Advances in Probabilistic Deep Learning

Raza Habib

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 20, 2022

I, Raza Habib, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis is concerned with methodological advances in probabilistic inference and their application to core challenges in machine perception and AI. Inferring a posterior distribution over the parameters of a model given some data is a central challenge that occurs in many fields ranging from finance and artificial intelligence to physics. Exact calculation is impossible in all but the simplest cases and a rich field of approximate inference has been developed to tackle this challenge. This thesis develops both an advance in approximate inference and an application of these methods to the problem of speech synthesis. In the first section of this thesis we develop a novel framework for constructing Markov Chain Monte Carlo (MCMC) kernels that can efficiently sample from high dimensional distributions such as the posteriors, that frequently occur in machine perception. We provide a specific instance of this framework and demonstrate that it can match or exceed the performance of Hamiltonian Monte Carlo without requiring gradients of the target distribution. In the second section of the thesis we focus on the application of approximate inference techniques to the task of synthesising human speech from text. By using advances in neural variational inference we are able to construct a state of the art speech synthesis system in which it is possible to control aspects of prosody such as emotional expression from significantly less supervised data than previously existing state of the art methods.

Impact Statement

The work presented in this thesis has had or has the potential to have academic, industrial and social impacts.

On the academic front, much of this work has been published at major machine learning conferences and has been cited over 50 times in follow-on-research. The MCMC methods presented in chapter 3 contribute to our understanding of approximate inference which has wide applicability both in machine learning but also science and statistics more broadly. The work presented in chapter 4 takes us a step closer to controllable speech synthesis and demonstrates how pure deep learning methods can be productively combined with probabilistic models.

In industry the methods developed for controllable speech synthesis have direct application to commercial products such as digital assistants and automated book reading.

There is also the potential for significant positive social impact since speech synthesis can be used to assist people who have lost the ability to speak through diseases such as ALS.

Acknowledgements

I'm very grateful to Professor David Barber for encouraging me to do a PhD and thus kicking off what I'm sure will be a lifelong passion for independent discovery. I'm also grateful to him for guidance, support and many fruitful hours spent bouncing around ideas. I'm also indebted to Matt Shannon and Soroosh Mariooryad for a crash course on speech synthesis and introducing me to a fascinating field. The PhD would not have been anywhere near as enjoyable without our wider research group and many ideas were hatched in discussions with Tom Bird, Jamie Townsend, Alex Botev and Peter Hayes. I would never have discovered machine learning had it not been for the incredible teaching of Professor Sir David Mackay. I'm grateful to my parents who have given me opportunities that few can hope for and to my wife, Alexandria, for enduring support and making it feel worthwhile. Finally, I'd like to thank Dada for always being an inspiration and encouraging me to see the thesis through to the end.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning (ML) has become a critically important technology for modern society and its significance is rising. It's already used by billions of people daily: It's used through recommendation engines that influence buying habits, and through machine translation systems that make communication between previously disconnected people possible. It's used in speech synthesis and recognition to create more accessible computer interfaces and digital assistants. In the physical world, computer vision and reinforcement learning will be crucial to producing autonomous vehicles that will likely change both how we travel and transport goods. In engineering, machine learning is used to better design experiments and rapidly approximate expensive physical simulation. The pharmaceutical industry has started to use machine learning in drug discovery and researchers have successfully applied it to related scientific problems such as protein folding. This is far from an exhaustive list of applications.

A remarkable aspect of machine learning is that similar learning algorithms and techniques can be used in diverse applications. It's not at all obvious that the same methods that are used in machine translation should be applicable to software that recognises objects in images, synthesises speech, answers questions or plays games. In practice, however, the state of the art in all of these applications share very similar foundations. That a lot of valuable and diverse applications stem from a small number

of approaches, makes research into fundamental methods (rather than directly into applications) a very high leverage activity. Improvements in core learning algorithms can lead to improvements across a very wide range of end uses and can be adopted quickly.

This thesis presents two distinct but interconnected attempts at making such a contribution: Auxiliary Variational Sampling (Chapter 3) and Controllable Speech Synthesis (Chapter 4). The unifying theme connecting this work is an attempt to make fundamental methodological improvements by combining deep learning and Bayesian machine learning.

Recently, deep learning has become the dominant approach in many sub-fields of machine learning. It began with multi-layer perceptrons, very simple models of neurons in the brain, but it has come to refer to a wider family of techniques characterised by the composition of simple differentiable primitives trained through stochastic gradient optimisation on large data sets. Deep learning has produced state of the art results in many areas of machine perception including computer vision and speech recognition and is increasingly the best performing approach in many natural language processing and reasoning tasks as well.

However, Deep learning has known limitations. Deep learning models are hard to interpret, require large volumes of annotated training data and don't explicitly model or represent uncertainty. The task of rapidly incorporating new data into an existing deep learning model or explicitly incorporating prior knowledge remains an open research problem.

Bayesian machine learning is an alternative approach to constructing learning algorithms that provides a principled way of combining prior knowledge with new data via Bayes rule. Bayesian learning can perform well with small data-sets, assumptions are stated explicitly and uncertainty is represented through probability distributions.

This thesis provides two contributions that demonstrate Deep learning and probabilistic machine learning have complementary strengths and weaknesses and that there

is much to be gained by building systems that incorporate elements of both. The core mathematical machinery used in both chapters is extremely similar and relies on using probabilistic models parameterised by neural networks that have come to be known as "Variational Autoencoders" [Rezende et al., 2014].

In chapter 3 we present a novel approach to learning MCMC kernels that are able to mix efficiently. In this chapter we are using deep learning methods to accelerate a more traditional form of probabilistic inference. Key to the success of our method is to parameterise the MCMC kernels with flexible function approximators. We show that using a probabilistic model parameterised by a deep neural network is a very natural choice.

In chapter 4 we show how augmenting a purely heuristic neural speech synthesis system with probabilistic latent variables allows us to have better control over the generated speech. Unlike chapter 3, in this chapter we are using probabilistic machine learning to improve an otherwise purely neural system.

The two main contributions are based on work originally published in the following papers:

- "Semi-Supervised Generative Modelling for Controllable Speech Synthesis." International Conference on Learning Representations. 2019.
- "Auxiliary variational MCMC." International Conference on Learning Representations. 2018.

The structure of the thesis is as follows: In chapter 2 we present the necessary background to understand the core contributions on the thesis and its context. In chapter 3 we present Auxiliary Variational MCMC, a framework for combining variational inference with MCMC to produce rapidly mixing MCMC kernels. We also present a specific instance of that framework. In chapter 4 we present a novel method for controllable speech synthesis that combines probabilistic latent variable models with neural speech synthesis.

Chapter 2

Background

This chapter aims to give a concise overview of the necessary background material for understanding the rest of the thesis and its context within the existing literature. In section 2.1 we give a brief introduction to the Bayesian approach to machine learning and explain the need for approximate inference. Then in section 2.2 we introduce Markov Chain Monte Carlo and some of its modern variants. We then give a minimal introduction to Variational Inference as alternative approach in section 2.3 as it is needed both to understand the MCMC framework in chapter 3 and the training of the model in chapter 4. We end the chapter with a brief background on neural speech speech synthesis which will be necessary to understand the contributions of chapter 4.

2.1 Bayesian Machine Learning

Probabilistic or Bayesian inference provides a principled way of combining prior knowledge with new data to update conclusions. The core equation in Bayesian inference is a simple rewriting of the definition of conditional probabilities, known as Bayes' rule

$$P(A|B) = P(A)\frac{P(B|A)}{P(B)}. \tag{2.1}$$

Often the random variable A represents some hypothesis about the world and B is some data (modelled itself as a random variable). In this case the rule is written

$$P(hypothesis|data) = P(hypothesis)\frac{P(data|hypothesis)}{P(data)}. \tag{2.2}$$

Written like this the equation shows how one can update a prior belief, $P(hypothesis)$, in light of some new data. The $P(data|hypothesis)$ is known as the likelihood and the equation tells us to adjust our prior belief by the ratio of how probable our data would be if the hypotheses were true compared to how probable they would be in general. The Bayesian approach to inference is to first summarise our knowledge of the world by choosing specific forms for the likelihood and prior distributions. Then, given some data, to compute either the posterior distribution or expectations of key quantities under this posterior.

Even though it appears extremely simple, the Bayesian approach comes with a range of impressive theoretical justifications. Quantifying our beliefs using probabilities is justified by "Cox's Axioms" [Jaynes, 1996], which show that under very mild desiderata the only consistent scalar representation of beliefs must obey the sum rule and product rule of elementary probability.[1] Bayesian reasoning also comes with an asymptotic consistency result which shows (under some regularity conditions) that in the limit of infinite data, model parameters will concentrate on the true data generating parameters or if the model class under consideration does not contain the true data generating process then to the closest possible model as measured by KL-Divergence [Ghahramani, 2015]. There are also consistency results that show that the impact of the assumed prior distributions vanishes in the limit of infinite data [Ghahramani, 2015]. Typically the likelihood and prior are specified with a particular parametric form (e.g Gaussian) and this can often be justified by appeal to De Finetti's theorem which states that any exchangeable set of random variables may be written as a (possibly infinite) parametric mixture [Kirsch, 2019].

[1] Here by consistency we mean that equivalent statements are given equivalent degrees of belief, all available evidence is considered and all reasoning methods give the same result

The rich theoretical justification for Bayesian methods has motivated a host of successful applications both across science and within the field of machine learning. Bayesian methods, like those developed in this thesis, have been used in physics to model the early cosmos [Jennings, 2019], in finance for time-series forecasting [Koop et al., 2010] and in machine learning. In machine learning they are often employed in situations where explicitly modelling uncertainty is vital, such as reinforcement learning [Russo et al., 2018] or active learning [Settles, 2009, Houlsby et al., 2011].

The challenge of a probabilistic approach to machine learning is that, despite its apparent simplicity, the computation of posterior expectations is rarely tractable. The calculation usually requires a summation over an infeasibly large set or a high dimensional integration that cannot be written in closed form. Consider the standard problem of calculating the predictive distribution over a new data-point x^N, given a statistical model with parameters θ and historical data $X^{1\ldots N-1}$

$$P(x^N|x^{<N}) = \int p(x^N|\theta)p(\theta|x^{<N})d\theta. \tag{2.3}$$

This equation requires calculating an average over all possible settings of the parameters of the model. The normaliser of the posterior itself requires a similar integral or sum.

$$p(\theta|x^{<N}) = p(\theta)\frac{p(x^{<N}|\theta)}{\int p(\theta)p(x^{<N}|\theta)d\theta} \tag{2.4}$$

In some special cases these integrals are feasible, such as if there is conjugacy between posterior and prior or in tree structured graphical models. However this is generally not the case and a rich field of study in approximate inference has been developed to solve these problems.

The vast majority of inference methods are either some form of sampling based approach like Markov Chain Monte Carlo (MCMC) or variational inference. Varia-

tional inference attempts to convert challenging integrals into optimisation problems by bounding the integrals in question. MCMC based methods, approximate the posterior integrals with finite averages over samples drawn from the posterior distribution.

2.2 Monte Carlo Methods

2.2.1 Simple Monte Carlo

The core idea of Monte Carlo methods is to approximate the intractable expectations that occur frequently in machine learning and statistics with averages under samples drawn from the distribution of interest. The simplest Monte Carlo method relies on the following result

$$\frac{1}{N}\sum_{n} f(x_n) \xrightarrow[N\to\infty]{a.s.} \int f(x)p(x)dx \tag{2.5}$$

where $x_n \sim p(x)$. This forms an unbiased estimate and, as long as variances are bounded appropriately, obeys a central limit theorem; the estimator's variance will scale as $\frac{1}{N}$. Though a $\frac{1}{N}$ reduction in variance might seem slow, the fact that this reduction is independent of dimension makes it an important and popular approach. Despite the simplicity and utility of the Monte-Carlo approximation, drawing the necessary samples is challenging for all but the most basic distributions.

2.2.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) introduces another layer of approximation above simple Monte Carlo and is a strategy for approximately drawing samples from otherwise intractable distributions. It has the advantage that it can be used in any case where the target distribution has a density which can be evaluated point-wise. The cost for the generality of the method is that the samples it generates are no longer independent and are only asymptotically guaranteed to be from the distribution of interest.

MCMC draws samples from a target distribution p by simulating a Markov chain

whose stationary distribution is p. The only requirement is that p can be evaluated point-wise, up to a multiplicative constant. It is easiest to understand for finite state-spaces, where the samples $x_n \sim p(x)$ can only take on a discrete set of values, $x \in 1 \dots K$. A stochastic process is a (first order) Markov chain iff:

$$p(x_t|x_1, x_2, x_3 \dots x_{t-1}) = p(x_t|x_{t-1}) \tag{2.6}$$

i.e it generates an ordered sequence of random variables each of whose distribution is independent of its history given the present state of the chain. For discrete state spaces $p(x_t|x_{t-1})$ can be represented by a KxK transition matrix, M, whose ij^{th} element represents the probability that $x_t = j|x_{t-1} = i$. If this transition matrix is constant, the chain is said to be homogeneous [2]. The marginal distribution on x_t in this chain is given by

$$p(x_t) = \sum_{x_{<t}} p(x_0, x_1, x_2, x_3 \dots x_t) \tag{2.7}$$

$$= \sum_{x_{<t}} p(x_0) \prod_{k=1}^{t} p(x_k|x_{k-1}) \tag{2.8}$$

$$= M^t p(x_0) \tag{2.9}$$

where $p(x_0)$ is a K dimensional vector whose i^{th} element represents the probability that $p(x_0 = i)$ and M^t represents the application of the transition matrix t times. It follows from equation (2.9) that if our target distribution p is an eigenvector of M with eigenvalue 1 then it will be left invariant by this transition. This condition (stationarity) alongside two other conditions on M (that it be a-periodic and irreducible) are sufficient to guarantee that no matter what the initial distribution $p(x_0)$ is, the marginal distribution of the chain will *eventually* converge to the target distribution

[2] Many authors define these transition matrices to be the transpose of my definition here but I prefer the notation in probability vectors are columns and transition matrices act to their right.

we wish to sample from. To draw samples from our target distribution, we need only simulate this chain and we are guaranteed that eventually the states the chain passes through will be samples from our target. The rate of convergence will depend on the spectral properties of the transition operator M. The challenge of MCMC is to construct transition operators M that have both the necessary properties but also have fast rates of convergence. If p is an eigenvector of M with an eigenvalue of 1, then the Perron-Frobenius theorem [Andrieu et al., 2003] tells us that the absolute value of the remaining eigenvalues is less than 1 and the rate of convergence will be determined by the size of the next largest eigenvalue.

Three conditions sufficient to guarantee convergence of the chain with transition operator M to the correct target distribution p are:

1. Stationarity: Mp = p

2. Irreducibility: For any state in the chain there must exist a non-zero probability of visiting every other state in a finite number of steps.

3. Aperiodicity. The period of a given state in the chain is defined as $gcd\{n : P(\text{return to i from i in n steps}) > 0\}$. The chain is a-periodic if the largest period of any state is 1. I.e the chain does not contain cycles.

Intuitively, the latter two conditions ensure that the marginal distribution of the chain will eventually reach p and the first condition ensures that if we get to p the marginal distribution stays there. The above discussion has focussed on finite-state spaces but can be easily carried over to continuous distributions by replacing the vector p with a continuous density and the matrix M with a continuous transition kernel $T(x'|x)$. The sums in the above equations then become integrals, for example the stationarity condition becomes $p(x) = \int_{x'} T(x|x')p(x')dx'$.

We have now stated sufficient conditions to construct a Markov chain whose stationary distribution converges asymptotically to some target distribution of interest and turn to the question of how to construct such a process in practice.

Algorithm 1: Metropolis Hastings

Initialize x_0
for $t = 1, \ldots, T$ **do**
  $x' \sim K(x'|x_{t-1})$
  $A \leftarrow \frac{p(x')K(x_{t-1}|x')}{p(x_{t-1})K(x'|x_{t-1})}$ ▷ acceptance ratio
  $u \sim U[0,1]$ ▷ sample u from uniform distribution
  if $u \leq A$ **then**
    $x_t \leftarrow x'$
  else
    $x_t \leftarrow x_{t-1}$
  end
end

2.2.3 Metropolis-Hastings Algorithm

An important and popular practical algorithm for constructing MCMC kernels with the necessary properties stated above is the Metropolis-Hastings algorithm. The Metroplis-Hastings algorithm provides a recipe for taking any Markov transition operator $K(x'|x)$ that shares the support of the target, $p(x)$, and converting it to a new transition operator $T(x'|x)$ that converges to $p(x)$.

The method is simple to state and we will outline the algorithm first before explaining why it guarantees convergence to the distribution of interest: We draw an initial state x_0 from any distribution we like and then conditioned on x_0 we sample a new state, x', from our arbitrary transition operator $K(x'|x)$. We then calculate the ratio $a = min\{1, \frac{p(x)K(x'|x)}{p(x')K(x|x')}\}$ and set $x_1 = x'$ with probability a and otherwise set $x_1 = x_0$. Iterating these two steps will eventually ensure that the marginal distribution on samples converges to the target distribution, $p(x)$. The algorithm is shown in detail above in algorithm 1.

To see that this algorithm does indeed converge to the distribution of interest we need to show it satisfies the three conditions of stationarity, a-periodicity and irreducibility. To ensure that $p(x)$ is a stationary point of the transition operator implied by the above process, $T(x'|x)$, the Metropolis-Hastings algorithm actually ensures a stronger condition known as detailed balance:

$$p(x)T(x'|x) = p(x')T(x|x') \tag{2.10}$$

It's straightforward to show that detailed balance is a sufficient condition for $p(x)$ to be stationary, since

$$\int T(x|x')p(x')dx' = \int T(x'|x)p(x)dx' = p(x)\int T(x'|x)dx' = p(x) \tag{2.11}$$

where in the second step we have used the detailed balance condition. The transition operator defined by the Metropolis-Hastings algorithm is given by

$$T(x'|x) = A(x',x)K(x'|x) + r(x)\delta(x-x') \tag{2.12}$$

where we have defined the acceptance probability $A(x',x) = min\{1, \frac{p(x)K(x'|x)}{p(x')K(x|x')}\}$ and the rejection probability $r(x) = \int_{x'} K(x'|x)(1-A(x',x))dx'$. To see that this transition operator satisfies detailed balance, first consider the case that $x' = x$. In this case $T(x'|x) = T(x'|x')$ is clearly symmetric and detailed balance is obviously satisfied. If $x' \neq x$ then

$$\begin{aligned}
p(x)T(x'|x) &= p(x)K(x'|x)A(x',x) && (2.13)\\
&= p(x)K(x'|x)min\left\{1, \frac{p(x')K(x|x')}{p(x)K(x'|x)}\right\} && (2.14)\\
&= min\left\{p(x)K(x'|x), p(x')K(x|x')\right\} && (2.15)\\
&= p(x')K(x|x')min\left\{1, \frac{p(x)K(x'|x)}{p(x')K(x|x')}\right\} && (2.16)\\
&= p(x')T(x|x') && (2.17)
\end{aligned}$$

and so again detailed balance is satisfied. Since detailed balance is sufficient for $p(x)$

to be stationary, we have shown that the Metropolis-Hastings transition operator is guaranteed to have $p(x)$ as a stationary distribution. Since rejection is always possible it is clear that the chain induced by $T(x'|x)$ is a-periodic and irreducibility is guaranteed by the requirement that $K(x'|x)$ shares the support of $p(x)$.

These three conditions taken together, ensure that the Metropolis-Hastings algorithm is guaranteed in theory to produce samples from any distribution of interest for any proposal kernel $K(x'|x)$ that shares the support of $p(x)$. This guarantee, however, only applies asymptotically and in practice the rate of convergence depends strongly on the choice of the proposal kernel $K(x'|x)$.

2.2.4 Random Walk Metropolis

A frequently used MCMC method is a special instance of the Metropolis-Hastings algorithm, known as Random Walk Metropolis (RWM). In RWM, the proposal distribution is chosen to be a Gaussian centred on the current sample from the Markov chain. ie. The proposal distribution is given by

$$p(x'|x) = |2\pi\Sigma|^{-\frac{1}{2}} e^{-\frac{1}{2}(x'-x)^T\Sigma^{-1}(x-x')} \tag{2.18}$$

RWM is popular largely because of its simplicity and ease of implementation. However, exactly because it explores the target distribution as a random walk it can be extremely slow to converge. In high-dimensional problems a large random perturbation from a typical point (a sample from the target distribution) will usually have low probability and so will rarely be accepted. If the perturbations are made small then it will take a long time for the MCMC algorithm to explore the typical set of a high dimensional target distribution and convergence will again be slow.

This idea can be approximately quantified by noting that a random walk in D dimensions will on average travel a distance of $\sqrt{T}\sigma$ in T steps (where σ is the average step size) independent of dimension. That means that if the largest length scale of a distribution is L, random walk Metropolis will need $\approx (\frac{L}{\sigma})^2$ steps to

Algorithm 2: Hamiltonian Monte Carlo

Initialize x_0, ε, L
$x \leftarrow x_0$
for $t = 1, \ldots, T$ **do**
  $v \sim N(0, I)$ ▷ Perform a Gibbs Sampling step for v
  $H \leftarrow E(x) + \frac{1}{2} v^T v$
  for $l = 1, \ldots, L$ ▷ Perform L leapfrog steps
  do
    $v \leftarrow v - \frac{\varepsilon}{2} \nabla_x E(x)|_x$
    $x \leftarrow x + \varepsilon v$
    $v \leftarrow v - \frac{\varepsilon}{2} \nabla_x E(x)|_x$
  end
  $H' \leftarrow E(x) + \frac{1}{2} v^T v$
  $\Delta H \leftarrow H' - H$
  if $\Delta H \leq 0$ ▷ Metropolis-Hastings accept-reject step
  then
    $x_t \leftarrow x$
  else
    $u \sim U[0,1]$ ▷ sample u from uniform distribution
    if $u \leq \exp(-\Delta H)$ **then**
      $x_t \leftarrow x$
    else
      $x_t \leftarrow x_{t-1}$
    end
  end
  $x \leftarrow x_t$
end

converge. As the dimensionality of the target distribution increases it becomes increasingly likely that the target distribution will be very constrained in some dimensions and not in others but Random Walk Metropolis will be limited by the length scale of the most constrained dimension.

In order to suppress random walk behaviour and produce more quickly converging samplers, a wide range of more advanced MCMC methods have been developed. These include adaptive samplers, MCMC methods with data driven proposals and auxiliary variable methods [Andrieu and Thoms, 2008]. We include a discussion below of a small selection of these methods that are most relevant to the new framework proposed in chapter 3.

2.2.5 Hamiltonian Monte Carlo and its Variants

HMC

Hamiltonian Monte Carlo (HMC) [Duane et al., 1987] is an auxiliary variable method that can be used to sample from distributions on continuous state spaces. Instead of directly sampling from the target distribution, $p(x)$, HMC augments the sample space with an auxiliary random variable, v, and draws samples from the joint distribution $p(x,v) = p(x)p(v|x)$. By construction, the marginal distribution on x is the target distribution of interest, so it is valid to simply discard the samples from $p(v)$. It may not at first be apparent that enlarging the sample space should make sampling easier. However, HMC uses the extra auxiliary variables as part of a carefully designed Metropolis-Hastings proposal that can approximately trace contours of equal probability in the joint x, v space. This makes it possible to accept proposals that travel long distances and so explore the target distribution more efficiently.

If we write the target density $p(x) \propto e^{-E(x)}$ and chose $p(z)$ to be a standard normal distribution. Then the joint density on x, v may be written as

$$p(x,v) \propto \exp\{-H(x,v)\} \tag{2.19}$$

where we have defined $H(x,v) = E(x) + \frac{1}{2}v^T v$. This choice of notation is deliberately suggestive of the Hamiltonian of classical physics which is typically written as the sum of a potential energy term, $E(x)$, and a kinetic energy term, $\frac{1}{2}v^T v$. Hamilton's equations of motion in classical physics are a pair of coupled continuous time differential equations that have the very desirable property that they evolve x and v whilst conserving the value of $H(x,v)$. Hamilton's equations are given by

$$\nabla_x H(x,v) = -\frac{\partial v}{\partial t} \tag{2.20}$$

$$\nabla_v H(x,v) = \frac{\partial x}{\partial t} \tag{2.21}$$

HMC alternates between sampling a value v from $p(v)$ and then proposing a new value of x by integrating Hamilton's equations for some time. As the dynamics preserve probability density, the start and end point of a trajectory will have the same probability and will be automatically accepted under Metropolis-Hastings. In some rare cases the equations can be exactly integrated but typically this is not possible and so a discrete time approximation is used. We start from a sample (x, v) and propose a new sample (x', v') by iterating the following set of equations, known as the leapfrog integrator

$$
\begin{aligned}
v &\leftarrow v - \frac{\varepsilon}{2}\nabla_x E(x)|_x \\
x &\leftarrow x + \varepsilon v \\
v &\leftarrow v - \frac{\varepsilon}{2}\nabla_x E(x)|_x
\end{aligned}
$$

where ε and the number of leapfrog steps used, L, are hyper-parameters of the method. The leapfrog integrator is used over other simpler options like Euler integration because it has the desirable property that it preserves both density and volume in the x, v space [Neal et al., 2011]. This volume preservation minimises the accumulation of error and so ensures that the method still approximately traces contours of equal probability density.

HMC uses a deterministic proposal $(x', v') = f(x, v)$, which is given by first performing L leapfrog steps starting at x, v and then negating the auxiliary variable v to $-v$. This final-step is technically necessary to ensure that the proposal is reversible but often ignored in practice because sampling v from $p(v)$ randomises v immediately afterwards. Formally, since the proposal is deterministic, it doesn't have a well defined density and the Metropolis-Hastings acceptance ratio must be replaced by the Metropolis-Hastings-Green [Green, 1995] acceptance ratio which accounts for

this:

$$A(x,v',x,v) = min\left\{1, \frac{p(x,v)}{p(x',v')}|J_f|\right\} \tag{2.22}$$

where J_f is the Jacobian of the overall leapfrog integrator. In practice, because the leapfrog integrator preserves volume, the determinant of the Jacobian is 1 [Neal et al., 2011] and the two acceptance ratios are the same. The full algorithm for HMC is shown in detail in algorithm 2.

During each HMC proposal, the motion of x has some persistence in the direction of v and so can travel long distances. Density preservation ensures the acceptance rate stays high. Compared to Random Walk Metropolis, HMC can make larger proposals and so can explore the space more efficiently and converge more quickly. HMC was orignaly introduced into the machine learning literature in order to sample from the high dimensional posterior distributions of small Bayesian neural networks and is successful in suppressing random walk behaviour [Neal, 2012].

HMC is not however a perfect solution. Its main deficits are a strong sensitivity to its two hyper-parameters and the requirement that the gradient of the target likelihood be computed many times per sample. The two hyper-parameters are, L, the number of leapfrog steps used per sample and, ε, the step-size used in the leapfrog dynamics. In practice if either of these parameters is poorly chosen than HMC can perform badly and some skill and experience is required to set them.

If ε is set too small then the average distance between proposals will be small and we re-inherit the problems of Random Walk Metropolis. If ε is chosen too large than discretisation error can lead to most proposals being rejected and narrow valleys of probability density may become inaccessible. Choosing a large number of leapfrog steps increases the number of gradient calculations required and can become expensive. Rules of thumb have been developed to help overcome these problems [Neal et al., 2011]. In general it is advisable to randomly vary ε from some

appropriately chosen distribution and to choose L based on examining trace-plots from preliminary runs. Trace-plots are graphs of key parameters (e.g an expected value) as a function of step number in the Markov chain.

Since its introduction to Machine Learning, HMC has become a popular algorithm and a large number of variants have been developed to mitigate its deficiencies [Levy et al., 2018, Hoffman and Gelman, 2014, Strathmann et al., 2015]. Two important research directions are methods for automated tuning, such as the No U-Turn Sampler (NUTS) [Hoffman and Gelman, 2014], and methods that deal with expensive gradient computation using stochastic gradients, such as Stochastic Gradient HMC [Ma et al., 2015] and Stochastic Gradient Langevin Dynamics [Welling and Teh, 2011].

NUTS

NUTS develops methods for setting both ε and L automatically. The number of leapfrog steps is set by repeatedly doubling the trajectory length until there is a part of the trajectory that turns back on itself and then choosing a sample from that trajectory. ε is adapted in early parts of sampling to achieve a desired acceptance rate and then held constant. Thanks to a high performance C implementation in the STAN library [Carpenter et al., 2017], NUTS has become an important and practically used HMC variant. Since NUTS is primarily a method for tuning HMC, well tuned HMC should match or exceed its performance [Neal] and we do not explicitly compare against it in our experiments.

Stochastic Gradient HMC

SGHMC and SGLD tackle a different problem that has become particularly important when sampling from posterior distributions in modern machine learning where large data-sets have become the norm. The posterior is proportional to the likelihood and in many models the likelihood decomposes into a product over data points. This means that the cost of likelihood evaluation and gradient evaluation scales linearly with data-set size and can become intractable for modern data-sets with millions of samples. This is a problem for all Metropolis methods, which require likelihood evaluation to compute acceptance rates, but is particularly serious for HMC

because potentially hundreds of gradient evaluations are required per MCMC sample. Stochastic Gradient HMC and Stochastic Gradient Langevin Dynamics are both methods that attempt to replace the full batch gradients in the leapfrog discretisation with approximate gradients created by subsampling the data. SGLD performs just one step of the leapfrog dynamics per sample and maintains ε sufficiently small that the Metropolis acceptance step may be omitted. The SGLD update has a particularly simple form, in that it is equivalent to stochastic gradient descent with added noise. Although it can scale to modern data sizes, because it only uses one leapfrog step it suffers from random walk like behaviour and slow mixing. SGHMC modifies the leapfrog dynamics such that in the limit that the step-size ε tends to 0, the target distribution remains invariant even with noisy gradients. However in practice it is typical to use a small finite step-size, ε, and neglect the expensive Metropolis-Hastings accept-reject step. In general, stochastic gradient MCMC methods present a trade off between expensive computation and biased samples.

2.2.6 Neural Samplers

In chapter 3, we introduce a novel MCMC scheme that like HMC uses auxiliary variables to try and produce Metropolis-Hastings proposals with large step-sizes and suppress random walk behaviour. Our method is motivated by similar ideas to HMC; We wish to construct proposals that can be large distances from the current state of the MCMC chain whilst still having similar probability density to the current point. However, we achieve this without needing to calculate gradients of the target distribtuion during sampling. In our experiments we parameterise our sampler with neural networks and discuss here related work that also uses the flexibility and experessive power of neural networks to accelerate MCMC sampling.

L2HMC

In their paper "Generalizing Hamiltonian Monte Carlo with Neural Networks", Levy et al. [2018] parameterise the leapfrog update equations of HMC and adapt the parameters during burn-in to optimise the expected squared distance moved between proposals. The resulting algorithm is referred to as L2HMC.

For Metropolis-Hastings with a deterministic proposal, it is necessary that the proposal both be reversible and have a Jacobian with a tractable determinant so that the acceptance probability, equation 2.22, may be computed. In the case of standard HMC this is straightforward because the leapfrog integrator is reversible and volume preserving. In L2HMC the authors overcome this challenge by using Real Non-Volume-Preserving Flows (RNVP) [Dinh et al., 2016] to enforce this condition whilst varying the leapfrog dynamics.

Real Non-Volume Preserving Flows are feed-forward neural networks that are carefully designed to both be reversible and have tractable determinants. They achieve this by partitioning the input to the network, x, into two parts, $x = [x_{1:d}, x_{d+1:D}]$ and updating each part dependent only on the other part. The function used is known as an "affine coupling layer"

$$y_{1:d} = x_{1:d} \tag{2.23}$$

$$y_{d+1:D} = \exp(f(x_{1:d})) \odot x_{d+1:D} + g(x_{1:d}) \tag{2.24}$$

where f and g are arbitrary differentiable functions and $\odot$ represents element-wise multiplication. These equations are straightforward to invert without needing the inverses of f or g and the Jacobian of the transformation is a block diagonal matrix

$$\begin{bmatrix} I_d & 0 \\ J_y & \text{diag}(\exp(f(x_{1:d}))) \end{bmatrix}$$

The determinant of this matrix depends only on the diagonal blocks and is straightforwardly given by the product of the elements of $\text{diag}(\exp(f(x_{1:d})))$.

L2HMC parameterises the leapfrog updates with RNVP and learns a proposal for Metropolis-Hastings that is initialised as equivalent to HMC but quickly deviates and does not necessarily preserve probability density. By optimising the squared

distance between proposals they are able to learn a sampler that can mix much more quickly than HMC and can cross regions of low probability between modes that HMC struggles with. Although their method removes the need to tune the HMC hyper-parameters (step-size and number of leapfrog steps) it introduces a new layer of neural network hyper-parameters and considerable additional complexity, which has hindered its adoption as a general method.

A-NICE-MCMC

Earlier work by Song et al. [2017], also attempts to use neural networks to learn fast mixing proposal distributions for Metropolis-Hastings. Rather than start from HMC, they parameterise the mean of a Gaussian proposal kernel with a feed-forward network and train using an adversarial objective similar to GANs [Goodfellow, 2016]. In order to calculate the Metropolis-Hastings acceptance probabilities, it's necessary to be able to evaluate the probability of transition in both directions. Song et al. [2017] achieve this by using volume-preserving invertable neural networks first introduced in Dinh et al. [2015]. There are essentially equivalent to RNVP except that the function f in equation 2.24 is 0 everywhere.

To train their adversarial objective they require samples from the target distribution, which ofcourse they do not have as that is the goal of sampling. To overcome this challenge they use a bootstrap procedure starting from a randomly initialised proposal and iteratively training on better and better samples. Although their method, A-NICE-MCMC, can learn fast mixing proposals its reliance on adversarial training makes it unstable and its dependence on volume preserving proposals means it can also struggle to mix between modes with significantly differing volumes.

2.3 Variational Inference

Variational Inference (VI) is an alternative approach to approximating the intractable integrals that arise in probabilistic machine learning. In chapter 3 we produce an MCMC method that attempts to combine Variational Inference and sampling based approaches. Here we provide the minimal background on variational methods needed to understand the contributions of subsequent chapters. The core idea of VI is to

convert challenging integrals into optimisation problems by bounding the integral in question and then optimising the bound.

Consider the common task of estimating the marginal likelihood of some generative model with data $X = \{x_1 \dots x_N\}$, parameters θ and latent variables $Z = \{z_1 \dots z_N\}$. We may write the following bound on the marginal likelihood

$$\log p(X) \geq L = \int q(Z,\theta) \log \left[\frac{p(X,Z,\theta)}{q(\theta,Z)} \right] dZ d\theta \tag{2.25}$$

$$= E_q [\log p(X,Z,\theta)] + H(q) \tag{2.26}$$

$$= \log p(X) - KL[q(Z,\theta)|p(Z,\theta|X)] \tag{2.27}$$

where H is the entropy of the distribution and $KL[q|p]$ is the Kullback-Leibler divergence. The inequality on the first line follows from a straightforward application of Jensen's inequality and $q(\theta,Z)$ is a variational distribution that has been introduced to approximate the posterior. Typically, the bound is optimised with respect to q over some constrained family of distributions. Looking at equation 2.21, we see that the lower bound will be tight iff $q(\theta,Z) = p(Z,\theta|X)$ and so this optimisation encourages q to be close to the true posterior. Typically the true posterior will not be in the family of variational distributions considered and so variational inference, unlike MCMC, will be irretrievably biased. Although MCMC is asymptotically exact, it can be slow to mix whilst Variational Inference is approximate but fast and so remains an important family of methods.

By choosing different constraints for the family that q belongs to we may derive many common variational algorithms. If we choose the constraint that $q(z,\theta) = q(z)q(\theta)$ factorises then we arrive at Variational Bayes [Fox and Roberts, 2012, Beal, 2003]. If we further constrain $q(\theta) = \delta(\theta - \theta_0)$ to be a point mass and $q(z)$ remains unconstrained then optimising over $q(z)$ and θ_0 by coordinate ascent yields the generalised Expectation Maximization (EM) algorithm [Dempster et al., 1977]. Two

typical forms of constraint on the distribution q are either that it factors in some useful way (e.g in graphical models q may be assumed to be tree structured) or that q belongs to some parametric family (e.g Gaussian).

2.3.1 Stochastic Gradient Variational Bayes

Stochastic Gradient Variational Bayes (SGVB) [Kingma and Welling, 2014, Rezende et al., 2014] is a variational method that was introduced to perform approximate inference over latent variables in modern deep learning models where large data-set sizes make the per data-point calculations of traditional EM intractable. SGVB uses stochastic gradient methods to optimise a parameteric bound of the loglikelihood with respect to both the parameters of the model and the variational distributions simultaneously.

SGVB [Kingma and Welling, 2014, Rezende et al., 2014] relies on the combination of three key approximations that emerged independently:

1. **Mini-batch gradients**

   Optimisation of the variational bound, equation 2.25, would naively require gradient computations that scale linearly with the size of the data-set. Hoffman et al. [2013] popularised the idea of applying stochastic gradient estimates to variational inference. At each optimisation step a noisy estimate of the gradient is formed form a mini-batch of the data. The mini-batch size can be kept small even as the overall data-set size grows and so the algorithm can scale to data sets of millions or even billions of data points.

2. **Monte Carlo estimates of expectations**

   The variational bound shown in equation 2.25 contains expectations under the variational posterior that may themselves be intractable. Graves [2011] introduced the idea of a second layer of approximation in which these expectations are estimated with Monte-Carlo samples. The key insight being that sampling from the variational posterior is much more straightforward than sampling from the true posterior. Monte Carlo approximations of the

marginal likelihood itself are very difficult but the variational bound may be approximated straightforwardly.

3. **Inference networks**

   Inference networks replace the per data-point posterior calculations that are needed in the EM algorithm with a neural network that can takes as input a datapoint and outputs the parameters of the approximate posterior on the latent variables for that data-point. Rezende et al. [2014] and Kingma and Welling [2014] combined inference networks with stochastic variationl inference to simultaneously optimise the variational lower bound with respect to the parameters of the inference network and posterior simultaneously.

To see these approximations in practice it's helpful to consider the following deep generative model for a datapoint x

$$p(x) = \int p(x|f_\theta(z))p(z)dz \tag{2.28}$$

where $p(z)$ is a standard normal distribution and $p(x|f_\theta(z),\theta)$ is an exponential family distribution with parameters $f_\theta(z)$ given by passing z through a feed forward neural network. Sampling from this model is straightforward but the integral over z is intractable.

To train this model, SGVB uses the bound of the generalised EM algorithm but parameterises the variational distribution $q = q_\phi(z|x)$. In the traditional EM algorithm, the optimisation over $q(Z) = \prod_n q_n(z_n)$ is often performed by setting $q_n(z_n) = p(z_n|x_n,\theta)$ for each data point. This requires a calculation that scales linearly with data-set size and which requires that the posterior on each latent variable z_n be tractable. By parameterising $q = q_\phi(z|x)$ as an explicit function of x, the per data-point calculations can be avoided. This parameterisation where a parametric conditional distribution $q_\phi(z|x)$ rather than $q(z)$ is used, is often referred to as amortised inference Kingma and Welling [2014]. The per-data point bound becomes:

$$\log p(x) \geq L(\theta, \phi) = E_{q_\phi}[\log p(x|f_\theta(z))] - KL[q_\phi(z|x)|p(z)] \tag{2.29}$$

Typically $q_\phi(z|x)$ and $p(z)$ can be chosen so that the KL maybe calculated in closed form. The expectation in the first term is intractable but we can approximate gradients of this expectation with samples. The gradient of the expectation with respect to θ is unproblematic because after exchanging the order of integration and differentiation it remains an expectation we can sample from

$$\begin{aligned}
\nabla_\theta E_{q_\phi(z|x)}[\log p(x|f_\theta(z))] &= \nabla_\theta \int \log p(x|f_\theta(z)) q_\phi(z|x) dz && (2.30)\\
&= \int \nabla_\theta \log p(x|f_\theta(z)) q_\phi(z|x) dz && (2.31)\\
&= E_{q_\phi(z|x)}[\nabla_\theta \log p(x|f_\theta(z))] && (2.32)\\
&\approx \frac{1}{N} \sum_n \nabla_\theta \log p(x|f_\theta(z_n)) && (2.33)
\end{aligned}$$

with $z_n \sim q_\phi(z|x)$. However, the gradient with respect to ϕ is more complex because the distribution $q_\phi(z|x)$ depends on ϕ. After exchanging the order of integration and differentiation the integral is no longer an expectation and so we can't simply form a Monte Carlo estimate.

$$\nabla_\phi E_{q_\phi}[\log p(x|f_\theta(z))] = \int \log p(x|f_\theta(z)) \nabla_\phi q_\phi(z|x) dz \tag{2.34}$$

We can however re-write the integral as an expectation so that we may once again use a Monte Carlo approximation. Two common ways of re-writing the integral are using the score function estimator (also known as the log-derivative trick or REINFORCE gradient) [Sutton et al., 1999, Williams, 1992] or using re-parameterised gradients [Kingma and Welling, 2014, Rezende et al., 2014].

The score function estimator uses the identity $\nabla_x f(x) = f(x)\nabla_x \log f(x)$ to re-write the integral as an expectation

$$\nabla_\phi \int \log p(x|f_\theta(z)) q_\phi(z|x) dz = \int q_\phi(z|x) \left[\log p(x|f_\theta(z)) \nabla_\phi \log q_\phi(z|x)\right] dz \tag{2.35}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} \log p(x|f_\theta(z_m)) \nabla_\phi \log q_\phi(z_m|x) \tag{2.36}$$

where $z_m \sim q_\phi(z|x)$. This estimator is unbiased and can be used for both discrete and continuous distributions but has been observed in practice to have high-variance [Paisley et al., 2012] and is usually used with variance reduction techniques such as control variates [Tucker et al., 2017].

To re-parameterise the gradient we seek to find a parameterless distribution $p(\varepsilon)$ such that we can write z as a function of ε, i.e $z = g(\varepsilon, \phi)$. If we can find such an ε and g, then we use a simple change of variables to rewrite the integral as an expectation and once again form a Monte Carlo estimate

$$\nabla_\phi \int \log p(x|f_\theta(z)) q_\phi(z|x) dz = \int p(\varepsilon) \left[\nabla_\phi \log p(x|f_\theta(g(\phi, \varepsilon)))\right] d\varepsilon \tag{2.37}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} p(x|f_\theta(g(\phi, \varepsilon_m))) \tag{2.38}$$

where $\varepsilon_m \sim p(\varepsilon)$. For example in the case that $q(z|x) \sim N_z(\mu(x), \Sigma(x))$ we can set $g(\varepsilon, \phi) = \mu + \Sigma^{\frac{1}{2}}\varepsilon$ for $\varepsilon \sim N(0, I)$.

It has been observed empirically that re-parameterised gradients typically produce lower variance estimates when compared to the score function estimator but this is still an active area of research and there are known cases where this pattern is reversed [Gal, 2016].

In chapter 3, we make use of SGVB to train variational approximation as part of a proposal distribution in MCMC. Variational approximations are typically fast to compute but irretrievably biased, by incorporating them into an MCMC proposal we attempt to produce a sampler that both mixes quickly and is asymptotically exact.

2.4 Deep Generative Models for Speech

In chapter 4 we introduce a method for controllable text-to-speech synthesis (TTS) that uses deep latent variable models trained by SGVB. The core contribution stems from the introduction of probabilistic latent variables to existing speech synthesis methods. Here we present a succinct summary of the relevant background on neural speech synthesis.

The task of synthesising human speech from corresponding text has been a goal of the research community for many decades [Taylor, 2009]. Across that time there have been differing paradigms of techniques. For many years the state-of-the art was concatenative synthesis with unit selection, in which small prerecorded snippets are stitched together [Wang et al., 2017]. Subsequently statistical parametric methods based on Hidden Markov Models (HMMs) dominated [Taylor, 2009]. These models predicted smooth trajectories of linguistic features which were then fed to a separate vocoder to generate wave forms. Parametric speech synthesis typically consists of a complex pipeline of independently trained parts: a linguistic front-end, an acoustic model, a duration model and finally a signal processing based vocoder [Taylor, 2009]. Each of these components typically requires extensive domain expertise and since they are trained independently, errors may accumulate across components. Starting with Wavenet [van den Oord et al., 2016], and subsequently Char2Wav [Sotelo et al., 2017] and Tacotron [Wang et al., 2017], neural network based models trained end-to-end directly on text-waveform pairs began to dominate TTS and today are the state-of-the-art.

Neural speech synthesis typically frames TTS as a sequence-to-sequence mapping problem in which we must train a model to map from a sequence of characters to a

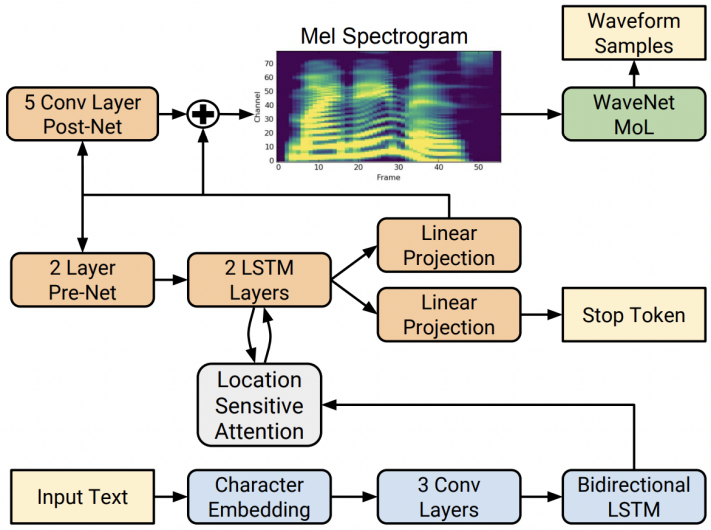


Figure 2.1: Schematic showing the basic structure of the Tacotron 2 model introduced by Shen et al. [2018]. Reproduced from the Original paper.

sequence of waveform amplitudes, given a training set of text and waveform pairs. Early neural TTS methods took inspiration from machine translation [Bahdanau et al., 2014], where recurrent neural networks with attention were successfully used to map from sequences of words in one language to sequences of words in another. The sequence-to-sequence mapping in TTS has significantly differing challenges though. In machine translation the two sequences are typically similar in length but in TTS the waveform must be sampled at high frequency [3] and so the output sequence is usually much longer than the input sequence. The longer sequences also require modelling very long range time dependencies which poses a challenge for training recurrent neural networks where vanishing or exploding gradients become more likely with increased sequence length. To overcome these challenges a range of innovations were introduced both to the neural network architectures and the training methods. Many of the innovations are empirically derived through trial and error over time.

An example of a typical neural TTS model is given by Tacotron 2 [Shen et al., 2018] whose structure is shown schematically in figure 2.1 and given in full detail in

[3] To faithfully recover a continuous signal Nyquist's theorem tells us that we must sample at a rate that is at least twice the highest frequency fourier component found in the signal.

chapter 4. The Tacotron model starts either from characters or phonemes (symbols representing a vocabulary of sounds). These characters are mapped to dense vectors which are then passed through an encoder neural network to produce a sequence of vectors the same length as the initial sequence. The encoder network in Tacotron 2 consists of 1-dimensional convolution layers [LeCun et al., 1995] followed by a bidirectional LSTM network [Hochreiter and Schmidhuber, 1997]. Conditioned on this sequence of vectors, the network then auto-regressivly predicts frames of a paired mel-frequency spectrogram.

A mel-frequency spectrogram is a non-linear transformation of the standard short-time-Fourier-transform magnitude [Taylor, 2009]. The transformation used is a heuristic that was crafted to over emphasise lower frequencies which are critical to humans perceiving the speech accurately and de-emphasise higher frequencies which are typically dominated by fricatives and noise bursts which do not need to be as faithfully modelled. The choice of mel-spectrogram as an intermediate model target was one of the key innovations in getting neural TTS to work [Wang et al., 2017]. It allows for a shorter target sequence which is invariant to phase within spectrogram frames and incorporates the prior knowledge that the human auditory system is more sensitive to lower frequencies [Taylor, 2009].

The length of the spectrogram differs significantly from the input sequence and so a learned "attention mechanism" [Bahdanau et al., 2014] is used to align these two sequences. At each time step of generation, the attention mechanism outputs a weighted sum of the input sequence vectors. The weights are allowed to depend on the spectrogram generated so far as well as the input sequence. Finally, the frames of the mel-spectrogram are fed through a separate vocoder network to produce wave forms. The model is trained by minimising the absolute difference between the synthesised mel-spectrogram frames and the true mel-spectrogram using stochastic gradient descent. A classification loss is used to simultaneously train the neural vocoder.

Tacotron is one of many neural TTS models that have been proposed in recent years

[Shen et al., 2018, Arik et al., 2017, Gibiansky et al., 2017, Ping et al., 2017, Vasquez and Lewis, 2019, Taigman et al., 2017] and forms the basis of the architecture we use in our experiments. Differentiating factors between these models include the degree of parallelism, with some models using Transformer based architectures [Ren et al., 2019], the choice of conditional independence assumptions made [Vasquez and Lewis, 2019] and the number of separately trained components [Gibiansky et al., 2017].

In our work we take a probabilistic view of the Tacotron model, treating the absolute difference loss as the conditional log-likelihood of an isotropic Laplace distribution with constant variance and viewing the optimisation as maximum likelihood learning. We introduce stochastic latent variables which make it possible generate multiple samples with varying prosody (duration, intonation and intensity) for a given text input and train using stochastic gradient variational Bayes.

Chapter 3

Auxiliary Variational Markov Chain Monte Carlo

This chapter is based on the paper 'Auxiliary Variational MCMC' (Habib and Barber, 2019), which was presented at the International Conference of Learning Representations (ICLR). As we discussed in the previous chapter, Markov Chain Monte Carlo (MCMC) and Variational Inference (VI) are well-established approaches to approximating expectations under the complex distributions p that frequently arise in machine learning and statistics [Wainwright and Jordan, 2008, Brooks et al., 2011]. VI is usually fast and cheap but often places strong restrictions on the class of approximating distributions leading to irreducible bias. MCMC on the other hand is asymptotically exact but may require an infeasible amount of computation to converge. Given the complimentary strengths and weaknesses of the two methods it is natural to wish to combine them [De Freitas et al., 2001, Salimans et al., 2015].

The most naive pairing of the two methods, simply using a variational approximation q as the proposal distribution in a Metropolis-Hastings sampler (see for example Gamerman and Lopes [2006] or De Freitas et al. [2001]), is known to scale poorly with the dimension of the target distribution [De Freitas et al., 2001] and mix inefficiently when p is multi-modal.

In this chapter we suggest an alternative approach to combining MCMC and Varia-

tional inference, inspired both by the successes of black box variational inference [Ranganath et al., 2014] and auxiliary variable MCMC methods, such as Hamiltonian Monte Carlo (HMC) [Duane et al., 1987, Girolami and Calderhead, 2011].

The key contributions of our work are :

- A general framework for marrying variational inference with Markov Chain Monte Carlo in a way likely to produce efficient samplers.
- The use of the auxiliary variational method to capture latent low-dimensional structure in our target distributions and exploit this structure to suppress random walk behaviour.
- The extension of the Metropolis-Hastings algorithm to continuous mixture proposals.
- The introduction and demonstration of a specific instance of our framework, the Auxiliary Variational Sampler (AVS). Our sampler takes advantage of flexible distributions parameterized by neural networks that can be trained in a fully black-box manner.

3.1 Auxiliary variational MCMC

The key idea behind Auxiliary Variational MCMC is to exploit structure present in the target distribution $p(x)$ by first fitting a parameterized variational approximation in an augmented space. This allows the sampler to leverage learned low-dimensional structure. In the subsequent sections we introduce the auxiliary variational method and describe how it can be combined with a carefully chosen class of proposal distributions to construct an efficient sampler.

3.1.1 Mixture proposal MCMC

To develop a valid MCMC algorithm we need to construct an ergodic Markov chain whose stationary distribution is our target distribution $p(x)$. In order to do this we introduce a Metropolis-Hastings [Gamerman and Lopes, 2006] like algorithm with

a specially chosen form of proposal distribution. This proposal can be naturally combined with the auxiliary variational method which we will introduce in section 3.1.2. We first consider a mixture proposal distribution of the following form[1]

$$\tilde{q}(x'|x) = \int \tilde{q}(x'|a)\tilde{q}(a|x)da \tag{3.1}$$

We prove below that the following forms a valid MCMC sampling step:

1. Sample a from $\tilde{q}(a|x)$
2. Sample x' from $\tilde{q}(x'|a)$
3. Accept the candidate sample x' with probability

$$\min\left\{1, \frac{\tilde{q}(x|a)\tilde{q}(a|x')p(x')}{\tilde{q}(x'|a)\tilde{q}(a|x)p(x)}\right\} \tag{3.2}$$

   otherwise reject x' and define the new sample as a copy of the current x, namely $x' = x$.

It is worth noting that the above procedure is not equivalent to simply performing Metropolis-Hastings in the joint (x, a) space, but is a sampler in x alone, specified by the marginalized proposal distribution given by (3.1).

The mixture proposal can be extended to an arbitrary number of auxiliary variables as long as the acceptance ratio is adjusted accordingly. For example with two auxiliary variables, the proposal becomes

$$\tilde{q}(x'|x) = \int \tilde{q}(x'|a')\tilde{q}(a'|a)\tilde{q}(a|x)dada' \tag{3.3}$$

where $\tilde{q}(a'|a)$ is another arbitrary proposal distribution. The acceptance probability is now given by

$$\min\left\{1, \frac{p(x')\tilde{q}(x|a')\tilde{q}(a'|a)\tilde{q}(a|x')}{p(x)\tilde{q}(x'|a)\tilde{q}(a|a')\tilde{q}(a'|x)}\right\} \tag{3.4}$$

[1] In general we use $\tilde{q}$ to denote a proposal distribution and q to denote a variational distribution.

For a proposal $\tilde{q}(a'|a)$ that is symmetric, $\tilde{q}(a'|a) = \tilde{q}(a|a')$, this simplifies to

$$\min\left\{1, \frac{p(x')\tilde{q}(x|a')\tilde{q}(a|x')}{p(x)\tilde{q}(x'|a)\tilde{q}(a'|x)}\right\} \tag{3.5}$$

We now show how the mixture proposal can be naturally combined with the auxiliary variational method.

3.1.2 The auxiliary variational method

The auxiliary variational method [Agakov and Barber, 2004, Ranganath et al., 2016]is a strategy for creating more expressive families of approximating distributions for use in variational inference. Instead of minimizing the Kullback-Leibler divergence between our approximating distribution $q(x)$ and our target $p(x)$, we instead minimize the divergence in an augmented space (x,a) with additional auxiliary variables a. In doing this, we first define a joint $q_\phi(x,a)$ in the augmented space and a joint $p(x,a) = p_\theta(a|x)p(x)$ where θ and ϕ are parameters. Marginalizing $p(x,a)$ over a recovers $p(x)$ by construction. We are free to decide on the dimension and form (e.g. continuous, discrete or mixed) for a and our objective is the Kullback-Leibler divergence between the joint approximation q and joint target p

$$(\phi^*, \theta^*) = \arg\min_{\phi,\theta} \mathrm{KL}\left(q_\phi(x,a)||p(x)p_\theta(a|x)\right) \tag{3.6}$$

Moving to the joint space allows us to tractably learn complex approximating distributions $q_\phi(x,a)$ whose marginal, $q_\phi(x) = \int q_\phi(x|a)q_\phi(a)da$, may be intractable.

Whilst sampling in high-dimensional spaces is difficult, in many cases of interest the high probability or typical regions may lie close to much lower dimensional manifolds. A key point of fitting an auxiliary variational distribution is that, by constraining a to have dimension much lower than x, we can then exploit this low dimensional structure to form a more efficient proposal distribution, staying close to the manifold of significant probability.

Algorithm 3: The Auxiliary Variational Sampler. The algorithm begins by fitting a mixture variational distribution to the target $p(x)$ by stochastic gradient descent, based on the auxiliary variational method.

Initialize ϕ and θ ▷ Fit the variational distribution
while *Not Converged* **do**
  $\{a_1,\ldots,a_N\} \sim q_\phi(a)$
  $\{x_1,\ldots,x_N\} \sim q_\phi(x|a_n)$ ▷ re-parameterized
  $L \leftarrow \frac{1}{N}\sum_n \log \frac{q_\phi(x_n|a_n)q_\phi(a_n)}{p(x_n)p_\theta(a_n|x_n)}$
  $\phi \leftarrow \phi - \eta\nabla_\phi L$
  $\theta \leftarrow \theta - \eta\nabla_\theta L$
end
$a \sim q_\phi(a)$ ▷ Mixture-Model MCMC sampling
$x_0 \sim q_\phi(x|a)$
for $t = 0,\ldots,T$ **do**
  $a \sim p_\theta(a|x_t)$
  $a' \sim N\left(a'|a,\sigma_a^2 I\right)$
  $x' \sim q_\phi(x|a')$
  $A \leftarrow \frac{p(x')p_\theta(a'|x')q_\phi(x_t|a)}{p(x_t)p_\theta(a|x_t)q_\phi(x'|a')}$ ▷ acceptance ratio
  $u \sim U[0,1]$ ▷ sample u from uniform distribution
  if $u \leq A$ **then**
    $x_{t+1} = x'$
  else
    $x_{t+1} = x_t$
  end
end

3.1.3 Combining auxiliary variational inference and MCMC

After fitting our variational approximation to $p(x)$, we will have three variational distributions: $q_*(x|a)$, $q_*(a)$ and $p_*(a|x)$ that are the solution of the optimization problem given in (3.6). These distributions approximately satisfy the following relationships

$$\int p(x)p_*(a|x)dx \approx q_*(a), \qquad \int q_*(x|a)q_*(a)da \approx p(x) \tag{3.7}$$

which become exact iff the divergence in (3.6) becomes zero. Here $p_*(a|x)$ is a learned stochastic mapping from the high-dimensional target space to the low-dimensional auxiliary space and $q_*(x|a)$ is a mapping in the opposite direction. These learned mappings can be composed in a variety of ways to form marginal proposal distributions of the kind discussed in section 3.1.1 and also joint proposals. We now

discuss some natural combinations and argue that these may form good proposal distributions in practice.

3.1.3.1 Naive proposal distributions

A simple proposal constructed from our variational distribution is to perform Metropolis-Hastings in the joint (x,a) space with an independent proposal given by:

$$\tilde{q}(x',a'|x,a) = q_*(x'|a')q_*(a') \tag{3.8}$$

If the variational approximation is accurate such that $q(x,a) \approx p(x)p(a|x)$ then one might expect this to have high acceptance probability. A potential downside of this scheme is that proposals are independent between time-steps and in high dimensions this may cause the acceptance ratio to become impractically low.

Another natural MCMC proposal distribution to consider is

$$\tilde{q}(x'|x) = \int q_*(x'|a)p_*(a|x)da \tag{3.9}$$

where we have replaced the arbitrary proposal distribution of (3.1) with the optimal variational distributions learned by minimizing the joint divergence given in (3.6). We might expect this to be a promising proposal distribution, with high acceptance probability, as it already approximately satisfies the stationarity criterion of our MCMC chain. That is

$$\begin{aligned}\int \tilde{q}(x'|x)p(x)dx &= \int q_*(x'|a)p_*(a|x)p(x)dxda \approx \int q_*(x'|a)q_*(x|a)q_*(a)dxda \\ &= \int q_*(x'|a)q_*(a)da \approx \int p_*(a|x')p_*(a)da = p(x')\end{aligned}$$

However, if the variational distribution truly captures underlying structure in $p(x)$, it has been our experience that $p_*(a|x)$ and $q_*(x|a)$ become approximate inverses causing $\tilde{q}(x'|x)$ to resemble the identity mapping and thereby slowing mixing in the chain.

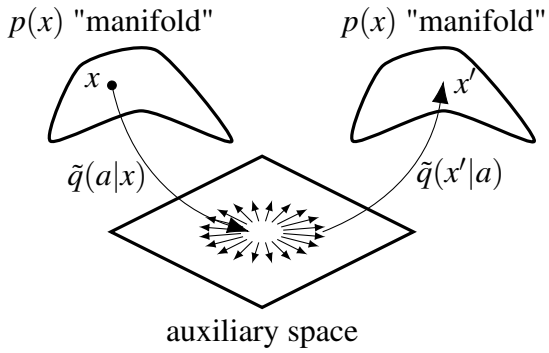


Figure 3.1: The auxiliary random walk proposal. The variational approximation allows for small steps in the auxiliary space to correspond to large steps in the target space. Our initial point starts on a manifold of high probability in the target space, is mapped down to the low-dimension auxiliary space, perturbed, and then mapped back up to the high probability manifold. Unlike a random-walk in the x-space, our random perturbations correspond to moves along the high probability manifold.

3.1.3.2 An auxiliary random walk proposal distribution

To avoid the above identity mapping issue and encourage the sampler to take large steps, we introduce an additional random perturbation in the auxiliary a-space. That is we use a proposal distribution

$$\tilde{q}(x'|x) = \int q_*(x'|a')\tilde{q}(a'|a)p_*(a|x)dada' \tag{3.10}$$

where $\tilde{q}(a'|a) = N\left(a'|a, \sigma_a^2 I\right)$ is a Gaussian proposal with mean a and isotropic covariance $\sigma_a^2 I$. The overall algorithm can be viewed as mapping from the high-dimensional x to the low-dimensional a, performing a random walk in the low dimensional auxiliary a-space and subsequently mapping back up to the high-dimensional target space x, see figure (3.1).

In high dimensions our target distribution is likely to have high probability only close to some low-dimensional manifold. The traditional Random-Walk-Metropolis Algorithm proposes new samples x' by perturbing the most recent sample in an arbitrary direction, usually based on a Gaussian proposal $\tilde{q}(x'|x) = N\left(x'|x, \sigma_x^2 I\right)$; however, in high-dimensional spaces almost all directions will correspond to steps off the manifold of high probability and thus out of the typical set. In our case, we

perform the random perturbation in the low-dimensional auxiliary space and our variational distribution $q_*(x'|a')$ ensures that this move remains within the manifold of high probability.

We depict this process in figure (3.1) and provide explicit examples of the learned latent structure of real distributions in section 3.2. Much like HMC, we are able to exploit the addition of auxiliary variables to encourage large moves of high probability in the target space but unlike HMC we do so by explicitly modelling the structure of the target distribution and do not require gradients of the target density when sampling. We are able to take large steps because small perturbations in a can correspond to large perturbations within the manifold of high probability; unlike methods that adapt to the local geometry of the probability manifold [Girolami and Calderhead, 2011, Strathmann et al., 2015], our variational fit allows for larger, non-local moves in the x-space. Note that, unlike HMC, in general there is no requirement that (x,a) are continuous random variables.

3.1.4 Choosing the variational family

Within the above framework, we still have to decide on the structure of the variational distributions, $q(a,x)$ and $p(a|x)$. We take inspiration from recent successes in the generative modeling of complex data distributions [Kingma and Welling, 2014], and propose to parameterize our variational distributions using deep neural networks. For continuous x we choose the following structure for each of our approximating distributions

$$q(a) = N(a|0,I) \tag{3.11}$$

$$q_\phi(x|a) = N\left(x|\mu_\phi(a),\Sigma_\phi(a)\right) \tag{3.12}$$

$$p_\theta(a|x) = N(a|\mu_\theta(x),\Sigma_\theta(x)) \tag{3.13}$$

where $q_\phi(x|a)$ and $p_\theta(a|x)$ are both diagonal Gaussian distributions whose means and covariances are parameterized by neural networks with parameters shared between the mean and covariance. In some experiments we also choose $p_\theta(a|x)$ to be a

(a) (b)

Figure 3.2: Target densities with a high degree of latent structure. (a) Mixture of Two Gaussians with highly separated means. (b) A ring of high density centered at the origin.

mixture of diagonal Gaussians (see the appendix). Key to the flexibility of the auxiliary variational method is that whilst $q(a,x)$ can be evaluated point-wise, the marginal $q(x)$ is a much richer approximating distribution whose density we typically cannot evaluate point-wise. Whilst we can thus evaluate our joint approximating density we still can't compute the objective

$$\mathrm{KL}\left(q_\phi(x|a)q(a)||p_\theta(a|x)p(x)\right) \tag{3.14}$$

However, we recognize that an unbiased estimator of the KL divergence can be obtained by sampling from $q_\phi(x,a)$. We use the standard re-parameterization trick of Kingma and Welling [2014] to reduce the variance in the corresponding gradient estimator. We refer to this version of the algorithm as the Auxiliary Variational Sampler (AVS). The full algorithm is given in algorithm (3). Within this framework, different mixture proposals could be considered, based on the fitted variational distribution, but our experience is that the Auxiliary Random Walk proposal is effective in our experiments.

3.2 Experiments

To demonstrate the benefit of fitting an auxiliary variational approximation we first test our sampler on a number of distributions with known low-dimensional structure. We show how AVS is both able to recover the latent structure and exploit it for the purposes of sampling. We then demonstrate the sampler on more realistic problems.

Ring density The first distribution is in two dimensions $x = (x_1, x_2)$. The distribution is a ring of high probability, centered at a fixed distance from the origin, figure (3.2b).

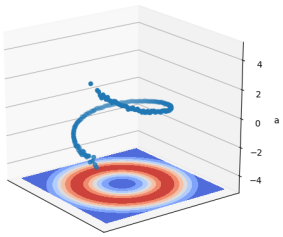
We construct the ring density to have explicit latent structure by defining the target distribution as

$$p(x) \propto \int_0^{2\pi} e^{-\frac{1}{2\sigma^2}(x-r\mu(\theta))^2} d\theta, \qquad \mu(\theta) = (\cos(\theta), \sin(\theta)) \tag{3.15}$$

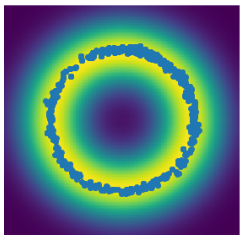
Since this ring distribution is highly constrained, pure random walk sampling in x will not efficiently move around the ring. However, provided we can find a variational fit to capture the ring structure, we expect our AVS sampler to be able to move more efficiently. To train our AVS sampler, we fit an auxiliary variational approximation with a 1 dimensional continuous mixture (see the appendix for details). We can examine the latent structure learned by the variational distribution by drawing samples from $p_*(a|x)$ as we vary $x = (\cos(\theta), \sin(\theta))$ for $\theta \in [0, 2\pi]$. Looking at figure (3.3), we can see that the variational distribution has successfully captured the latent structure, automatically learning to map points of high probability in x onto distinct auxiliary variables. Used with the mixture proposal (3.10) this results in an effective sampler that is able to take much larger step sizes than random walk Metropolis (RWM) as shown in figure (3.3).

Mixtures of Gaussians and Student T The second pair of densities we wish to sample from is a two dimensional mixture of Gaussians with highly separated means (with a distance of 20 between the means and a standard deviation of 1), figure (3.2), and a two dimensional mixture of Student T distributions. These distributions have discrete latent structure and are challenging both for random walk Metropolis (RWM) [Gamerman and Lopes, 2006] and more advanced samplers such as HMC, which fail to find more than one mode in any reasonable amount of time or can struggle with the heavy tails. Our sampler has no problem finding both modes and, as shown by the plot of consecutive samples in figure (3.4), is able to hop between modes in single steps. We use a 1-dimensional continuous auxiliary variable.

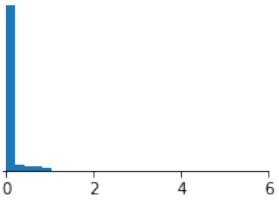
Posterior sampling in Bayesian models The low-dimensional distributions above demonstrate the ability of our sampler to learn and exploit latent structure. Here we analyze the performance of our sampler on the more realistic problem of posterior



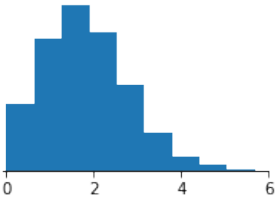
(a) Samples from $p_*(a|x)$



(b) MCMC samples from $p(x)$



(c) Histogram of distance moved by RWM



(d) Histogram of distance moved by AVS

Figure 3.3: (a) Samples drawn from the learned variational distribution $p_\theta(a|x)$ for x on the ring of high probability. The auxiliary space is plotted on the vertical axis, with the target space plotted in the plane. We can see that the auxiliary variational approximation has recovered the low dimensional structure of the target distribution, so that random perturbations in the auxiliary space will mostly correspond to proposals within the region of high target probability and large moves. (b): Samples from the trained Auxiliary Variational Sampler (AVS) algorithm (3). (c,d) show the distance moved per sampling step for RWM and AVS for a ring of radius 5, thus demonstrating the benefit of exploiting structure.

sampling in Bayesian logistic regression. We use the heart data-set used by Song et al. [2017], which has 13 covariates and 270 data-points. We tune all algorithms to maximize effective sample size (see supplement (3.5.3)) at convergence. In the case of HMC we use a number of initial runs to select an appropriate step-size and then tune the number of leapfrog steps. In all cases, we run a chain for 10000 burn-in steps and then draw 20000 samples to calculate diagnostics. In order to assess convergence and ensure the validity of the results, we monitor the Gelman-Rubin statistic [Gelman and Rubin, 1992] and visually inspect trace-plots of parameters.

3.2.1 Evaluating performance

Evaluation of MCMC methods is notoriously difficult [Gelman and Rubin, 1992]. Even diagnosing convergence is challenging and there is no clear agreement on the

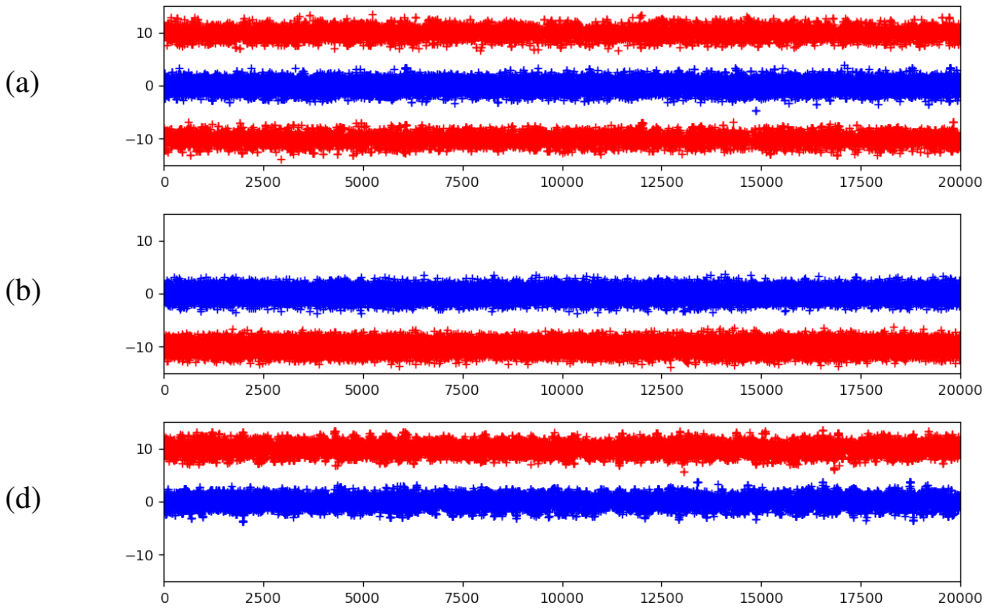


Figure 3.4: Traceplot showing 20000 consecutive samples of x_1 (blue), x_2 (red), drawn from the mixture of Gaussians, figure 3.2, using (a) AVS given in algorithm 3, (b) HMC and (c) RWM. AVS can easily move between the modes at $x_1 = 10$ and $x_1 = -10$, whilst RWM and HMC get stuck in one mode.

"correct" metric of performance. In practice there are two factors that are of primary concern to an end-user: How easy is the sampler to implement and tune? and how computationally efficient is the resulting sampler? We evaluate our performance by comparison to RWM, HMC and two recently proposed neural samplers: A-NICE-MC [Song et al., 2017] and L2HMC [Levy et al., 2018], discussed further in section 3.3. For learned samplers there is an additional cost involved in training the sampler that may be offset by improved performance.

We choose here to focus our quantitative evaluation on the effective Sample Size (ESS), which measures how many independent samples would be equivalent to an MCMC sample, and the training-time for the learned samplers. We choose not to investigate burn-in since AVS is initialized by the variational distribution and therefore has short burn-in time compared to other methods. For ESS calculations, we use the batch-means estimator applied to a single long chain as it has been shown to be more reliable than many alternatives [Thompson, 2010], exact details are given in the appendix. For the experiments with mixtures of Gaussians we do

not present ESS for HMC and RWM as both algorithms clearly fail to converge, getting trapped in one of the local modes. To ensure consistency in timing, all methods were implemented using Tensorflow 1.10 [Abadi and Agarwal, 2015] and run on a single Tesla K80 GPU. Code to reproduce all experiments can be found at github.com/AVMCMC. For L2HMC and A-NICE-MCMC, we perform a random search over hyper-parameters ensuring we include the parameters used in the original papers wherever possible. The figures reported are the mean of 10 independent runs alongside their standard deviations.

Table 3.1: ESS calculated using the batch-means estimator.

	AVS	AVS-IND	HMC	RWM	ANICE	L2HMC
Ring	0.176 ± 0.005	NA	$\mathbf{0.612 \pm 0.120}$	0.024 ± 0.005	0.541 ± 0.105	0.247 ± 0.062
Mix. of Gauss.	0.178 ± 0.042	0.009 ± 0.004	NA	NA	$\mathbf{0.322 \pm 0.103}$	0.170 ± 0.10
Mix. of Student T.	0.047 ± 0.026	NA	0.0020 ± 0.0002	0.002 ± 0.001	0.071 ± 0.026	NA
Log. regression	0.066 ± 0.027	NA	0.070 ± 0.015	0.013 ± 0.002	0.257 ± 0.035	$\mathbf{0.562 \pm 0.071}$

Table 3.2: ESS/s taking into account the total time including both training time and sampling time for 20000 samples (left) and taking account only the sampling time (right). Absolute times are provided in the appendix.

	AVS	ANICE	L2HMC
Ring	**5.30e-3**	1.25e-3	7.54e-5
Mix. of Gauss.	**7.64e-2**	2.50e-4	2.88e-5
Mix. of Student T	**1.45e-2**	2.17e-4	NA
Log. regression	**6.55e-2**	4.30e-4	6.24e-5

	AVS	ANICE	L2HMC	HMC	RWM
Ring	35.42	**807.42**	6.34	624.97	688.32
Mix. of Gauss.	66.08	**288.10**	3.24	NA	NA
Mix. of Student T	74.19	45.42	NA	1.16	6.12
Log. regression	48.12	**2660.35**	87.11	180.43	1858.15

3.3 Discussion and Related Work

The results demonstrate that Auxiliary MCMC can be used to construct a practical sampler that is able to exploit low dimensional structure and mix between modes. AVS produces competitive ESS without requiring gradients of the target distribution. Initializing from a variational approximation reduces the need to burn-in and offers a straightforward method to interpolate between pure VI and MCMC. We found that the naive baseline (AVS-IND), independent Metropolis-Hastings in the joint auxiliary-target space, failed to produce reasonable samples for all but the Mixture of Gaussians. Though the more recent neural samplers have higher effective sample sizes, this comes at a high cost in training time that, in our moderately sized problems, overwhelms their benefit. It was also our experience that training L2HMC is very sensitive to correct tuning of many hyper-parameters. In a search over 42 training

configurations only 2 configurations converged to samplers able to mix between modes in the Mixture of Gaussian experiments. The difficulty of tuning is hard to quantify and will no doubt vary with the experience of the practitioner. We nonetheless see this as a possible barrier to the adoption of more complex adaptive methods and a possible explanation for the popularity of RWM. We found qualitatively that AVS and A-NICE-MC were more straightforward to tune with good performance from many hyper-parameter configurations.

The work most similar to ours is Variational MCMC [De Freitas et al., 2001], in which a variational approximation is used as a proposal distribution in independent Metropolis-Hastings. To overcome the poor scaling with dimension of the independent Metropolis algorithm, the authors interleave their variational proposal distribution with traditional RWM and make block proposals. Although their use of a variational distribution aids the convergence of MCMC, their method still struggles to mix efficiently even in relatively low dimensional problems. By introducing low dimensional auxiliary variables, we are able both to fit a more accurate approximating distribution and to leverage the learned low dimensional structure to take very large steps in the target space, even hopping between modes. In Salimans et al. [2015] the authors also consider a combination of MCMC and auxiliary variational inference but take a different approach, choosing to use MCMC kernels to design more flexible variational distributions rather than constructing an MCMC sampler at all.

There have also been other recent attempts to parameterize MCMC kernels with neural networks that don't start from variational inference. A-NICE-MCMC [Song et al., 2017], attempts to learn MCMC transition kernels parameterized by volume preserving flows [Dinh et al., 2015]. They craft an adversarial objective to train their kernels and use a discriminator that examines pairs of samples, in order to encourage fast mixing. To ensure they have a valid sampler, they use a bootstrap procedure starting from an existing MCMC algorithm. L2HMC [Levy et al., 2018] is an extension of HMC, that parameterizes a scaling and shift of Hamiltonian dynamics

with neural networks and trains this parameterization to optimize the expected squared step-size. They show that the introduction of flexible neural networks to HMC vastly improves the ability to mix between modes but still require access to the gradient of the target density.

As with any learned MCMC method, great care has to be taken to preserve the ergodicity of the chain during adaptation [Andrieu and Thoms, 2008]. To overcome this, all of the above methods, including ours, stop adaptation before collecting any samples. A potential problem with this strategy is that if the learned sampler is specialized to only a part of the distribution or the variational approximation has entirely missed regions of high probability, then such regions are unlikely to be explored. A possible avenue for future work would be to investigate iterative improvement of our variational approximation with samples drawn from the true distribution rather than our variational approximation. Another potential avenue for ensuring good coverage of the variational approximation, would be to combine our reverse KL objective, (3.14), with the KL divergence in the forward direction as this is known to encourage moment matching rather than mode seeking behavior [Minka, 2005].

3.4 Conclusion

We introduced a novel framework for combining MCMC and Variational Inference that makes use of the auxiliary variational method to capture low-dimensional latent structure. We have explored a particular black-box instance of this framework and demonstrated that it can be used to create a fast mixing sampler without the need to take gradients of the target distribution. The method is competitive with other recent geometry-learning approaches and opens up additional avenues for exploring how to combine the best of variational inference and sampling.

3.5 Appendix

3.5.1 Metropolis-Hastings with a Mixture proposal

We define a mixture proposal using an auxiliary variable a as

$$\tilde{q}(x'|x) = \int \tilde{q}(x'|a)\tilde{q}(a|x)da \tag{3.16}$$

We consider the transition kernel

$$q(x',a|x) = \tilde{q}(x',a|x)f(x',a,x) + \delta\left(x',x\right)\tilde{q}(a|x)\left(1 - \int \tilde{q}(x'',a'|x)f(x'',a',x)dx''da'\right) \tag{3.17}$$

It is trivial to check that this defines a valid distribution $q(x',a|x)$. We wish to set $f(x',a,x)$ such that $p(x)$ is a stationary distribution of $q(x'|x)$. That is

$$p(x') = \int q(x',a|x)p(x)dxda$$

The right hand side of the above equation can be written as

$$\begin{aligned}&\int \tilde{q}(x',a|x)f(x',a,x)p(x)dxda \\ &\quad + \int \delta\left(x',x\right)\tilde{q}(a|x)\left(1 - \int \tilde{q}(x'',a'|x)f(x'',a',x)dx''da'\right)p(x)dxda\end{aligned} \tag{3.18}$$

which simplifies to

$$\int \tilde{q}(x',a|x)f(x',a,x)p(x)dxda \;+\; p(x')\left(1 - \int \tilde{q}(x'',a'|x')f(x'',a',x')dx''da'\right) \tag{3.19}$$

For the above to hold we require (changing the integration variable x'' to x and a' to a)

$$\int \tilde{q}(x',a|x)f(x',a,x)p(x)dxda = \int \tilde{q}(x,a|x')f(x,a,x')p(x')dxda \tag{3.20}$$

Writing

$$\tilde{q}(x',a|x) = \tilde{q}(x'|a)\tilde{q}(a|x) \tag{3.21}$$

and considering the function

$$f(x',a,x) = \min\left(1, \frac{\tilde{q}(x|a)\tilde{q}(a|x')p(x')}{\tilde{q}(x'|a)\tilde{q}(a|x)p(x)}\right) \tag{3.22}$$

one can readily verify that

$$f(x',a,x)\tilde{q}(x'|a)\tilde{q}(a|x)p(x) = f(x,a,x')\tilde{q}(x|a)\tilde{q}(a|x')p(x') \tag{3.23}$$

meaning that (3.20) is satisfied.

We can then sample from $q(x',a|x)$ by first sampling a from $\tilde{q}(a|x)$ and then sampling from $\tilde{q}(x'|a)$ and accepting with probability $f(x',a,x)$. That is :

1. Sample a from $\tilde{q}(a|x)$

2. Sample x' from $\tilde{q}(x'|a)$

3. Accept the candidate sample x' with probability

$$\min\left(1, \frac{\tilde{q}(x|a)\tilde{q}(a|x')p(x')}{\tilde{q}(x'|a)\tilde{q}(a|x)p(x)}\right) \tag{3.24}$$

   otherwise reject x' and define the new sample as a copy of the current x, namely $x' = x$.

The extension to the case of more than 1-auxiliary variable follows naturally using the same argument as above.

If we were to perform Metropolis-Hastings in the joint space of (x,a) then the acceptance probability would be given by:

$$\min\left(1, \frac{p(a'|x')p(x')\tilde{q}(a,x|a',x')}{p(a|x)p(x)\tilde{q}(x',a'|a,x)}\right)$$

which can be seen to be different from (3.24).

3.5.2 Exact Parameterization of the variational distributions

Low dimensional examples

For the mixtures of Gaussians and ring density experiments we used an auxiliary dimension of 1 and the target dimension was 2. For the mixture of Gaussians, the variational distributions had the following form:

$$q(a) = N(a|0,I) \tag{3.25}$$

$$q_\phi(x|a) = N\left(x|\mu_\phi(a), \Sigma_\phi(a)\right) \tag{3.26}$$

$$p_\theta(a|x) = N(a|\mu_\theta(x), \Sigma_\theta(x)) \tag{3.27}$$

Where the $\mu_\phi(x)$ and $\Sigma_\phi(a)$ are 3 layer feed-forward neural networks with 10 neurons in each layer, all but the last layer were shared between them. Similarly $\mu_\theta(x)$ and $\Sigma_\theta(x)$ were also 3 layer feed-forward neural networks with all but the last layer shared. We used tanh non-linearities in all but the final layer which was simply linear.

In the case of the ring density experiments $p_\theta(a|x) = \Sigma_k \pi_k N\left(a|\mu_\theta^k(x), \Sigma_\theta^k(x)\right)$ was a mixture of 2 Gaussians, each parameterized as above. The mixture weights π_k were also learned and were paramterized as $\pi_1 = \sigma(\tau)$, $\pi_2 = 1 - \pi_1$ to allow for unconstrained optimization.

Regression examples

For Bayesian logistic regression, the target dimension was 14 and the auxiliary dimension used was 2 dimensional. The structure was otherwise the same as for the low dimensional experiments.

3.5.3 Calculation of the effective sample size

The effective sample size is calculated as the reciprocal of the auto-correlation-time. It is intended to represent the number of truly independent samples that would be equivalent to a correlated sample drawn using MCMC, in terms of the variance of

estimated quantities. One definition of the auto-correlation-time, for a 1-dimensional chain, is:

$$\rho = 1 + 2\sum_{\tau=1}^{\infty} C_\tau \tag{3.28}$$

where $C_\tau = E[x_t, x_{t+\tau}]$ is the lag-τ auto-correlation of the stationary converged MCMC chain. Though multivariate definitions of the auto-correlation exist, it is common practice to report the lowest effective-sample-size across all dimensions. For ease of comparison, we adopt this practice.

There are numerous methods for estimating ρ [Thompson, 2010]. It is worth noting however that simply estimating C_τ from multiple chains and substituting the estimates into the above formula does not yield a consistent estimator except in very simple cases when the auto-correlations are guaranteed to be positive. In general however, the auto-correlations can be of both negative and positive variance. The estimator formed by substituting the empirically estimated auto-correlations does not have a variance that goes to 0 as the sequence length goes to infinity[Thompson, 2010]. Instead we use the batch-means estimator.

3.5.4 batch-means estimator

An estimator of the auto-correlation time is computed using "batch-means" [Thompson, 2010]. A single long sequence is split into m sub-sequences and the mean of each sub-sequence is calculated. The auto-correlation-time is then estimated by using the ratio of the variance of the batch-means to the variance of the overall sequence. The estimator is given by:

$$\hat{\rho} = m\frac{s_m^2}{s^2} \tag{3.29}$$

where s_m^2 is the variance of the batch-means and s^2 is the variance of the entire chain.

Chapter 4

Semi-Supervised Generative Modelling for Controllable Speech Synthesis

The following chapter is based on the paper 'Semi-Supervised Modeling for Controllable Speech Synthesis' (Habib et al, 2019), which was presented at the International Conference of Learning Representations (ICLR). The work was started whilst interning at Google AI but substantially completed upon return to UCL. The paper was co-authored with Saroosh Maryooryad and Matt Shannon. Soroosh provided advice, extensive support in reproducing and modifying the Tacotron model [Wang et al., 2017] and helped in conducting human evaluations. Matt provided the code to calculate the MCD-DTW evaluation metric, helped train the classifiers used in evaluation and provided advice.

The ability to reliably control high level attributes of speech, such as emotional expression (affect) or speaking rate, is often desirable in speech synthesis applications. Achieving this control however is made difficult by the necessity of acquiring a large quantity of high quality labels. In this chapter we show that semi-supervised latent variable models can take us a significant step closer towards solving this problem.

4.1 Introduction

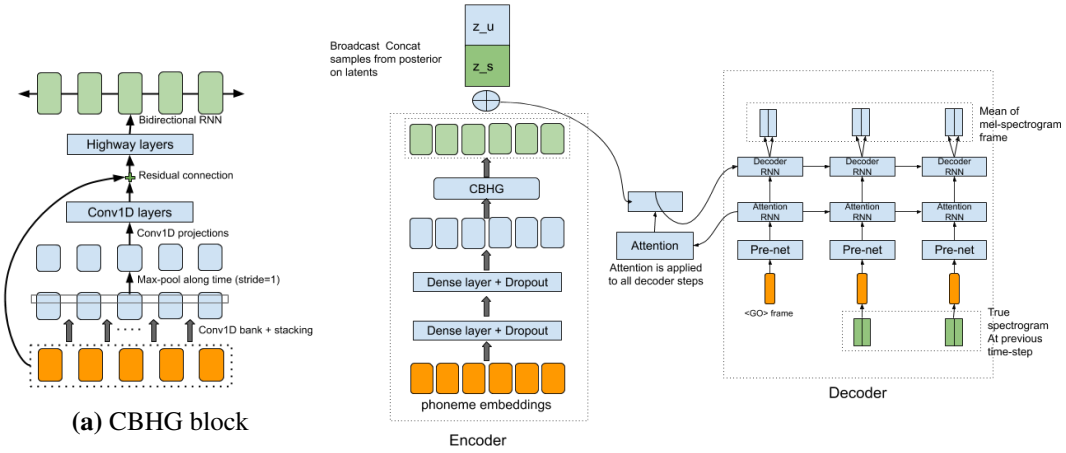
Combining state-of-the-art neural text-to-speech (TTS) systems with probabilistic latent variable models provides a natural framework for discovering aspects of speech that are rarely labelled or even difficult to describe. Both inferring the latent prosody and generating samples with sufficient variety requires reasoning about uncertainty and is thus a natural fit for deep generative models.

There has been recent progress in applying stochastic gradient variational Bayes (SGVB) [Kingma and Welling, 2014, Rezende et al., 2014] to training probabilistic neural TTS models. Battenberg et al. [2019] and Hsu et al. [2018] have shown that it is possible to use latent variable models to discover features such as speaking style, speaking rate, arousal, gender and even the quality of the recording environment.

However, these models are formally non-identifiable [Hyvärinen and Pajunen, 1999] and this implies that repeated training runs will not reliably discover the same latent attributes. Even if they did, a lengthy human post-processing stage is necessary to identify what the model has learned on any given training run. In order to be of practical use for control, it is not enough for the models to discover latent attributes, they need to do so reliably and in a way that is robust to random initialization and to changes in the model. We demonstrate that the addition of even modest amounts of supervision can be sufficient to achieve this reliability.

By augmenting state-of-the art neural TTS with semi-supervised deep generative models within the VAE framework [Kingma et al., 2014, Narayanaswamy et al., 2017], we show that it is possible to not only discover latent attributes of speech but to do so in a reliable and controllable manner. In particular we are able to achieve reliable control over affect, speaking rate and F0 variation (F0 is the fundamental frequency of oscillation of the vocal folds). Further, we provide demonstrations that it is possible to transfer controllability to speakers for whom we have no labels. Our core contributions are:

- To combine semi-supervised latent variable models with Neural TTS systems,



(a) CBHG block

(b) seq-to-seq network.

Figure 4.1: Schematic showing how we parameterize the conditional likelihood $p(x|y, z_u, z_s)$. Left: A block of 1-d convolutions and RNNs originally introduced by Wang et al. [2017] and described in detail in the appendix. Right: Schematic of the sequence-to-sequence network that outputs the means of our auto-regressive distribution. At each decoder time step, the network outputs the means for the next two spectrogram frames.

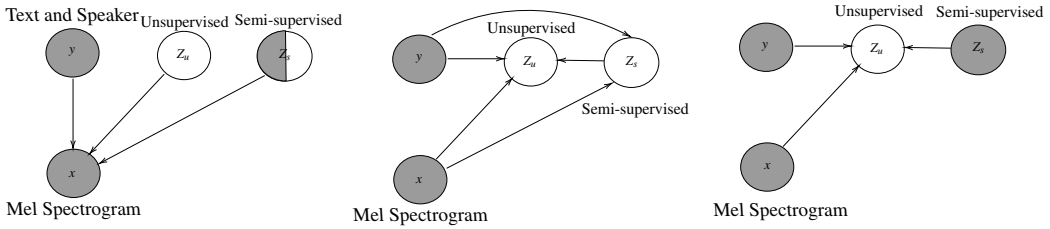
producing a system that can *reliably* discover attributes of speech we wish to control.

- To demonstrate that as little as 10 minutes of supervision can be sufficient to improve prosody and allow control over speaking rate, fundamental frequency (F0) variation and affect, a problem of interest to the speech community for well over two decades [Schröder, 2001].

- To embue TTS models with control over affect, F0 and speaking rate whilst still maintaining prosodic variation when sampling.

4.2 Generative Model

Our generative model, shown in figures 4.1 and 4.2a, consists of an autoregressive distribution over a sequence of acoustic features, $x_{1...t}$, that are generated conditioned on a sequence of text, $y_{1...k}$, and on two latent variables, z_u and z_s. The latent variables can be discrete or continuous. z_s represents the variations in prosody that we seek to control and is semi-supervised. z_u is fully unobserved and represents latent variations in prosody (intonation, rhythm, stress) that we wish to model but not explicitly



(a) Generative model **(b)** Unsupervised posterior **(c)** Supervised posterior

Figure 4.2: Left: The graphical model showing the conditional independence assumptions between each of the stochastic variables. Centre: The structure of the variational distribution used to approximate the posterior for fully unsupervised data points and Right: supervised points.

control. In full our model may be written

$$p(x_{1\ldots t}|y_{1\ldots t}) = \prod_{t=1}^{T} \int p(x_t|x_{<t}, y_{<t}, z_u, z_s) p(z_u) p(z_s) dz_s dz_u. \tag{4.1}$$

For each sequence of length T we have two latent variables, z_u and z_s that represent variations in prosody. In our experiments the sequences are typically 2-5s long. Once trained, our model can be used to synthesize acoustic features from text. Similar to Tacotron 2 [Shen et al., 2018], we then generate waveforms by training a second network such as WaveNet [van den Oord et al., 2016] or WaveRNN [Kalchbrenner et al., 2018] to act as a vocoder. In our case we use WaveRNN.

We parameterize our likelihood $p(x_{1\ldots t}|y_{1\ldots k}, z_u, z_s, \theta)$ by a sequence-to-sequence neural network with attention [Shen et al., 2018, Graves, 2013, Bahdanau et al., 2014] that is shown schematically in figure 4.1. Details largely follow Tacotron [Wang et al., 2017] and are given in appendix 4.7.1. At each time step we model a mel-spectrogram frame with a fixed variance isotropic Laplace distribution whose mean is output by the neural network. We condition each of the latent variables by concatenating the vectors z_u and z_s to the representation of the text-encoder, before the application of the attention mechanism. In the case of continuous z we use a standard normal prior and in the case of discrete z we use a uniform categorical prior with one-hot encoding.

4.2.1 Semi-Supervised Training

Following Kingma et al. [2014] and Narayanaswamy et al. [2017], we train our model via stochastic gradient variational Bayes (SGVB). That is we approximately maximize the log-likelihood of our training data by maximizing a variational lower bound using stochastic gradient ascent. Since we are training with semi-supervision we in fact need two lower bounds: one for the data points for which z_s is observed; one for the case where z_s is unobserved. In our models the fully unobserved latent variable z_u is always continuous but the semi-supervised latent z_s can be continuous or discrete. The conditional independence structure of our variational distributions is shown in figures 4.2b and 4.2c. On supervised data, the per-datapoint bound is:

$$\begin{aligned}
\log p(x, z_s|y) &= \log \int p(x, z_u, z_s|y, \theta) dz_u \\
&\geq E_{q(z_u|x,y,z_s,\phi)} \left[\log \left(\frac{p(x|y, z_u, z_s, \theta) p(z_u) p(z_s)}{q(z_u|x, y, z_s, \phi)} \right) \right] \\
&= E_{q(z_u|x,y,z_s,\phi)} \left[\log p(x|y, z_u, z_s, \theta) \right] + \log p(z_s) - D_{KL}(q(z_u|x, y, z_s, \phi) \| p(z_u)) \\
&= \mathscr{L}_s(\theta, \phi)
\end{aligned}$$

Where $q(z_u|x, y, z_s, \phi)$ is a parametric variational distribution introduced to approximately marginalize z_u. θ are the parameters of the generative model and ϕ are the parameters of the variational distributions. The intractable integrals are approximated with reparameterized samples. For the cases where z_s is unobserved and discrete, the bound is:

$$\log p(x|y) = \log \int \sum_{z_s} p(x, z_u, z_s|y) dz_u \tag{4.2}$$

$$\geq \sum_{z_s} [q(z_s|x, y, \phi) \mathscr{L}_s(\theta, \phi)] + H(q(z_s|x, y, \phi)) \tag{4.3}$$

$$= \mathscr{L}_u(\theta, \phi) \tag{4.4}$$

and when z_s is continuous we replace the sum above with an integral and again approximate with reparameterized samples. The variational distributions are parameterized by a neural network that takes as input the text, spectrograms and other conditioning variables and outputs the parameters of the distribution. The exact structure of this network is given in appendix 4.7.1. We have implicitly assumed that $q(z_u, z_s|x, y, \phi)$ may be factorized as $q(z_u, z_s|x, y, \phi) = q(z_u|x, y, z_s, \phi)q(z_s|x, y, \phi)$ with shared parameters between these two distributions (see appendix 4.7.1). Optimizing the variational objective with respect to the parameters ϕ encourages the variational distributions to match the posterior of the generative model $p(z_u, z_s|x, y, \theta)$. Unlike previous work [Hsu et al., 2018], we do not assume that the posterior on the latents is independent of the text, as this dependence likely exists in the model due to explaining away. That is to say that although the text and the latents are independent in our prior, observing the spectrogram correlates them in the posterior because they both explain variation in the spectrogram. This has been shown to be significant by Battenberg et al. [2019].

If we define

$$\tilde{q}(z_s|x, y) = \begin{cases} q(z_s|x, y, \phi) & \text{if unsupervised} \\ \gamma\delta(z_s - z_{s_{observed}}) & \text{if supervised} \end{cases} \tag{4.5}$$

then we can write the overall objective over both the supervised and unsupervised points succinctly as[1]

$$\mathscr{L}(\theta, \phi) = E_{x,y,z_s}\left[\sum_{z_s}[\tilde{q}(z_s|x, y, \phi)\mathscr{L}_s(\theta, \phi)] + H(\tilde{q}(z_s|x, y, \phi))\right] \tag{4.6}$$

where summation would again be replaced by integration for continuous z_s and γ (shown in equation 4.5) is a weighting factor that pre-multiplies the loss for any supervised point. This weighting was also used in previous work such as

[1] We define the differential entropy of the delta function to be 0

Narayanaswamy et al. [2017], who showed it to be beneficial at very low levels of supervision.

Writing the objective in this form allows an intuitive interpretation for the semi-supervised training procedure. When supervision is provided, our objective function is evaluated at the observed value of z_s. When supervision is not provided, we evaluate the objective function for every possible value of z_s and take a (potentially infinite for continuous z_s) weighted average. The weighting in the average is given by $q(z_s|x,y,\phi)$, which is simultaneously trained to approximate the posterior $p(z_s|x,y,\theta)$. In other words, on unsupervised utterances, we evaluate our objective for each possible value of the latent attribute and weight by the (approximate) posterior probability that this value of the latent was responsible for generating the utterance.

As $q(z_s|x,y,\phi)$ is trained to approximate $p(z_s|x,y,\theta)$ we can expect it to become a reasonable classifier/regressor for the semi-supervised latent attribute as the model improves. For example when z_s represents an affect label, $p(z_s|x,y,\theta)$ is the posterior probability, of the model, over affect given text and speech. By taking the most likely posterior class, this distribution can be used as an affect classifier. However, this variational distribution is only trained on unsupervised training points and so does not benefit directly from the supervised data. To overcome this problem we follow Kingma et al. [2014] and add a classification loss to our objective. The overall objective becomes

$$\mathscr{L}_{total}(\theta,\phi) = \mathscr{L}(\theta,\phi) + \alpha E_{x,y,z_s}[\log q(z_s|x,y,\phi)] \tag{4.7}$$

where α is a hyper parameter which adjusts the contribution of this term and the expecation is over supervised data points.

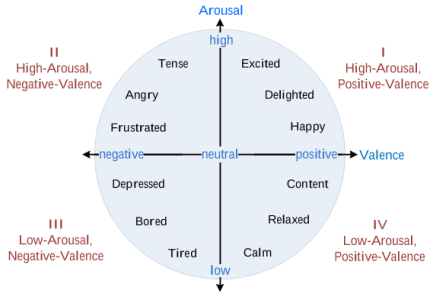


Figure 4.3: The circumplex model of emotion. Each possible emotion is represented in a 2 dimensional plane consisting of an arousal dimension and valence dimension. This figure is borrowed from Munoz-de Escalona and Canas [2017].

4.3 Data

We have used a proprietary high quality labeled data-set of 40 English speakers. The training set consists of 72,405 utterances with durations of at most 5 seconds (45 hours). The validation and test sets each contain 745 utterances or roughly 30 minutes of data. We vary the amount of supervision in the experiments below. We also experimented with transferring controllability to a fully unlabeled data-set of audiobook recordings by Catherine Byers (the speaker from the 2013 Blizzard Challenge), which exhibits high variation in affect and prosody and to other speakers who were less expressive. We strongly encourage the reader to listen to the synthesized samples on our demo page[2].

In this work we chose to focus on learning to control affect with a discrete representation, as well as speaking rate and F0 variation with a continuous representation, as these are challenging aspects of prosody to control. Our method could be applied to other factors without modification.

4.3.1 Affect Control

The best way to represent emotion is an actively researched area and many models of affect exist. In this work we chose to follow the circumplex model of emotion [Russell, 1980] which posits that most affective states can be represented in a 2 dimensional plane with one axis representing arousal and the other axis representing valence. Arousal measures the level of excitement or energy and valence mea-

[2]Sound demos are available at `https://tts-demos.github.io/`.

sures positivity or negativity. Figure 4.3, shows a chart of emotions plotted in the arousal-valence plane where we can see that, for example, high arousal and high valence corresponds to joy or happiness whereas high arousal and low valence might correspond to anger or frustration.

Our data-set was recorded under studio conditions with trained voice actors who were prompted to read dialogues in one of three valences:-2, -1, +2 and two arousal values:-2 (low), +2 (high). This was achieved by prompting the actors to read dialogues in either a happy, sad or angry voice at two levels of arousal. This results in 6 possible affective states which we chose to model as discrete and use as our supervision labels.

4.3.2 Speaking Rate and F0 Variation Control

In order to demonstrate that we can control continuous attributes we also created approximate real-valued labels for speaking rate and arousal for all of our data. We generate the approximate speaking rate as number of syllables per second in each utterance. F0, also known as the fundamental frequency, measures the frequency of vibration of the vocal folds during voiced sounds. Variation in F0 is highly correlated with arousal and roughly measures how expressive an utterance is. To create approximate arousal labels we extracted the F0 contour from each of our utterances, using the YIN algorithm [De Cheveigné and Kawahara, 2002], and measured its standard deviation. We then performed a whitening transform on these two approximate labels in order to match it to our standard normal prior.

These artificial labels would of course be cheap to obtain for the entire data-set and would not justify the use of semi-supervision in real applications. But, our objective here is to evaluate/demonstrate the efficacy of semi-supervision rather than to specifically control a particular attribute. We have chosen syllable rate and F0 standard deviation, because they both correspond to subjectively distinct variations of interest, and they are more easily quantifiable than affect and so provide strong evidence of controllability. For the continuous latents we are not only able to interpolate speaking-rates and F0 variations but also to extrapolate outside of our

training data. We provide examples on our demo page of samples with significantly greater/lower speed and F0 variation than typically observed in natural speech.

4.4 Experiments and Results

To evaluate the efficacy of semi-supervised latent variable models for controllable TTS we trained the model described in section 4.2 on the above data-sets at varying levels of supervision as well as for varying settings of the hyper-parameters: α which controls the supervision loss and γ, which over emphasizes supervised training points. We found that a value of $\alpha = 1$ was optimal for the discrete experiments and $\alpha = 0$ for the continuous experiments, which corresponds to simply optimizing the ELBO. For each experiment we report the results for the best γ found, and $\gamma = 1$. $\gamma = 1$ corresponds to experiments with no over-weighting of the supervised points. All models were trained using the ADAM optimizer with learning rate of 10^{-3} and run for $300,000$ training steps with a batch size of 256, distributed across 32 Google Cloud TPU chips. All models were implemented using tensorflow 1 [Abadi and Agarwal, 2015].

Assessing the degree of control is challenging as interpreting affect is subjective. We used two objective metrics of control as well as subjective evaluation from human raters and a third objective metric of overall quality. For affect, the first objective metric we introduced was the test-set accuracy of a 6-class affect classifier trained on the ground truth training data and applied to generated samples from the model (shown in figure 4.4a). The classifier is a convolutional neural network whose structure mirrors the posterior network $q(z_s|x,y,\phi)$ and its exact architecture is given in appendix 4.7.1. We also provide subjective metrics of controllability, shown in table 4.1. For speaking rate control, we are able to measure the syllable rate and so report the mean syllable rate error on a held out test-set. The syllable rate error is calculated as the absolute difference in syllable rate between the desired syllable rate and that measured from the synthesized sample. We calculate an analogous error rate for F0 variation.

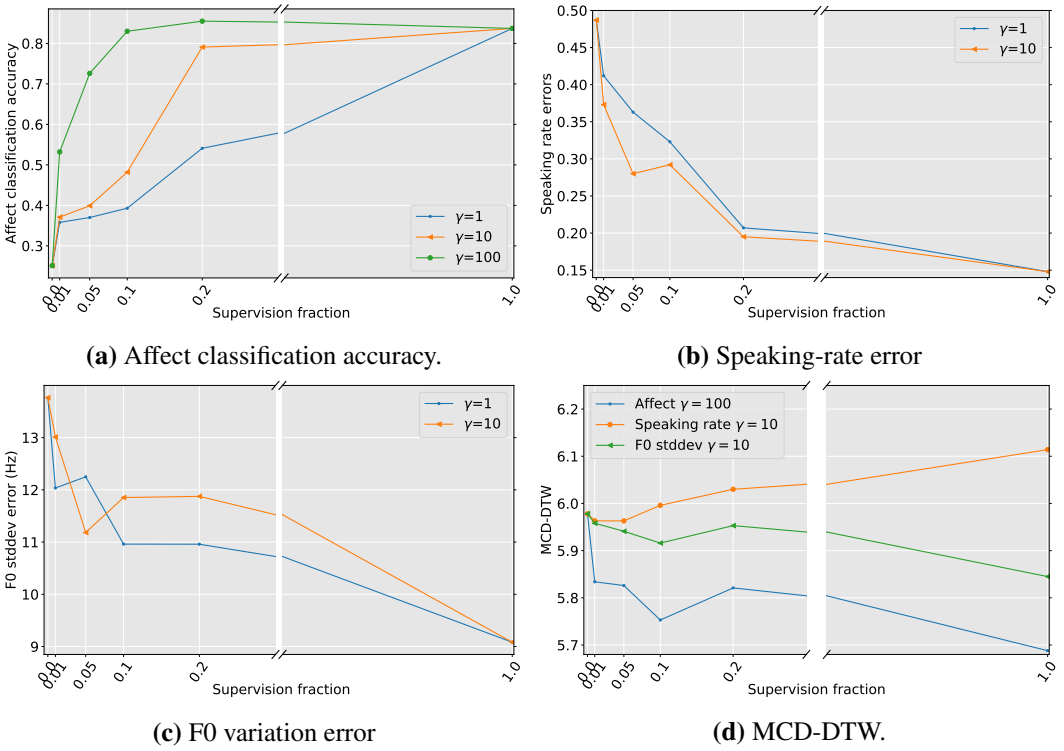
		Valence			Arousal
		baseline vs. angry	baseline vs. sad	baseline vs. happy	low vs. high
preference score	27 min (1%)	-0.20 ± 0.10	-0.60 ± 0.08	-0.43 ± 0.09	-0.50 ± 0.09
	135 min (5%)	-0.74 ± 0.07	-0.83 ± 0.06	-0.83 ± 0.06	-0.57 ± 0.08
	270 min (10%)	-0.71 ± 0.07	-0.86 ± NA	-0.61 ± 0.08	-0.59 ± 0.08

Table 4.1: Subjective metrics for affect control. Negative is a preference for the controlled model. +1 indicates a preference for sample A and -1 indicates a preference for sample B. For valence, raters are told that a sample is intended to convey a particular emotion, e.g. happy, and then presented with sample from baseline without control (A), and controlled model (B), and asked to choose between them. For arousal, raters are told to choose the sample that is more vocally aroused, and presented with controlled samples in low (A) and high (B) arousal. To avoid bias, the orders are randomly altered during rating. We show preference score and 95% confidence intervals at multiple supervision levels.

Whilst the two metrics above measure controllability they don't tell us if this comes at the expense of a degradation in synthesis quality. To probe quality we use three further metrics. The first was Mel-Cepstral-Distortion-Dynamic-Time-Warping (MCD-DTW) [Kubichek, 1993] on a held out test-set, shown in figure 4.4d. MCD-DTW is a measure of the difference between the ground-truth spectrogram and the synthesized mel spectrogram that is known to correlate well with human perception [Kubichek, 1993]. The second metric of quality was crowd sourced mean-opinion-scores (MOS). The third metric of quality is speech recognition word error rate (WER) and character error rate (CER) on audio samples. The MOS and speech recognition results are summarized in table 4.2.

To demonstrate that semi-supervision by including unlabelled data is beneficial, we also provide MOS and speech recognition errors for fully supervised subsets of the data in table 4.2. These show that at least close to 5 hours of data is required to train a reasonable quality TTS model, far above the 30 minutes supervision needed to control prosodic aspects of speech.

We provide further details of all of these metrics in the appendix 4.7.2, and sample spectrograms are provided in appendix 4.7.3.



(a) Affect classification accuracy.

(b) Speaking-rate error

(c) F0 variation error

(d) MCD-DTW.

Figure 4.4: Objective evaluation metrics as a function of supervision fraction. 100% supervision corresponds to 45 hours of supervised training data and 0% supervision corresponds to base tacotron. For MCD-DTW and error-rates lower is better.

4.5 Discussion

The classification accuracy (see figure 4.4a), subjective metrics (see table 4.1) and error-rate results (see figure 4.4b-4.4c) provide a clear demonstration that using semi-supervised latent variables, we are able to achieve control of both continuous and discrete attributes of speech. There is not a significant degradation in the overall quality and this is evidenced by the mean opinion scores which are above the baseline, Tacotron, and also speech recognition errors (see table 4.2). We also include a baseline of our Tacotron model augmented only by the unsupervised latent z_s, to aid comparison. The MCD-DTW scores for F0 variation and affect are improved at all levels of supervision (figure 4.4d). Whilst the MCD-DTW is degraded for speaking rate, this is likely a misleading metric when targeting changes in timing as the dynamic-time-warping component of MCD-DTW changes exactly the aspect we wish to control. For speaking rate the combination of MOS and samples is a better indication of the overall quality. We are able to reduce the supervision level to

				Semi-Supervised (10% supervision)		
				continuous latent		discrete latent
	ground truth	baseline	baseline with z_u	F0	speaking-rate	affect
MOS	4.52±0.07	4.09±0.09	4.24±0.08	4.28±0.07	4.16±0.08	4.17±0.09
WER	4.49	4.93	4.93	4.06	2.53	6.09
CER	2.11	2.38	2.22	1.80	1.14	3.12

Table 4.2: Metrics of overall quality: Mean Opinion Scores (MOS) alongside 95% confidence intervals, speech recognition word error rate (WER), and chararacter error rate (CER). The results show no degradation in performance compared to the baseline.

	27 min (1%)	54 min (2%)	108 min (4%)	135 min (5%)	270 min (10%)	45 hours (100%)
MOS	unintelligible	unintelligible	3.20±0.13	3.52±0.11	4.03±0.08	4.08±0.09
WER	91.95	96.31	19.83	7.55	5.56	4.93
CER	74.57	78.9	12.54	4.46	2.79	2.22

Table 4.3: Metrics of overall quality for fully supervised data at varying data-set sizes, showing significant degradation below 270 minutes.

levels as low as 1% or 30 minutes and still have a significant degree of control. We show on our demo page[3] that even at 3 minutes of supervision we can still achieve control of speaking rate and that we are able to extrapolate outside the range of values seen during training. On the affect data our classification accuracy doesn't degrade significantly until we reach 10% (300 minutes) supervision and remains significantly above chance down to levels as low as 1% (30 minutes), see figure 4.4a and table 4.1. Obtaining 30 minutes of supervised data is likely within reach of most teams constructing TTS systems. Unlike previous work on generative modelling for control [Hsu et al., 2018, Wang et al., 2018], we do not require a post-processing stage to determine what our latent variables control and we can pre-determine what aspects we wish to control through choice of data. By separating our latent variables into those that are partially supervised and those which are fully unsupervised we retain the ability to model other latent aspects of prosody; this means that we can still draw samples of varying prosody whilst holding constant the affect or speaking rate.

We observe the greatest degree of affect control, as measured by classifier accuracy, when $\alpha = 1$ and $\gamma = 100$. This means that to achieve the highest controllability we

[3] https://tts-demos.github.io

needed to 1) provide extra information to our approximate posterior $q(z_s|x,y,\phi)$ and 2) to over-represent the supervised data at low levels of supervision. Although both of these hyper-parameters have been used in the literature before [Narayanaswamy et al., 2017, Kingma et al., 2014] and shown to be either beneficial or necessary, they aren't strictly required by our probabilistic framework and so it is worth considering why they are needed. There are three potential sources of error in any generative model trained with SGVB: the model itself may be mis-specified such that the true data-generating distribution is not in the model class, the parametric family chosen to approximate the posterior may be overly restrictive and finally the optimization landscape may contain undesirable local minima. These problems have afflicted previous work with deep latent variable models trained with SGVB, resulting in models that don't use their latent variables unless trained with complex annealing schedules [Bowman et al., 2015]. In our case we believe that the necessity to set α and γ arises from a combination of model mis-specification and local minima. If α is set to 0, then at the start of training $q(z_s|x,y,\phi)$ is trained only to approximate $p(z_s|x,y,\theta_0)$, which is randomly initialized. We found empirically that in our discrete-latent experiments this resulted in $q(z_s|x,y,\phi)$ collapsing early in training to a point-mass on a single class for every single training example. Having ended up in this undesirable local minimum the posterior distribution never recovered, despite this being an obviously poor approximation to the model posterior later in training. The addition of the classification loss and supervision weighting were sufficient to overcome this collapse and allow q to continue to model the posterior.

The optimization landscape is strongly affected by the relative size of the conditional likelihood and KL terms in our objective. These are in turn strongly affected by our choice of conditional independence assumptions and output-distributions. Thus, a natural direction for further work is to increase the expressivity of the conditional likelihood $p(x|y,z_s,z_u,\phi)$ to reduce model mis-specification. This could be done by learning the variance of the Laplace-distribution we currently use or by parameterizing more expressive output distributions that do not assume conditional independence across spectrogram channels. We conjecture that with more expressive

output distributions, it may be possible to reduce the need for the α and γ terms in the objective. In this work we chose to use quite simple unconditional diagonal Gaussian priors, as our primary goal was to demonstrate the practicality of semi-supervision. Another natural extension would be to use conditional-priors $p(z|y)$ and to use more expressive priors such as mixtures as was done in Hsu et al. [2018].

4.5.1 Related Work

There has been enormous recent progress in neural TTS with numerous novel models proposed in recent years to synthesize speech directly from characters or phonemes [Shen et al., 2018, Arik et al., 2017, Gibiansky et al., 2017, Ping et al., 2017, Vasquez and Lewis, 2019, Taigman et al., 2017]. Differentiating factors between these models include the degree of parallelism, with some models using Transformer based architectures [Ren et al., 2019], the choice of conditional independence assumptions made [Vasquez and Lewis, 2019] or the number of separately trained components [Gibiansky et al., 2017]. Our work here is largely orthogonal to the exact structure of the conditional likelihood $p(x|y, z_s, z_u)$ and could be combined with all of the above methods.

Much of the recent research focus has been on modeling latent aspects of prosody. Early attempts include Global Style Tokens [Wang et al., 2018] which attempted to learn a trainable set of style-embeddings. Wang et al. [2018] condition the Tacotron decoder on a linear combination of embedding vectors whose weights during training are predicted from the ground-truth spectrogram. They were able to achieve prosodic control but there is no straightforward way to sample utterances of varying prosody. More recently, attempts have also been made to combine probabilistic latent variable models trained using SGVB [Akuzawa et al., 2018, Wan et al., 2019]. These models use a fully unsupervised and non-identifiable approach, which makes it difficult to disentangle or interpret their latent variables for control. Hsu et al. [2018] attempt to overcome this problem by using a Gaussian mixture as the latent prior and so perform clustering in the latent space. Battenberg et al. [2019] introduce a hierarchical latent variable model to separate the modelling of style from prosody. However, all of these

methods are fully unsupervised and this results in latents that can be hard to interpret or require complex post-processing.

The work most similar to ours is Wu et al. [2019] which also attempts to achieve affect control using semi-supervision with a heuristic approach based on Global Style Tokens [Wang et al., 2018]. Wu et al. [2019] add a cross-entropy objective to the weightings of the style-tokens that encourages them to be one-hot on points with supervision. Similar to our method, they are able to achieve control over affect but unlike our method they do not have a principled probabilistic interpretation nor the ability to simultaneously model aspects of prosody other than emotion. The result is that their method is not able to draw samples of varying prosody for the same utterance with fixed emotion. Furthermore, whilst our method can be applied to both continuous and discrete controllable factors, its not clear how to extend the style-token based approach to handle continuous latent factors.

In the wider generative modelling literature, the combination of semi-supervision and deep latent variable models was first introduced in Kingma et al. [2014] who focus on using unlabelled data to improve classification accuracy. The potential to use the same technique for controllable generation was recognized by Narayanaswamy et al. [2017] who also provided demonstrations on image synthesis tasks. Since that work, interest in learning disentangled latent variables has grown but generally pursued alternate directions such as re-weighting the ELBO [Higgins et al., 2017], augmenting the objective to encourage factorization [Kim and Mnih, 2018] or using adversarial training [Mathieu et al., 2016]. The ability to transfer controllability to speakers for whom we do not have supervision is referred to as domain transfer and our model bears similarities to that introduced by Ilse et al. [2019] but they use a mixture in their latent space more similar to Hsu et al. [2018].

4.5.2 Ethical Considerations

As with many advances in speech synthesis, progress in controllability raises the prospect that bad actors may misuse the technology either for misinformation or to commit fraud. Improvements in data efficiency and realism increase these risks and,

when publishing, a consideration has to be made as to whether the benefits of the developments outweigh the risks. It is my opinion in this case that, since the focus of this work is on improved prosody, with potential benefits to human-computer interfaces, the benefits likely outweigh the risks. We nonetheless urge the research community to take seriously the potential for misuse both of this work and broader advances in TTS.

4.6 Conclusion

We have shown that the combination of semi-supervised latent variable models with neural TTS presents a practical and principled path towards building speech synthesizers we can control. Unlike previous fully unsupervised methods, we are able to consistently and reliably learn to control predetermined aspects of prosody. Our method can be applied to any latent attribute of speech for which a modest amount of labelling can be obtained, whether it be continuous or discrete. In our experiments we found that 30 minutes of supervision was sufficient, a volume of data that is within the reach of most research teams. We are able to learn to control subtle characteristics of speech such as affect and for continuous attributes we have provided demonstrations of extrapolation to ranges never seen during training, and to speakers with no supervision. Augmenting existing state-of-the-art TTS systems with latent variables does not degrade synthesis quality and we evidence this with crowd sourced mean opinion scores. Unlike similar heuristic methods, our probabilistic formulation, allows us to draw samples of varying prosody whilst holding constant some attribute we wish to control.

4.7 Appendix

4.7.1 Neural Network Architecture

Module	Hyper-parameters
Input	Text normalized phonemes
Phoneme embedding	256-D
Pre-net	FC-256-Relu-Dropout(0.5) → FC-128-Relu-Dropout(0.5)
CHBG Text Encoder	Conv1D bank: K=16, conv-k-128-Relu →Max pooling with stride=1 width=2 → Conv1D projections: conv-3-128-Relu → conv-3-128-Linear → Highway net: 4 layers of FC-128-Relu → Bidirectional GRU: 128 cells
Attention type	5 component GMM attention w/ softplus [Graves, 2013]
Attention RNN	LSTM-256-Zoneout(0.1) → FC-128-tanh
DecoderRNN	2-layer residual-LSTM-265-zoneout(0.1) → FC-80-Linear
Frames-per-timestep (reduction factor)	2
WaveRNN	5 layers DilatedConv1D-512 → 2 layers TransposeConv + ReLu → GRU-768 conditioned on 5 previous samples → FC-768-relu → 3 component MoL, 24kHz sample rate
Variational Posterior	Spectrogram → 6 Conv-layers 32-32-64-64-128-128 → LSTM-128 → FC-128-tanh
Optimizer	ADAM with learning rate 10^{-3}, batch-size 256
Speaker embedding	64-D

Table 4.4: Summary of the hyper-parameters described below.

Sequence-to-Sequence model Our sequence-to-sequence network is modelled on Tacotron [Wang et al., 2018] but uses some modifications introduced in Skerry-Ryan et al. [2018]. Input to the model consists of sequences of phonemes produced by a text normalization pipeline rather than character inputs. The CBHG text encoder from Wang et al. [2017] is used to convert the input phonemes into a sequence of text embeddings. The phoneme inputs are converted to learned 256-dimensional embeddings and passed through a pre-net composed of two fully connected ReLU layers (with 256 and 128 units, respectively), with dropout of 0.5 applied to the output of each layer, before being fed to the encoder. For multi-speaker models, a learned embedding for the target speaker is broadcast-concatenated to the output of the text encoder. The attention module uses a single LSTM layer with 256 units and zoneout of 0.1 followed by an MLP with 128 tanh hidden units to compute

parameters for the monotonic 5-component GMM attention window. Instead of using the exponential function to compute the shift and scale parameters of the GMM components as in Graves [2013], we use the softplus function, which we found leads to faster alignment and more stable optimization. The attention weights predicted by the attention network are used to compute a weighted sum of output of the text encoder, producing a context vector. The context vector is concatenated with the output of the attention LSTM layer before being passed to the first decoder LSTM layer. The autoregressive decoder module consists of 2 LSTM layers each with 256 units, zoneout of 0.1, and residual connections between the layers. The spectrogram output is produced using a linear layer on top of the 2 LSTM layers, and we use a reduction factor of 2, meaning we predict two spectrogram frames for each decoder step. The decoder is fed the last frame of its most recent prediction (or the previous ground truth frame during training) and the current context as computed by the attention module. Before being fed to the decoder, the previous prediction is passed through a pre-net with the same same structure used before the text encoder above but its own parameters.

CHBG Text Encoder We reuse the CHGB text encoder introduced in Wang et al. [2018]. The text encoder consists of a bank of 1-D convolutional filters, followed by highway networks and a bidirectional gated recurrent unit (GRU) recurrent neural net (RNN). The input sequence is first convolved with K sets of 1-D convolutional filters, where the k-th set contains C_k filters of width k. The convolution outputs are stacked together and further max pooled, preserving time. As in the original paper we use a stride of 1 to preserve the original time resolution. We further pass the processed sequence to a few fixed-width 1-D convolutions, whose outputs are added with the original input sequence via residual connections. Batch normalization is used for all convolutional layers. The convolution outputs are fed into a multi-layer highway network to extract high-level features. Finally, we stack a bidirectional GRU RNN on top to extract sequential features from both forward and backward context.

Variational Posteriors The variational distributions $q(z_s|x,y)$ and $q(z_u|x,y,z_s)$ are both structured as diagonal Gaussian distributions whose mean and variance are parameterized by neural networks. For discrete supervision we replace $q(z_s|x,y)$ by a categorical distribution and use the same network to output just the mean. The input to the distribution starts from the mel spectrogram x and passes it through a stack of 6 convolutional layers, each using ReLU non-linearities, 3x3 filters, 2x2 stride, and batch normalization. The 6 layers have 32, 32, 64, 64, 128, and 128 filters, respectively. The output of this convolution stack is fed into a unidirectional LSTM with 128 units. We pass the final output of this LSTM (and potentially vectors describing the text and/or speaker) through an MLP with 128 tanh hidden units to produce the parameters of the diagonal Gaussian posterior which we sample from. All but the last linear layer of these networks is shared between the two distributions $q(z_s|x,y)$ and $q(z_u|x,y,z_s)$. The resulting sample is broadcast-concatenated to the output of the text encoder. In our experiments z_u is always 32-dimensional and z_s is either a one-hot vector across 6 classes or a 1 dimensional continuous value.

Conditional inputs When providing information about the text to the variational posterior, we pass the sequence of text embeddings produced by the text encoder to a unidirectional RNN with 128 units and use its final output as a fixed-length text summary that is passed to the posterior MLP. Speaker information is passed to the posterior MLP via a learned speaker embedding.

WaveRNN We used a WaveRNN model similar to that described in Kalchbrenner et al. [2018] as our vocoder. Our WaveRNN uses discretized mixture of logistics output as described in Salimans et al. [2017] instead of the dual softmax from that paper, and conditions on 5 previous samples at each step instead of only 1 previous. We trained the network to map from synthesized mel-spectrograms to waveforms, training on 900 sample windows. A conditioning stack of dilated convolution and transpose convolutions is applied to the input spectrogram before tiling to upsample to the audio sample rate.

4.7.2 Evaluation

mel spectrograms The mel spectrograms the model predicts are computed from 24 kHz audio using a frame size of 50 ms, a hop size of 12.5 ms, an FFT size of 2048, and a Hann window. From the FFT energies, we compute 80 mel bins distributed between 80 Hz and 12 kHz.

MCD-DTW To compute mel cepstral distortion (MCD) [Kubichek, 1993], we use the same mel spectrogram parameters described above and take the discrete-cosine-transform to compute the first 13 Mel Frequency Cepstral Coefficients (MFCCs) (not including the 0th coefficient). The MCD between two frames is the Euclidean distance between their MFCC vectors. Then we use the dynamic time warping (DTW) algorithm [Kubichek, 1993] (with a warp penalty of 1.0) to find an alignment between two spectrograms that produces the minimum MCD cost (including the total warp penalty). We report the average per-frame MCD-DTW.

Affect Classifier The affect classifier has a very similar structure to the variational posterior. The input to the classifier starts from the mel spectrogram x and passes it through a stack of 6 convolutional layers, each using ReLU non-linearities, 3x3 filters, 2x2 stride, and batch normalization. The 6 layers have 32, 32, 64, 64, 128, and 128 filters, respectively. The output of this convolution stack is fed into a unidirectional LSTM with 128 units. The final output of the LSTM is then passed through a softmax non-linearity to get logits over the training classes. We use the same data splits described in section 4.3 to train and evaluate the classifier. The classifier is tuned on the validation set achieving 84.33% classification accuracy, generalizing well to the test set with 83.94% accuracy.

Mean Opinion Scores We use a human rating service similar to Amazon's Mechanical Turk, with a large pool of English speakers to collect MOS evaluations. The MOS template is shown in figure 4.5. A human rater is presented with a single speech sample and is asked to rate perceived naturalness on a scale of 1 to 5, where 1 is "Bad" and 5 is "Excellent". We have selected the utterances of one male, and one female speaker in our test set, totalling 371 utterances to evaluate. For each sample, we collect 1 rating, and no rater is used for more than 6 items in a single

evaluation set. In total, 270 unique raters completed the 6 evaluation sets presented in table 4.2. Since raters are randomly selected for each set, some raters have assessed multiple methods. Across the 6 evaluation sets, the average and median of total number of ratings per rater was 8.24, and 6, respectively. To analyze the data from these subjective tests, we average the scores and compute 95% confidence intervals. Natural human speech is typically rated around 4.5. Samples used for MOS from our model were drawn using the mean of z_u, whilst sampling z_s.

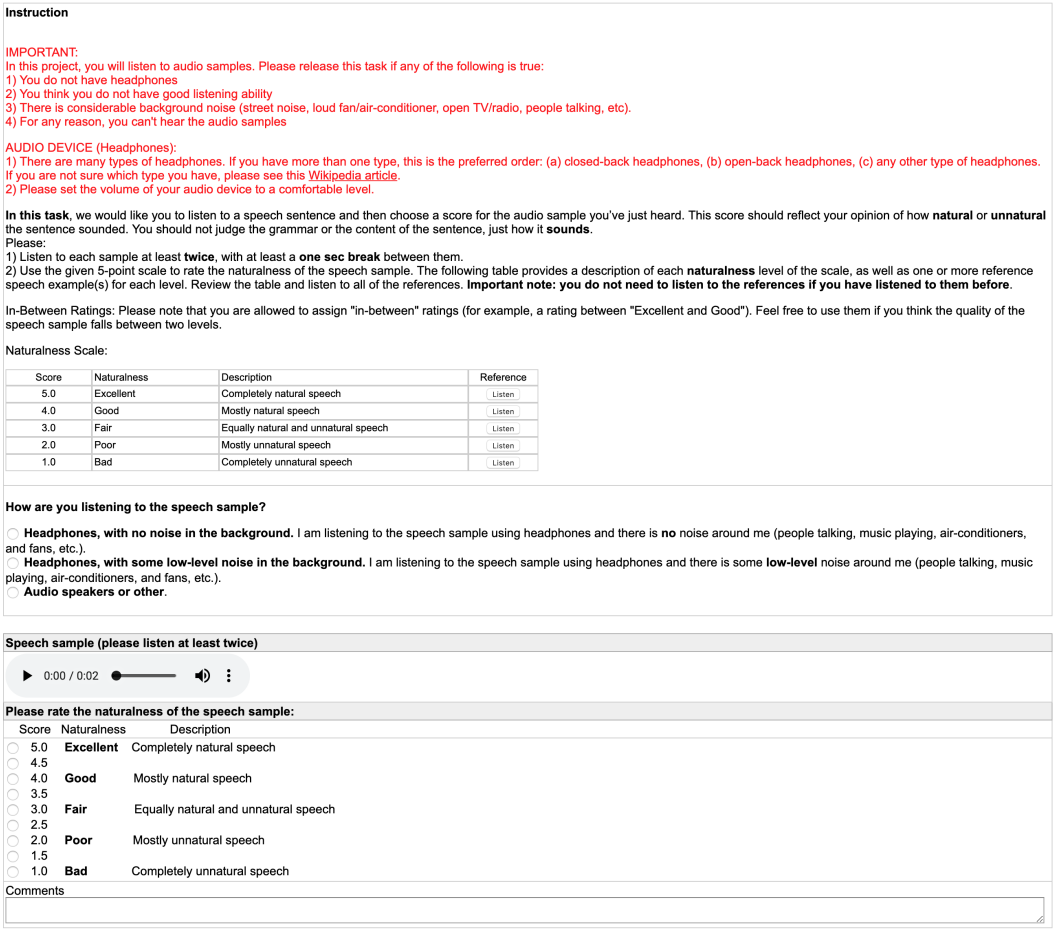


Figure 4.5: Mean opinion score (MOS) evaluation template. For each utterance, the human raters assign a 1–5 score of the perceived naturalness, with 1 being "Bad" and 5 being "Excellent".

Subjective affect control evaluation We use the same rater pool, and the same set of 371 utterances used for the MOS evaluations. The A/B template is shown in figure 4.6. For each utterance, the human rater is presented with a pair of utterances to choose the one that better conveys the target emotion (e.g., happy in the figure). Both

utterances are generated with the same text. To evaluate the control over valence, we present baseline (i.e, no control) against utterances generated in specific valence category (angry, happy and sad). To evaluate the control over arousal, we present samples generated at low arousal against samples generated at high arousal, and ask the rater to choose the utterance that is more vocally aroused. We use mean of z_u to generate all the samples.

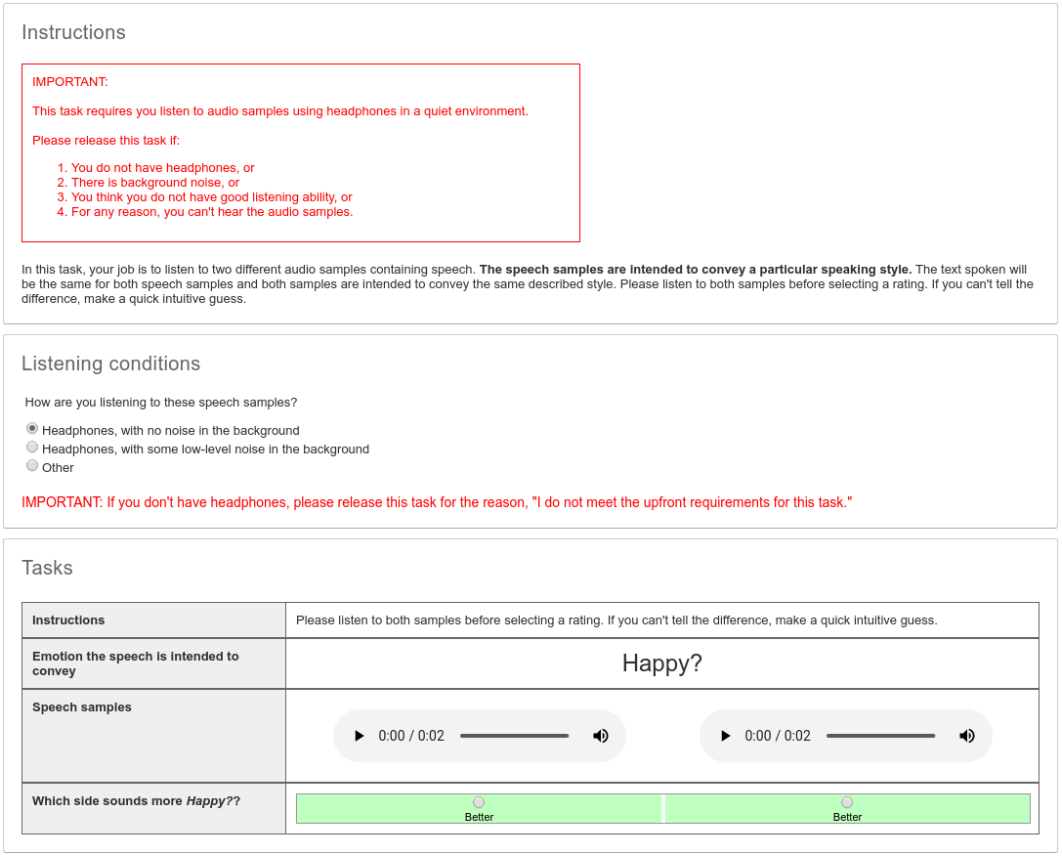


Figure 4.6: A/B evaluation affect control evaluation template. The emotion label (Happy in the figure) varies depending on the task.

4.7.3 Sample Spectrograms

Controlling Affect Table 4.5 shows the effect of varying the valence and arousal latent variable on the spectrogram and F0 track. We can see that a low valence for sadness corresponds to the flattest F0 track, and high arousal manifests in higher F0 values and variations.

Controlling Speaking Rate and Pitch Variations Table 4.6 shows the effect of varying speaking rate, and F0 variation control variables on a sample spectrogram and

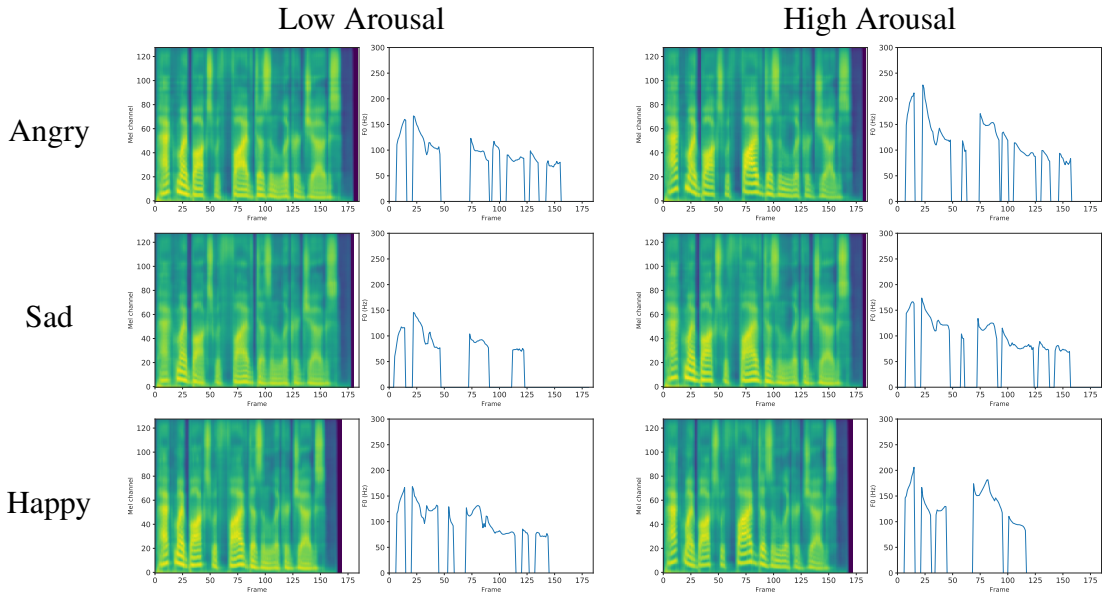


Table 4.5: Sample spectrogram and F0 track plots, generated by varying affect labels, valence in y-axis, and arousal in x-axis.

F0 track. When controlling speaking rate (first column), the duration reduces as we increase the input speaking rate control, while the F0 variation remains stable. When controlling the F0 variation (second column), the pitch dynamic range increases, while the duration remains constant, which demonstrates controllablity and also some degree of disentanglement.

4.7.4 Reproducing results on LibriTTS public dataset

To verify the reproduciblity of our results on a public dataset, we trained models to control speaking rate and F0 variation on clean subset of LibriTTS dataset [Zen et al., 2019]. We only use the utterances below 5 seconds, which is 62 hours of data. We have done no tuning on this dataset, and directly used the hyperparametrs we used for our proprietary dataset. Figures 4.7a and 4.7b show the errors of producing the desired speaking rate, and F0 standard deviation, which generally go down as function of supervision level, with exception of 10% supervision for controlling F0 variation. Given, this is a lower quality dataset, with many more speakers and with much smaller data per speaker and also the fact that we have done zero hyperparameter tuning on this dataset, this result look very encouraging.

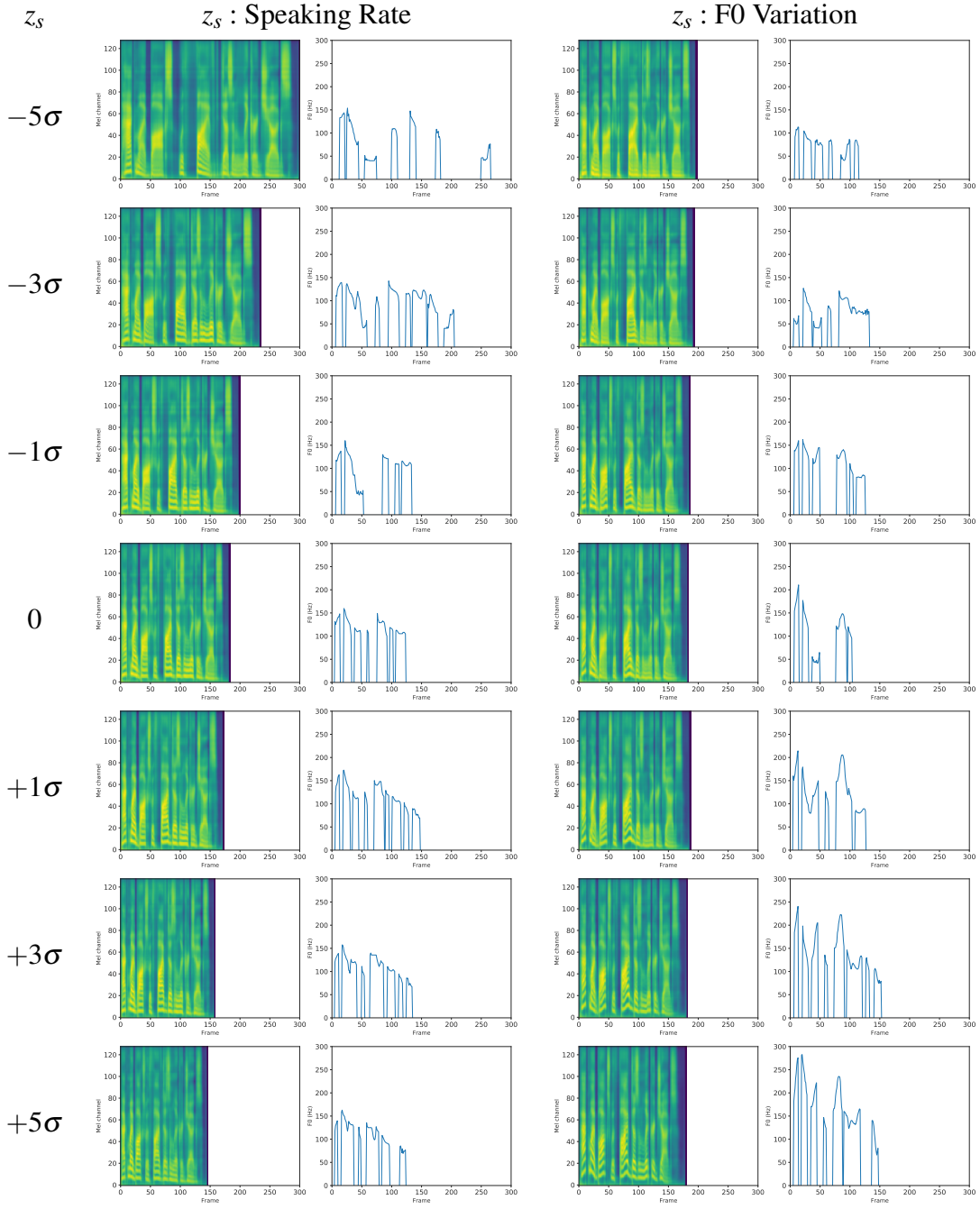
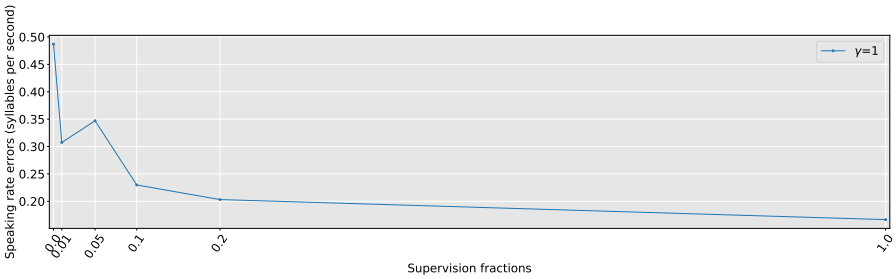
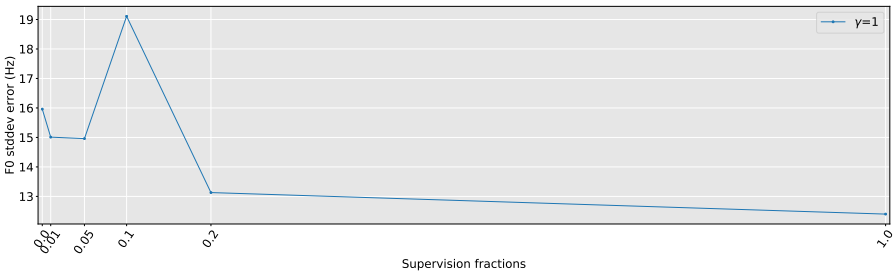


Table 4.6: Sample spectrogram and F0 track plots, generated by varying the speaking rate (first column) and F0 variation (second column). We use strand normal prior for these factors and this table demos varying the control factor from -5σ to 5σ, demonstrating the controllability, interpolation and extrapolation of conditional generation, and also disentanglement of these factors.



(a) Speaking-rate error as a function of supervision level.



(b) F0 variation error as a function of supervision level

Figure 4.7: Objective controllability of speaking rate and F0 variation evaluation metrics presented at multiple supervision levels, on LibriTTS [Zen et al., 2019] datasets. 100% supervision corresponds to 62 hours of supervised data.

Chapter 5

General Conclusions

The work in this thesis has presented two distinct contributions to probabilistic deep learning. In chapter 3, we introduced a novel framework for combining MCMC and Variational Inference that makes use of the auxiliary variational method to capture low-dimensional latent structure. We explored a particular instance of this framework which used neural networks to parameterise adaptive proposal distributions and demonstrated that it can be used to create a fast mixing sampler. Our methods are competitive with other recent geometry-learning approaches [Strathmann et al., 2015, Song et al., 2017] and open up additional avenues for exploring how to combine the best of variational inference and sampling. In chapter 4, we demonstrated that taking a probabilistic view of neural TTS methods [Wang et al., 2017, 2018, Shen et al., 2018] allows us to take steps towards controllable speech synthesis. By augmenting a neural TTS pipeline with stochastic latent variables we demonstrated a method that was able to control latent aspects of prosody with as little as 3 minutes of supervision whilst achieving state of the art synthesis quality.

The work in chapter 3 fits into a larger effort within the research community to produce approximate inference methods with the speed and flexibility of Variational Inference but the asymptotic guarantees of MCMC [Ranganath et al., 2016, De Freitas et al., 2001]. It was also amongst the first of a small number of approaches demonstrating that neural methods could be effectively used to produce adaptive

MCMC samplers [Levy et al., 2018, Song et al., 2017]. Since its publication there have been many works building in a similar direction. For example, Hoffman et al. [2019] also use variational methods to learn structure and guide a sampler. They train a bijective mapping parameterised by inverse auto-regressive flows [Kingma et al., 2016] between a normal distribution and their target density and perform HMC in the transformed space of their variational approximation. The method is conceptually very similar to auxiliary variational sampling but rather than use a low dimensional latent they use a square system and learn a better geometry for HMC. Francesco Ruiz and Michalis Titsias have also produced a series of papers attempting to marry variational inference and MCMC in similar ways [Ruiz and Titsias, 2019, Titsias and Dellaportas, 2019, Dellaportas and Titsias, 2019].

Possible avenues for future work would be to investigate iterative improvement of our variational approximation with samples drawn from the true distribution rather than our variational approximation. Since the original publication of our method, there has been enormous progress in the expressiveness of generative neural models [Bond-Taylor et al., 2021] and another natural direction for future work would be to incorporate newer network architectures into our MCMC framework.

The work on TTS presented in chapter 4 was amongst the first papers to augment neural TTS models with probabilistic latent variables [Hsu et al., 2018, Battenberg et al., 2019] but is part of a much wider effort to produce speech synthesisers that are both high quality and controllable [Wang et al., 2018, Akuzawa et al., 2018]. Since its publication, the work has been cited numerous times and many more papers focused on controllablity or style transfer have used probabilistic methods when previously heuristic approaches dominated [Sun et al., 2020, Morrison et al., 2020].

A natural direction for further work would be to increase the expressivity of the conditional likelihood in order to reduce model mis-specification. This could be done by learning the variance of the Laplace-distribution we currently use or by parameterising more expressive output distributions that do not assume conditional independence across spectrogram channels. We conjecture that with more expres-

sive output distributions, it may be possible to reduce the dependence on ad-hoc parameters that over emphasise the supervised data.

Beyond the direct contributions of the work presented here, it is also my hope that this thesis provides evidence of the potential benefits of combining probabilistic methods with deep learning and will inspire more work at the intersection of these fields.

Bibliography

M. Abadi and A. Agarwal. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

F. V. Agakov and D. Barber. An auxiliary variational method. *International Conference on Neural Information Processing*, pages 561–566, 2004.

K. Akuzawa, Y. Iwasawa, and Y. Matsuo. Expressive speech synthesis via modeling expressions with variational autoencoder. *arXiv preprint arXiv:1804.02135*, 2018.

C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and computing*, pages 343–373, 2008.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1):5–43, 2003.

S. Ö. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, and J. Raiman. Deep voice: Real-time neural text-to-speech. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

E. Battenberg, S. Mariooryad, D. Stanton, R. Skerry-Ryan, M. Shannon, D. Kao, and T. Bagby. Effective use of variational embedding capacity in expressive end-to-end speech synthesis. *arXiv preprint arXiv:1906.03402*, 2019.

M. J. Beal. *Variational algorithms for approximate Bayesian inference*. University of London, University College London (United Kingdom), 2003.

S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *arXiv preprint arXiv:2103.04922*, 2021.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

S. Brooks, A. Gelman, G. Jones, and M. Xiao-Li. *Handbook of Markov chain Monte Carlo*. CRC press, 2011.

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

A. De Cheveigné and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, pages 1917–1930, 2002.

N. De Freitas, P. Højen-Sørensen, M. I. Jordan, and S. Russell. Variational MCMC. *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 120–127, 2001.

P. Dellaportas and M. Titsias. Gradient-based adaptive markov chain monte carlo. 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019.

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 (Methodological)*, 39(1):1–22, 1977.

L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation. *International Conference in Learning Representations*, 2015.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *International Conference on Learning Representations*, 2016.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 1987.

C. W. Fox and S. J. Roberts. A tutorial on variational bayesian inference. *Artificial intelligence review*, 38(2):85–95, 2012.

Y. Gal. *Uncertainty in deep learning*. PhD thesis, 2016.

D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo*. Chapman & Hall, 2006.

A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 1992.

Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In *Advances in neural information processing systems*, pages 2962–2970, 2017.

M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 123–214, 2011.

I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

A. Graves. Practical variational inference for neural networks. volume 24, 2011.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, December 1995.

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

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

M. Hoffman, P. Sountsov, J. V. Dillon, I. Langmore, D. Tran, and S. Vasudevan. Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport. *arXiv preprint arXiv:1903.03704*, 2019.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 2014.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *CoRR*, 2011.

W. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, et al. Hierarchical generative modeling for controllable speech synthesis. *International Conference On Learning Representations*, 2018.

A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 1999.

M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling. Diva: Domain invariant variational autoencoders. *arXiv preprint arXiv:1905.10427*, 2019.

E. Jaynes. *Probability theory: the logic of science*. Washington University St. Louis, MO, 1996.

W. D. Jennings. *The Emulation Game: Modelling and Machine Learning for the Epoch of Reionization*. PhD thesis, UCL (University College London), 2019.

N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Van den Oord, S. Dieleman, and K. Kavukcuoglu. Efficient neural audio synthesis. *International Conference on Machine Learning*, 2018.

H. Kim and A. Mnih. Disentangling by factorising. *International Conference on Machine Learning*, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *International Conference for Learning Representations*, 2014.

D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

W. Kirsch. An elementary proof of De Finetti's theorem. *Statistics & Probability Letters*, 151:84–88, 2019.

G. Koop, D. Korobilis, et al. Bayesian multivariate time series methods for empirical macroeconomics. *Foundations and Trends in Econometrics*, 3(4):267–358, 2010.

R. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, 1993.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

D. Levy, M. D. Hoffman, and J. Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. *International Conference on Learning Representations*, 2018.

Y.-A. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient MCMC. *Advances in neural information processing systems*, 28, 2015.

M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

T. Minka. Divergence measures and message passing. Technical report, Microsoft Research Ltd, 2005.

M. Morrison, Z. Jin, J. Salamon, N. J. Bryan, and G. J. Mysore. Controllable neural prosody synthesis. *arXiv preprint arXiv:2008.03388*, 2020.

E. Munoz-de Escalona and J. J. Canas. Online measuring of available resources. Technical report, 2017.

S. Narayanaswamy, B. T. Paige, J. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems*, pages 5925–5935, 2017.

R. Neal. https://radfordneal.wordpress.com/2012/01/21/no-u-turns-for-hamiltonian-monte-carlo-comments-on-a-paper-by-hoffman-and-gelman/.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

R. M. Neal et al. MCMC using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

J. Paisley, D. M. Blei, and M. I. Jordan. Variational bayesian inference with stochastic search. International Converence on Machine Learning, 2012.

W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv preprint arXiv:1710.07654*, 2017.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. *Artificial Intelligence and Statistics*, pages 814–822, 2014.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333, 2016.

Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T. Liu. Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*, 2019.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *International Conference on Machine Learning*, 2014.

F. Ruiz and M. Titsias. A contrastive divergence for combining variational inference and MCMC. 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 5537–5545, 2019.

J. A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 1980.

D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, et al. A tutorial on thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

T. Salimans, D. Kingma, and M. Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. *International Conference on Machine Learning*, pages 1218–1226, 2015.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *International Conference on Learning Representations*, 2017.

M. Schröder. Emotional speech synthesis: A review. In *Seventh European Conference on Speech Communication and Technology*, 2001.

B. Settles. Active learning literature survey. 2009.

J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, and R. Skerrv-Ryan. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

R. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. Weiss, R. Clark, and R. A. Saurous. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

J. Song, S. Zhao, and S. Ermon. A-NICE-MC: Adversarial training for MCMC. *Advances in Neural Information Processing Systems 30*, pages 5140–5150, 2017.

J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio. Char2wav: End-to-end speech synthesis. 2017.

H. Strathmann, D. Sejdinovic, S. Livingstone, Z. Szabó, and A. Gretton. Gradient-free Hamiltonian Monte Carlo with efficient kernel exponential families. *Advances in Neural Information Processing Systems*, 2015.

G. Sun, Y. Zhang, R. J. Weiss, Y. Cao, H. Zen, and Y. Wu. Fully-hierarchical fine-grained prosody modeling for interpretable speech synthesis. In *ICASSP 2020-2020 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6264–6268. IEEE, 2020.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Y. Taigman, L. Wolf, A. Polyak, and E. Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv preprint arXiv:1707.06588*, 2017.

P. Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009.

M. B. Thompson. A comparison of methods for computing autocorrelation time. *arXiv preprint arXiv:1011.0175*, 2010.

M. Titsias and P. Dellaportas. Gradient-based adaptive markov chain monte carlo. *Advances in Neural Information Processing Systems*, 32, 2019.

G. Tucker, A. Mnih, C. J. Maddison, J. Lawson, and J. Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

S. Vasquez and M. Lewis. Melnet: A generative model for audio in the frequency domain. *arXiv preprint arXiv:1906.01083*, 2019.

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

V. Wan, C. Chan, T. Kenter, J. Vit, and R. Clark. Chive: Varying prosody in speech synthesis with a linguistically driven dynamic hierarchical conditional variational network. *arXiv preprint arXiv:1905.07195*, 2019.

Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. *arXiv preprint arXiv:1803.09017*, 2018.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

P. Wu, Z. Ling, L. Liu, Y. Jiang, H. Wu, and L. Dai. End-to-end emotional speech synthesis using style tokens and semi-supervised training. *arXiv preprint arXiv:1906.10859*, 2019.

H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.