Denoising Adversarial Autoencoders

Antonia Creswell and Anil Anthony Bharath

***Abstract*— Unsupervised learning is of growing interest because it unlocks the potential held in vast amounts of unlabeled data to learn useful representations for inference. Autoencoders, a form of generative model, may be trained by learning to reconstruct unlabeled input data from a latent representation space. More robust representations may be produced by an autoencoder if it learns to recover clean input samples from corrupted ones. Representations may be further improved by introducing regularization during training to shape the distribution of the encoded data in the latent space. We suggest *denoising adversarial autoencoders* (AAEs), which combine denoising and regularization, shaping the distribution of latent space using adversarial training. We introduce a novel analysis that shows how denoising may be incorporated into the training and sampling of AAEs. Experiments are performed to assess the contributions that denoising makes to the learning of representations for classification and sample synthesis. Our results suggest that autoencoders trained using a denoising criterion achieve higher classification performance and can synthesize samples that are more consistent with the input data than those trained without a corruption process.**

***Index Terms*— Image analysis, pattern recognition, semisupervised learning, unsupervised learning.**

I. Introduction

MODELING and drawing data samples from complex, high-dimensional distributions are challenging. *Generative models* may be used to capture an underlying statistical structure from real-world data. A good generative model is not only able to draw samples from the distribution of data being modeled but should also be useful for inference.

Modeling complicated distributions may be made easier by learning the parameters of conditional probability distributions that map intermediate, *latent*, [2] variables from simpler distributions to more complex ones [4]. Often, the intermediate representations that are learned can be used for tasks, such as retrieval or classification [20], [24], [26], [30].

Typically, to train a model for classification, a deep neural network may be constructed, demanding large labeled data sets to achieve high accuracy [15]. Large labeled data sets may be expensive or difficult to obtain for some tasks. However, many state-of-the-art generative models can be trained without labeled data sets [9], [12], [14], [24]. For example, autoencoders learn a generative model, referred to as a *decoder*, by recovering inputs from corrupted [5], [12], [30] or *encoded* [14] versions of themselves.

Two broad approaches to learning the state-of-the-art generative autoencoders that do not require labeled training data include: 1) introduction of a denoising criterion [5], [30], [31], where the model learns to reconstruct clean samples from corrupted ones and 2) regularization of the latent space to match a prior [14], [20]; for the latter, the priors take a simple form, such as multivariate normal distributions.

The denoising variational autoencoder (DVAE) [12] combines both denoising and regularization in a single generative model. However, introducing a denoising criterion makes the variational cost function—used to match the latent distribution to the prior—analytically intractable. Reformulation of the cost function makes it tractable but only for certain families of prior and posterior distributions. We propose using adversarial training [9] to match the posterior distribution to the prior. Taking this approach expands the possible choices for families of prior and posterior distributions.

When a denoising criterion is introduced to an adversarial autoencoder (AAE), we have a choice to either shape the conditional distribution of latent variables given *corrupted* samples to match the prior (as was done using a variational approach [12]) or to shape the *full* posterior conditional on the *original* data samples to match the prior. Shaping the posterior distribution over corrupted samples does not require additional sampling during training, but trying to shape the full conditional distribution with respect to the original data samples does. We explore both the approaches using adversarial training to avoid the difficulties posed by analytically intractable cost functions.

In addition, a model that has been trained using the posterior conditioned on the corrupted data requires an iterative process for synthesizing samples, whereas using the full posterior conditioned on the original data does not. Similar challenges exist for the DVAE but were not addressed by Im *et al.* [12]. We analyze and address these challenges for AAEs, introducing a novel sampling approach for synthesizing samples from trained models.

In summary, our contributions include: 1) two types of denoising AAEs: one which is more efficient to train and one which is more efficient to draw samples from; 2) methods to draw synthetic data samples from denoising AAEs through Markov chain (MC) sampling; and 3) an analysis of the quality of features learned with denoising AAEs through their application to discriminative tasks.

Manuscript received July 3, 2017; revised January 3, 2018, March 17, 2018, and June 24, 2018; accepted June 25, 2018. Date of publication August 16, 2018; date of current version March 18, 2019. This work was supported by the Engineering and Physical Sciences Research Council through a Doctoral Training Studentship under Grant EP/L504786/1. *(Corresponding author: Antonia Creswell.)*

The authors are with the Biologically Inspired Computer Vision Group, Imperial College London, London SW7 2AZ, U.K. (e-mail: ac2211@ic.ac.uk; a.bharath@imperial.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2852738

II. Background

A. Autoencoders

In a supervised learning setting, given a set of training data $\{(y_i, x_i)\}_{i=1}^N$, we wish to learn a model $f_\psi(y|x)$ that maximizes the likelihood $\mathbb{E}_{p(y|x)} f_\psi(y|x)$ of the true label y given an observation x. In the supervised setting, there are many ways to calculate and approximate the likelihood because there is a ground-truth label for every training data sample.

When trying to learn a generative model $p_\theta(x)$ in the absence of a ground truth, calculating the likelihood of the model under the observed data distribution $\mathbb{E}_{x\sim p(x)} p_\theta(x)$ is challenging. Autoencoders introduce a two-step learning process that allows the estimation $p_\theta(x)$ of $p(x)$ via an auxiliary variable z. The variable z may take many forms, and we shall explore several of these in this section. The two-step process involves first learning a *probabilistic encoder* [14] $q_\phi(z|x)$ conditioned on observed samples and a second *probabilistic decoder* [14] $p_\theta(x|z)$ conditioned on the auxiliary variables. Using the probabilistic encoder, we may form a training data set $\{(z_i, x_i)\}_{i=1}^N$ where x_i is the ground truth output for $x \sim p(x|z_i)$ with the input being $z_i \sim q_\phi(z|x_i)$. The probabilistic decoder $p_\theta(x|z)$ may then be trained on this data set in a supervised fashion. By sampling $p_\theta(x|z)$ conditioning on the suitable z values, we may obtain a joint distribution $p_\theta(x, z)$, which may be marginalized by integrating over all z values to obtain to $p_\theta(x)$. Note that a deterministic autoencoder is a special case of a probabilistic one.

In some situations, the encoding distribution is chosen rather than learned [5], and in other situations, the encoder and the decoder are learned simultaneously [12], [14], [20].

B. Denoising Autoencoders

Bengio *et al.* [5] treat the encoding process as a local corruption process that does not need to be learned. In the corruption process, defined as $c(\tilde{x}|x)$ where $\tilde{x}$, the corrupted x is the auxiliary variable (instead of z). The decoder $p_\theta(x|\tilde{x})$ is therefore trained on the data pairs $\{(\tilde{x}_i, x_i)\}_{i=1}^N$.

By using a local corruption process (e.g., additive white Gaussian noise [5]), both $\tilde{x}$ and x have the same number of dimensions and are close to each other. This makes it very easy to learn $p_\theta(x|\tilde{x})$. Bengio *et al.* [5] show how the learned model may be sampled using an iterative process but does not explore how representations learned by the model may transfer to other applications such as classification.

Hinton and Salakhutdinov [11] show that when auxiliary variables of an autoencoder have lower dimension than the observed data, the encoding model learns representations that may be useful for tasks, such as classification and retrieval.

Rather than treating the corruption process $c(\tilde{x}, x)$ as an encoding process [5]—missing out on potential benefits of using a lower dimensional auxiliary variable—Vincent *et al.* [30], [31] learn an encoding distribution $q_\phi(z|\tilde{x})$ conditioned on the corrupted samples. The decoding distribution $p_\theta(x|z)$ learns to reconstruct images from encoded, corrupted images, see the denoising autoencoders (DAEs) in Fig. 1. Vincent *et al.* [30], [31] show that compared with regular autoencoders, DAEs learn representations that are more useful and robust for tasks such as classification. Parameters ϕ and θ are learned simultaneously by minimizing the reconstruction error for the training set $\{(\tilde{x}_i, x_i)\}_{i=1}^N$, which does not include z_i. The ground truth z_i for given $\tilde{x}_i$ is unknown. The form of the distribution over z, to which x samples are mapped, $p_\phi(z)$ is also unknown, making it difficult to draw novel data samples from the decoder model $p_\theta(x|z)$.

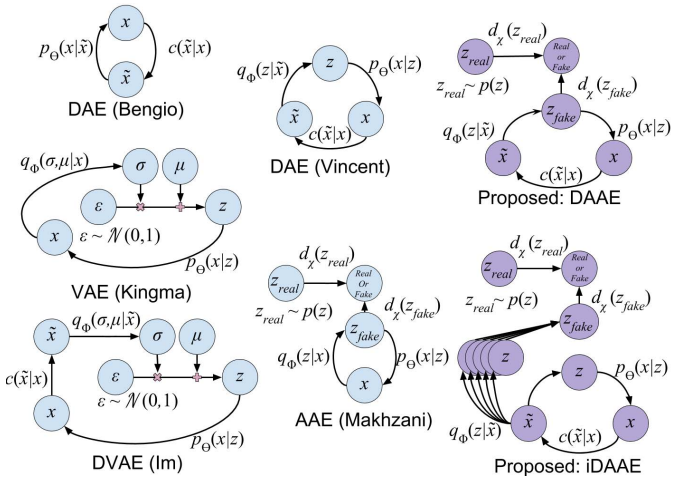


Fig. 1. Comparison of autoencoding models. Previous works include DAEs [5], [30], VAEs [14], AAEs [20], and DVAEs [12]. Our contributions are the DAAE and iDAAE models. Arrows represent mappings implemented using trained neural networks.

C. Variational Autoencoders

VAEs [14] specify a prior distribution, $p(z)$ to which $q_\phi(z|x)$ should map all x samples, by formulating and maximizing a variational lower bound on the log-likelihood of $p_\theta(x)$.

The variational lower bound on the log-likelihood of $p_\theta(x)$ is given by [14]

$$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL[q_\phi(z|x)||p(z)]. \quad (1)$$

The term $p_\theta(x|z)$ corresponds to the likelihood of a reconstructed x value given the encoding z of a data sample x. This formulation of the variational lower bound does not involve a corruption process. The term $KL[q_\phi(z|x)||p(z)]$ is the Kullback–Leibler (KL) divergence between $q_\phi(z|x)$ and $p(z)$. Samples are drawn from $q_\phi(z|x)$ via a reparametrisation trick (see the VAE in Fig. 1).

If $q_\phi(z|x)$ is chosen to be a parameterized multivariate Gaussian $\mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$ and the prior is chosen to be a Gaussian distribution, then $KL[q_\phi(z|x)||p(z)]$ may be computed analytically. KL divergence may only be computed analytically for certain (limited) choices of prior and posterior distributions.

VAE training encourages $q_\phi(z|x)$ to map observed samples to the chosen prior $p(z)$. Therefore, novel observed data samples may be generated via the following simple sampling process: $z_i \sim p(z)$, $x_i \sim p_\theta(x|z_i)$ [14].

Note that despite the benefits of the denoising criterion shown by Vincent *et al.* [30], [31] for regular

autoencoders, no corruption process was introduced by Kingma and Welling [14] for VAEs.

D. Denoising Variational Autoencoders

Adding the denoising criterion to a VAE is nontrivial because the variational lower bound becomes intractable.

Consider the conditional probability density function $\tilde{q}_\phi(z|x) = \int q_\phi(z|\tilde{x})c(\tilde{x}|x)d\tilde{x}$, where $q_\phi(z|\tilde{x})$ is the probabilistic encoder conditioned on the corrupted x samples $\tilde{x}$, and $c(\tilde{x}|x)$ is a corruption process. The variational lower bound may be formed in the following way [12]:

$$\log p_\theta(x) \geq \mathbb{E}_{\tilde{q}_\phi(z|x)} \log \left[\frac{p_\theta(x,z)}{q_\phi(z|\tilde{x})} \right] \geq \mathbb{E}_{\tilde{q}_\phi(z|x)} \log \left[\frac{p_\theta(x,z)}{\tilde{q}_\phi(x|z)} \right].$$

If $q_\phi(z|\tilde{x})$ is chosen to be Gaussian, then in many cases, $\tilde{q}_\phi(z|x)$ will be a mixture of Gaussians. If this is the case, there is no analytical solution for $KL[\tilde{q}_\phi(z|x)||p(z)]$, and so the denoising variational lower bound becomes analytically intractable. However, there may still be an analytical solution for $KL[q_\phi(z|\tilde{x})||p(z)]$. The DVAE therefore maximizes $\mathbb{E}_{\tilde{q}(x|z)} \log[(p_\theta(x,z)/q_\phi(z|\tilde{x}))]$. We refer to the model which is trained to maximize this objective as a DVAE (see Fig. 1). Im *et al.* [12] show that the DVAE achieves lower negative variational lower bounds than the regular VAE on the test data set.

However, note that $q_\phi(z|\tilde{x})$ is matched to the prior $p(z)$ rather than $\tilde{q}_\phi(z|x)$. This means that generating novel samples using $p_\theta(z|x)$ is not as simple as the process of generating samples from a VAE. To generate novel samples, we should sample $z_i \sim \tilde{q}_\phi(z|x)$ and $x_i \sim p_\theta(x|z_i)$, which is difficult because of the need to evaluate $\tilde{q}_\phi(z|x)$. Im *et al.* [12] do not address this problem.

For both DVAEs and VAEs, there is a limited choice of prior and posterior distributions for which there exists an analytic solution for the KL divergence. Alternatively, adversarial training may be used to learn a model that matches samples to an arbitrarily complicated target distribution—provided that samples may be drawn from both the target and model distributions.

III. Related Work

In this section, we introduce adversarial training and AAEs, on which this paper builds directly.

A. Adversarial Training

Adversarial training, as introduced by Goodfellow *et al.* [9], involves learning a mapping from a latent sample v to a data sample w. However, at a more abstract level, w may be thought of as a sample from any chosen target distribution and v as a sample from any distribution that we wish to map to w.

More formally, in adversarial training [9], a model $g_\phi(w|v)$ is trained to produce output samples w that match a target probability distribution $t(w)$. This is achieved by iteratively training two competing models: a generative model $g_\phi(w|v)$ and a discriminative model $d_\chi(w)$. The discriminative model is fed with the samples either from the generator (i.e., "fake" samples) or with samples from the target distribution (i.e., "real" samples) and trained to correctly predict whether the samples are "real" or "fake." The generative model—fed with input samples v, drawn from a chosen prior distribution $p(v)$—is trained to generate output samples w that are indistinguishable from target w samples in order to "fool" [24] the discriminative model into making incorrect predictions. This may be achieved by the following minimax objective [9]:

$$\min_g \max_d \mathbb{E}_{w\sim t(w)}[\log d_\chi(w)] + \mathbb{E}_{w\sim g_\phi(w|v)}[\log(1 - d_\chi(w))].$$

It has been shown that for an optimal discriminative model, optimizing the generative model is equivalent to minimizing the Jensen–Shannon divergence between the generated and target distributions [9]. In general, it is reasonable to assume that, during training, the discriminative model quickly achieves near optimal performance [9]. This property is useful for learning distributions for which the Jensen–Shannon divergence may not be easily calculated.

The generative model is optimal when the distribution of the generated samples matches the target distribution. Under these conditions, the discriminator is maximally confused and cannot distinguish "real" samples from "fake" ones. As a consequence of this, adversarial training may be used to capture very complicated data distributions and has been shown to be able to synthesize images of handwritten digits and human faces that are almost indistinguishable from real data [24].

B. Adversarial Autoencoders

Makhzani *et al.* [20] introduce the AAE, where $q_\phi(z|x)$ is both the probabilistic encoding model in an autoencoder framework and the generative model in an adversarial framework.

A new discriminative model $d_\chi(z)$ is introduced. This discriminative model is trained to distinguish between latent samples drawn from $p(z)$ and $q_\phi(z|x)$. The cost function used to train the discriminator $d_\chi(z)$ is

$$\mathcal{L}_{\text{dis}} = -\frac{1}{N}\sum_{i=0}^{N-1} \log d_\chi(z_i) - \frac{1}{N}\sum_{j=N}^{2N-1} \log(1 - d_\chi(z_j))$$

where $z_{i=0:N-1} \sim p(z)$ and $z_{j=N:2N-1} \sim q_\phi(z|x)$ and N is the size of the training batch.

Adversarial training is used to match $q_\phi(z|x)$ to an arbitrarily chosen prior $p(z)$. The cost function for matching $q_\phi(z|x)$ to prior $p(z)$ is as follows:

$$\mathcal{L}_{\text{prior}} = \frac{1}{N}\sum_{i=0}^{N-1} \log(1 - d_\chi(z_i)) \tag{2}$$

where $z_{i=0:N-1} \sim q_\phi(z|x)$ and N is the size of a training batch. If both $\mathcal{L}_{\text{prior}}$ and $\mathcal{L}_{\text{dis}}$ are optimized, $q_\phi(z|x)$ will be indistinguishable from $p(z)$.

In Makhzani *et al.*'s [20] AAE, $q_\phi(z|x)$ is specified by a neural network whose input is x and whose output is z. This allows $q_\phi(z|x)$ to have arbitrary complexity, unlike the VAE where the structure of $q_\phi(z|x)$ is usually limited to a multivariate Gaussian. In an AAE, the posterior does not have to be analytically defined because an adversary is used to

match the prior, avoiding the need to analytically compute a KL divergence.

Makhzani *et al.* [20] demonstrate that AAEs are able to match $q_\phi(z|x)$ to several different priors $p(z)$ including a mixture of 10 2-D Gaussian distributions. We explore another direction for AAEs, by extending them to incorporate a denoising criterion.

IV. Denoising Adversarial Autoencoder

We propose denoising AAEs, DAEs, that use adversarial training to match the distribution of auxiliary variables z to a prior distribution $p(z)$.

We formulate two versions of a denoising AAE, which are trained to approximately maximize the denoising variational lower bound [12]. In the first version, we directly match the posterior $\tilde{q}_\phi(z|x)$ to the prior $p(z)$ using adversarial training. We refer to this as an integrating denoising AAE (iDAAE). In the second, we match the intermediate conditional probability distribution $q_\phi(z|\tilde{x})$ to the prior $p(z)$. We refer to this as a DAAE.

In the iDAAE, adversarial training is used to bypass analytically intractable KL divergences [12]. In the DAAE, using adversarial training broadens the choice for prior and posterior distributions beyond those for which the KL divergence may be analytically computed.

A. Construction

The distribution of encoded data samples is given by $\tilde{q}_\phi(z|x) = \int q_\phi(z|\tilde{x})c(\tilde{x}|x)d\tilde{x}$ [12]. The distribution of decoded data samples is given by $p_\theta(x|z)$. Both $q_\phi(z|x)$ and $p_\theta(x|z)$ may be trained to maximize the likelihood of the reconstructed sample, by minimizing the reconstruction cost function $\mathcal{L}_{rec} = 1/N \sum_{i=0}^{N-1} \log p_\theta(x|z_i)$, where the z_i values are obtained via the following sampling process $x_{i=0:N-1} \sim p(x)$, $\tilde{x}_i \sim c(\tilde{x}|x_i)$, and $z_i \sim q_\phi(z|\tilde{x}_i)$, and $p(x)$ is the distribution of the training data.

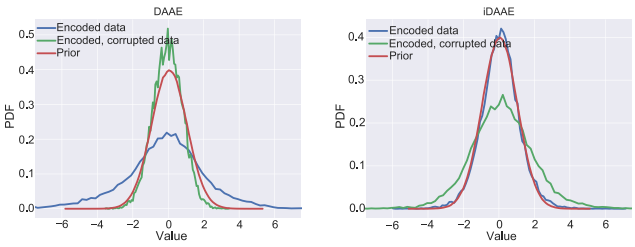
We also want to match the distribution of auxiliary variables z to a prior $p(z)$. When doing so, there is a choice to match either $\tilde{q}_\phi(z|x)$ or $q_\phi(z|\tilde{x})$ to $p(z)$. Each choice has its own tradeoffs either during training or during sampling.

1) iDAAE (Matching $\tilde{q}_\phi(z|x)$ to a Prior): In DVAEs, there is often no analytical solution for the KL divergence between $\tilde{q}_\phi(z|x)$ and $p(z)$ [12], thus making it difficult to match $\tilde{q}_\phi(z|x)$ to $p(z)$. Rather, we propose using adversarial training to match $\tilde{q}_\phi(z|x)$ to $p(z)$, thus requiring samples to be drawn from $\tilde{q}_\phi(z|x)$ during training.

It is challenging to draw samples directly from $\tilde{q}_\phi(z|x) = \int q_\phi(z|\tilde{x})c(\tilde{x}|x)d\tilde{x}$, but it is easy to draw samples from $q_\phi(z|\tilde{x})$ and so $\tilde{q}_\phi(z|x)$ may be approximated by $1/M \sum_{j=1}^{M} q_\phi(z|\tilde{x}_j)$, $\tilde{x}_{j=1:M} \sim c(\tilde{x}|x_0)$, and $x_0 \sim p(x)$ (see Fig. 1). Matching is achieved by minimizing the following cost function:

$$\mathcal{L}_{\text{prior}} = \frac{1}{N} \sum_{i=0}^{N-1} \log(1 - d_\chi(\hat{z}_i))$$

where $\hat{z}_{i=0:N-1} = (1/M) \sum_{j=1}^{M} z_{i,j}$, $z_{i=0:N-1,j=1:M} \sim q_\phi(z|\tilde{x}_{i,j})$, $\tilde{x}_{i=0:N-1,j=1:M} \sim c(\tilde{x}|x_j)$, and $x_{i=0:N-1} \sim p(x)$.



(a) (b)

Fig. 2. Compare how iDAAE and DAAE match encodings to the prior when trained on the CelebA data set. Encoding refers to $q_\phi(z|x)$, prior refers to the normal prior $p(z)$, and encoded corrupted data refers to $q_\phi(z|\tilde{x})$. (a) DAAE: encoded corrupted data samples match the prior. (b) iDAAE: encoded data samples match the prior.

2) DAAE (Matching $q_\phi(z|\tilde{x})$ to a Prior): Since drawing samples from $q_\phi(z|\tilde{x})$ is trivial, $q_\phi(z|\tilde{x})$ may be matched to $p(z)$ via adversarial training. This is more efficient than matching $\tilde{q}_\phi(z|x)$ since a Monte Carlo integration step (in Section IV-A1) is not needed (see Fig. 1). In using adversarial training in place of KL divergence, the only restriction is that we must be able to draw samples from the chosen prior. Matching may be achieved by minimizing the following loss function:

$$\mathcal{L}_{\text{prior}} = \frac{1}{N} \sum_{i=0}^{N-1} \log(1 - d_\chi(z_i))$$

where $z_{i=0:N-1} \sim q_\phi(z|\tilde{x}_i)$.

Though more computationally efficient to train, there are drawbacks when trying to synthesise novel samples from $p_\theta(x)$ if $q_\phi(z|\tilde{x})$—rather than $\tilde{q}_\phi(z|x)$—is matched to the prior. The effects of using a DAAE rather than an iDAAE may be visualized by plotting the empirical distribution of encodings of both data samples and corrupted data samples with the desired prior, and these are shown in Fig. 2.

V. Synthesizing Novel Samples

In this section, we review several techniques used to draw samples from trained autoencoders and identify a problem with sampling DAAEs, which interestingly, also applies to DVAEs [12]. We propose a novel approach to sampling DAAEs; we draw strongly on previous work by Bengio *et al.* [4], [5].

A. Drawing Samples From Autoencoders

New samples may be generated by sampling a learned $p_\theta(x|z)$ value conditioning on z drawn from a suitable distribution. In the case of VAE [14] and AAE [20], the choice of this distribution is simple, because during training, the distribution of auxiliary variables is matched to a chosen prior distribution $p(z)$. It is therefore easy and efficient to sample both VAE and AAE via the following process: $z \sim p(z)$, $x \sim p_\theta(x|z)$ [14], [20].

The process for sampling DAEs is more complicated. In the case where the auxiliary variable is a corrupted image $\tilde{x}$ [3], the sampling process is as follows: $x_0 \sim p(x)$, $\tilde{x}_0 \sim c(\tilde{x}|x_0)$,

$x_1 \sim p_\theta(x|\tilde{x}_0)$ [5]. In the case where the auxiliary variable is an encoding [30], [31], the sampling process is the same, with $p_\theta(x|\tilde{x})$ encompassing both the encoding and decoding processes.

However, since a DAE [5] is trained to reconstruct corrupted versions of its inputs, and the sample x_1 is likely to be very similar to x_0. Bengio *et al.* [5] propose a method for iteratively sampling DAEs by defining an MC whose stationary distribution—under certain conditions—exists and is equivalent, under certain assumptions, to training the data distribution. This approach is generalized and extended by Bengio *et al.* [4] to introduce a latent distribution with no prior assumptions on z.

We now consider the implication for drawing samples from denoising AAEs introduced in Section IV-A. By using the iDAAE formulation (see Section IV-A1), where $\tilde{q}_\phi(z|x)$ is matched to the prior over z, then x samples may be drawn from $p_\theta(x|z)$, conditioning on $z \sim p(z)$. However, if we use the DAAE—matching $q_\phi(z|\tilde{x})$ to a prior—sampling becomes nontrivial.

On the surface, it may appear easy to draw samples from DAAEs (see Section IV-A2), by first sampling the prior $p(z)$ and then sampling $p_\theta(x|z)$. However, the full posterior distribution is given by $\tilde{q}_\phi(z|x) = \int q_\phi(z|\tilde{x})c(\tilde{x}|x)d\tilde{x}$, but only $q_\phi(z|\tilde{x})$ is matched to $p(z)$ during training (see Fig. 2). The implication of this is that when attempting to synthesize novel samples from $p_\theta(x|z)$, drawing samples from the prior $p(z)$ is unlikely to yield samples consistently with $p(x)$. This will become more clear in Section V-B.

B. Proposed Method for Sampling DAAEs

Here, we propose a method for synthesizing novel samples using trained DAAEs. In order to draw samples from $p_\theta(x|z)$, we need to be able to draw samples from $\tilde{q}_\phi(z|x)$.

To ensure that we draw novel data samples, we do not want to draw samples from the training data at any point during sample synthesis. This means that we cannot use data samples from our training data to approximately draw samples from $\tilde{q}_\phi(z|x)$.

Instead, similar to Bengio *et al.* [5], we formulate an MC, which we show that it has the necessary properties to converge and that the chain converges to $\mathcal{P}(z) = \int \tilde{q}_\phi(z|x)p(x)dx$. Unlike Bengio's formulation, our chain is initialized with a random vector of the same dimensions as the latent space, rather than a sample drawn from the training set.

We define an MC by the following sampling process:

$$\begin{aligned} &z^{(0)} \sim \mathbb{R}^a, \quad x^{(t)} \sim p_\theta(x|z^{(t)} \\ &\tilde{x}^{(t)} \sim c(\tilde{x}|x^{(t)}), \quad z^{(t+1)} \sim q_\phi(z|\tilde{x}^{(t)}) \\ &\quad t \geq 0. \end{aligned} \tag{3}$$

Notice that our first sample is any real vector of dimension a, where a is the dimension of the latent space. This MC has the transition operator

$$\begin{aligned} &T_{\theta,\phi}(z^{(t+1)}|z^{(t)}) \\ &\quad = \int q_\phi(z^{(t+1)}|\tilde{x}^{(t)})c(\tilde{x}^{(t)}|x^{(t)})p_\theta(x^{(t)}|z^{(t)})dx\,d\tilde{x}. \end{aligned} \tag{4}$$

We will now show that under certain conditions, this transition operator defines an ergodic MC that converges to $\mathcal{P}(z) = \int \tilde{q}_\phi(z|x)p(x)dx$ in the following steps: 1) we will show that there exists a stationary distribution $\mathcal{P}(z)$ for $z^{(0)}$ drawn from a specific choice of initial distribution (see Lemma 1); 2) the MC is homogeneous, because the transition operator is defined by a set of distributions whose parameters are fixed during sampling; 3) we will show that the MC is also ergodic (see Lemma 2); and 4) since the chain is both homogeneous and ergodic, there exists a unique stationary distribution to which the MC will converge [22].

Step 1) shows that one stationary distribution is $\mathcal{P}(z)$, which we now know by 2) and 3) to be the unique stationary distribution. So the MC converges to $\mathcal{P}(z)$.

In this section, only we use a change of notation, where the training data probability distribution, previously represented as $p(x)$, is represented as $\mathcal{P}(x)$; this is to help make distinctions between "natural system" probability distributions and the learned distributions. Furthermore, note that $p(z)$ is the prior, while the distribution required for sampling $\mathcal{P}(x|z)$ is $\mathcal{P}(z)$ such that

$$\mathcal{P}(x) = \int \mathcal{P}(x|z)\mathcal{P}(z)dz \approx \int p_\theta(x|z)\mathcal{P}(z)dz. \tag{5}$$

$$\begin{aligned} \mathcal{P}(z) &= \int \tilde{q}_\phi(z|x)\mathcal{P}(x)dx \\ &= \int\int q_\phi(z|\tilde{x})c(\tilde{x}|x)d\tilde{x}\mathcal{P}(x)dx. \end{aligned} \tag{6}$$

Lemma 1: $\mathcal{P}(z)$ is a stationary distribution for the MC defined by the sampling process in (3).

For proof, see the Appendix.

Lemma 2: The MC defined by the transition operator $T_{\theta,\phi}(z_{t+1}|z_t)$ (4) is ergodic, provided that the corruption process is additive Gaussian noise and that the adversarial pair $q_\phi(z|\tilde{x})$ and $d_\chi(z)$ are optimal within the adversarial framework.

For proof, see the Appendix.

Theorem 1: Under the assumptions that $p_\theta(x|z) = \mathcal{P}(x|z)$ and that the adversarial pair $q_\phi(z|x)$ and $d_\chi(x)$ are optimal, the transition operator $T_{\theta,\phi}(z^{(t+1)}|z^{(t)})$ defines an MC whose stationary distribution is $\mathcal{P}(z) = \int \tilde{q}_\phi(z|x)\mathcal{P}(x)dx$.

Proof: This follows from Lemmas 1 and 2. □

This sampling method uncovers the distribution $\mathcal{P}(z)$ on which samples drawn from $p_\theta(x|z)$ must be conditioned in order to sample $p_\theta(x)$. Assuming that $p_\theta(x|z) = \mathcal{P}(x|z)$, this allows us to draw samples from $\mathcal{P}(x)$.

For completeness, we would like to acknowledge that there are several other methods that use MCs during the training of autoencoders [2], [21] to improve the performance. Our approach for synthesizing samples using the DAAE is focused on sampling only from trained models; the MC sampling is not used to update model parameters.

VI. Implementation

The analyses of Sections IV and V were deliberately general; they did not rely on any specific implementation choice to capture the model distributions. In this section, we consider a specific implementation of denoising AAEs and apply them to

the task of learning models for image distributions. We define an encoding model that maps corrupted data samples to a latent space $E_\phi(\tilde{x})$ and $R_\theta(z)$ which maps samples from a latent space to an image space. These, respectively, draw samples according to the conditional probabilities $q_\phi(z|\tilde{x})$ and $p_\theta(x|z)$. We also define a corruption process $C(x)$, which draws samples according to $c(\tilde{x}|x)$.

The parameters θ and ϕ of models $R_\theta(z)$ and $E_\phi(\tilde{x})$ are learned under an autoencoder framework; the parameter ϕ is also updated under an adversarial framework. The models are trained using large data sets of unlabeled images.

A. Autoencoder

Under the regular (nondenoising) autoencoder framework, $E_\phi(x)$ is the encoder, and $R_\theta(z)$ is the decoder. We used neural networks for both the encoder and the decoder. Rectifying linear units (ReLUs) were used between all intermediate layers to encourage the networks to learn representations that capture multimodal distributions. In the final layer of the decoder network, a sigmoid activation function is used so that the output represents the pixels of an image. The final layer of the encoder network is left as a linear layer so that the distribution of encoded samples is not restricted.

As described in Section IV-A, the autoencoder is trained to maximize the log-likelihood of the reconstructed image given the corrupted image. Although there are several ways in which one may evaluate this log-likelihood, we chose to measure pixelwise binary cross entropy between the reconstructed sample $\hat{x}$ and the original samples before corruption x. During training, we aim to learn parameters ϕ and θ that minimize the binary cross entropy between $\hat{x}$ and x. The training process is summarized by lines 1–9 in Algorithm 1 in the Appendix.

The elements of the vectors that output by the encoder may take any real values, and so minimizing reconstruction error is not sufficient to match either $q_\phi(z|\tilde{x})$ or $\tilde{q}_\phi(z|x)$ to the prior $p(z)$. For this, parameter ϕ must also be updated under the adversarial framework.

B. Adversarial Training

To perform adversarial training, we define the discriminator $d_\chi(z)$, described in Section III-A to be a neural network, which we denote $D_\chi(z)$. The output of $D_\chi(z)$ is a "probability" because the final layer of the neural network has a sigmoid activation function, constraining the range of $D_\chi(z)$ to be between (0, 1). Intermediate layers of the network have ReLU activation functions to encourage the network to capture highly nonlinear relations between z and the labels, {'real', 'fake'}.

How adversarial training is applied depends on whether $\tilde{q}_\phi(z|x)$ or $q_\phi(z|\tilde{x})$ is being fit to the prior $p(z)$. z_{fake} refers to the samples drawn from the distribution that we wish to fit to $p(z)$, and z_{real} samples drawn from the prior $p(z)$. The discriminator $D_\chi(z)$ is trained to predict whether the values of z are "real" or "fake." This may be achieved by learning the parameters χ that maximize the probability of the correct labels being assigned to z_{fake} and z_{real}. This training procedure is shown in Algorithm 1 on lines 14–16.

Drawing samples z_{real} involves sampling some prior distributions $p(z)$, often a Gaussian. Now, we consider how to draw fake samples z_{fake}. How these samples are drawn depends on whether $q_\phi(z|\tilde{x})$ (DAAE) or $\tilde{q}_\phi(z|x)$ (iDAAE) is being fit to the prior. Drawing samples z_{fake} is easy if $q_\phi(z|\tilde{x})$ is being matched to the prior, as these are simply obtained by mapping corrupted samples though the encoder: $z_{\text{fake}} = E_\phi(\tilde{x})$.

However, if $\tilde{q}(z|x)$ is being matched to the prior, we must use Monte Carlo sampling to approximate z_{fake} samples (see Section IV-A1). The process for calculating z_{fake} is given by Algorithm 2 in the Appendix and detailed in Section IV-A1.

Finally, in order to match the distribution of z_{fake} samples to the prior $p(z)$, adversarial training is used to update parameters ϕ while the holding parameters χ fixed. Parameter ϕ is updated to minimize the likelihood that $D_\chi(\cdot)$ correctly classifies z_{fake} as being "fake." The training procedure is laid out in lines 18 and 19 of Algorithm 1.

Algorithm 1 shows the steps taken to train an iDAAE. To train a DAAE instead, all lines in Algorithm 1 are the same except Line 11, which may be replaced by $z_{\text{fake}} = E_\phi(\tilde{x})$.

Algorithm 1 Algorithm for Training an iDAAE. This Algorithm May Be Altered for DAAE Training, by Replacing Line 11 With $z_{fake} = E_\phi(\tilde{x})$.

1 # Draw a batch of samples from the training data:
2 $\mathbf{x} = \{x_0, x_2, \ldots, x_{N-1}\} \sim p(x)$
3 **for** $k = 1$ *to NoEpoch* **do**
4 $\quad \tilde{\mathbf{x}} = C(\mathbf{x})$ # Corrupt all samples
5 $\quad \mathbf{z} = E_\phi(\tilde{\mathbf{x}})$ # Encode all corrupted samples
6 $\quad \hat{\mathbf{x}} = R_\theta(\mathbf{z})$ # Reconstruct
7 $\quad$ # Minimize reconstruction cost
8 $\quad \mathcal{L}_{rec} = -\frac{1}{N}\sum_{i=0}^{N-1}(\hat{x}_i \log x_i + (1-\hat{x}_i)\log(1-x_i))$
9 $\quad \phi \leftarrow \phi - \alpha\nabla_\phi \mathcal{L}_{rec}$
10 $\quad \theta \leftarrow \theta - \alpha\nabla_\theta \mathcal{L}_{rec}$
11 $\quad$ # Match $\tilde{q}_\phi(z|x)$ to $p(z)$ using adversarial training
12 $\quad \mathbf{z}_{fake} = \text{approx\_z}(\mathbf{x})$ # Draw samples for $\tilde{q}_\phi(z|x)$
13 $\quad \mathbf{z}_{real} \sim p(z)$ # Draw samples from prior $p(z)$
14 $\quad$ # Train the discriminator:
15 $\quad \mathcal{L}_{dis} = -\frac{1}{N}\big[\sum_{i=0}^{N-1}\log D_\chi(z_{real_i}) +$
16 $\qquad \sum_{i=0}^{N-1}\log(1 - D_\chi(z_{fake_i}))\big]$
17 $\quad \chi \leftarrow \chi - \alpha\nabla_\chi \mathcal{L}_{dis}$
18 $\quad$ # Train the decoder to match the prior:
19 $\quad \mathcal{L}_{prior} = \frac{1}{N}\sum_{i=0}^{N-1}\log(1 - D_\chi(z_{fake_i}))$
20 $\quad \phi \leftarrow \phi - \alpha\nabla_\phi \mathcal{L}_{prior}$
21 **end**

C. Sampling

Although the training process for matching $\tilde{q}_\phi(z|x)$ to $p(z)$ is less computationally efficient than matching $q_\phi(z|\tilde{x})$ to $p(z)$, it is very easy to draw samples when $\tilde{q}_\phi(z|x)$ is matched to the prior (iDAAE). We simply draw a random $z^{(0)}$ value from $p(z)$ and calculate $x^{(0)} = R_\theta(z^{(0)})$, where $x^{(0)}$ is a new sample. When drawing samples, parameters θ and ϕ are fixed.

If $q_\phi(z|\tilde{x})$ is matched to the prior (DAAE), an iterative sampling process is needed in order to draw new samples

from $p(x)$. This sampling process is described in Section V-B. To implement this, sampling process is trivial. A random sample $z^{(0)}$ is drawn from any distribution; the distribution does not have to be the chosen prior $p(z)$. New samples $z^{(t+1)}$ are obtained by iteratively decoding, corrupting, and encoding $z^{(t)}$, such that $z^{(t+1)}$ is given by: $z^{(t+1)} = E_\phi(C(R_\theta(z^{(t)})))$.

In Section IV, we evaluate the performance of denoising AAEs on three image data sets: a synthetic color image data set of tiny images (Sprites) [25], a complex data set of handwritten characters [17], and color faces (CelebA) [19]. Some results on handwritten digits (MNIST) are presented in the Appendix. The denoising and nondenoising AAEs are compared for tasks, such as reconstruction, generation, and classification.

VII. Experiments and Results

A. Code Available Online

We make our PyTorch [23] code available at the following link: https://github.com/ToniCreswell/pyTorch_DAAE.[1]

B. Data Sets

We evaluate our denoising AAE on three image data sets of varying complexity. Here, we describe the data sets and their complexity in terms of variation within the data set, the number of training examples, and the size of the images.

1) Data Sets (Omniglot): The Omniglot data set is a handwritten character data set consisting of 1623 categories of character from 50 different writing systems, with only 20 examples of each character. Each example in the data set is 105 $\times$ 105 pixels, taking values {0,1}. The data set is split such that 19 examples from 964 categories make up the training data set, while one example from each of those 964 categories makes up the testing data set. The 20 characters from each of the remaining 659 categories make up the evaluation data set. This means that experiments may be performed to reconstruct or classify samples from categories not seen during training of the autoencoders.

2) Data Sets (Sprites): The sprites data set is made up of 672 unique humanlike characters. Each character has seven attributes, including hair, body, armor, trousers, arm, and weapon type, as well as gender. For each character, there 20 animations consisting of 6–13 frames each. There are between 120 and 260 examples of each character; however, every example is in a different pose. Each sample is 60 $\times$ 60 pixels and is in color. The training, validation, and test data sets are split to have frames from 500, 72, and 100 unique characters each, with no two sets having frames containing the same character.

3) Data Sets (CelebA): The CelebA data set consists of 250k images of faces in color. Though a version of the data set with tightly cropped faces exists, we use the uncropped data set. We use 1000 samples for testing and the rest for training. Each example has dimensions 64 $\times$ 64 and a set of labeled facial attributes, for example, "No Beard," "Blond Hair," and "Wavy Hair." This face data set is more complex than the Toronto Face data set used by Makhzani *et al.* [20] for training the AAE.

[1]An older version of our code in Theano available at https://github.com/ToniCreswell/DAAE_ with our results presented in iPython notebooks. Since this is a revised version of this paper and Theano is no longer being supported, our new experiments on the CelebA data sets were performed using PyTorch.

C. Architecture and Training

For each data set, we detail the architecture and training parameters of the networks used to implement each of the denoising AAEs. For each data set, several DAAEs, iDAAEs, and AAEs are trained. In order to compare models trained on the same data sets, the same network architectures, batch size, learning rate, annealing rate, and size of latent code are used for each.

Each set of models were trained using the same optimization algorithm. The trained AAE [20] models act as a benchmark, allowing us to compare our proposed DAAEs and iDAAEs.

1) Architecture and Training (Omniglot): The decoder, encoder, and discriminator networks consisted of 6, 3, and 2 fully connected layers, respectively, each layer having 1000 neurons. We found that deeper networks than those proposed by Makhazni *et al.* [20] (for the MNIST data set) led to better convergence. The networks are trained for 1000 epochs, using a learning rate of 10^{-5}, a batch size of 64, and the Adam [13] optimization algorithm. We used a 200-D Gaussian for the prior and additive Gaussian noise with a standard deviation of 0.5 for the corruption process. When training the iDAAE, we use $M = 5$ steps of Monte Carlo integration (see Algorithm 2 in the Appendix).

2) Architecture and Training (Sprites): Both the encoder and the discriminator are two-layer fully connected neural networks with 1000 neurons in each layer. For the decoder, we used a three-layer fully connected network with 1000 neurons in the first layer and 500 in each of the last layers, and this configuration allowed us to capture complexity in the data without overfitting. The networks were trained for 5 epochs, using a batch size of 128, a learning rate of 10^{-4}, and the Adam [13] optimization algorithm. We used an encoding 200 units, 200-D Gaussian for the prior, and additive Gaussian noise with a standard deviation of 0.25 for the corruption process. The iDAAE was trained with $M = 5$ steps of Monte Carlo integration.

3) Architecture and Training (CelebA): The encoder and the decoder were constructed with convolutional layers, rather than fully connected layers since the CelebA data set is more complex than the Toronto face data set use by Makhzani *et al.* [20]. The encoder and the decoder consisted of four convolutional layers with a similar structure to that of the deep convolutional generative adversarial network proposed by Radford *et al.* [24]. We used a three-layer fully connected network for the discriminator. Networks were trained for 100 epochs with a batch size of 64 using RMSprop with a learning rate of 10^{-4} and a momentum of $\rho = 0.1$ for training the discriminator. We found that using smaller momentum values leads to more blurred images, and however, larger momentum values prevented the network from converging and made training unstable. When using Adam instead of RMSprop (on the CelebA data set specifically), we found that the values in

the encodings became very large and were not consistent with the prior. The encoding was made up of 200 units, and we used a 200-D Gaussian for the prior. We used additive Gaussian noise for the corruption process. We experimented with different noise levels σ between [0.1, 1.0], finding several values in this range to be suitable. For our classification experiments, we fixed $\sigma = 0.25$, and for synthesis from the DAAE, to demonstrate the effect of sampling, we used $\sigma = 1.0$. For the iDAAE, we experimented with $M = 2, 5, 20, 50$. We found that $M < 5$ (when $\sigma = 1.0$) was not sufficient to train an iDAAE. By comparing the histograms of encoded data samples to histograms of the prior (see Fig. 2), for an iDAAE trained with a particular M value, we are able to see whether M is sufficiently larger or not. We found $M = 5$ to be sufficiently large for most experiments.

D. Sampling DAAEs and iDAAEs

Samples may be synthesized using the decoder of a trained iDAAE or AAE by passing latent samples drawn from the prior through the decoder. On the other hand, if we pass samples from the prior through the decoder of a trained DAAE, the samples are likely to be inconsistent with the training data. To synthesize more consistent samples using the DAAE, we draw an initial $z^{(0)}$ value from any random distribution—we use a Gaussian distribution for simplicity[2]—and decode, corrupt, and encode the sample several times for each synthesized sample. This process is equivalent to sampling an MC where one iteration of the MC includes decoding, corrupting, and encoding to get a $z^{(t)}$ value after t iterations. The sample $z^{(t)}$ may be used to synthesize a novel sample, which we call $x^{(t)}$. $x^{(0)}$ is the sample generated when $z^{(0)}$ is passed through the decoder.

To evaluate the quality of some synthesized samples, we calculated the log-likelihood $\mathbb{L}_\theta$ of real (hold-out) samples under the model [20]. This is achieved by fitting a Parzen window to a number of synthesized samples. Further details of how the log-likelihood is calculated for each data set are given in Appendix G.

We expect initial samples $x^{(0)}$ values drawn from the DAAE to have a lower (worse) log-likelihood than those drawn from the AAE, and however, we expect MC sampling to improve synthesized samples, such that $x^{(t)}$ for $t > 0$ should have larger log-likelihood than the initial samples. It is not clear whether $x^{(t)}$ for $t > 0$ drawn using a DAAE will be better than samples drawn from an iDAAE. The purpose of these experiments is to demonstrate the challenges associated with drawing samples from denoising AAEs and show that our proposed methods for sampling a DAAE and training iDAAEs allow us to address these challenges. We also hope to show that iDAAE and DAAE samples are competitive with those drawn from an AAE.

1) Sampling (Omniglot): Here, we explore the Omniglot data set, where we look at the log-likelihood score on both the testing and evaluation data sets. Recall (see Section VII-B1) that the testing data set has samples from the same classes as the training data set and the evaluation data set has samples from different classes.

[2]Which happens to be equivalent to our choice of prior

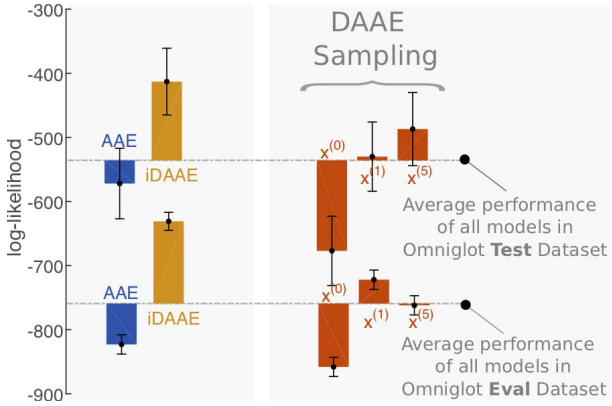


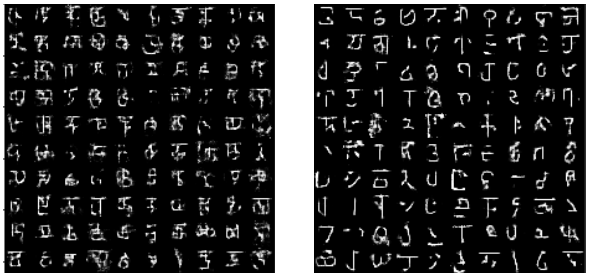
Fig. 3. Omniglot mean log-likelihood $\mathbb{L}_\theta$ compared on the testing and evaluation data sets. The training and evaluation data sets have samples from *different* handwritten character classes. All models were trained using a 200-D Gaussian prior. The training and testing data sets have samples from the *same* handwritten character classes. Error bars denote the standard error.

First, we discuss the results on the evaluation data set. The results, as shown in Fig. 3, are consistent with what is expected of the models. The iDAAE outperformed the AAE, with a higher (better) log-likelihood. The initial samples drawn using the DAAE had more smaller (worse) log-likelihood values than samples drawn using the AAE. However, after one iteration of MC sampling, the synthesized samples have increasing (better, i.e., moving away from $-\infty$) log-likelihood values than those from the AAE. Additional iterations of MC sampling led to worse results, possibly because synthesized samples tending toward multiple modes of the data generating distribution, appearing to be more like samples from classes represented in the training data.

The Omniglot testing data set consists of one example of every category in the training data set. This means that if multiple iterations of MC sampling cause synthesized samples to tend toward modes in the training data, the likelihood score on the testing data set is likely to increase. The results shown in Fig. 3 confirm this expectation; the log-likelihood for the fifth sample is higher (better) than for the first sample. These apparently conflicting results (in Fig. 3)—whether sampling improves or worsens synthesized samples—highlights the challenges involved with evaluating generative models using the log-likelihood, discussed in more depth by Theis *et al.* [29]. For this reason, we also show qualitative results.

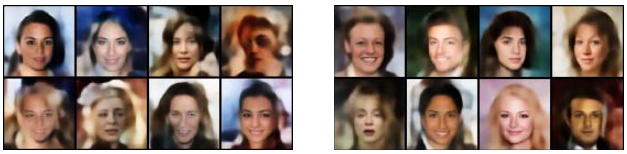
Fig. 4(a) shows an set of initial samples ($x^{(0)}$) drawn from a DAAE and samples synthesized after nine iterations ($x^{(9)}$) of MC sampling in Fig. 4(b), and these samples display good variation, capturing multiple modes of the data generating distribution.

2) Sampling (Sprites): In alignment with expectation, the iDAAE model synthesizes samples with higher (better) log-likelihood 2122 ± 5 than the AAE 2085 ± 5. The initial image samples drawn from the DAAE model underperform compared with the AAE model 2056 ± 5, and however, after just one iteration of sampling, the synthesized samples have



(a) (b)

Fig. 4. Omniglot MC sampling. (a) Initial sample $x^{(0)}$ and (b) *corresponding* samples $x^{(9)}$ after nine iterations of MC sampling. The chain was initialized with $z^{(0)} \sim \mathcal{N}(0, I)$.



(a) (b)

Fig. 5. CelebA iDAAE samples. (a) AAE (no noise) with $\rho = 0.1$. (b) iDAAE (with noise) with $\rho = 0.1$, $\sigma = 0.25$, and $M = 5$.

(a) (b) (c)

Fig. 6. DAAE face samples $\sigma = 1.0$. (a) $x^{(0)}$. (b) $x^{(5)}$. (c) $x^{(20)}$.

higher log-likelihood than samples from the AAE. Results also show that the synthesized samples drawn using the DAAE after one iteration of MC sampling have higher log-likelihood 2261 ± 5 than the samples drawn using either the iDAAE or AAE models.

When more than one step of MC sampling is applied, the log-likelihood decreases, in a similar way to results on the Omniglot evaluation data set. This may be related to how the training and test data are split; each data set has a unique set of characters, so combinations seen during training will not be presented in the testing data set. These results further suggest that MC sampling pushes synthesized samples toward the modes in the training data.

3) Sampling (CelebA): In Fig. 5, we compare the samples synthesized using an AAE to those synthesized using an iDAAE trained using $M = 5$ integration steps. Fig. 5(b) shows the samples drawn from the iDAAE, which improve upon those drawn from the AAE model.

In Fig. 6, we show samples synthesized from a DAAE using the iterative approach described in Section V-B for $M = \{0, 5, 20\}$. We see that the initial samples $x^{(0)}$ have blurry artifacts, while the final samples $x^{(20)}$ are sharper and free from the blurry artifacts.

When drawing samples from iDAAE and DAAE models trained on CelebA, a critical difference between the two models emerges: samples synthesized using a DAAE have good structure but appear to be quite similar to each other, while the iDAAE samples have poorer structure but appear to have variation. The lack of variation in the DAAE samples may be related to the sampling procedure, which according to theory presented by Alain and Bengio [1], would be similar to taking steps toward the highest density regions of the distribution (i.e., the mode), explaining why samples appear to be quite similar.

When comparing DAAE or iDAAE samples to samples from other generative models such as GANs [9], we may notice that samples are less sharp. However, GANs often suffer from "mode collapse"; this is where all synthesized samples are very similar. The iDAAE does not suffer mode collapse and does not require any additional procedures to prevent mode collapse [26]. Furthermore, (vanilla) GANs do not offer an encoding model. Other GAN variants, such as Bi-GAN [6] and adversarially learned inference (ALI) [7], do offer encoding models, but the fidelity of reconstruction is very poor. The AAE, DAAE, and iDAAE models are able to reconstruct samples faithfully. We will explore fidelity of reconstruction in Section VII-E and compare with the state-of-the-art ALI that has been modified to have improved reconstruction fidelity, known as Adversarially Learned Inference With Conditional Entropy (ALICE) [18].

We conclude this section on sampling by making the following observations; samples synthesized using iDAAEs outperformed AAEs on all data sets, where $M = 5$. It is convenient that relatively small M yields improvement, as the time needed to train an iDAAE may increase with M (see Fig. 11 in the Appendix). We also observed that initial samples synthesized using the DAAE are poor, and in all cases, even just one iteration of MC sampling improves image synthesis.

Finally, evaluating generated samples is challenging: log-likelihood is not always reliable [29], and qualitative analysis is subjective. For this reason, we provided both the quantitative and qualitative results to communicate the benefits of introducing MC sampling for a trained DAAE and the advantages of iDAAEs over AAEs.

E. Reconstruction

The reconstruction task involves passing a sample from the test data set through the trained encoder and decoder to recover a sample similar to the (uncorrupted) original. The reconstruction is evaluated by computing the mean squared error between the reconstruction and original samples.

We are interested in reconstruction for several reasons. The first is that if we wish to use encodings for downstream tasks, for example, classification, a good indication of whether the encoding is representing the sample well is to check the reconstructions. For example, if the reconstructed image is missing certain features that were present in the original, it may be that this information is not preserved in the encoding. The second reason is that checking sample reconstructions is also a method to evaluate whether the model has overfit to training samples. The ability to reconstruct samples not seen during training suggests that a model has not overfit. The final

TABLE I

RECONSTRUCTION SHOWS THE MEAN SQUARED ERROR FOR RECONSTRUCTIONS OF CORRUPTED TEST DATA SAMPLES. THIS TABLE SERVES TWO PURPOSES: 1) TO DEMONSTRATE THAT IN MOST CASES, THE DAAE AND IDAAE ARE BETTER ABLE TO RECONSTRUCT IMAGES COMPARED WITH THE AAE AND 2) TO MOTIVATE WHY WE ARE INTERESTED IN AAEs, AS OPPOSED TO OTHER GAN [9] RELATED APPROACHES. WE COMPARE RECONSTRUCTION ERROR ON MNIST FOR THE STATE-OF-THE-ART GAN VARIANT, ALICE [18], DESIGNED TO IMPROVE RECONSTRUCTION FIDELITY IN GAN-LIKE MODELS. THE MNIST DATA SET AND EXPERIMENTS ARE DESCRIBED IN THE APPENDIX

Model	Omniglot	Sprite	MNIST	CelebA
AAE	0.047	0.019	0.017	0.500
DAAE	**0.029**	0.019	**0.015**	0.501
iDAAE	0.031	**0.018**	0.018	**0.495**
ALICE [18]	-	-	0.080	

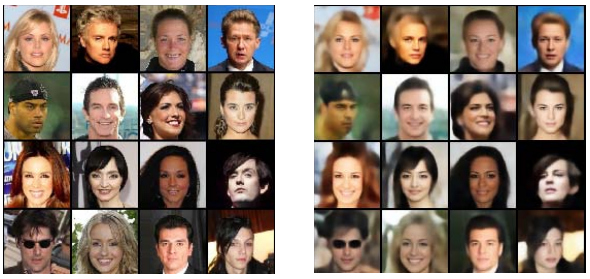
reason is to further motivate AAE, DAAE, and iDAAE models as alternatives to GAN-based models that are augmented with encoders [18], for downstream tasks that require good sample reconstruction.

We expect that adding noise during training would both prevent overfitting and encourage the model to learn more robust representations; therefore, we expect that the DAAE and iDAAE would outperform the AAE.

1) Reconstruction (Omniglot): Table I compares the reconstruction errors of the AAE, DAAE, and iDAAE trained on the Omniglot data set. The reconstruction errors for both the iDAAE and the DAAE are less than the AAE. The results suggest that using the denoising criterion during training helps the network learn more robust features compared with the nondenoising variant. The smallest reconstruction error was achieved by the DAAE rather than the iDAAE; qualitatively, the reconstructions using the DAAE captured small details while the iDAAE lost some. This is likely to be related to the multimodal nature of $\tilde{q}_\phi(z|x)$ in the DAAE compared with the unimodal nature of $\tilde{q}_\phi(z|x)$ in an iDAAE.

2) Reconstruction (Sprites): Table I shows the reconstruction error on samples from the sprite test data set for models trained on the sprite training data. In this case, only the iDAAE model outperformed the AAE and the DAAE performed as well as the AAE.

3) Reconstruction (CelebA): Table I shows the reconstruction error on the CelebA data set. We compare AAE, DAAE, and iDAAE models trained with momentum $\rho = 0.1$, where the DAAE and iDAAE have corruption $\sigma = 0.1$ and the iDAAE is trained with $M = 10$ integration steps. We also experimented with $M = 5$ but better results were obtained using $M = 10$. While the DAAE performs similarly well to the AAE, the iDAAE outperforms both. Fig. 7 shows examples of reconstructions obtained using the iDAAE. Although the reconstructions are slightly blurred, they are highly faithful, suggesting that facial attributes are correctly encoded by the iDAAE model.



(a) (b)

Fig. 7. CelebA reconstruction with an iDAAE. (a) Original. (b) Reconstructions.

TABLE II

OMNIGLOT CLASSIFICATION ON ALL 964 TEST SET CLASSES AND ON 20 EVALUATION CLASSES

Model	Test Acc.	Eval Acc. %
AAE	18.36%	78.75 %
DAAE	31.74%	**83.00**%
iDAAE	**34.02**%	78.25%
PCA	31.02%	76.75%
Random chance	0.11%	5%

F. Classification

We are motivated to understand the properties of the representations (latent encoding) learned by the DAAE and iDAAE trained on the unlabeled data. A particular property of interest is the separability, in latent space, between objects of different classes. To evaluate separability, rather than training in a semisupervised fashion [20], we obtain class predictions by training an SVM on top of the representations, in a similar fashion to that of Kumar *et al.* [16].

1) Classification (Omniglot): Classifying samples in the Omniglot data set is very challenging: the training and testing data sets consists of 946 classes, with only 19 examples of each class in the training data set. The 946 classes make up 30 writing systems, where symbols between writing systems may be visually indistinguishable. Previous work has focused on only classifying 5, 15, or 20 classes from within a single writing system [8], [17], [27], [32]; instead, we attempt to perform classification across all 946 classes. The Omniglot training data set is used to train SVMs [with radial basis function (RBF) kernels] on encodings extracted from the encoding models of the trained DAAE, iDAAE, and AAE models. Classification scores are reported on the Omniglot evaluation data set (see Table II).

Results show that the DAAE and iDAAE outperform the AAE on the classification task. The DAAE and iDAAE also outperform a classifier trained on encodings obtained by applying principle component's analysis (PCA) to the image samples, while the AAE does not, further showing the benefits of using denoising.

We perform a separate classification task using only 20 classes from the Omniglot evaluation data set (each class has 20 examples). This second test is performed for two key reasons: 1) to study how well autoencoders trained on only a subset of classes can generalize as feature extractors for

classifiers of classes not seen during *autoencoder* training and 2) to facilitate performance comparisons with previous work [8]. A linear SVM classifier is trained on the 19 samples from each of the 20 classes in the evaluation data set and tested on the remaining one sample from each class. We perform the classification 20 times, leaving out a different sample from each class, in each experiment. The results are shown in Table II. For comparison, we also show classification scores when PCA is used as a feature extractor instead of a learned encoder.

Results show that the DAAE model outperforms the AAE model, while the iDAAE performs less well, suggesting that features learned by the DAAE transfer better to new tasks than those learned by the iDAAE. The AAE, iDAAE, and DAAE models also outperform PCA.

2) Classification (CelebA): We perform a more extensive set of experiments to evaluate the linear separability of encodings learned on the celebA data set and compare to the state-of-the-art methods, including the VAE [14] and the β-VAE[3] [10].

We train a linear SVM on the encodings of a DAAE (or iDAAE) to predict labels for facial attributes, for example, "Blond Hair" and "No Beard." In our experiments, we compare classification accuracy on 12 attributes obtained using the AAE, DAAE, and iDAAE to previously reported results obtained for the VAE [14] and the β-VAE [10], and these are shown in Fig. 8. The results for the VAE and β-VAE were obtained using a similar approach to ours and were reported by Kumar *et al.* [16]. We used the same hyperparameters to train all models and a fixed noise level of $\sigma = 0.25$, and the iDAAE was trained with $M = 10$. Fig. 8 shows that the AAE, iDAAE, and DAAE models outperform the VAE and β-VAE models on most facial attribute categories.

To compare models more easily, we ask the question, "On how many facial attributes does one model outperform another?" in the context of facial attribute classification. We ask this question for various combinations of model pairs, and the results are shown in Fig. 9. Fig. 9(a) and (b), comparing the AAE to DAAE and iDAAE, respectively, demonstrates that for some attributes, the denoising models outperform the nondenoising models. Moreover, the particular attributes for which the DAAE and iDAAE outperform the AAE are (fairly) consistent, and both DAAE and iDAAE outperform the AAE on the (same) attributes: "Attractive," "Blond Hair," "Wearing Hat," and "Wearing Lipstick." The iDAAE outperforms on an additional attribute, "Arched Eyebrows."

There are various hyperparameters that may be chosen to train these models; for the DAAE and iDAAE, we may choose the level of corruption, and for the iDAAE, we may additionally choose the number of integration steps M used during training. We compare attribute classification results for three vastly different choices of parameter settings. The results are presented as a bar chart in Fig. 10 for the DAAE. Additional results for the iDAAE are shown in the Appendix (see Fig. 15).

[3]The β-VAE [10] weights the KL term in the VAE cost function with $\beta > 1$ to encourage better organization of the latent space, factorizing the latent encoding into interpretable, independent components.

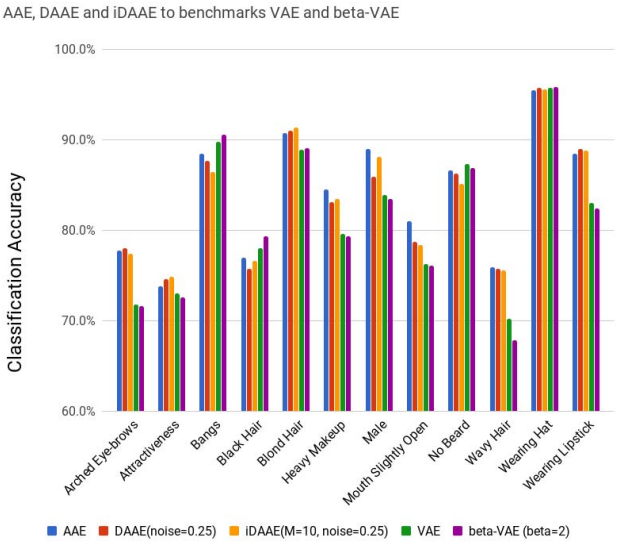


Fig. 8. Facial Attribute Classification. Comparison of classification scores for an AAE, DAAE, and iDAAE compared with the VAE [14] and β-VAE [10]. A Linear SVM classifier is trained on encodings to demonstrate the linear separability of representation learned by each model. The attribute classification values for the VAE and β-VAE were obtained from Kumar *et al.* [16].

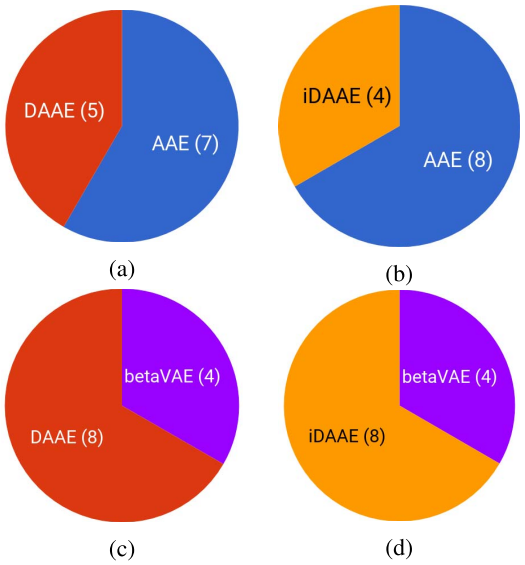


Fig. 9. *On how many facial attributes does one model outperform another?* For each chart (a)–(d), each portion shows the number of facial attributes that each model outperforms the other model in the same chart. The number of attributes is shown in brackets.

Figs. 10 and 15 show that the models perform well under various different parameter settings. Fig. 10 suggests that the model performs better with a smaller amount of noise $\sigma = \{0.1, 0.25\}$ rather than with $\sigma = 1.0$, and however, it is important to note that a large amount of noise does not "break" the model. These results demonstrate that the model works well for various hyperparameters, and fine-tuning is not necessary to achieve reasonable results (when compared with the VAE for example). It is possible that further fine-tuning may be done to achieve better results, and however, a full parameter sweep is highly computationally expensive.

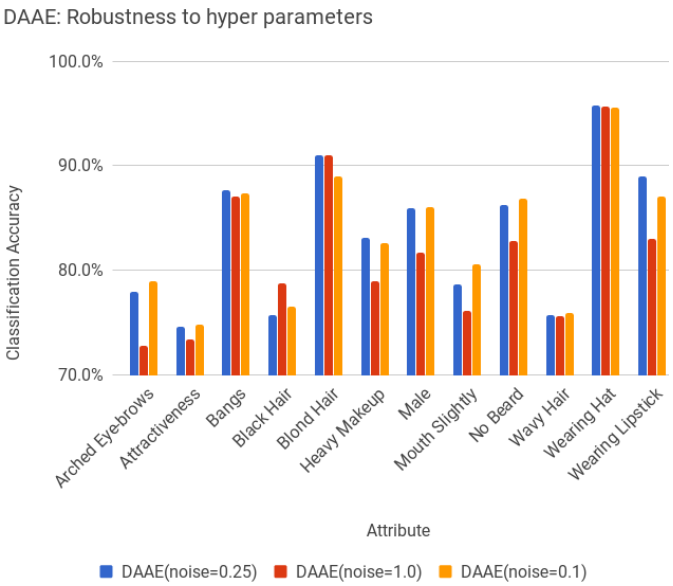


Fig. 10. DAAE Robustness to hyperparameters.

From this section, we may conclude that with the exception of three facial attributes, AAEs and variations of AAEs are able to outperform the VAE and β-VAE on the task of facial attribute classification. This suggests that AAEs and their variants are interesting models to study in the setting of learning linearly separable encodings. We also show that for a *specific set* of several facial attribute categories, the iDAAE or DAAE performs better than the AAE. This consistency suggests that there are some specific attributes that the denoising variants of the AAE learn better than the nondenoising AAE.

G. Tradeoffs in Performance

The results presented in this section suggest that both the DAAE and the iDAAE outperform the AAE models on most generation and some reconstruction tasks and suggest that it is sometimes beneficial to incorporate denoising into the training of AAEs. However, it is less clear which of the two new models, DAAE or iDAAE, are better for classification. When evaluating which one to use, we must consider both the practicalities of training and for generative purposes, the practicalities—primarily computational load—of each model.

The integrating steps required for training an iDAAE means that it may take longer to train than a DAAE (see Fig. 11 in the Appendix). On the other hand, it is possible to perform the integration process in parallel, provided that the sufficient computational resource is available. Furthermore, once the model is trained, the time taken to compute encodings for classification is the same for both the models. Finally, results suggest that using as few as $M = 5$ integrating steps during training leads to an improvement in classification score. This means that for some classification tasks, it may be worthwhile to train an iDAAE rather than a DAAE.

For generative tasks, neither the DAAE nor the iDAAE model consistently outperforms the other in terms of log-likelihood of synthesized samples. The choice of model may be more strongly affected by the computational effort required during training or sampling. In terms of log-likelihood on the synthesized samples, an iDAAE using even a small number of integration steps ($M = 5$) during training of an iDAAE leads to better quality images being generated, and similarly, using even one step of sampling with a DAAE leads to better generations.

Conflicting log-likelihood values of generated samples between testing and evaluation data sets means that these measurements are not a clear indication of how the number of sampling iterations affects the visual quality of samples synthesized using a DAAE. In some cases, it may be necessary to visually inspect samples in order to assess the effects of multiple sampling iterations (see Fig. 4).

VIII. Conclusion

We propose two types of DAEs, where a posterior is shaped to match a prior using adversarial training. In the first, we match the posterior conditional on corrupted data samples to the prior; we call this model a DAAE. In the second, we match the posterior, conditional on original data samples, to the prior. We call the second model an iDAAE because the approach involves using Monte Carlo integration during training.

Our first contribution is the extension of AAEs to denoising AAEs (DAAEs and iDAAEs). Our second contribution includes identifying and addressing challenges related to synthesizing data samples using the DAAE models. We propose synthesizing data samples by iteratively sampling a DAAE according to an MC transition operator, defined by the learned encoder and decoder of the DAAE model, and the corruption process used during training.

Finally, we present results on three data sets for three tasks that compare representations of both DAAE and iDAAE to AAE models. The data sets include: handwritten characters (Omniglot [17]), a collection of humanlike sprite characters (Sprites [25]), and a data set of faces (CelebA [19]). The tasks are reconstruction, classification, and sample synthesis.

Appendix A
Classification Results

Table III shows the numerical facial attribute classification results, corresponding to Fig. 8.

Appendix B
Time Comparison for Various M Values

Fig. 11 shows the average times taken to run a 100 training iterations of batch size 128 for an iDAAE with different numbers of integration steps M. Note that for $M = 1$, the iDAAE is equivalent to the DAAE. Models were trained on a Linux machine running Ubuntu 14.0, using an Nvidia Tesla K80 GPU (11.4GB) and CUDA 8.0.61.

Appendix C
Proofs

Lemma 1: $\mathcal{P}(z)$ is a stationary distribution for the MC defined by the sampling process in (3).

TABLE III
FACIAL ATTRIBUTE CLASSIFICATION RESULTS. COMPARISON OF (%) CLASSIFICATION SCORES FOR AN AAE, DAAE, AND IDAAE COMPARED WITH THE VAE [14] AND β-VAE [10]. A LINEAR SVM CLASSIFIER IS TRAINED ON ENCODINGS TO DEMONSTRATE THE LINEAR SEPARABILITY OF REPRESENTATION LEARNED BY EACH MODEL. THE ATTRIBUTE CLASSIFICATION VALUES FOR THE VAE AND β-VAE WERE OBTAINED FROM KUMAR *et al.* [16]

Facial Attribute	AAE	DAAE	iDAAE	β-VAE
Arched Eye-brows	77.8	**78.0**	77.4	71.6
Attractiveness	73.8	74.6	**74.9**	72.6
Bangs	88.5	87.7	86.5	**90.6**
Black Hair	77.0	75.7	76.6	**79.3**
Blond Hair	90.8	91.0	**91.4**	89.1
Heavy Makeup	**84.5**	83.1	83.5	79.3
Male	**89.0**	85.9	88.1	83.5
Mouth Slightly Open	**81.0**	78.7	78.4	76.1
No Beard	86.6	86.3	85.1	**86.9**
Wavy Hair	**75.9**	75.7	75.6	67.8
Wearing Hat	95.5	95.8	95.6	**95.9**
Wearing Lipstick	88.5	**89.0**	88.8	82.4

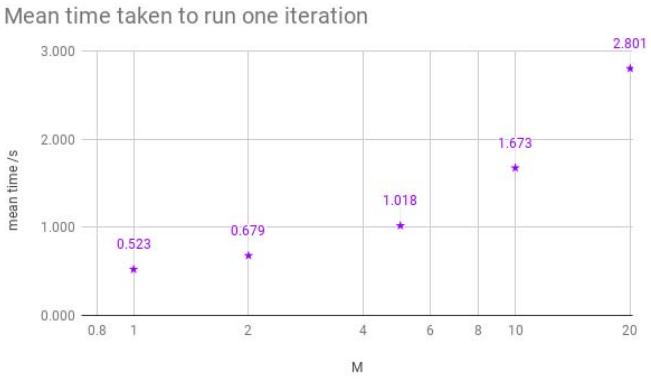


Fig. 11. Time take to run experiments for the iDAAE. M is the number of integration steps. Note that an iDAAE with $M = 1$ is equivalent to a DAAE. The batch size used in these experiments was 128.

Proof: Consider the case where $z^{(0)} \sim \mathcal{P}(z)$. $x^{(0)} \sim p_\theta(x|z^{(0)})$ is from $\mathcal{P}(x)$, by (5). Following the sampling process, $\tilde{x}^{(0)} \sim c(\tilde{x}|x^{(0)})$, $z^{(0)} \sim q_\phi(z|\tilde{x}^{(0)})$, $z^{(1)}$ is also from $\mathcal{P}(z)$, by (6). *See Bengio et al. [4] for a related argument*. Therefore, $\mathcal{P}(z)$ is a stationary distribution of the MC defined by (3). □

Lemma 2: The MC defined by the transition operator $T_{\theta,\phi}(z_{t+1}|z_t)$ (4) is ergodic, under the assumption that the corruption process is additive Gaussian noise and that the adversarial pair $q_\phi(z|\tilde{x})$ and $d_\chi(z)$ are optimal within the adversarial framework.

Proof: Consider $X = \{x : \mathcal{P}(x) > 0\}$, $\tilde{X} = \{\tilde{x} : c(\tilde{x}|x) > 0\}$, and $Z = \{z : \mathcal{P}(z) > 0\}$, where $Z \subseteq \{z : p(z) > 0\}$ and $X \subseteq \tilde{X}$. We make the following assumptions:

1) Assuming that $p_\theta(x|z)$ is a good approximation of the underlying probability distribution $\mathcal{P}(x|z)$, then $\forall x_j \sim \mathcal{P}(x)\ \exists\ z_i \sim \mathcal{P}(z)$, such that $p_\theta(x_j|z_i) > 0$.
2) Assuming that adversarial training has shaped the distribution of $q_\phi(z|\tilde{x})$ to match the prior $p(z)$, then $\forall z_i \sim p(z)\ \exists\ \tilde{x}_j$, such that $q_\phi(z_i|\tilde{x}_j) > 0$. This holds because if not all points in $p(z)$ could be visited, $q_\phi(z|\tilde{x})$ would not have matched the prior.

1) suggests that every point in X may be reached from a point in Z and 2) suggests that every point in Z may be reached from a point in $\tilde{X}$.

Under the assumption that $c(\tilde{x}|x)$ is an additive Gaussian corruption process, then $\tilde{x}_i$ is likely to lie within a (hyper) spherical region around x_i. If the corruption process is sufficiently large such that (hyper) spheres of nearby x samples overlap, for x_i and x_{i+m} $\exists$ a set $\{x_{i+1}, \ldots, x_{i+m-1}\}$ such that $\text{supp}(c(\tilde{x}|x_i)) \cap \text{supp}(c(\tilde{x}|x_{i+1})) \neq \emptyset, \forall i = 1, \ldots, (m-1)$ and where supp is the support.

Then, it is possible to reach any z_i from any z_j (including the case $j = i$). Therefore, the chain is both irreducible and positive recurrent.

To be ergodic, the chain must also be aperiodic: between any two points x_i and x_j, there is a boundary, where x values between x_i and the boundary are mapped to z_i, and the points between x_j and the boundary are mapped to z_j. By applying the corruption process to $x^{(t)} = x_i$, followed by the reconstruction process, there are always at least two possible outcomes, because we assume that all (hyper) spheres induced by the corruption process overlap with at least one other (hyper) sphere: either $\tilde{x}^{(t)}$ is not pushed over the boundary and $z^{(t+1)} = z_i$ remains the same, or $\tilde{x}^{(t)}$ is pushed over the boundary and $z^{(t+1)} = z_j$ moves to a new state. The probability of either outcome is positive, and so there is always more than one route between two points, thus avoiding periodicity provided that for $x_i \neq x_j$ and $z_i \neq z_j$. Considering the case where for $x_i \neq x_j$ and $z_i = z_j$, then it would not be possible to recover both x_i and x_j using $\mathcal{P}(x|z)$, and so if $\mathcal{P}(x_i|z) > 0$, then $\mathcal{P}(x_j|z) = 0$ (and vice versa), which is a contradiction to 1). □

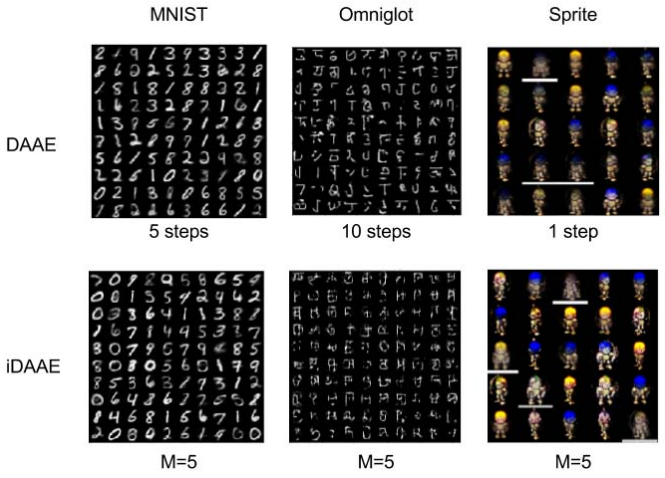


Fig. 12. Examples of synthesized samples. Examples of randomly synthesized data samples.

APPENDIX D
EXAMPLES OF SYNTHESIZED SAMPLES

See Fig. 12.

APPENDIX E
ALGORITHM FOR MONTE CARLO INTEGRATION

See Algorithm 2.

Algorithm 2 Drawing Samples From $\tilde{q}_\phi(z|x)$

function: approx_z($\{x_0, x_2, \ldots x_{N-1}\}$)
for $i = 0$ *to* $N-1$ **do**
  $\hat{z}_{i+1} = [\ ]$
  for $j = 1$ *to* M **do**
    $\tilde{x}_{i,j} = C(x_i)$
    $z_{i+1,j} = E_\phi(\tilde{x}_j)$
  end
  $\hat{z}_{i+1} = \frac{1}{M}\sum_{j=1}^{M} z_j$
end
return $\hat{\mathbf{z}} = \{\hat{z}_1, \hat{z}_2 \ldots \hat{z}_N\}$

APPENDIX F
MNIST RESULTS

A. Data Sets (MNIST)

The MNIST data set consists of gray-scale images of handwritten digits between 0 and 9, with 50k training samples, 10k validation samples, and 10k testing samples. The training, validation, and testing data sets have an equal number of samples from each category. The samples are 28×28 pixels. The MNIST data set is a simple data set with a few classes and many training examples, making it a good data set for proof-of-concept. However, because the data set is very simple, it does not necessarily reveal the effects of subtle, but potentially important, changes to algorithms for training or sampling. For this reason, we consider two data sets with greater complexity.

B. Architecture and Training (MNIST)

For the MNIST data set, we train a total of five models detailed in Table IV. The encoder, decoder, and discriminator networks each have two fully connected layers with 1000 neurons each. For most models, the size of the encoding is 10 units, and the prior distribution that the encoding is being matched to is a 10-D Gaussian.

All networks are trained for 100 epochs on the training data set, with a learning rate of 0.0002 and a batch size of 64. The standard deviation of the additive Gaussian noise used during training is 0.5 for all iDAAE and DAAE models.

An additional DAAE model is trained using the same training parameters and networks as described earlier but with a mixture of 10 2-D Gaussians for the prior. Each 2-D Gaussian with a standard deviation of 0.5 is equally spaced with its mean around a circle of radius 4 units. This results in a prior with 10 modes separated from each other by large regions of very low probability. This model of the prior is very unrealistic, as it assumes that MNIST digits occupy distinct regions of image probability space. In reality, we may expect two numbers that are similar to exist side-by-side in image probability space, and for there to exist a smooth transition between handwritten digits. As a consequence of this, this model is intended specifically for evaluating how well the posterior distribution over latent space may be matched to a mixture of Gaussians—something that could not be achieved easily by a VAE [14].

TABLE IV
MODELS TRAINED ON MNIST. FIVE MODELS ARE TRAINED ON THE MNIST DATA SET. *CORRUPTION* INDICATES THE STANDARD DEVIATION OF GAUSSIAN NOISE ADDED DURING THE CORRUPTION PROCESS $c(\tilde{x}|x)$. *PRIOR* INDICATED THE PRIOR DISTRIBUTION IMPOSED ON THE LATENT SPACE. M IS THE NUMBER OF MONTE CARLO INTEGRATION STEPS (SEE ALGORITHM 2 IN THE APPENDIX) USED DURING TRAINING—THIS APPLIES ONLY TO THE IDAAE

ID	Model	Corruption	Prior	M
1	AAE	0.0	10D Gaussian	-
2	DAAE	0.5	10D Gaussian	-
3	DAAE	0.5	10-GMM	-
4	iDAAE	0.5	10D Gaussian	5
5	iDAAE	0.5	10D Gaussian	25

TABLE V
MNIST RECONSTRUCTION. *RECON.* SHOWS THE MEAN SQUARED ERROR FOR RECONSTRUCTIONS OF CORRUPTED TEST DATA SAMPLES ACCOMPANIED BY THE STANDARD ERROR. *CORRUPTION* IS THE STANDARD DEVIATION OF THE ADDITIVE GAUSSIAN NOISE USED DURING TRAINING AND TESTING

Model & Training				Recon.
Model	Corruption	Prior	M	Mean $\pm$ s.e.
AAE	0.0	10D Gaussian	-	0.017±0.001
DAAE	0.5	10D Gaussian	-	0.023±0.001
DAAE	0.5	2D 10-GMM	-	0.043±0.001
iDAAE	0.5	10D Gaussian	5	0.022±0.001
iDAAE	0.5	10D Gaussian	25	0.026±0.001

C. Reconstruction (MNIST)

Table V shows the reconstruction error on samples from the testing data set, for each of the models trained on the MNIST training data set. Reconstruction error for the DAAE and iDAAE trained with a 10-D Gaussian prior does not outperform the AAE. However, the reconstruction task for the AAE model was less challenging than that of the DAAE and iDAAE models because samples were not corrupted before the encoding step. The best model for reconstruction was the iDAAE with $M = 5$, increasing M to 25 led to worse reconstruction error, as reconstructions tended to appear more like mean samples.

Reconstruction error for the DAAE trained using a 2-D mixture of 10 Gaussians underperformed compared with the rest of the models. All reconstructions looked like mean images, which may be expected given the nature of the prior.

D. Sampling (MNIST)

To calculate the log-likelihood of samples drawn from DAAE, iDAAE, and AAE models trained on the MNIST data

TABLE VI

MNIST LOG-LIKELIHOOD $\mathbb{L}_\theta$. TO CALCULATE THE LOG-LIKELIHOOD $\mathbb{L}_\theta$ OF A MODEL $p_\theta(x)$, A PARZEN WINDOW WAS FIT TO 10^4 GENERATED SAMPLES AND THE MEAN LOG-LIKELIHOOD WAS REPORTED FOR THE TESTING DATA SET. THE BANDWIDTH USED FOR THE PARZEN WINDOW WAS DETERMINED USING A VALIDATION SET. THE TRAINING, TEST, AND VALIDATION DATA SETS HAD DIFFERENT SAMPLES

Model & Training				Log-likelihood	
Model	Corruption	Prior	M	$x^{(0)}$	$x^{(5)}$
AAE	0.0	10D Gaussian	-	529	-
DAAE	0.5	10D Gaussian	-	529	444
DAAE	0.5	2D 10-GMM	-	9	205
iDAAE	0.5	10D Gaussian	5	532	-
iDAAE	0.5	10D Gaussian	25	508	-

set, a Parzen window is fitted to 1×10^4 generated samples from a single trained model. The bandwidth for constructing the Parzen window was selected by choosing one of 10 values, evenly space along a log axis between -1 and 0, that maximizes the likelihood on the validation data set. The log-likelihood for each model was then evaluated on the test data set.

Table VI shows the log-likelihood on the test data set for samples drawn from models trained on the MNIST training data set. First, we will discuss models trained with a 10-D Gaussian prior. The best log-likelihood was achieved by the iDAAE with $M = 5$. Synthesized samples generated by MC sampling from the DAAE model caused the log-likelihood to decrease. This may be because the samples tended toward mean samples and dropping modes, causing the log-likelihood to decrease. Initial samples and those obtained after five iterations of sampling are shown in Fig. 13. The samples in 13(d) are clearer than in 13(c). Although mode dropping is not immediately apparent, note that the digit 4 is not present after five iterations.

Now, we consider the DAAE model trained with a 2-D, 10-Gaussian mixture model prior. Samples drawn from the prior are shown in Fig. 14(a). The purpose of this experiment was to show the effects of MC sampling, where the distribution from which initial samples of z_0 are drawn is significantly different to the prior. Samples of z_0 were drawn from a normal distribution and passed through the decoder of the DAAE model to produce initial image samples [see Fig. 14(c)]. A further nine steps of MC sampling were applied to synthesize the samples shown in Fig. 14(d). As expected, the initial image samples do not look like MNIST digits, and the MC sampling improves samples dramatically. Unfortunately, many of the samples appear to correspond to the samples at modes of the data distribution. In addition, several modes appear to be missing from the model distribution. This may be attributed to the nature of the prior since we did not encounter this problem to the same extent when using a 10-D Gaussian prior (see Fig. 13).

Synthesizing MNIST samples is fairly trivial since there are many training examples and a few classes. For this reason, it is difficult to see the benefits of using DAAE or iDAAE models

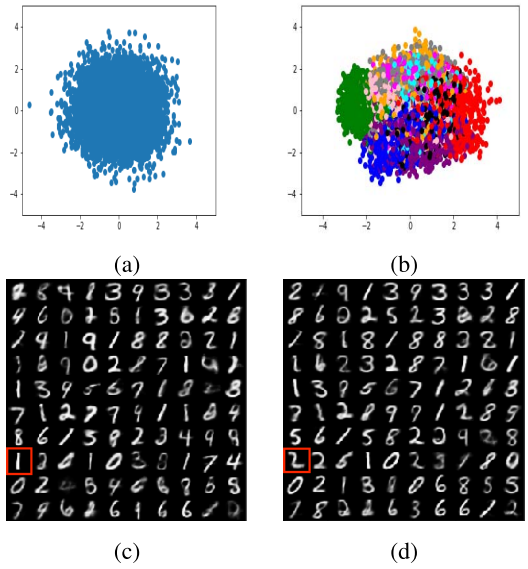


Fig. 13. MNIST MC sampling with 10-D Gaussian prior. (a) Two dimensions of the 10-D Gaussian prior used to train the DAAE. (b) Samples drawn from $q_\phi(z|\tilde{x})$, projected into 2-D space, and color coded by the digit labels. (c) Initial samples $x^{(0)}$ generated from the prior illustrated by (a). (d) Corresponding samples after five iterations of MC sampling. Notice how the highlighted "1" changes to a "2" after five iterations of MC sampling.

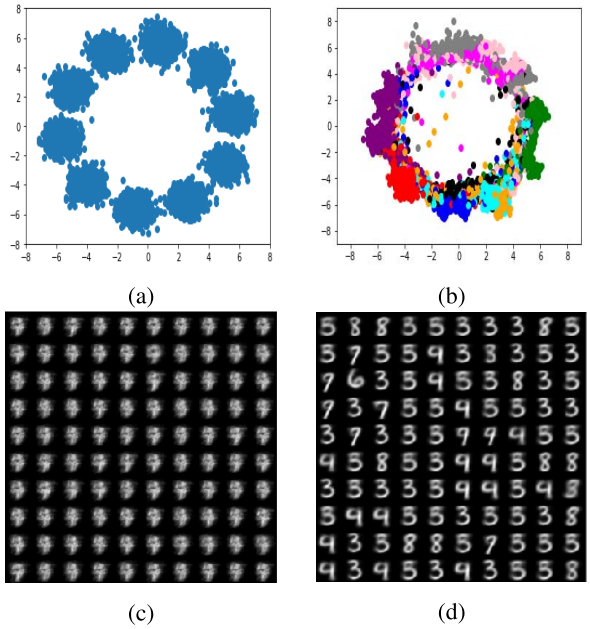


Fig. 14. MNIST MC sampling. (a) 2-D Gaussian mixture prior used to train a DAAE. (b) Samples drawn from $q_\phi(z|\tilde{x})$ and color coded by the digit labels. (c) Initial sample $x^{(0)}$ generated by a normal distribution. (d) *Corresponding* samples generated after nine further iterations of MC sampling.

compared with AAE models. Now, we focus on the two more complex data sets, Omniglot and Sprites. We find that iDAAE models and correctly sampled DAAE models may be used to synthesize samples with higher log-likelihood than samples synthesized using an AAE.

1) Classification (MNIST): The MNIST data set consists of ten classes $[0, 9]$, and the classification task involves

TABLE VII
MNIST CLASSIFICATION: SVM CLASSIFIERS WITH RBF KERNELS WERE TRAINED ON ENCODED MNIST TRAINING DATA SAMPLES. THE SAMPLES WERE ENCODED USING THE ENCODER OF THE TRAINED AAE, DAAE, OR IDAAE MODELS. CLASSIFICATION SCORES ARE GIVEN FOR THE MNIST TEST DATA SET

Model & Training				Accuracy
Model	Corruption	Prior	M	%
AAE	0.0	10D Gaussian	-	95.19%
DAAE	0.5	10D Gaussian	-	95.28%
DAAE	0.5	2D 10-GMM	-	73.81%
iDAAE	0.5	10D Gaussian	5	96.48%
iDAAE	0.5	10D Gaussian	25	96.24%
PCA + SVM				94.73%

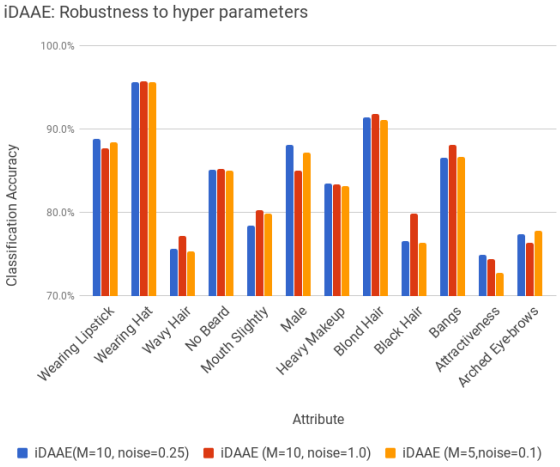


Fig. 15. iDAAE—Robustness to hyperparameters.

correctly predicting a label in this interval. For the MNIST data set, the SVM classifier is trained on encoded samples from the MNIST training data set and evaluated on encoded samples from the MNIST testing data set; results are shown in Table VII.

First, we consider the results for DAAE, iDAAE, and AAE models trained with a 10-D Gaussian prior. Classifiers trained on encodings extracted from the encoders of trained DAAE, iDAAE, or AAE models outperformed classifiers trained on PCA of image samples. Classifiers trained on the encodings extracted from the encoders of learned DAAE and iDAAE models outperformed those trained on the encodings extracted from the encoders of the AAE model.

The differences in classification score for each model on the MNIST data set are small; this might be because it is relatively easy to classify MNIST digits with very high accuracy [28].

APPENDIX G
DETAILS FOR CALCULATING LOG-LIKELIHOOD

A. Omniglot

To calculate log-likelihood of samples, a Parzen window was fit to 1×10^3 synthesized samples, where the bandwidth was determined on the testing data set in a similar way to that in Appendix F-D. The log-likelihood was evaluated on both the evaluation data set and the testing data set. To compute the log-likelihood on the of the testing data set, a Parzen window was fit to a new set of synthesized samples, different to those used to calculate the bandwidth. The results are shown in Fig. 3.

B. Sprites

To calculate log-likelihood of samples, a Parzen window was fit to 1×10^3 synthesized samples. The bandwidth was set as previously described in Appendix F-D; we found the optimal bandwidth to be 1.29.

APPENDIX H
IDAAE ROBUSTNESS TO HYPERPARAMETERS

See Fig. 15.

ACKNOWLEDGMENT

The authors would like to thank K. Arulkumaran for interesting discussions and managing the compute cluster on which many experiments were performed. They would like to thank Dr. L. Hadjilucas for discussions on the SVM results. They would also like to thank N. Pawlowski and M. Rajchl for additional help with cluster access.

REFERENCES

[1] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data-generating distribution," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3563–3593, 2014.
[2] P. Bachman and D. Precup, "Variational generative stochastic networks with collaborative shaping," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 1964–1972.
[3] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.
[4] Y. Bengio, É. Thibodeau-Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *Proc. 31st Int. Conf. Mach. Learn.*, vol. 32, 2014, pp. II-226–II-234.
[5] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized denoising auto-encoders as generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 899–907.
[6] J. Donahue, P. Krähenbühl, and T. Darrell. (2016). "Adversarial feature learning." [Online]. Available: https://arxiv.org/abs/1605.09782
[7] V. Dumoulin *et al.* (2016). "Adversarially learned inference." [Online]. Available: https://arxiv.org/abs/1606.00704
[8] H. Edwards and A. Storkey. (2016). "Towards a neural statistician." [Online]. Available: https://arxiv.org/abs/1606.02185
[9] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[10] I. Higgins *et al.*, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[12] D. J. Im, S. Ahn, R. Memisevic, and Y. Bengio, "Denoising criterion for variational auto-encoding framework," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2059–2065.
[13] D. P. Kingma and L. J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.
[14] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[16] A. Kumar, P. Sattigeri, and A. Balakrishnan. (2017). "Variational inference of disentangled latent concepts from unlabeled observations." [Online]. Available: https://arxiv.org/abs/1711.00848

[17] B. M. Lake, R. Salakhutdinov, and J. B. Tenebaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.
[18] C. Li *et al.*, "Alice: Towards understanding adversarial learning for joint distribution matching," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5495–5503.
[19] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 3730–3738.
[20] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. (2015). "Adversarial autoencoders." [Online]. Available: https://arxiv.org/abs/1511.05644
[21] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. (2016). "Plug & play generative networks: Conditional iterative generation of images in latent Space." [Online]. Available: https://arxiv.org/abs/1612.00005
[22] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Sampling* (Markov Chain Convergence). Harrisburg, PA, USA: Harrisburg Univ. Sci. Technol., 2005, ch. 1, p. 5.
[23] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
[24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.
[25] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1252–1260.
[26] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. (2016). "Improved techniques for training GANs." [Online]. Available: https://arxiv.org/abs/1606.03498
[27] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. (2016). "One-shot learning with memory-augmented neural networks." [Online]. Available: https://arxiv.org/abs/1605.06065
[28] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. ICDAR*, vol. 3, Aug. 2003, pp. 958–962.
[29] L. Theis, A. van de Oord, and M. Bethge, "A note on the evaluation of generative models," in *Proc. Int. Conf. Learn. Represent.*, 2015.
[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ACM 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[31] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.
[32] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3630–3638.

Antonia Creswell received the M.Eng. degree in biomedical engineering from Imperial College London, London, U.K., where she is currently pursuing the Ph.D. degree with the Department of Bioengineering.

Anil Anthony Bharath received the B.Eng. degree in electronic and electrical engineering from University College London, London, U.K., in 1988, and the Ph.D. degree in signal processing from Imperial College London, London, in 1993.

He was an Academic Visitor with the Signal Processing Group, University of Cambridge, Cambridge, U.K., in 2006. He is a Co-Founder of Cortexica Vision Systems, London. He is currently a Reader with the Department of Bioengineering, Imperial College London, where he is also a fellow of the Data Science Institute. His current research interests include deep architectures for visual inference.

Dr. Bharath is a fellow of the Institution of Engineering and Technology.