$ be the negative log-likelihood that we are trying to minimize then the updates are of the form

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}}(-\ell(\boldsymbol{\theta}|\boldsymbol{x}_i)), \tag{2.21}$$

where $\gamma$ is the learning rate of the algorithm.

Stochastic Gradient Descent is a similar algorithm to GD that instead of calculating the gradient with respect to the whole dataset calculates an approximate gradient, as it only takes a subset of the data into account at each update. If we let $I_t$ be a random subset of the indices of $\{(\boldsymbol{x}_n)_{n=1}^N\}$ of size $m$ then SGD does the following update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \gamma \frac{1}{m} \sum_{i \in I_t} \nabla_{\boldsymbol{\theta}}(-\ell(\boldsymbol{\theta}|\boldsymbol{x}_i)) \tag{2.22}$$

[31][8, p. 240].

### 2.4.2 ADAM

SGD has found widespread use within the machine learning community due to strong experimental results and ease of use, especially in deep learning. However there are notable alternatives that try to improve on SGD such as RMSProp[32] and AdaGrad[33].

Adam takes inspiration from RMSProp and AdaGrad. Technically, Adam keeps an exponential running average of the first and second order statistics of

the gradient, using these to calculate an adaptive learning rate.

As laid out in the original paper by Kingma et al. [34], ADAM operates on a stochastic objective function $f_t(\boldsymbol{\theta})$, where $t$ denotes the time step, similarly $g_t = \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta})$. $m_t$ and $v_t$ denotes the first and second order moments of the gradients, however, these are biased and are corrected in the algorithm. The whole algorithm is as follows.

**Algorithm 2** The ADAM algorithm

**Require:** $\alpha$: Stepsize
**Require:** $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates of the moment estimates
**Require:** $f(\boldsymbol{\theta})$: Stochastic objective function with parameters $\boldsymbol{\theta}$
**Require:** $\boldsymbol{\theta}_0$: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize $1^{\text{st}}$ moment vector)
$v_0 \leftarrow 0$ (Initialize $2^{\text{nd}}$ moment vector)
$t \leftarrow 0$ (Initialize timestep)
**while** $\boldsymbol{\theta}_t$ not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_{\boldsymbol{\theta}} f_t(\boldsymbol{\theta}_{t-1})$ (get gradients w.r.t stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_1 \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_1 \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \cdot \hat{m}_i / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
**return** $\boldsymbol{\theta}_t$ (Resulting parameters)

Experimentally Adam has shown very good results on training various deep learning models such as MLP's, CNN's and RNN's [34], we will use it to train all our models.

### 2.4.3 Automatic Differentiation

The AutoDiff framework is a general theory for efficiently calculating gradients with respect to a loss function in general computational graphs. It includes the famed backpropagation [8, 35] algorithm as a special instance.

In machine learning we most often deal with many-to-one mappings with a high-dimensional input $\boldsymbol{x}$ to a scalar output in the form of a loss. In terms of machine learning models, AutoDiff takes a loss function $L(\boldsymbol{\theta})$ and returns an exact

value up to machine accuracy for the gradient $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})|_{\hat{\boldsymbol{\theta}}}$. There are two different modes of AutoDiff, forward and reverse, although in practice reverse is preferred.

Reverse mode AutoDiff is efficient in the way that calculating the gradient of $L(\boldsymbol{\theta})$ is guaranteed to take at most 5 times the time it takes to compute $L(\boldsymbol{\theta})$ [36], enabling neural network libraries to calculate the gradients needed for optimization algorithms such as SGD and ADAM in a swift and automatic manner.

# Chapter 3

# Methods and Theory

This chapter deals with the theory needed in order to use VAE in our setting. It will also deal with extending the theory and necessary calculations.

## 3.1 Variational Inference

Consider the following general graphical model of a latent variable $\boldsymbol{z}$ and an observed variable $\boldsymbol{x}$ parametrised by $\boldsymbol{\theta}$:

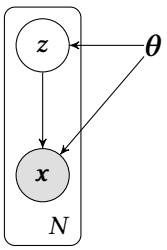



**Figure 3.1:** Graphical model of a latent variable model parametrised by $\boldsymbol{\theta}$.

implying the joint distribution

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}). \tag{3.1}$$

In order to train the model using optimisation of $\ell(\boldsymbol{\theta}|\mathcal{D})$ we need to be able to to find the gradient and value of $\ell(\boldsymbol{\theta}|\boldsymbol{x})$ which are used by ADAM. We can rewrite

the likelihood for a datapoint $\boldsymbol{x}$ as

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{x}) = \int_{\mathcal{Z}} p(\boldsymbol{x}, z)\, \mathrm{d}z \tag{3.2}$$

by using the sum rule of 2.1.3. For many models, this integral is either not available in a closed form or requires exponential time to compute [4]. While for some special cases it is possible to find a local optimum of the likelihood by using the EM algorithm [37], in general it is too restrictive since we need to be able to find $p(\boldsymbol{z}|\boldsymbol{x})$ analytically.

Generally it is possible to use sampling based techniques such as MCMC [38] to sample from $p(\boldsymbol{z}|\boldsymbol{x})$ but in practice it is often slow and depending on the problem may be less than ideal. Especially for problems where we use complex models and have large datasets it's not possible in practice to use MCMC [4].

Variational inference approaches the problem of optimizing the intractable log-likelihood by positing a variational family of distributions, $\boldsymbol{Q}$ and introducing a variational distribution $q \in \boldsymbol{Q}$. The log-likelihood is then bounded from below using the concavity of the $\log(\cdot)$ function in order to apply Jensen's inequality and the fact that $q$ is a distribution:

$$\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \log \int_{\mathcal{Z}} p_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{x})\, \mathrm{d}\boldsymbol{z} \\
&= \log \int_{\mathcal{Z}} q_{\boldsymbol{\varphi}}(\boldsymbol{z}) \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z} \\
&= \log \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z})} \left[ \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z})} \right] \\
&\geq \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z})} \left[ \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z})} \right] \\
&= \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z})} \left[ \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}) \right].
\end{aligned}$$

We call the lower bound the Evidence Lower Bound Objective (ELBO),

$$\text{ELBO}(q_{\boldsymbol{\varphi}}(\boldsymbol{z})) = \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z})\right] \tag{3.3}$$

and we get the following inequality

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \text{ELBO}(q_{\boldsymbol{\varphi}}(\boldsymbol{z})). \tag{3.4}$$

Rewriting the ELBO in terms of the log-likelihood

$$\text{ELBO}(q_{\boldsymbol{\varphi}}(\boldsymbol{z})) = \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z})||p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})) \tag{3.5}$$

we see how the choice of $q_{\boldsymbol{\varphi}}(\boldsymbol{z})$ affects the tightness of the lower bound through the closeness to the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ in terms of the KL-divergence [4].

## 3.2 VAE and SGVB

The Variational Autoencoder by Kingma et al. [5] introduces a recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ to a latent variable model of the form (3.1), that serves as an approximation to the true posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. We let $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\varphi}}(\boldsymbol{x}))$ such that the pair of vectors $(\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\varphi}}(\boldsymbol{x}))$ is the output of a neural network, and let this Gaussian have a diagonal covariance structure with the vector $\boldsymbol{\sigma}^2_{\boldsymbol{\varphi}}(\boldsymbol{x})$ as the diagonal.

In the following we assume that $\boldsymbol{x}$ is an observed datapoint from our dataset and $\boldsymbol{z}$ is the corresponding latent variable. If we consider the ELBO again we can rewrite it in terms that only involve known functions and quantities

$$\text{ELBO}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})) = \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})). \tag{3.6}$$

Since this depends on both the parameters of the generative model and the recog-

nition model, $(\boldsymbol{\theta}, \boldsymbol{\varphi})$ we make this explicit and write

$$\mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}|\boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})). \tag{3.7}$$

Note that for most models we are not able to get an analytical form of the expectation over $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ due to the non-linear relationship between $\boldsymbol{x}$ and $\boldsymbol{z}$. Instead we sample $\boldsymbol{z}$ to estimate this expectation using normal Monte Carlo estimation.

While it would be possible to differentiate and optimize the lower bound $\mathcal{L}^{\text{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}|\boldsymbol{x})$ jointly with respect to both the variational and generative parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ by using a straightforward Monte Carlo gradient estimator with respect to $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$, this gradient exhibits very high variance and is impractical compared to the reparametrisation trick which we will introduce below [5].

### 3.2.1 Reparametrisation Trick

Under certain mild regularity conditions, for a chosen approximate posterior $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ we can express the distribution of $\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ in terms of a simpler distribution using reparametrisation, called the reparametrisation trick [5]. For our case, let transformation $g_{\boldsymbol{\varphi}}(\boldsymbol{\epsilon}, \boldsymbol{x})$ be such that

$$\boldsymbol{z} = g_{\boldsymbol{\varphi}}(\boldsymbol{\epsilon}, \boldsymbol{x}), \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{3.8}$$

In our case we use (2.3) in order to write $g_{\boldsymbol{\varphi}}(\boldsymbol{\epsilon}, \boldsymbol{x}) = \boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}) + \boldsymbol{\sigma}_{\boldsymbol{\varphi}}(\boldsymbol{x}) \odot \boldsymbol{\epsilon}$.

This means that we can form Monte Carlo estimates of the ELBO by sampling $\boldsymbol{z}$ through sampling $\boldsymbol{\epsilon}$ and using the differentiable transformation

$$\mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})\right] \simeq \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}^s) - \log q_{\boldsymbol{\varphi}}(\boldsymbol{z}^s|\boldsymbol{x}) \tag{3.9}$$

where $\boldsymbol{z}^s = g_{\boldsymbol{\varphi}}(\boldsymbol{\epsilon}^s, \boldsymbol{x})$ and $\boldsymbol{\epsilon}^s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for all $s \in \{1, \ldots, S\}$.

The right hand side of equation (3.9) is the Stochastic Gradient Variational

Bayes estimator of the ELBO, which we call $\mathcal{L}^{SGVB}(\boldsymbol{\theta}, \boldsymbol{\varphi}|\boldsymbol{x})$ and is an unbiased estimator of the ELBO since we sample $\boldsymbol{z}$ from the recognition model.

### 3.2.2 AEVB algorithm

The AEVB algorithm optimises the SGVB equation (3.9) in order to in drive up the ELBO and push the lower bound of the the log-likelihood up. If the recognition model and the prior are chosen to be from appropriate distributions it is possible to get an analytical form of the KL-divergence. Choosing $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ to be Gaussianly distributed enables us to write the SGVB in an alternative form which typically has less variance than the SGVB form of equation (3.9):

$$\mathcal{L}^{SGVB}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = -D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})) + \frac{1}{S}\sum_{s=1}^{S} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}^s) \tag{3.10}$$

where $\boldsymbol{z}^s = g_{\boldsymbol{\varphi}}(\boldsymbol{\epsilon}^s, \boldsymbol{x})$ and $\boldsymbol{\epsilon}^s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We let $S = 1$ following the advice of Kingma et al. [5].

## 3.3 Models

Here we lay out the graphical models we will use in order to model language generation. All of our models are latent variable models with the prior on $\boldsymbol{z}$ being unit normal, $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$.

### 3.3.1 Mono-Language Model

The generative model is the same as in the figure for latent variable models , except that the distribution of $\boldsymbol{z}$ does not depend on $\boldsymbol{\theta}$:

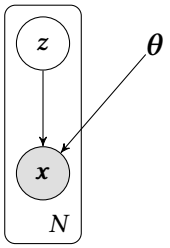



**Figure 3.2:** Graphical representation of the Mono-language model. The prior over the latent is independent of $\boldsymbol{\theta}$.

where $z$ is the latent variable representing the latent sentence structure and the joint factorizes as $p_\theta(\boldsymbol{x}, z) = p_\theta(\boldsymbol{x}|z)p(z)$.

The Mono-language model is the same model specified in [5]. Bowman et al. has successfully showed that it can be applied to natural language, learning meaningful representations of language in the latent dimension [7] while improving on previous models such as RNNLM [39].

### 3.3.2 Twin-Language Model

Using two languages adds and extra observed variable $\boldsymbol{y}$, a sentence belonging to $\mathcal{Y}$. The probabilistic model looks like:

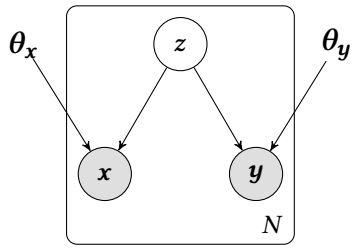



**Figure 3.3:** Twin-language generative latent variable model. The prior over the latent is independent of $\boldsymbol{\theta_x}$ and $\boldsymbol{\theta_x}$.

The graphical model implies that the joint distribution factorises as

$$p_\theta(z, \boldsymbol{x}, \boldsymbol{y}) = p(z)p_{\theta_x}(\boldsymbol{x}|z)p_{\theta_y}(\boldsymbol{y}|z) \tag{3.11}$$

and we make the following distributional assumption

$$p_{\theta_x}(\boldsymbol{x}|z) = \mathcal{N}(\boldsymbol{\mu}_{\theta_x}(z), \boldsymbol{\sigma}^2_{\theta_x}(z)) \tag{3.12}$$

$$p_{\theta_y}(\boldsymbol{y}|z) = \mathcal{N}(\boldsymbol{\mu}_{\theta_y}(z), \boldsymbol{\sigma}^2_{\theta_y}(z)) \tag{3.13}$$

The notation $\mathcal{N}(\boldsymbol{\mu}_\theta(z), \boldsymbol{\sigma}^2_\theta(z))$ means that the mean and variance diagonal of the normal are represented by a functional relationship $f : z \mapsto (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ where $f$ is a function represented a a learnable neural network with parameters $\boldsymbol{\theta}$.

The recognition model parametrises the probability distribution $q_{\boldsymbol{\varphi}}(z|\boldsymbol{x}, \boldsymbol{y})$. We are free to choose this parametrisation and distributional form however we like, but the closer this is to the actual conditional distribution over the latent, $p(z|\boldsymbol{x}, \boldsymbol{y})$ in the KL sense, the tighter the inequality in (3.4) will be as shown by equation (3.5). We let the distributions be Gaussians to have analytical tractability,

$$\begin{aligned} q_{\boldsymbol{\varphi}}(z|\boldsymbol{x}, \boldsymbol{y}) &= q_{\boldsymbol{\varphi}_x}(z|\boldsymbol{x}) q_{\boldsymbol{\varphi}_y}(z|\boldsymbol{y}) \\ &= \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi}_x}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\varphi}_x}(\boldsymbol{x})) \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\varphi}_y}(\boldsymbol{y}), \boldsymbol{\sigma}^2_{\boldsymbol{\varphi}_y}(\boldsymbol{y})). \end{aligned}$$

This is again a Gaussian distribution

$$q_{\boldsymbol{\varphi}}(z|\boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(z|\mu_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y}), \sigma^2_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y})) \tag{3.14}$$

which reduces to the expressions (2.5) for the mean and (2.4) for the covariance as shown in the background section on the Gaussian distribution. Pulling all of these statements together, we get that the SGVB of the translation case reduces to the

equation

$$\mathcal{L}^{SGVB}(\boldsymbol{\theta}, \boldsymbol{\varphi}|\boldsymbol{x}, \boldsymbol{y}) = (\frac{1}{S}\sum_{s=1}^{S}\log(\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\theta_x}(z), \boldsymbol{\sigma}^2_{\theta_x}(z))) + \log(\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_{\theta_y}(z), \boldsymbol{\sigma}^2_{\theta_y}(z))))+ \frac{1}{2}\sum_{j=1}^{J}(1 + \log((\boldsymbol{\sigma}_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y})_j)^2) - (\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y})_j^2 - (\boldsymbol{\sigma}_{\boldsymbol{\varphi}}(\boldsymbol{x}, \boldsymbol{y})_j^2)) \tag{3.15}$$

where $j$ goes over all of the elements of the mean and the diagonal covariance vector. The second expression on the right is the analytical form of $-D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})||p(\boldsymbol{z}))$ [5].

### 3.3.3 Training

Variational Autoencoders in NLP have a tendency to collapse the KL term in the SGVB objective to zero, effectively being reduced a RNNLM. As the KL can be seen as a measure of how much information is encoded in $\boldsymbol{z}$, the KL-collapse makes the use of VAE's to model language-agnostic features in the latent space useless. One of the objectives is to make sure this KL-collapse doesn't happen.

We use KL-annealing where we anneal the ELBO by a pseudo-objective

$$ELBO_{\beta_t} = \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \beta_t * D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{3.16}$$

where $\beta_t$ is an annealing term going from 0 to 1 as we train. We do this by choosing a warm-up $\beta$ and let $\beta_t = \min(1, \frac{t}{\beta})$ [7], letting this warm up be done linearly until we recover the original ELBO objective, similar to normal simulated annealing [40].

The training is then carried out using ADAM on the objective $\mathcal{L}^{SGVB}$ where we take into account the KL-annealing. In order to calculate the gradients efficiently we use Theano which has automatic support for AutoDiff, laid out in section 2.4.3.

# Chapter 4

# Experiments

Here we specify the setup of the experiments, the data sets and preprocessing done and interpret the results. We divide the experiments into Mono-language generation and reconstruction and Twin-language generation, reconstruction. Finally we also specify our contribution in terms of software engineering.

## 4.1 Data

### 4.1.1 Dataset

We evaluate the models on two different datasets.

**Europarl** A parallel corpus consisting of the proceedings of the European Parliament, comprising in total the 11 official languages of the European Union [41]. We use the French-English language pair and preprocess it. The English dataset used for the training of the Mono-language models has 625′213 train sentences and 69′469 test sentences. The French-English dataset used for the Twin-language models, has 534′635 train sentences and 59′404 test sentences. The reason why the Mono-language dataset is bigger than the Twin-language dataset is because for Twin-language models we take the intersection of the sentences that is between 2 and 20 words long for French and English, while for Mono-language models we only need to consider the

English sentences on their own.

**Custom** Our custom dataset is made from merging three different datasets; Europarl [41], News Commentary and UN corpus [42] [1]. This dataset is considerably bigger than the Europarl dataset. We only use it on the twin-language model. After preprocessing the dataset it has 4′268′791 train sentences and 474′311 test sentences.

### 4.1.2 Preprocessing

The raw data is unfit for use directly. For one thing the raw data is in the form of strings and in order to use neural networks we need to transform this into a form using vectors in some space $\mathbb{R}^m$.

We do the following preprocessing steps:

- We specify the maximum and minimum length of the words. This enables us to pad sentences to be the same length with zeros.
- We calculate the word frequencies in order to sort all of the words in the dataset in terms of how often it appear in absolute terms. This is then used to only retain the 30′000 most common words. Words which are not part of this list gets replaced by an `<UNK>` token, specifying that it's an unknown word outside of the dictionary.
- Newline characters were removed and replaced by `<EOS>`, end-of-sentence tokens, signifying the end of a sentence.
- When preprocessing the dataset for Twin-language models, we let the preprocessed data be the intersection of the sentences in English and French that pass the above criteria.

[1] Taken from the wmt14 translation challenge: `http://www.statmt.org/wmt14/translation-task.html`

- Unicode is needed for French due to diacritics and accents which are not included in the normal ASCII encoder-decoder scheme for characters. We make sure that encoding and decoding is done correctly.

## 4.2 software

### 4.2.1 Libraries

All of the models were trained using Theano [43] and Lasagne [44] built on top of the Python (2.7) programming language. All of the plots were generated using Matplotlib [45] and Seaborn [46]. Data processing was done with Numpy [47] and Pandas [48]. The software repository can be found at `http://www.github.com/isakfalk/project`.

### 4.2.2 Engineering

Due to the nature of the project, a lot of work went into software engineering. We were fortunate to inherit a code base that implemented the necessary base for Mono-language generation and reconstruction. This code base was refactored and reconfigured for our purposes. We ported the code from the Mono-language setting to the Twin-language setting and created programs for preprocessing the data.

For decoding the sentences we implemented a beam search algorithm [2, p. 249]. For a given latent $\boldsymbol{z}$ there is an optimal sentence $\boldsymbol{x}_{opt}$ such that $\boldsymbol{x}_{opt}$ maximises the probability $p_{\theta}(\boldsymbol{x}|\boldsymbol{z})$. Using a brute force search strategy fails since the number of branches in the search tree grows exponentially. Beam search prunes the tree at each stage and only keeps the $k$ best paths from the root to the leaves at any point. In this way it allows us to decode the sentences from $\boldsymbol{z}$ efficiently and such that the generated sentences are close to the optimal sentence.

## 4.3 Design

Working within the VAE framework forces us to choose in addition to the generative model also the form of the recognition model. For Mono-language models we will need to specify the generative model $p_\theta(\boldsymbol{x}|z)$ and the recognition model $q_\varphi(z|\boldsymbol{x})$. For Twin-language models we need to specify the generative model $p_\theta(\boldsymbol{x},\boldsymbol{y}|z) = p_{\theta_x}(\boldsymbol{x}|z)p_{\theta_y}(\boldsymbol{y}|z)$ and the recognition model $q_\varphi(z|\boldsymbol{x},\boldsymbol{y})$. We assume that the recognition model can be factorised into $q_\varphi(z|\boldsymbol{x},\boldsymbol{y}) = q_{\varphi_x}(z|\boldsymbol{x})q_{\varphi_y}(z|\boldsymbol{y})$. For the Twin-language models we assume that the generative and recognition models for $\boldsymbol{x}$ and $\boldsymbol{y}$ are the same in terms of architecture, but let them be optimised independently.

Specifically, for the recognition models, we have a Gaussian distribution of the form

$$q_\varphi(z|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}_\varphi(\boldsymbol{x}), \boldsymbol{\sigma}^2_\varphi(\boldsymbol{x}))$$

where the pair of vectors $(\boldsymbol{\mu}_\varphi(\boldsymbol{x}), \boldsymbol{\sigma}^2_\varphi(\boldsymbol{x}))$ are the output of some specified neural network followed by an affine layer. The Twin-language case is similar but combines the pairs of vectors $(\boldsymbol{\mu}_{\varphi_x}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\varphi_x}(\boldsymbol{x}))$, $(\boldsymbol{\mu}_{\varphi_y}(\boldsymbol{y}), \boldsymbol{\sigma}^2_{\varphi_y}(\boldsymbol{y}))$ into a third set of pair $(\boldsymbol{\mu}_\varphi(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{\sigma}^2_\varphi(\boldsymbol{y},\boldsymbol{x}))$ as specified by equations (2.6) and (2.7).

We let the prior be a unit normal for all models:

$$p(z) = \mathcal{N}(z|\mathbf{0}, \boldsymbol{I}). \tag{4.1}$$

Furthermore we specify the dimensionality of the word embeddings, $E$, and the latent dimension, $dim(z)$. All of the activation functions for the neural networks are ELU activation function (SELU with $\alpha = 1, \lambda = 1$), unless specified otherwise.

### 4.3.1 Mono-language models

We let $E = 300$ and $dim(z) = 50$ for all models. We have three different models that we use, we label them $\mathcal{M}_{M1}$, $\mathcal{M}_{M2}$ and $\mathcal{M}_{M3}$. All recognition models are con-

nected to a final MLP of depth 3 and with layers of 1000 units specified otherwise. We train the models for 200′000 iterations each, taking approximately 2 days per model to train with MLP being the fastest by a half. The generative model for all of the Mono-language models is the WaveNet with dilations $(1, 2, 4, 8)$, 2 dilation channels and 4 residual channels. We thus only specify the recognition models:

| Model | Recognition model | Depth | Hidden dim. | Nonlinearity |
|---|---|---|---|---|
| $\mathcal{M}_{M1}$ | MLP | 4 | 1000 | ELU |
| Model | Recognition model | Depth | Hidden dim. | Cell |
| $\mathcal{M}_{M2}$ | RNN | 1000 | 1000 | LSTM |
| Model | Recognition mode | dilations | dilation channels | residual channels |
| $\mathcal{M}_{M3}$ | WaveNet | $(1, 2, 4, 8)$ | 200 | 400 |

**Table 4.1:** Recognition models for all of the Mono-language models.

### 4.3.2 Twin-language models

The quality of the trained models highly depend on the number of iterations trained. As MLP was much faster for training the Mono-language models (by a factor of 2) compared to the other neural network architectures we tried, while being similar in performance to the others, we use it as the recognition model for all Twin-language models. We train all models for 500′000 iterations taking approximately 5 days each.

For all Twin-language models we let $E = 300$ and $dim(z) = 100$. We train 4 models in total, labelling them $\mathcal{M}_{T1}$, $\mathcal{M}_{T2}$, $\mathcal{M}_{T3}$ and $\mathcal{M}_{T4}$. All of the models were trained on the Europarl dataset (French-English) except for $\mathcal{M}_{T4}$ which was trained on the custom bigger dataset. The architecture of the recognition model is an MLP of depth 4, with layers of a 1′000 hidden units and with ELU as the activation function. All the generative models are WaveNet of varying hyperparameters:

| Model | dilations | dilation channels | residual channels |
|---|---|---|---|
| $\mathcal{M}_{T1}$ | $(1, 2, 4, 8, 16) \times 3$ | 4 | 8 |
| $\mathcal{M}_{T2}$ | $(1, 2, 4, 8, 16) \times 5$ | 2 | 4 |
| $\mathcal{M}_{T3}$ | $(1, 2, 4, 8, 16) \times 5$ | 4 | 8 |
| $\mathcal{M}_{T4}$ | $(1, 2, 4, 8, 16) \times 5$ | 4 | 8 |

**Table 4.2:** Generative models for all of the Twin-language models.

## 4.4 Results

We use several scores to measure how well the models are doing.

**ELBO (SGVB estimate)** The SGVB estimate of the ELBO from the VAE framework. This is a lower bound on the log-likelihood and our objective function. We want to maximise this.

**KL** KL in the VAE framework measures how much the outputted distribution from the recognition model differ from the prior over the latent variable. In this sense it tells us how much the model is relying on the latent variable $z$ when generating sentences compared to the generative model. We want to ensure that this quantity doesn't go to zero.

**Perplexity (PP)** For corpus data $\mathcal{D} = \{\boldsymbol{x}^n\}_{n=1}^N$ and total number of words $W$, assuming independence of sentences, we define the perplexity of a model to be

$$\mathrm{PP}(\mathcal{D}) = \sqrt[W]{\frac{1}{\prod_{n=1}^N p(\boldsymbol{x}^n)}}. \tag{4.2}$$

Taking exponent and logarithm we get the following easier expression

$$\exp(-\frac{1}{W}\sum_{n=1}^{N}\log p(\boldsymbol{x}^n)) \tag{4.3}$$

where $\log p(\boldsymbol{x}^n) = \sum_{l=1}^{L_n} \log p(\boldsymbol{x}_l^n | \boldsymbol{x}_{l-1}^n, \ldots, \boldsymbol{x}_1^n)$ and $L_n$ is the length of the $n$'th sentence. Since we don't have access to the log-likelihood directly we instead use the SGVB estimate of the ELBO to bound the perplexity by above. This

follows from the fact that

$$\mathcal{L}^{ELBO}(\boldsymbol{x}) \leq \log p(\boldsymbol{x}) \implies \exp(-\frac{1}{W}\log \mathcal{L}^{ELBO}(\boldsymbol{x})) \geq \exp(-\frac{1}{W}\log p(\boldsymbol{x}))$$

as $\exp(\cdot), \log(\cdot)$ are monotonically increasing functions. We thus use an approximate upper bound on the perplexity.

### 4.4.1 Metrics

| Model | ELBO | KL | PP |
|---|---|---|---|
| $\mathcal{M}_{M1}$ | −56.1135 | 9.0132 | 57.4400 |
| $\mathcal{M}_{M2}$ | −55.9835 | 8.5231 | 56.9031 |
| $\mathcal{M}_{M3}$ | −56.4169 | 4.9505 | 58.7118 |

**Table 4.3:** Results for the different Mono-language models.

| Model | ELBO | KL | PP (EN, FR) |
|---|---|---|---|
| $\mathcal{M}_{T1}$ | −145.8284 | 33.3129 | 264.7696 |
| $\mathcal{M}_{T2}$ | −148.6798 | 32.3151 | 295.6363 |
| $\mathcal{M}_{T3}$ | −143.5361 | 35.5259 | 243.8360 |
| $\mathcal{M}_{T4}$ | −122.2952 | 26.7563 | 237.2852 |

**Table 4.4:** Results for the different Twin-language models.

## 4.5 Discussion

We tried three different recognition models for the Mono-language models. These all performed relatively equal in performance as can be seen by how close the ELBO estimates are to each other, (-56.1135, -55.9835, -56.4169) for model ($M1, M2$ and $M3$) respectively. Since the recognition model have the same form for all of the models in this thesis, $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}), \boldsymbol{\sigma}^2_{\boldsymbol{\varphi}}(\boldsymbol{x}))$, this essentially puts a bottleneck on the information that can be encoded in $\boldsymbol{z}$ which would explain why they all reach similar level of performance, even though the different architectures (MLP, RNN, WaveNet) have different predictive capabilities. Given that the models

perform similar for different recognition models we deliberately pick MLP as the recognition model for all Twin-language models, as this lets us train for longer. From table 4.3 We note that the KL is similar for the MLP and the RNN while very low for the WaveNet, which explains why the sentences for reconstructing sentences using the recognition model is worse for $\mathcal{M}_{M3}$ as it relies too much on the generative model.

As for the quality, inspecting the sentences in appendix A, we can see that they all perform well for both generating new sentences from the prior distribution over $z$ but also are able to do reconstruction to some extent. Reconstructed sentences are not directly comparable to the input sentences, but often pick up on words and form sentences around these.

The Twin-language models are all of a similar form, MLP as the recognition model with the generative model being different forms of WaveNet. It is interesting to note that WaveNet has the capability to mimic the form of an MLP for shallow stacked convolutions, while being closer to an RNN for deeper stacks, justifying choosing WaveNet as a generative model. From table 4.3 we see that as we make the models deeper and stack more convolutions we get a smaller ELBO estimate ($\mathcal{M}_{M1}$ and $\mathcal{M}_{M2}$ does not progress strictly in terms of how powerful they are since the channels and dilations differ in a way that can't really be ordered). This is in line with results that state that CNN's are able to model more long term dependencies in data as the receptive field grows larger.

Referring to the appendix A we can see that the models does not perform as well as the Mono-language model in terms of reconstruction but learns to do generation well. This makes sense since the data is now split up in two parts which are structurally very different. In a sense we are trying to learn two different distributions over languages instead of one. It is very good on reconstructing certain sentences, phrases and words that occurs frequently in the dataset, such as `Mr. President, Europe` and `Commission` pointing that it learns some of these through

memorisation. We see that KL stays away from zero in all cases avoiding collapse. Finally we see that all models, Twin and Mono-language perform better at the start of all the sentences and also on shorter sentences implying that the generative model forgets.

We tried to do translation by sampling $\boldsymbol{z}$ from $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \propto q_{\boldsymbol{\varphi}_x}(\boldsymbol{z}|\boldsymbol{x})$ in the case of translating from language $\mathcal{X}$ to $\mathcal{Y}$ but noted that it didn't work at all. There are several reasons why this might have failed. Since the reconstructed sentences when conditioning on both $\boldsymbol{x}$ and $\boldsymbol{y}$ are not great in themselves, conditioning on only one and then transferring to the other language will be even worse. Since the model is trained using both $\boldsymbol{x},\boldsymbol{y}$ together this also means that it doesn't really optimize for this case directly, even though we could imagine that it could be the outcome that a trained model would force $q_{\boldsymbol{\varphi}_x}(\boldsymbol{z}|\boldsymbol{x})$ and $q_{\boldsymbol{\varphi}_y}(\boldsymbol{z}|\boldsymbol{y})$ to be similar in distribution since $\boldsymbol{x}$ and $\boldsymbol{y}$ represents the same meaning in different languages.

This could be explained by the fact training we did not experience any kind of KL collapse, with the KL actually being bigger with respect to the ELBO than for the Mono-language models. When training we want the sentences generated for $\mathcal{X}$ and $\mathcal{Y}$ to be close to the input sentences $\boldsymbol{x}$ and $\boldsymbol{y}$ which would force the model to rely on the latent space rather than purely outputting probable sentences which does not rely on $\boldsymbol{z}$ or only do so very weakly by the generative model forgetting about it.

# Chapter 5

# Conclusions

## 5.1 Discussion

In this thesis, we have shown how the VAE framework can be used for language reconstruction and generation. We extend it from a single language latent variable model to a model with two output language, efficiently enabling doing multi-language generation by sampling the latent. We apply the model two French and English and show using reconstructed sentences that the twin-language model learns to encode information in the latent space. Using the recognition model to sample the latent we are able to do both language reconstruction and generation at the same time. We try to apply the model to machine translation, however this fails.

## 5.2 Future work

Building on the results in this thesis, it would be fruitful to explore more combinations of recognition models and generative models. While the normal yields analytical forms of the KL, other distributions might be better for generating the latent, especially if the conditional distribution over the latent is multimodal.

Furthermore it would be interesting to see if applying regularisation such as dropout on the generative model would force the model to rely more on the en-

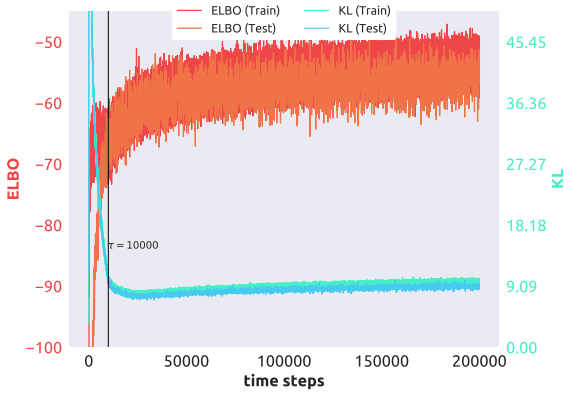
coded information in the latent space compared to the generative neural networks.

In particular there has recently been improvements in for performing inference using a certain kind of normalising flows using autoregressive neural networks [49]. Similarly there have been advances in using semi-supervised learning [50] for deep generative models which would allow us to train the models on non-parallel data and make use of mono-language data for while training multi-language models.

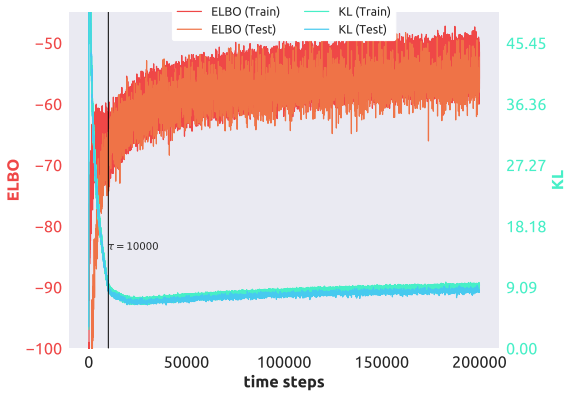
Appendix A

# Plots and generated text

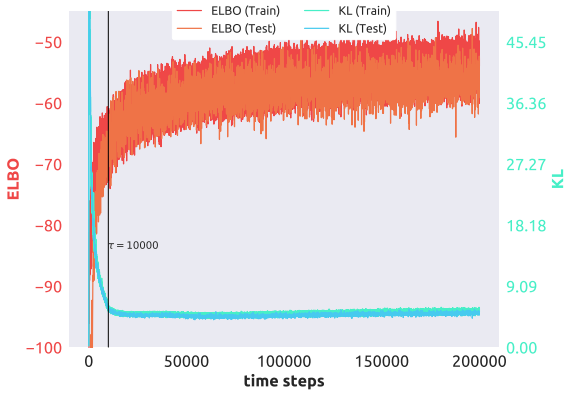
## A.1 Plots




**(a)** Plot for $\mathcal{M}_{M1}$

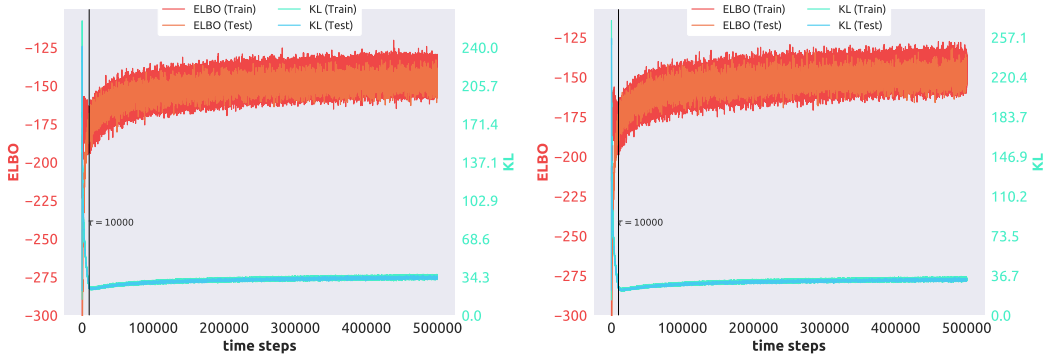



**(b)** Plot for $\mathcal{M}_{M2}$




**(c)** Plot for $\mathcal{M}_{M3}$

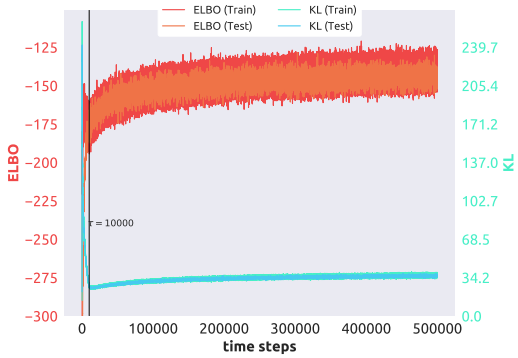
**Figure A.1:** Plots of the SGVB estimate of ELBO (Labelled ELBO in plots) together with the KL divergence $D_{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{z}))$ for the different models for Mono-language models. The vertical line specifies the end of the KL-annealing.



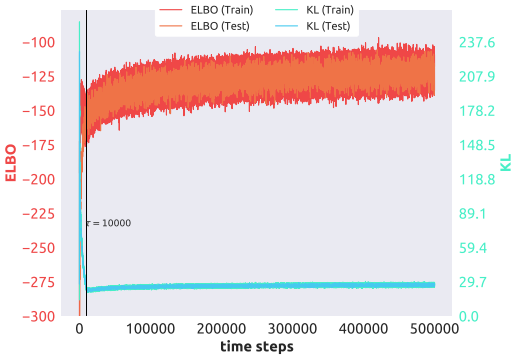

**(a)** Plot for $\mathcal{M}_{T1}$




**(b)** Plot for $\mathcal{M}_{T2}$




**(c)** Plot for $\mathcal{M}_{T3}$




**(d)** Plot for $\mathcal{M}_{T4}$

**Figure A.2:** Plots of the SGVB estimate of ELBO (Labelled ELBO in plots) together with the KL divergence $D_{KL}(q_{\phi}(z|x)||p_{\theta}(z))$ for the different models for Twin-language models. The vertical line specifies the end of the KL-annealing.

## A.2 Generated Sentences

We let (T) stand for the true sentence, and (G) for the generated sentence.

| Sampled sentences from prior |
|---|
| the vote will take place on the debate , the council , and the commission . |
| i have already received a number of questions . |
| why , we have to be able to get the right to get the right ? |

**Table A.1:** Sampled sentences (EN) using the prior $p(z)$ of model $\mathcal{M}_{M1}$.

| Sampled sentences from posterior |
|---|
| (T) faced with this situation , europe has a decision to make .<br>(G) in the council , we are faced with a situation . |
| (T) now , that process certainly requires a reliable police force .<br>(G) this is a new process , the european union is needed . |
| (T) as i am pleased to say that the commission is going to be done .<br>(G) as i say , i shall be here with the presidency . |

**Table A.2:** Sampled sentences (EN) using the recognition model $q_{\varphi}(z|x)$ of model $\mathcal{M}_{M1}$.

| Sampled sentences from prior |
|---|
| what is the result of the vote , and i am not going to be taken . |
| as you believe that the commission is right , we need to be transparent . |
| i am not a great deal of a great deal of a great deal . |

**Table A.3:** Sampled sentences (EN) using the prior $p(z)$ of model $\mathcal{M}_{M2}$.

| Sampled sentences from posterior |
|---|
| (T) i hope that we will continue to support the commission .<br>(G) i hope we shall be able to maintain this cooperative approach throughout the subsequent presidencies . |
| (T) i agree with the need to support efficiency and effectiveness in the various education systems .<br>(G) i believe that we need to improve the need for greater transparency and efficiency . |
| (T) this decline is encouraging news .<br>(G) the situation is a very important . |

**Table A.4:** Sampled sentences (EN) using the recognition model $q_{\varphi}(z|x)$ of model $\mathcal{M}_{M2}$.

| Sampled sentences from prior |
|---|
| the commission is not the same thing , and the same . |
| ( the sitting was closed at UNK p.m . ) |
| the european union 's aim is to be welcomed . |

**Table A.5:** Sampled sentences (EN) using the prior $p(z)$ of model $\mathcal{M}_{M3}$.

| Sampled sentences from posterior |
|---|
| (T) the eu should stop these payments immediately .<br>(G) we must not be taken into account . |
| (T) that is the inescapable conclusion we must draw .<br>(G) the commission is very important . |
| (T) mr president , thank you very much for this debate .<br>(G) i would like to thank you for your attention . |

**Table A.6:** Sampled sentences (EN) using the recognition model $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x})$ of model $\mathcal{M}_{M3}$.

| Sampled sentences from prior |
|---|
| in my view the european union has been said . |
| now we are all UNK ! |
| as far ' s and UNK to being UNK to UNK UNK , to the UNK UNK . |

**Table A.7:** Sampled sentences (EN) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T1}$.

| Sampled sentences from posterior |
|---|
| (T) i find it sad that there are meps who are trying to weaken the commission ' s proposal .<br>(G) i find it is that the commission , however , the commission is not in the commission . |
| (T) mr president , commissioner , ladies and gentlemen , what is this debate actually about ?<br>(G) mr president , commissioner , ladies and gentlemen , what is going to do about ? |
| (T) mrs UNK was the rapporteur at the time .<br>(G) mrs UNK has to be of no to UNK . |

**Table A.8:** Sampled sentences (EN) using the recognition model $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ of model $\mathcal{M}_{T1}$.

| Sampled sentences from prior |
|---|
| madame la présidente , ce problème n'est pas tout une intervention des femmes . |
| pouviez , notre groupe aussi pour un communication pour cette approche . |
| lançons , UNK a la parole de la commission . |

**Table A.9:** Sampled sentences (FR) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T1}$.

| Sampled sentences from posterior |
|---|
| (T) il est regrettable , selon moi , que certains députés essayent d'affaiblir l'initiative de la commission .<br>(G) je suis néanmoins qu'il est que la commission que l'on est une proposition pas de la commission . |
| (T) monsieur le président , madame la commissaire , chers collègues , de quoi traite vraiment ce débat ?<br>(G) monsieur le président , madame la présidente , monsieur le commissaire , dont nous avons ce débat ? |
| (T) à l'époque , c'est mme UNK qui était rapporteur .<br>(G) à occupé de UNK , cela était de UNK . |

**Table A.10:** Sampled sentences (FR) using the recognition model $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ of model $\mathcal{M}_{T1}$.

| Sampled sentences from prior |
|---|
| mr president , i see on this report , in the vote . |
| i would like to thank wreckage to hope to be for the european union to be made . |
| you must be able to make for one of one . |

**Table A.11:** Sampled sentences (EN) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T2}$.

| Sampled sentences from posterior |
|---|
| (T) there is a real guerrilla war in progress .<br>(G) there is a great is the union is . |
| (T) however , later on , while i was asleep , i dreamt of mrs roth-behrendt .<br>(G) however , as of course , i have , i welcome , i have about . |
| (T) mr president , there is essentially nothing surprising about the situation today .<br>(G) mr president , there is no longer in the european parliament . |

**Table A.12:** Sampled sentences (EN) using the recognition model $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ of model $\mathcal{M}_{T2}$.

| Sampled sentences from prior |
|---|
| je tiens à présent à la commission à la commission . |
| nous UNK donc cette décision au conseil . |
| par conséquent ce sont les citoyens . |

**Table A.13:** Sampled sentences (FR) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T2}$.

| Sampled sentences from posterior |
|---|
| (T) une guérilla énorme est en cours .<br>(G) la russie est en effet est . |
| (T) puis , je me suis quand même UNK , et j'ai rêvé de mme roth-behrendt .<br>(G) et je tigre , je me UNK , je me UNK , et je me UNK . |
| (T) monsieur le président , la situation d'aujourd'hui n'a au fond rien d'étonnant .<br>(G) monsieur le président , la situation n'est pas la situation sont la situation . |

**Table A.14:** Sampled sentences (FR) using the recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ of model $\mathcal{M}_{T2}$.

| Sampled sentences from prior |
|---|
| to those member states , for proposal for UNK would not support the most people . |
| we would like to make a new measures that of this point . |
| that is something we are now today . |

**Table A.15:** Sampled sentences (EN) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T3}$.

| Sampled sentences from posterior |
|---|
| (T) it is a compromise .<br>(G) this is a compromise . |
| (T) that is why this aid programme needs to be made a top priority .<br>(G) that is why the only to do is about to be in UNK . |
| (T) yet it is a country that is important to europe .<br>(G) yet it is an UNK of an UNK for europe . |

**Table A.16:** Sampled sentences (EN) using the recognition model $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ of model $\mathcal{M}_{T3}$.

| Sampled sentences from prior |
|---|
| il convient est souvent de nouvelles à cet aspect pas les UNK à cet accord . |
| cependant , nous sommes tous , tout , à bien se fait . |
| c'est pourquoi j'ai également que notre travail à faire dans notre dans ce sens . |

**Table A.17:** Sampled sentences (FR) using the prior $p(\boldsymbol{z})$ of model $\mathcal{M}_{T3}$.

| Sampled sentences from posterior |
|---|
| (T) c'est un compromis .<br>(G) c'est un compromis . |
| (T) c'est pourquoi ce programme d'assistance doit avant tout être prioritaire .<br>(G) c'est pourquoi ce programme doit être une question au niveau . |
| (T) mais c'est un pays important pour l'europe .<br>(G) mais il s'agit d'un pays pour l'europe . |

**Table A.18:** Sampled sentences (FR) using the recognition model $q_{\varphi}(z|x, y)$ of model $\mathcal{M}_{T3}$.

| Sampled sentences from prior |
|---|
| mrs UNK has been UNK UNK UNK to the present in the general assembly . |
| this is also that this on this countries . |
| and that time , with the situation by the international community of the international organizations of international organizations . |

**Table A.19:** Sampled sentences (EN) using the prior $p(z)$ of model $\mathcal{M}_{T4}$.

| Sampled sentences from posterior |
|---|
| (T) development project for indigenous communities<br>(G) international convention for development |
| (T) working party of general safety ( grsg )<br>(G) working group of united nations took ) |
| (T) in these new circumstances , UNK solidarity is no longer enough , at least for the socialists .<br>(G) in the UNK , UNK , UNK , they on the UNK of UNK of UNK of UNK . |

**Table A.20:** Sampled sentences (EN) using the recognition model $q_{\varphi}(z|x, y)$ of model $\mathcal{M}_{T4}$.

| Sampled sentences from prior |
|---|
| de plus que ont et de la proposition et ne et ont été sont et du rapport . |
| mecque pays en programmes de UNK par a |
| enthousiasme du conseil de sécurité |

**Table A.21:** Sampled sentences (FR) using the prior $p(z)$ of model $\mathcal{M}_{T4}$.

| Sampled sentences from posterior |
|---|
| (T) projet de développement des communautés autochtones<br>(G) projet de rapporte pour le développement |
| (T) rapport du comité du programme et de la coordination ( UNK ( chap .<br>(G) rapport du comité pour le programme et de la coordination ( UNK ) ( UNK ) . |
| (T) en ces nouvelles circonstances , la solidarité UNK n'est plus suffisante , du moins de l'avis des socialistes .<br>(G) en ce sont , UNK ont , ont des UNK et ont , sont des UNK de UNK . |

**Table A.22:** Sampled sentences (FR) using the recognition model $q_{\varphi}(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$ of model $\mathcal{M}_{T4}$.

# Bibliography


[1] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

[5] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. arXiv: 1312.6114.

[6] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*, January 2014.

[7] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*, November 2015. arXiv: 1511.06349.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA, 2012.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[11] George Casella and Roger Berger. *Statistical Inference*. Duxbury Resource Center, June 2001.

[12] Yoav Goldberg. A primer on neural network models for natural language processing, 2015. cite arxiv:1510.00726.

[13] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.

[14] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989.

[15] David Barber. Lecture notes in applied machine learning, September 2016.

[16] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017.

[17] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[19] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[20] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, page 2000, 2000.

[21] Lenhart Schubert. Computational linguistics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2015 edition, 2015.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[25] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[26] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[27] Krzysztof Wo lk and Krzysztof Marasek. Neural-based machine translation for medical text domain. Based on European Medicines Agency leaflet texts. *Procedia Computer Science*, 64:2–9, 2015. arXiv: 1509.08644.

[28] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144 [cs]*, September 2016. arXiv: 1609.08144.

[29] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*, September 2014. arXiv: 1409.1259.

[30] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016.

[31] Léon Bottou. Stochastic gradient descent tricks. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks*

*of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2012.

[32] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[33] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

[35] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[36] David Barber. Deep learning: Autodiff, parameter tying and backprop through time, 2015.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[38] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[39] Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE, 2011.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[41] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT.

[42] Micha l Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. The united nations parallel corpus v1.0. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[43] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[44] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, et al. Lasagne: First release., August 2015.

[45] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[46] Michael Waskom, Olga Botvinnik, Paul Hobson, John B. Cole, Yaroslav Halchenko, Stephan Hoyer, Alistair Miles, Tom Augspurger, Tal Yarkoni, Tobias Megies, Luis Pedro Coelho, Daniel Wehner, cynddl, Erik Ziegler, diego0020, Yury V. Zaytsev, Travis Hoppe, Skipper Seabold, Phillip Cloud, Miikka Koskinen, Kyle Meyer, Adel Qalieh, and Dan Allan. seaborn: v0.5.0 (november 2014), November 2014.

[47] Stefan van der Walt, S. Chris Colbert, and Gael Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30, March 2011.

[48] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[49] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016.

[50] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.