IMPROVING SAMPLING FROM GENERATIVE AUTOENCODERS WITH MARKOV CHAINS

Antonia Creswell, Kai Arulkumaran & Anil A. Bharath
Department of Bioengineering
Imperial College London
London SW7 2BP, UK
{ac2211,ka709,aab01}@ic.ac.uk

ABSTRACT

We focus on generative autoencoders, such as variational or adversarial autoencoders, which jointly learn a generative model alongside an inference model. Generative autoencoders are those which are trained to softly enforce a prior on the latent distribution learned by the inference model. We call the distribution to which the inference model maps observed samples, the *learned latent distribution*, which may not be consistent with the prior. We formulate a Markov chain Monte Carlo (MCMC) sampling process, equivalent to iteratively decoding and encoding, which allows us to sample from the learned latent distribution. Since, the generative model learns to map from the *learned* latent distribution, rather than the prior, we may use MCMC to improve the quality of samples drawn from the **generative** model, especially when the learned latent distribution is far from the prior. Using MCMC sampling, we are able to reveal previously unseen differences between generative autoencoders trained either with or without a denoising criterion.

1 INTRODUCTION

Unsupervised learning has benefited greatly from the introduction of deep generative models. In particular, the introduction of generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma & Welling, 2014; Rezende et al., 2014) has led to a plethora of research into learning latent variable models that are capable of generating data from complex distributions, including the space of natural images (Radford et al., 2015). Both of these models, and their extensions, operate by placing a prior distribution, $P(Z)$, over a latent space $Z \subseteq \mathbb{R}^b$, and learn mappings from the latent space, Z, to the space of the observed data, $X \subseteq \mathbb{R}^a$.

We are interested in autoencoding generative models, models which learn not just the generative mapping $Z \mapsto X$, but also the inferential mapping $X \mapsto Z$. Specifically, we define *generative autoencoders* as autoencoders which softly constrain their latent distribution, to match a specified prior distribution, $P(Z)$. This is achieved by minimising a loss, $\mathcal{L}_{prior}$, between the latent distribution and the prior. This includes VAEs (Kingma & Welling, 2014; Rezende et al., 2014), extensions of VAEs (Kingma et al., 2016), and also adversarial autoencoders (AAEs) (Makhzani et al., 2015). Whilst other autoencoders also learn an encoding function, $e : \mathbb{R}^a \to Z$, together with a decoding function, $d : \mathbb{R}^b \to X$, the latent space is not necessarily constrained to conform to a specified probability distribution. This is the key distinction for generative autoencoders; both e and d can still be deterministic functions (Makhzani et al., 2015).

The functions e and d are defined for any input from $\mathbb{R}^a$ and $\mathbb{R}^b$ respectively, however the outputs of the functions may be constrained practically by the type of functions that e and d are, such that e maps to $Z \subseteq \mathbb{R}^b$ and d maps to $X \subseteq \mathbb{R}^a$. During training however, the encoder, e is only fed with training data samples, $\mathbf{x} \in X$ and the decoder, d is only fed with samples from the encoder, $\mathbf{z} \in Z$, and so the encoder and decoder learn mappings between X and Z.

The process of encoding and decoding may be interpreted as sampling the conditional probabilities $Q_\phi(Z|X)$ and $P_\theta(X|Z)$ respectively. The conditional distributions may be sampled using the encoding and decoding functions $e(X;\phi)$ and $d(Z;\theta)$, where ϕ and θ are learned parameters of the

encoding and decoding functions respectively. The decoder of a generative autoencoder may be used to generate new samples that are consistent with the data. There are two traditional approaches for sampling generative autoencoders:

Approach 1 (Bengio et al., 2014):

$$\mathbf{x}_0 \sim P(X), \quad \mathbf{z}_0 \sim Q_\phi(Z|X = \mathbf{x}_0), \quad \mathbf{x}_1 \sim P_\theta(X|Z = \mathbf{z}_0)$$

where $P(X)$ is the data generating distribution. However, this approach is likely to generate samples similar to those in the training data, rather than generating novel samples that are consistent with the training data.

Approach 2 (Kingma & Welling, 2014; Makhzani et al., 2015; Rezende et al., 2014):

$$\mathbf{z_0} \sim P(Z), \quad \mathbf{x_0} \sim P_\theta(X|Z = \mathbf{z}_0)$$

where $P(Z)$ is the prior distribution enforced during training and $P_\theta(X|Z)$ is the decoder trained to map samples drawn from $Q_\phi(Z|X)$ to samples consistent with $P(X)$. This approach assumes that $\int Q_\phi(Z|X)P(X)dX = P(Z)$, suggesting that the encoder maps all data samples from $P(X)$ to a distribution that matches the prior distribution, $P(Z)$. However, it is not always true that $\int Q_\phi(Z|X)P(X)dX = P(Z)$. Rather $Q_\phi(Z|X)$ maps data samples to a distribution which we call, $\hat{P}(Z)$:

$$\int Q_\phi(Z|X)P(X)dX = \hat{P}(Z)$$

where it is not necessarily true that $\hat{P}(Z) = P(Z)$ because the prior is only softly enforced. The decoder, on the other hand, is trained to map encoded data samples (i.e. samples from $\int Q_\phi(Z|X)P(X)dX$) to samples from X which have the distribution $P(X)$. If the encoder maps observed samples to latent samples with the distribution $\hat{P}(Z)$, rather than the desired prior distribution, $P(Z)$, then:

$$\int P_\theta(X|Z)P(Z)dZ \neq P(X)$$

This suggests that samples drawn from the decoder, $P_\theta(X|Z)$, conditioned on samples drawn from the prior, $P(Z)$, may not be consistent with the data generating distribution, $P(X)$. However, by conditioning on $\hat{P}(Z)$:

$$\int P_\theta(X|Z)\hat{P}(Z)dZ = P(X)$$

This suggests that to obtain more realistic generations, latent samples should be drawn via $\mathbf{z} \sim \hat{P}(Z)$ rather than $\mathbf{z} \sim P(Z)$, followed by $\mathbf{x} \sim P_\theta(X|Z)$. A limited number of latent samples may be drawn from $\hat{P}(Z)$ using the first two steps in Approach 1 - however this has the drawbacks discussed in Approach 1. We introduce an alternative method for sampling from $\hat{P}(Z)$ which does not have the same drawbacks.

Our main contribution is the formulation of a Markov chain Monte Carlo (MCMC) sampling process for generative autoencoders, which allows us to sample from $\hat{P}(Z)$. By iteratively sampling the chain, starting from an arbitrary $\mathbf{z}_{t=0} \in \mathbb{R}^b$, the chain converges to $\mathbf{z}_{t\to\infty} \sim \hat{P}(Z)$, allowing us to draw latent samples from $\hat{P}(Z)$ after several steps of MCMC sampling. From a practical perspective, this is achieved by iteratively decoding and encoding, which may be easily applied to existing generative autoencoders. Because $\hat{P}(Z)$ is optimised to be close to $P(Z)$, the initial sample, $\mathbf{z}_{t=0}$ can be drawn from $P(Z)$, improving the quality of the samples within a few iterations.

When interpolating between latent encodings, there is no guarantee that $\mathbf{z}$ stays within high density regions of $\hat{P}(Z)$. Previously, this has been addressed by using spherical, rather than linear interpolation of the high dimensional Z space (White, 2016). However, this approach attempts to keep $\mathbf{z}$

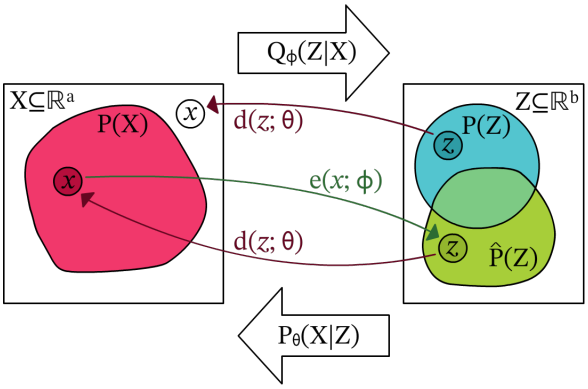
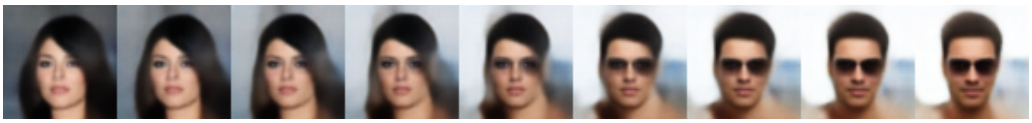
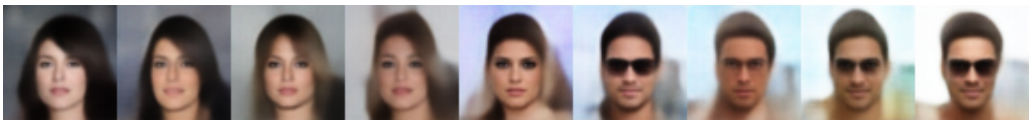


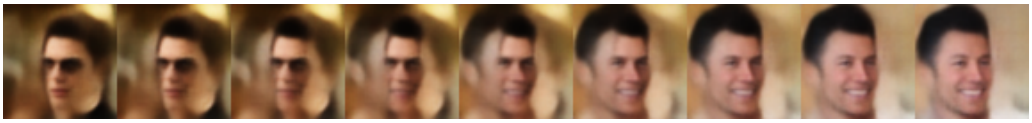
Figure 1: $P(X)$ is the data generating distribution. We may access some samples from $P(X)$ by drawing samples from the training data. $Q_\phi(Z|X)$ is the conditional distribution, modeled by an encoder, which maps samples from $\mathbb{R}^a$ to samples in $\mathbb{R}^b$. An ideal encoder maps samples from $P(X)$ to a known, prior distribution $P(Z)$: in reality the encoder maps samples from $P(X)$ to an unknown distribution $\hat{P}(Z)$. $P_\theta(X|Z)$ is a conditional distribution, modeled by a decoder, which maps samples from $\mathbb{R}^b$ to $\mathbb{R}^a$. During training the decoder learns to map samples drawn from $\hat{P}(Z)$ to $P(X)$ rather than samples drawn from $P(Z)$ because the decoder only sees samples from $\hat{P}(Z)$. Regularisation on the latent space only encourages $\hat{P}(Z)$ to be close to $P(Z)$. Note that if $\mathcal{L}_{prior}$ is optimal, then $\hat{P}(Z)$ overlaps fully with $P(Z)$.



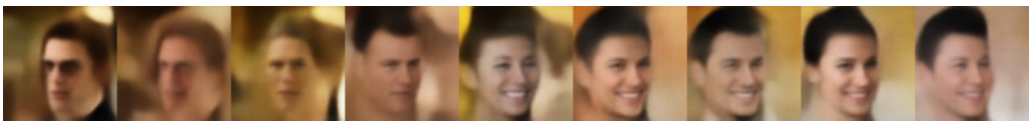
(a) VAE (initial)



(b) VAE (5 steps)



(c) VAE (initial)



(d) VAE (5 steps)

Figure 2: **Prior work:** Spherically interpolating (White, 2016) between two faces using a VAE (a, c). In (a), the attempt to gradually generate sunglasses results in visual artifacts around the eyes. In (c), the model fails to properly capture the desired change in orientation of the face, resulting in three partial faces in the middle of the interpolation. **This work:** (b) and (d) are the result of 5 steps of MCMC sampling applied to the latent samples that were used to generate the original interpolations, (a) and (c). In (b), the discolouration around the eyes disappears, with the model settling on either generating or not generating glasses. In (d), the model moves away from multiple faces in the interpolation by producing new faces with appropriate orientations.

within $P(Z)$, rather than trying to sample from $\hat{P}(Z)$. By instead applying several steps of MCMC sampling to the interpolated $\mathbf{z}$ samples before sampling $P_\theta(X|Z)$, unrealistic artifacts can be reduced (see Figure 2). Whilst most methods that aim to generate realistic samples from X rely on adjusting encodings of the observed data (White, 2016), our use of MCMC allows us to walk any latent sample to more probable regions of the learned latent distribution, resulting in more convincing generations. We demonstrate that the use of MCMC sampling improves generations from both VAEs and AAEs with high-dimensional Z; this is important as previous studies have shown that the dimensionality of Z should be scaled with the intrinsic latent dimensionality of the observed data.

Our second contribution is the modification of the proposed transition operator for the MCMC sampling process to denoising generative autoencoders. These are generative autoencoders trained using a denoising criterion, (Seung, 1997; Vincent et al., 2008). We reformulate our original MCMC sampling process to incorporate the noising and denoising processes, allowing us to use MCMC sampling on denoising generative autoencoders. We apply this sampling technique to two models. The first is the denoising VAE (DVAE) introduced by Im et al. (2015). We found that MCMC sampling revealed benefits of the denoising criterion. The second model is a denoising AAE (DAAE), constructed by applying the denoising criterion to the AAE. There were no modifications to the cost function. For both the DVAE and the DAAE, the effects of the denoising crtierion were not immediately obvious from the initial samples. Training generative autoencoders with a denoising criterion reduced visual artefacts found both in generations and in interpolations. The effect of the denoising criterion was revealed when sampling the denoising models using MCMC sampling.

2 BACKGROUND

One of the main tasks in machine learning is to learn explanatory factors for observed data, commonly known as inference. That is, given a data sample $\mathbf{x} \in X \subseteq \mathbb{R}^a$, we would like to find a corresponding latent encoding $\mathbf{z} \in Z \subseteq \mathbb{R}^b$. Another task is to learn the inverse, generative mapping from a given $\mathbf{z}$ to a corresponding $\mathbf{x}$. In general, coming up with a suitable criterion for learning these mappings is difficult. Autoencoders solve both tasks efficiently by jointly learning an inferential mapping $e(X;\phi)$ and generative mapping $d(Z;\theta)$, using unlabelled data from X in a self-supervised fashion (Kingma & Welling, 2014). The basic objective of all autoencoders is to minimise a reconstruction cost, $\mathcal{L}_{reconstruct}$, between the original data, X, and its reconstruction, $d(e(X;\phi);\theta)$. Examples of $\mathcal{L}_{reconstruct}$ include the squared error loss, $\frac{1}{2}\sum_{n=1}^{N} \|d(e(\mathbf{x}_n;\phi);\theta) - \mathbf{x}_n\|^2$, and the cross-entropy loss, $\mathcal{H}[P(X)\|P(d(e(X;\phi);\theta))] = -\sum_{n=1}^{N} \mathbf{x}_n \log(d(e(\mathbf{x}_n;\phi);\theta)) + (1-\mathbf{x}_n)\log(1-d(e(\mathbf{x}_n;\phi);\theta))$.

Autoencoders may be cast into a probablistic framework, by considering samples $\mathbf{x} \sim P(X)$ and $\mathbf{z} \sim P(Z)$, and attempting to learn the conditional distributions $Q_\phi(Z|X)$ and $P_\theta(X|Z)$ as $e(X;\phi)$ and $d(Z;\theta)$ respectively, with $\mathcal{L}_{reconstruct}$ representing the negative log-likelihood of the reconstruction given the encoding (Bengio, 2009). With any autoencoder, it is possible to create novel $\mathbf{x} \in X$ by passing a $\mathbf{z} \in Z$ through $d(Z;\theta)$, but we have no knowledge of appropriate choices of $\mathbf{z}$ beyond those obtained via $e(X;\phi)$. One solution is to constrain the latent space to which the encoding model maps observed samples. This can be achieved by an additional loss, $\mathcal{L}_{prior}$, that penalises encodings far away from a specified prior distribution, $P(Z)$. We now review two types of generative autoencoders, VAEs (Kingma & Welling, 2014; Rezende et al., 2014) and AAEs (Makhzani et al., 2015), which each take different approaches to formulating $\mathcal{L}_{prior}$.

2.1 GENERATIVE AUTOENCODERS

Consider the case where e is constructed with stochastic neurons that can produce outputs from a specified probability distribution, and $\mathcal{L}_{prior}$ is used to constrain the distribution of outputs to $P(Z)$. This leaves the problem of estimating the gradient of the autoencoder over the expectation $\mathbb{E}_{Q_\phi(Z|X)}$, which would typically be addressed with a Monte Carlo method. VAEs sidestep this by constructing latent samples using a deterministic function and a source of noise, moving the source of stochasticity to an input, and leaving the network itself deterministic for standard gradient calculations—a technique commonly known as the reparameterisation trick (Kingma & Welling, 2014). $e(X;\phi)$ then consists of a deterministic function, $e_{rep}(X;\phi)$, that outputs parameters for a probability distribution, plus a source of noise. In the case where $P(Z)$ is a diagonal covariance Gaussian, $e_{rep}(X;\phi)$

Figure 3: Reconstructions of faces from a DVAE trained with additive Gaussian noise: $Q(\tilde{X}|X) = \mathcal{N}(X, 0.25\mathbf{I})$. The model successfully recovers much of the detail from the noise-corrupted images.

maps $\mathbf{x}$ to a vector of means, $\boldsymbol{\mu} \in \mathbb{R}^b$, and a vector of standard deviations, $\boldsymbol{\sigma} \in \mathbb{R}^b_+$, with the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Put together, the encoder outputs samples $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$, where $\odot$ is the Hadamard product. VAEs attempt to make these samples from the encoder match up with $P(Z)$ by using the KL divergence between the parameters for a probability distribution outputted by $e_{rep}(X; \phi)$, and the parameters for the prior distribution, giving $\mathcal{L}_{prior} = D_{KL}[Q_\phi(Z|X) \| P(Z)]$. A multivariate Gaussian has an analytical KL divergence that can be further simplified when considering the unit Gaussian, resulting in $\mathcal{L}_{prior} = \frac{1}{2} \sum_{n=1}^{N} \boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - \mathbf{1}$.

Another approach is to deterministically output the encodings $\mathbf{z}$. Rather than minimising a metric between probability distributions using their parameters, we can turn this into a density ratio estimation problem where the goal is to learn a conditional distribution, $Q_\phi(Z|X)$, such that the distribution of the encoded data samples, $\hat{P}(Z) = \int Q_\phi(Z|X)P(X)dX$, matches the prior distribution, $P(Z)$. The GAN framework solves this density ratio estimation problem by transforming it into a class estimation problem using two networks (Goodfellow et al., 2014). The first network in GAN training is the discriminator network, D_ψ, which is trained to maximise the log probability of samples from the "real" distribution, $\mathbf{z} \sim P(Z)$, and minimise the log probability of samples from the "fake" distribution, $\mathbf{z} \sim Q_\phi(Z|X)$. In our case $e(X; \phi)$ plays the role of the second network, the generator network, G_ϕ, which generates the "fake" samples.[1] The two networks compete in a minimax game, where G_ϕ receives gradients from D_ψ such that it learns to better fool D_ψ. The training objective for both networks is given by $\mathcal{L}_{prior} = \operatorname{argmin}_\phi \operatorname{argmax}_\psi \mathbb{E}_{P(Z)}[\log(D_\psi(Z))] + \mathbb{E}_{P(X)}[\log(1 - D_\psi(G_\phi(X)))] = \operatorname{argmin}_\phi \operatorname{argmax}_\psi \mathbb{E}_{P(Z)}[\log(D_\psi(Z))] + \mathbb{E}_{Q_\phi(Z|X)P(X)} \log[1 - D_\psi(Z)]$. This formulation can create problems during training, so instead G_ϕ is trained to minimise $-\log(D_\psi(G_\phi(X)))$, which provides the same fixed point of the dynamics of G_ϕ and D_ψ. The result of applying the GAN framework to the encoder of an autoencoder is the deterministic AAE (Makhzani et al., 2015).

2.2 Denoising autoencoders

In a more general viewpoint, generative autoencoders fulfill the purpose of learning useful representations of the observed data. Another widely used class of autoencoders that achieve this are denoising autoencoders (DAEs), which are motivated by the idea that learned features should be robust to "partial destruction of the input" (Vincent et al., 2008). Not only does this require encoding the inputs, but capturing the statistical dependencies between the inputs so that corrupted data can be recovered (see Figure 3). DAEs are presented with a corrupted version of the input, $\tilde{\mathbf{x}} \in \tilde{X}$, but must still reconstruct the original input, $\mathbf{x} \in X$, where the noisy inputs are created through sampling $\tilde{\mathbf{x}} \sim C(\tilde{X}|X)$, a corruption process. The denoising criterion, $\mathcal{L}_{denoise}$, can be applied to any type of autoencoder by replacing the straightforward reconstruction criterion, $\mathcal{L}_{reconstruct}(X, d(e(X; \phi); \theta))$, with the reconstruction criterion applied to noisy inputs: $\mathcal{L}_{reconstruct}(X, d(e(\tilde{X}; \phi); \theta))$. The encoder is now used to model samples drawn from $Q_\phi(Z|\tilde{X})$. As such, we can construct *denoising generative autoencoders* by training autoencoders to minimise $\mathcal{L}_{denoise} + \mathcal{L}_{prior}$.

One might expect to see differences in samples drawn from denoising generative autoencoders and their non-denoising counterparts. However, Figures 4 and 6 show that this is not the case. Im et al.

[1] We adapt the variables to better fit the conventions used in the context of autoencoders.

(2015) address the case of DVAEs, claiming that the noise mapping requires adjusting the original VAE objective function. Our work is orthogonal to theirs, and others which adjust the training or model (Kingma et al., 2016), as we focus purely on sampling from generative autoencoders after training. We claim that the existing practice of drawing samples from generative autoencoders conditioned on $\mathbf{z} \sim P(Z)$ is suboptimal, and the quality of samples can be improved by instead conditioning on $\mathbf{z} \sim \hat{P}(Z)$ via MCMC sampling.

3 MARKOV SAMPLING

We now consider the case of sampling from generative autoencoders, where $d(Z;\theta)$ is used to draw samples from $P_\theta(X|Z)$. In Section 1, we showed that it was important, when sampling $P_\theta(X|Z)$, to condition on $\mathbf{z}$'s drawn from $\hat{P}(Z)$, rather than $P(Z)$ as is often done in practice. However, we now show that for any initial $\mathbf{z}_0 \in Z_0 = \mathbb{R}^b$, Markov sampling can be used to produce a chain of samples $\mathbf{z}_t$, such that as $t \to \infty$, produces samples $\mathbf{z}_t$ that are from the distribution $\hat{P}(Z)$, which may be used to draw meaningful samples from $P_\theta(X|Z)$, conditioned on $\mathbf{z} \sim \hat{P}(Z)$. To speed up convergence we can initialise $\mathbf{z}_0$ from a distribution close to $\hat{P}(Z)$, by drawing $\mathbf{z}_0 \sim P(Z)$.

3.1 MARKOV SAMPLING PROCESS

A generative autoencoder can be sampled by the following process:

$$\mathbf{z}_0 \in Z_0 = \mathbb{R}^b, \quad \mathbf{x}_{t+1} \sim P_\theta(X|Z_t), \quad \mathbf{z}_{t+1} \sim Q_\phi(Z|X_{t+1})$$

This allows us to define a Markov chain with the transition operator

$$T(Z_{t+1}|Z_t) = \int Q_\phi(Z_{t+1}|X) P_\theta(X|Z_t) dX \tag{1}$$

for $t \geq 0$.

Drawing samples according to the transition operator $T(Z_{t+1}|Z_t)$ produces a Markov chain. For the transition operator to be homogeneous, the parameters of the encoding and decoding functions are fixed during sampling.

3.2 CONVERGENCE PROPERTIES

We now show that the stationary distribution of sampling from the Markov chain is $\hat{P}(Z)$.

Theorem 1. *If $T(Z_{t+1}|Z_t)$ defines an ergodic Markov chain, $\{Z_1, Z_2 ... Z_t\}$, then the chain will converge to a stationary distribution, $\Pi(Z)$, from any arbitrary initial distribution. The stationary distribution $\Pi(Z) = \hat{P}(Z)$.*

The proof of Theorem 1 can be found in (Rosenthal, 2001).

Lemma 1. *$T(Z_{t+1}|Z_t)$ defines an ergodic Markov chain.*

Proof. For a Markov chain to be ergodic it must be both irreducible (it is possible to get from any state to any other state in a finite number of steps) and aperiodic (it is possible to get from any state to any other state without having to pass through a cycle). To satisfy these requirements, it is more than sufficient to show that $T(Z_{t+1}|Z_t) > 0$, since every $\mathbf{z} \in Z$ would be reachable from every other $\mathbf{z} \in Z$. We show that $P_\theta(X|Z) > 0$ and $Q_\phi(Z|X) > 0$, giving $T(Z_{t+1}|Z_t) > 0$, providing the proof of this in Section A of the supplementary material. □

Lemma 2. *The stationary distribution of the chain defined by $T(Z_{t+1}|Z_t)$ is $\Pi(Z) = \hat{P}(Z)$.*

Proof. For the transition operator defined in Equation (1), the asymptotic distribution to which $T(Z_{t+1}|Z_t)$ converges to is $\hat{P}(Z)$, because $\hat{P}(Z)$ is, by definition, the marginal of the joint distribution $Q_\phi(Z|X)P(X)$, over which the $\mathcal{L}_{prior}$ used to learn the conditional distribution $Q_\phi(Z|X)$. □

Using Lemmas 1 and 2 with Theorem 1, we can say that the Markov chain defined by the transition operator in Equation (1) will produce a Markov chain that converges to the stationary distribution $\Pi(Z) = \hat{P}(Z)$.

3.3 Extension to denoising generative autoencoders

A denoising generative autoencoder can be sampled by the following process:

$$\mathbf{z}_0 \in Z_0 = \mathbb{R}^b, \quad \mathbf{x}_{t+1} \sim P_\theta(X|Z_t), \quad \tilde{\mathbf{x}}_{t+1} \sim C(\tilde{X}|X_{t+1}), \quad \mathbf{z}_{t+1} \sim Q_\phi(Z|\tilde{X}_{t+1}).$$

This allows us to define a Markov chain with the transition operator

$$T(Z_{t+1}|Z_t) = \int Q_\phi(Z_{t+1}|\tilde{X})C(\tilde{X}|X)P_\theta(X|Z_t)dXd\tilde{X} \tag{2}$$

for $t \geq 0$.

The same arguments for the proof of convergence of Equation (1) can be applied to Equation (2).

3.4 Related work

Our work is inspired by that of Bengio et al. (2013); denoising autoencoders are cast into a probabilistic framework, where $P_\theta(X|\tilde{X})$ is the *denoising* (decoder) distribution and $C(\tilde{X}|X)$ is the *corruption* (encoding) distribution. $\tilde{X}$ represents the space of corrupted samples. Bengio et al. (2013) define a transition operator of a Markov chain – using these conditional distributions – whose stationary distribution is $P(X)$ under the assumption that $P_\theta(X|\tilde{X})$ perfectly denoises samples. The chain is initialised with samples from the training data, and used to generate a chain of samples from $P(X)$. This work was generalised to include a corruption process that mapped data samples to latent variables (Bengio et al., 2014), to create a new type of network called Generative Stochastic Networks (GSNs). However in GSNs (Bengio et al., 2014) the latent space is not regularised with a prior.

Our work is similar to several approaches proposed by Bengio et al. (2013; 2014) and Rezende et al. (Rezende et al., 2014). Both Bengio et al. and Rezende et al. define a transition operator in terms of X_t and X_{t-1}. Bengio et al. generate samples with an initial X_0 drawn from the observed data, while Rezende et al. reconstruct samples from an X_0 which is a corrupted version of a data sample. In contrasts to Bengio et al. and Rezende et al., in this work we define the transition operator in terms of Z_{t+1} and Z_t, initialise samples with a Z_0 that is drawn from a prior distribution we can directly sample from, and then sample X_1 conditioned on Z_0. Although the initial samples may be poor, we are likely to generate a novel X_1 on the first step of MCMC sampling, which would not be achieved using Bengio et al.'s or Rezende et al.'s approach. We are able draw initial Z_0 from a prior because we constrain $\hat{P}(Z)$ to be close to a prior distribution $P(Z)$; in Bengio et al. a latent space is either not explicitly modeled (Bengio et al., 2013) or it is not constrained (Bengio et al., 2014).

Further, Rezende et al. (2014) explicitly assume that the distribution of latent samples drawn from $Q_\phi(Z|X)$ matches the prior, $P(Z)$. Instead, we assume that samples drawn from $Q_\phi(Z|X)$ have a distribution $\hat{P}(Z)$ that does not necessarily match the prior, $P(Z)$. We propose an alternative method for sampling $\hat{P}(Z)$ in order to improve the quality of generated image samples. Our motivation is also different to Rezende et al. (2014) since we use sampling to generate improved, novel data samples, while they use sampling to denoise corrupted samples.

3.5 Effect of regularisation method

The choice of $\mathcal{L}_{prior}$ may effect how much improvement can be gained when using MCMC sampling, assuming that the optimisation process converges to a reasonable solution. We first consider the case of VAEs, which minimise $D_{KL}[Q_\phi(Z|X)\|P(Z)]$. Minimising this KL divergence penalises the model $\hat{P}(Z)$ if it contains samples that are outside the support of the true distribution $P(Z)$, which might mean that $\hat{P}(Z)$ captures only a part of $P(Z)$. This means that when sampling

$P(Z)$, we may draw from a region that is not captured by $\hat{P}(Z)$. This suggests that MCMC sampling can improve samples from trained VAEs by walking them towards denser regions in $\hat{P}(Z)$.

Generally speaking, using the reverse KL divergence during training, $D_{KL}[P(Z)\|Q_\phi(Z|X)]$, penalises the model $Q_\phi(Z|X)$ if $P(Z)$ produces samples that are outside of the support of $\hat{P}(Z)$. By minimising this KL divergence, most samples in $P(Z)$ will likely be in $\hat{P}(Z)$ as well. AAEs, on the other hand are regularised using the JS entropy, given by $\frac{1}{2}D_{KL}[P(Z)\|\frac{1}{2}(P(Z)+Q_\phi(Z|X))]+\frac{1}{2}D_{KL}[Q_\phi(Z|X)\|\frac{1}{2}(P(Z)+Q_\phi(Z|X))]$. Minimising this cost function attempts to find a compromise between the aforementioned extremes. However, this still suggests that some samples from $P(Z)$ may lie outside $\hat{P}(Z)$, and so we expect AAEs to also benefit from MCMC sampling.

4 EXPERIMENTS

4.1 MODELS

We utilise the deep convolutional GAN (DCGAN) (Radford et al., 2015) as a basis for our autoencoder models. Although the recommendations from Radford et al. (2015) are for standard GAN architectures, we adopt them as sensible defaults for an autoencoder, with our encoder mimicking the DCGAN's discriminator, and our decoder mimicking the generator. The encoder uses strided convolutions rather than max-pooling, and the decoder uses fractionally-strided convolutions rather than a fixed upsampling. Each convolutional layer is succeeded by spatial batch normalisation (Ioffe & Szegedy, 2015) and ReLU nonlinearities, except for the top of the decoder which utilises a sigmoid function to constrain the output values between 0 and 1. We minimise the cross-entropy between the original and reconstructed images. Although this results in blurry images in regions which are ambiguous, such as hair detail, we opt not to use extra loss functions that improve the visual quality of generations (Larsen et al., 2015; Dosovitskiy & Brox, 2016; Lamb et al., 2016) to avoid confounding our results.

Although the AAE is capable of approximating complex probabilistic posteriors (Makhzani et al., 2015), we construct ours to output a deterministic $Q_\phi(Z|X)$. As such, the final layer of the encoder part of our AAEs is a convolutional layer that deterministically outputs a latent sample, $\mathbf{z}$. The adversary is a fully-connected network with dropout and leaky ReLU nonlinearities. $e_{rep}(X;\phi)$ of our VAEs have an output of twice the size, which corresponds to the means, $\boldsymbol{\mu}$, and standard deviations, $\boldsymbol{\sigma}$, of a diagonal covariance Gaussian distribution. For all models our prior, $P(Z)$, is a 200D isotropic Gaussian with zero mean and unit variance: $\mathcal{N}(\mathbf{0},\mathbf{I})$.

4.2 DATASETS

Our primary dataset is the (aligned and cropped) CelebA dataset, which consists of 200,000 images of celebrities (Liu et al., 2015). The DCGAN (Radford et al., 2015) was the first generative neural network model to show convincing novel samples from this dataset, and it has been used ever since as a qualitative benchmark due to the amount and quality of samples. In Figures 7 and 8 of the supplementary material, we also include results on the SVHN dataset, which consists of 100,000 images of house numbers extracted from Google Street view images (Netzer et al., 2011).

4.3 TRAINING & EVALUATION

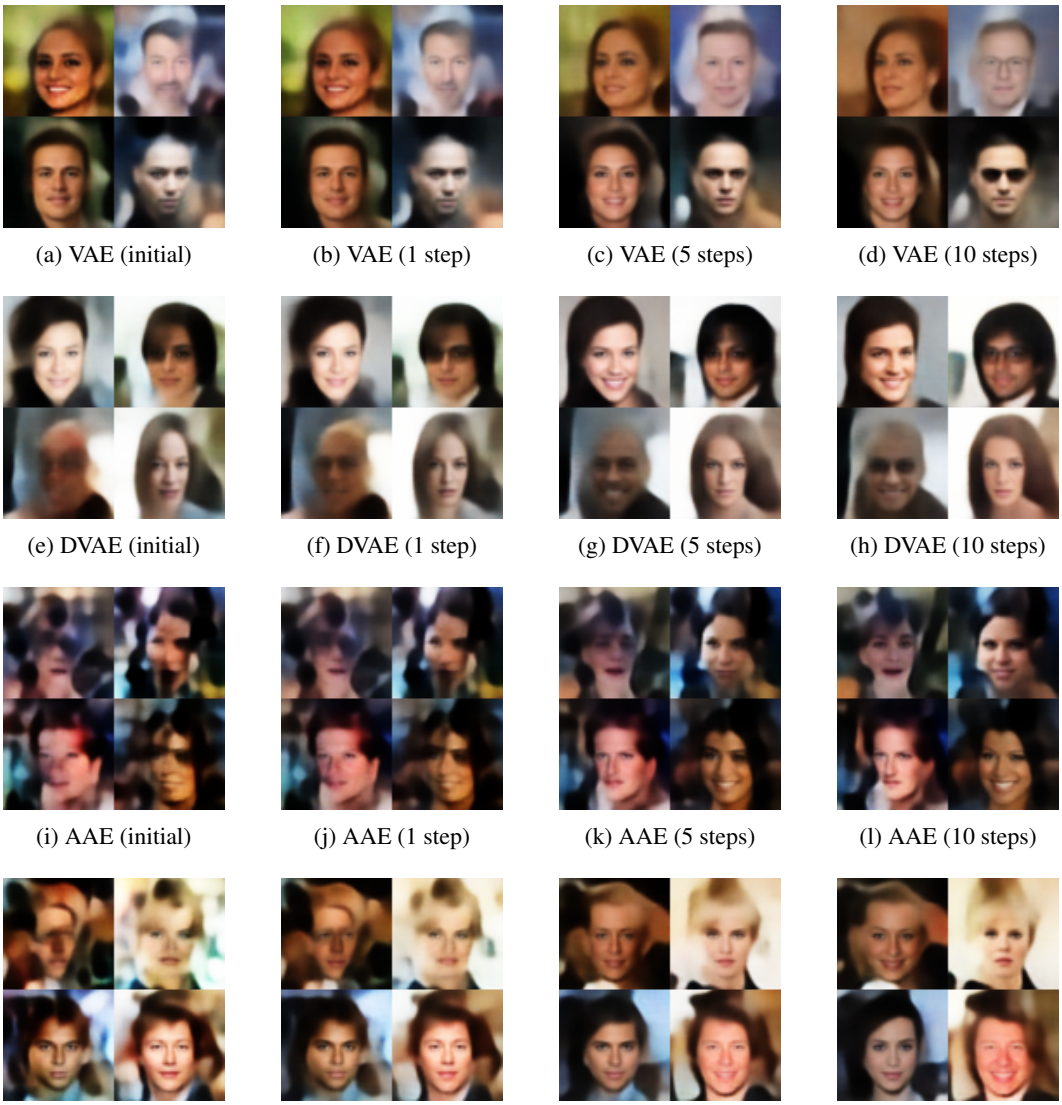
For all datasets we perform the same preprocessing: cropping the centre to create a square image, then resizing to 64×64px. We train our generative autoencoders for 20 epochs on the training split of the datasets, using Adam (Kingma & Ba, 2014) with $\alpha = 0.0002$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The denoising generative autoencoders use the additive Gaussian noise mapping $C(\tilde{X}|X) = \mathcal{N}(X, 0.25\mathbf{I})$. All of our experiments were run using the Torch library (Collobert et al., 2011).[2]

For evaluation, we generate novel samples from the decoder using $\mathbf{z}$ initially sampled from $P(Z)$; we also show spherical interpolations (White, 2016) between four images of the testing split, as depicted in Figure 2. We then perform several steps of MCMC sampling on the novel samples and interpolations. During this process, we use the training mode of batch normalisation (Ioffe &

[2]Example code is available at `https://github.com/Kaixhin/Autoencoders`.

Szegedy, 2015), i.e., we normalise the inputs using minibatch rather than population statistics, as the normalisation can partially compensate for poor initial inputs (see Figure 4) that are far from the training distribution. We compare novel samples between all models below, and leave further interpolation results to Figures 5 and 6 of the supplementary material.

4.4 SAMPLES



(a) VAE (initial) (b) VAE (1 step) (c) VAE (5 steps) (d) VAE (10 steps)

(e) DVAE (initial) (f) DVAE (1 step) (g) DVAE (5 steps) (h) DVAE (10 steps)

(i) AAE (initial) (j) AAE (1 step) (k) AAE (5 steps) (l) AAE (10 steps)

(m) DAAE (initial) (n) DAAE (1 step) (o) DAAE (5 steps) (p) DAAE (10 steps)

Figure 4: Samples from a VAE (a-d), DVAE (e-h), AAE (i-l) and DAAE (m-p) trained on the CelebA dataset. (a), (e), (i) and (m) show initial samples conditioned on $\mathbf{z} \sim P(Z)$, which mainly result in recognisable faces emerging from noisy backgrounds. After 1 step of MCMC sampling, the more unrealistic generations change noticeably, and continue to do so with further steps. On the other hand, realistic generations, i.e. samples from a region with high probability, do not change as much. The adversarial criterion for deterministic AAEs is difficult to optimise when the dimensionality of Z is high. We observe that during training our AAEs and DAAEs, the empirical standard deviation of $\mathbf{z} \sim Q_\phi(Z|X)$ is less than 1, which means that $\hat{P}(Z)$ fails to approximate $P(Z)$ as closely as was achieved with the VAE and DVAE. However, this means that the effect of MCMC sampling is more pronounced, with the quality of all samples noticeably improving after a few steps. As a side-effect of the suboptimal solution learned by the networks, the denoising properties of the DAAE are more noticeable with the novel samples.

5 CONCLUSION

Autoencoders consist of a decoder, $d(Z;\theta)$ and an encoder, $e(X;\phi)$ function, where ϕ and θ are learned parameters. Functions $e(X;\phi)$ and $d(Z;\theta)$ may be used to draw samples from the conditional distributions $P_\theta(X|Z)$ and $Q_\phi(Z|X)$ (Bengio et al., 2014; 2013; Rezende et al., 2014), where X refers to the space of observed samples and Z refers to the space of latent samples. The encoder distribution, $Q_\phi(Z|X)$, maps data samples from the data generating distribution, $P(X)$, to a latent distribution, $\hat{P}(Z)$. The decoder distribution, $P_\theta(X|Z)$, maps samples from $\hat{P}(Z)$ to $P(X)$. We are concerned with *generative autoencoders*, which we define to be a family of autoencoders where regularisation is used during training to encourage $\hat{P}(Z)$ to be close to a known prior $P(Z)$. Commonly it is assumed that $\hat{P}(Z)$ and $P(Z)$ are similar, such that samples from $P(Z)$ may be used to sample a decoder $P_\theta(X|Z)$; we do not make the assumption that $\hat{P}(Z)$ and $P(Z)$ are "sufficiently close" (Rezende et al., 2014). Instead, we derive an MCMC process, whose stationary distribution is $\hat{P}(Z)$, allowing us to directly draw samples from $\hat{P}(Z)$. By conditioning on samples from $\hat{P}(Z)$, samples drawn from $\mathbf{x} \sim P_\theta(X|Z)$ are more consistent with the training data.

In our experiments, we compare samples $\mathbf{x} \sim P_\theta(X|Z = z_0)$, $\mathbf{z}_0 \sim P(Z)$ to $\mathbf{x} \sim P_\theta(X|Z = z_i)$ for $i = \{1, 5, 10\}$, where $\mathbf{z}_i$'s are obtained through MCMC sampling, to show that MCMC sampling improves initially poor samples (see Figure 4). We also show that artifacts in $\mathbf{x}$ samples induced by interpolations across the latent space can also be corrected by MCMC sampling see (Figure 2). We further validate our work by showing that the denoising properties of denoising generative autoencoders are best revealed by the use of MCMC sampling.

Our MCMC sampling process is straightforward, and can be applied easily to existing generative autoencoders. This technique is orthogonal to the use of more powerful posteriors in AAEs (Makhzani et al., 2015) and VAEs (Kingma et al., 2016), and the combination of both could result in further improvements in generative modeling. Finally, our basic MCMC process opens the doors to apply a large existing body of research on sampling methods to generative autoencoders.

ACKNOWLEDGEMENTS

We would like to acknowledge the EPSRC for funding through a Doctoral Training studentship and the support of the EPSRC CDT in Neurotechnology.

REFERENCES

Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1): 1–127, 2009.

Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, and Jason Yosinski. Deep generative stochastic networks trainable by backprop. In *Journal of Machine Learning Research: Proceedings of the 31st International Conference on Machine Learning*, volume 32, 2014.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A *matlab*-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Daniel Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. *arXiv preprint arXiv:1511.06406*, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 448–456, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR-2015), arXiv preprint arXiv:1412.6980*, 2014. URL https://arxiv.org/pdf/1412.6980v8.pdf.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR-2015), arXiv preprint arXiv:1312.6114*, 2014. URL https://arxiv.org/abs/1312.6114.

Diederik P Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Alex Lamb, Vincent Dumoulin, and Aaron Courville. Discriminative regularization for generative models. *arXiv preprint arXiv:1602.03220*, 2016.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of The 33rd International Conference on Machine Learning, arXiv preprint arXiv:1512.09300*, pp. 1558–1566, 2015. URL http://jmlr.org/proceedings/papers/v48/larsen16.pdf.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. URL https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/37648.pdf.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR) 2016, arXiv preprint arXiv:1511.06434*, 2015. URL https://arxiv.org/pdf/1511.06434.pdf.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning, arXiv preprint arXiv:1401.4082*, 2014. URL https://arxiv.org/pdf/1401.4082.pdf.

Jeffrey S Rosenthal. A review of asymptotic convergence for general state space markov chains. *Far East J. Theor. Stat*, 5(1):37–50, 2001.

H Sebastian Seung. Learning continuous attractors in recurrent networks. In *NIPS Proceedings*, volume 97, pp. 654–660, 1997.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1096–1103. ACM, 2008.

Tom White. Sampling generative networks: Notes on a few effective techniques. *arXiv preprint arXiv:1609.04468*, 2016.

Supplementary Material

A PROOF THAT $T(Z_{t+1}|Z_t) > 0$

For $P_\theta(X|Z) > 0$ we require that all possible $\mathbf{x} \in X \subseteq \mathbb{R}^a$ may be generated by the network. Assuming that the model $P_\theta(X|Z)$ is trained using a sufficient number of training samples, $\mathbf{x} \in X_{train} = X$, and that the model has infinite capacity to model $X_{train} = X$, then we should be able to draw any sample $\mathbf{x} \in X_{train} = X$ from $P_\theta(X|Z)$. In reality $X_{train} \subseteq X$ and it is not possible to have a model with infinite capacity. However, $P_\theta(X|Z)$ is modeled using a deep neural network, which we assume has sufficient capacity to capture the training data well. Further, deep neural networks are able to interpolate between samples in very high dimensional spaces (Radford et al., 2015); we therefore further assume that if we have a large number of training samples (as well as large model capacity), that almost any $\mathbf{x} \in X$ can be drawn from $P_\theta(X|Z)$.

Note that if we wish to generate human faces, we define X_{all} to be the space of all possible faces, with distribution $P(X_{all})$, while X_{train} is the space of faces made up by the training data. Then, practically even a well trained model which learns to interpolate well only captures an X, with distribution $\int P_\theta(X|Z)\hat{P}(Z)dZ$, where $X_{train} \subseteq X \subseteq X_{all}$, because X additionally contains examples of interpolated versions of $\mathbf{x} \sim P(X_{train})$.

For $Q_\phi(Z|X) > 0$ it must be possible to generate all possible $\mathbf{z} \in Z \subseteq \mathbb{R}^b$. $Q_\phi(Z|X)$ is described by the function $e(\cdot;\phi) : X \to Z$. To ensure that $Q_\phi(Z|X) > 0$, we want to show that the function $e(X;\phi)$ allows us to represent all samples of $z \in Z$. VAEs and AAEs each construct $e(X;\phi)$ to produce $z \in Z$ in different ways.

The output of the encoder of a VAE, $e_{VAE}(X;\phi)$ is $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The output of a VAE is then always Gaussian, and hence there is no limitation on the $\mathbf{z}$'s that $e_{VAE}(X;\phi)$ can produce. This ensures that $Q_\phi(Z|X) > 0$, provided that $\boldsymbol{\sigma} \neq \mathbf{0}$.

The encoder of our AAE, $e_{AAE}(X;\phi)$, is a deep neural network consisting of multiple convolutional and batch normalisation layers. The final layer of the $e_{AAE}(X;\phi)$ is a fully connected layer without an activation function. The input to each of the M nodes in the fully connected layer is a function $f_{i=1...M}(\mathbf{x})$. This means that $\mathbf{z}$ is given by: $\mathbf{z} = \mathbf{a}_1 f_1(\mathbf{x}) + \mathbf{a}_2 f_2(\mathbf{x}) + ... + \mathbf{a}_M f_M(\mathbf{x})$, where $\mathbf{a}_{i=1...M}$ are the learned weights of the fully connected layer. We now consider three cases:

Case 1: If $\mathbf{a}_i$ are a complete set of bases for Z then it is possible to generate any $\mathbf{z} \in Z$ from an $\mathbf{x} \in X$ with a one-to-one mapping, provided that $f_i(\mathbf{x})$ is not restricted in the values that it can take.

Case 2: If $\mathbf{a}_i$ are an overcomplete set of bases for Z, then the same holds, provided that $f_i(\mathbf{x})$ is not restricted in the values that it can take.

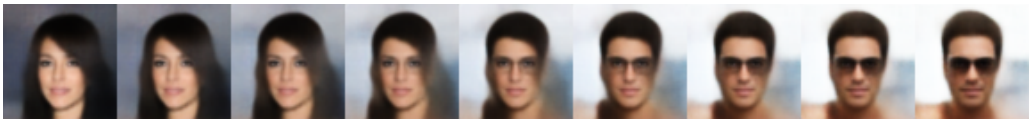
Case 3: If $\mathbf{a}_i$ are an undercomplete set of bases for Z then it is not possible to generate all $\mathbf{z} \in Z$ from $\mathbf{x} \in X$. Instead there is a many (X) to one (Z) mapping.

For $Q_\phi(Z|X) > 0$ our network must learn a complete or overcomplete set of bases and $f_i(x)$ must not be restricted in the values that it can take $\forall i$. The network is encouraged to learn an overcomplete set of bases by learning a large number of $\mathbf{a}_i$'s—specifically $M = 8192$ when basing our network on the DCGAN architecture (Radford et al., 2015)—more that 40 times the dimensionality of Z. By using batch normalisation layers throughout the network, we ensure that values of $f_i(x)$ are spread out, capturing a *close-to-Gaussian* distribution (Ioffe & Szegedy, 2015), encouraging infinite support.

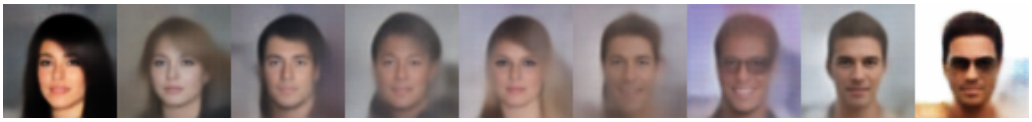
We have now shown that, under certain reasonable assumptions, $P_\theta(X|Z) > 0$ and $Q_\phi(Z|X) > 0$, which means that $T(Z_{t+1}|Z_t) > 0$, and hence we can get from any Z to any another Z in only one step. Therefore the Markov chain described by the transition operator $T(Z_{t+1}|Z_t)$ defined in Equation (1) is both irreducible and aperiodic, which are the necessary conditions for ergodicity.

B CELEBA

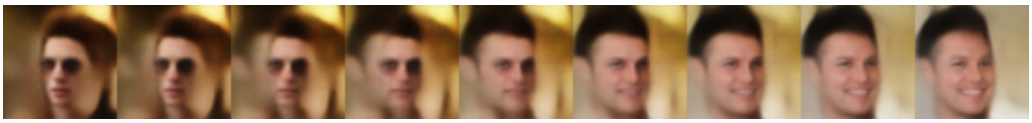
B.1 INTERPOLATIONS



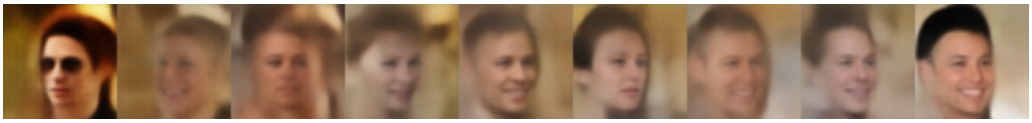
(a) DVAE (initial)



(b) DVAE (5 steps)

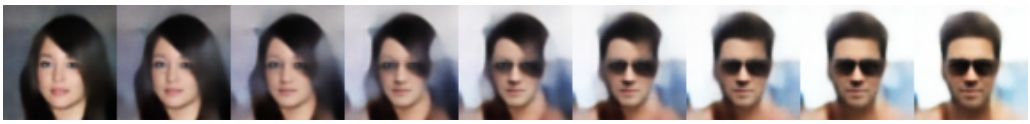


(c) DVAE (initial)

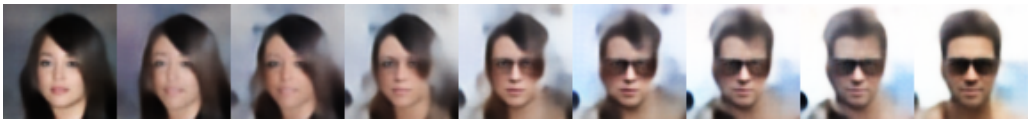


(d) DVAE (5 steps)

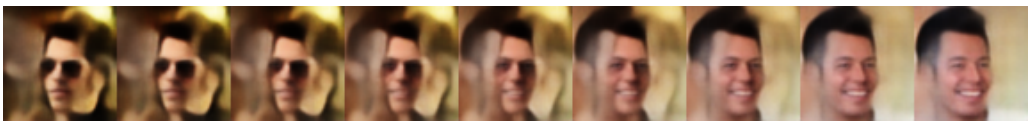
Figure 5: Interpolating between two faces using (a-d) a DVAE. The top rows (a, c) for each face is the original interpolation, whilst the second rows (b, d) are the result of 5 steps of MCMC sampling applied to the latent samples that were used to generate the original interpolation. The only qualitative difference when compared to VAEs (see Figure 4) is a desaturation of the generated images.



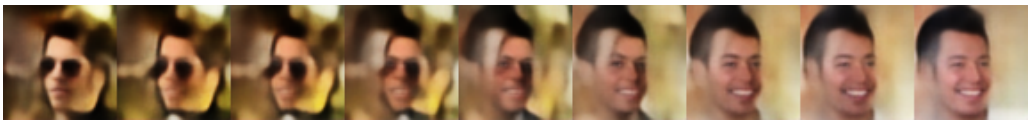
(a) AAE (initial)



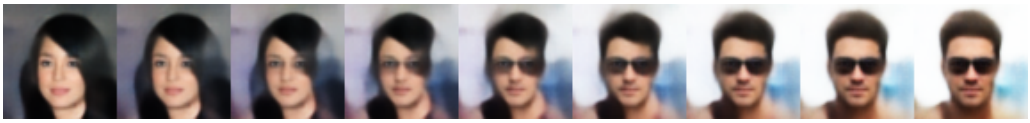
(b) AAE (5 steps)



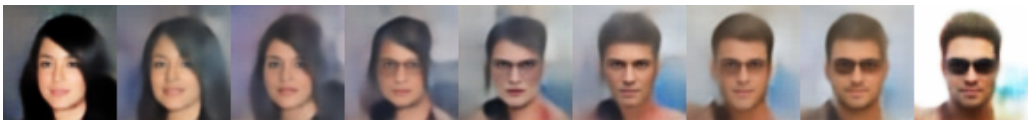
(c) AAE (initial)



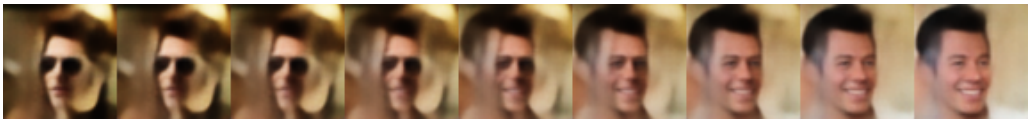
(d) AAE (5 steps)



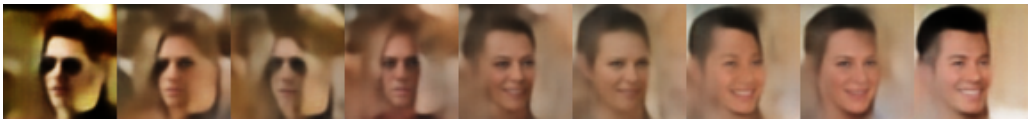
(e) DAAE (initial)



(f) DAAE (5 steps)



(g) DAAE (initial)



(h) DAAE (5 steps)

Figure 6: Interpolating between two faces using (a-d) an AAE and (e-h) a DAAE. The top rows (a, c, e, g) for each face is the original interpolation, whilst the second rows (b, d, f, h) are the result of 5 steps of MCMC sampling applied to the latent samples that were used to generate the original interpolation. Although the AAE performs poorly (b, d), the regularisation effect of denoising can be clearly seen with the DAAE after applying MCMC sampling (f, h).

C STREET VIEW HOUSE NUMBERS

C.1 SAMPLES

(a) VAE (initial) (b) VAE (1 step) (c) VAE (5 steps) (d) VAE (10 steps)

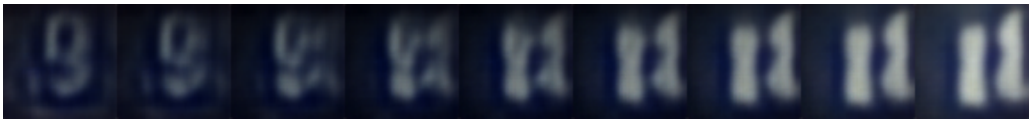
(e) DVAE (initial) (f) DVAE (1 step) (g) DVAE (5 steps) (h) DVAE (10 steps)

(i) AAE (initial) (j) AAE (1 step) (k) AAE (5 steps) (l) AAE (10 steps)

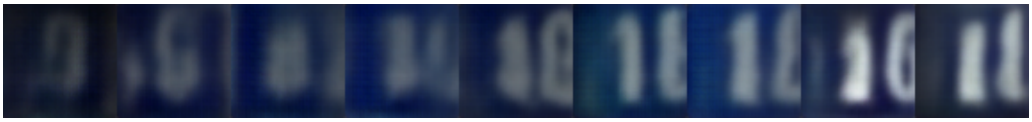
(m) DAAE (initial) (n) DAAE (1 step) (o) DAAE (5 steps) (p) DAAE (10 steps)

Figure 7: Samples from a VAE (a-d), DVAE (e-h), AAE (i-l) and DAAE (m-p) trained on the SVHN dataset. The samples from the models imitate the blurriness present in the dataset. Although very few numbers are visible in the initial sample, the VAE and DVAE produce recognisable numbers from most of the initial samples after a few steps of MCMC sampling. Although the AAE and DAAE fail to produce recognisable numbers, the final samples are still a clear improvement over the initial samples.

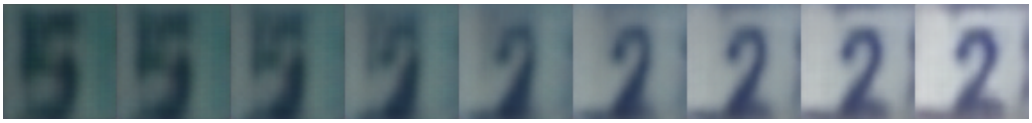
C.2 INTERPOLATIONS



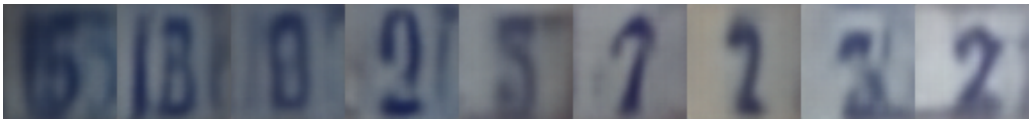
(a) VAE (initial)



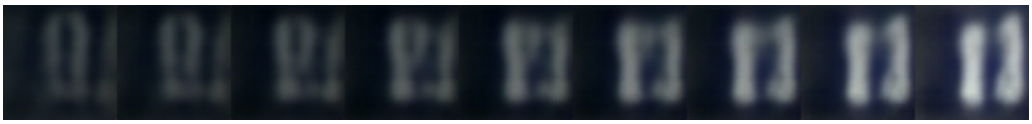
(b) VAE (5 steps)



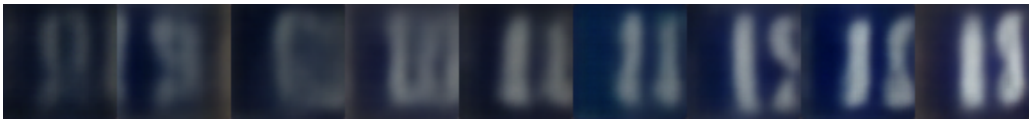
(c) VAE (initial)



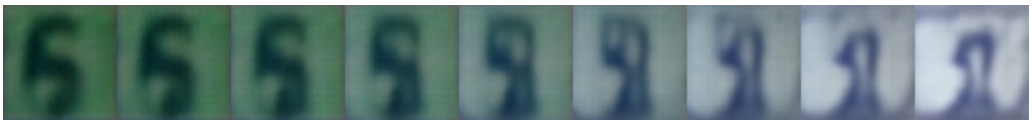
(d) VAE (5 steps)



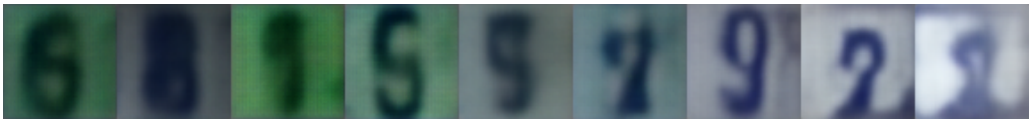
(e) DVAE (initial)



(f) DVAE (5 steps)



(g) DVAE (initial)



(h) DVAE (5 steps)

Figure 8: Interpolating between Google Street View house numbers using (a-d) a VAE and (e-h) a DVAE. The top rows (a, c, e, g) for each house number are the original interpolations, whilst the second rows (b, d, f, h) are the result of 5 steps of MCMC sampling. If the original interpolation produces symbols that do not resemble numbers, as observed in (a) and (e), the models will attempt to move the samples towards more realistic numbers (b, f). Interpolation between 1- and 2-digit numbers in an image (c, g) results in a meaningless blur in the middle of the interpolation. After a few steps of MCMC sampling the models instead produce more recognisable 1- or 2-digit numbers (d, h). We note that when the contrast is poor, denoising models in particular can struggle to recover meaningful images (h).