Deep AutoRegressive Networks

Karol Gregor KAROLG@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Andriy Mnih AMNIH@GOOGLE.COM
Charles Blundell CBLUNDELL@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

Abstract

We introduce a deep, generative autoencoder capable of learning hierarchies of distributed representations from data. Successive deep stochastic hidden layers are equipped with autoregressive connections, which enable the model to be sampled from quickly and exactly via ancestral sampling. We derive an efficient approximate parameter estimation method based on the minimum description length (MDL) principle, which can be seen as maximising a variational lower bound on the log-likelihood, with a feedforward neural network implementing approximate inference. We demonstrate state-of-the-art generative performance on a number of classic data sets: several UCI data sets, MNIST and Atari 2600 games.

1. Introduction

Directed generative models provide a fully probabilistic account of observed random variables and their latent representations. Typically either the mapping from observation to representation or representation to observation is intractable and hard to approximate efficiently. In contrast, autoencoders provide an efficient two-way mapping where an encoder maps observations to representations and a decoder maps representations back to observations. Recently several authors (Ranzato et al., 2007; Vincent et al., 2008; Vincent, 2011; Rifai et al., 2012; Bengio et al., 2013b) have developed probabilistic versions of regularised autoencoders, along with means of generating samples from such models. These sampling procedures are often iterative, producing correlated *approximate* samples from previous approximate samples, and as such explore the full distribution slowly, if at all.

In this paper, we introduce Deep AutoRegressive Networks (DARNs), deep generative autoencoders that in contrast to the aforementioned models efficiently generate independent, *exact* samples via ancestral sampling. To produce a sample, we simply perform a top-down pass through the decoding part of our model, starting at the deepest hidden layer and sampling one unit at a time, layer-wise. Training a DARN proceeds by minimising the total information stored for reconstruction of the original input, and as such follows the minimum description length principle (MDL; Rissanen, 1978). This amounts to backpropagating an MDL cost through the entire joint encoder/decoder.

There is a long history of viewing autoencoders through the lens of MDL (Hinton & Van Camp, 1993; Hinton & Zemel, 1994), yet this has not previously been considered in the context of deep autoencoders. MDL provides a sound theoretical basis for DARN's regularisation, whilst the justification of regularised autoencoders was not immediately obvious. Learning to encode and decode observations according to a compression metric yields representations that can be both concise and irredundant from an information theoretic point of view. Due to the equivalence of compression and prediction, compressed representations are good for making predictions and hence also good for generating samples. Minimising the description length of our model coincides exactly with minimising the Helmholtz variational free energy, where our encoder plays the role of the variational distribution. Unlike many other variational learning algorithms, our algorithm is not an expectation maximisation algorithm, but rather a stochastic gradient descent method, jointly optimising all parameters of the autoencoder simultaneously.

DARN and its learning algorithm easily stack, allowing ever deeper representations to be learnt, whilst at the same time compressing the training data — DARN allows for alternating layers of stochastic hidden units and deterministic non-linearities. Each stochastic layer within DARN is

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

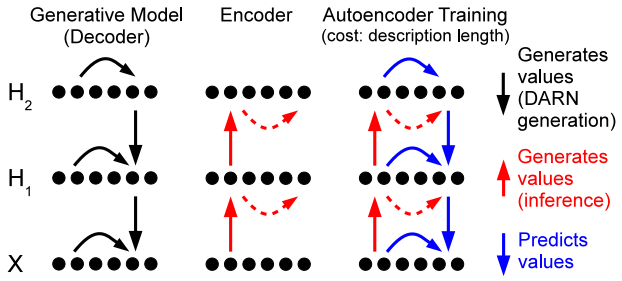

Figure 1. **Left:** DARN's decoder as a generative model. Top-down, ancestral sampling through DARN's decoder starts with the deepest stochastic hidden layer H_2, sampling each unit in turn before proceeding downwards to lower layers, ending by producing an observation X. **Centre:** DARN's encoder as inference. Conditioned upon the observation X, and sampling left-to-right, bottom-up, DARN's encoder infers the representation H_1, H_2 of an observation. **Right:** DARN as an autoencoder. During training, the encoder infers a suitable representation H_1, H_2 and the decoder predicts this representation and the observation. The parameters of the encoder and decoder are then simultaneously minimised with respect to the implied coding cost. This cost equals the Helmholtz variational free energy.

autoregressive: each unit receives input both from the preceding layer and the preceding units within the same layer. Autoregressive structure captures much of the dependence among units within the same layer, at very little computational cost during both learning and generation. This is in marked contrast to other mechanisms for lateral connections, such as introducing within-layer undirected edges, which often come at a prohibitively high computational cost at training and/or generation time.

Recently, several authors have exploited autoregression for distribution modelling (Larochelle & Murray, 2011; Gregor & LeCun, 2011; Uria et al., 2013). Unlike these models, DARN can have *stochastic* hidden units, and places autoregressive connections among these hidden units. Depending upon the architecture of the network, this can yield gains in both statistical and computational efficiency.

The remainder of the paper is structured as follows. In Section 2 we describe the architecture of our model, Section 3 reviews the minimum description length principle and its application to autoencoders. Section 4 describes the approximate parameter estimation algorithm. Section 5 has the results of our model on several data sets, and we conclude with a brief summary in Section 6.

2. Model Architecture

Our model is a deep, generative autoencoder; an example is shown in Figure 1 with two hidden layers. DARN has three components: the encoder $q(H|X)$ that picks a representation H for a given observation X, a decoder prior $p(H)$ which provides a prior distribution on representations H for generation, and a decoder conditional $p(X|H)$ which, given a representation H, produces an observation X. We shall use uppercase letters for random variables and lowercase for their values. We shall first describe our model with a single stochastic hidden layer and later show how this is easily generalised to multiple layers.

We begin by describing the decoder prior on the representation h. The decoder prior on the representation h is an autoregressive model. Let $h_{1:j}$ denote the vector $(h_1, h_2, \ldots, h_j)$ where each $h_i \in \{0, 1\}$, then

$$p(h) = \prod_{j=1}^{n_h} p(h_j|h_{1:j-1}) \tag{1}$$

where $h = (h_1, h_2, \ldots, h_{n_h})$ denotes the representation with n_h hidden stochastic units. $p(h_j|h_{1:j-1})$ is the probability mass function of h_j conditioned upon the activities of the previous units in the representation $h_{1:j-1}$.

In DARN, we parameterise the conditional probability mass of h_j in a variety of ways, depending upon the complexity of the problem. Logistic regression is the simplest:

$$p(H_j = 1|h_{1:j-1}) = \sigma(W_j^{(H)} \cdot h_{1:j-1} + b_j^{(H)}), \tag{2}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$. The parameters are $W_j^{(H)} \in \mathbb{R}^{j-1}$, which is the weight vector, and $b_j^{(H)} \in \mathbb{R}$, which is a bias parameter.

The conditional distributions of the decoder $p(X|H)$ and of the encoder $q(H|X)$ have similar forms:

$$p(x|h) = \prod_{j=1}^{n_x} p(x_j|x_{1:j-1}, h), \tag{3}$$

$$q(h|x) = \prod_{j=1}^{n_x} q(h_j|h_{1:j-1}, x), \tag{4}$$

where, as with the decoder prior, the conditional probability mass functions can be parameterised as in Eq. 2 (we shall explore some more elaborate parameterisations later). Consequently,

$$p(X_j = 1|x_{1:j-1}, h) = \sigma(W_j^{(X|H)} \cdot (x_{1:j-1}, h) + b_j^{(X|H)}) \tag{5}$$

$$q(H_j = 1|h_{1:j-1}, x) = \sigma(W_j^{(H|X)} \cdot x + b_j^{(H|X)}) \tag{6}$$

where $(x_{1:j-1}, h)$ denotes the concatenation of the vector $x_{1:j-1}$ with the vector h, $W_j^{(X|H)} \in \mathbb{R}^{j-1+n_h}$ and $W_j^{(H|X)} \in \mathbb{R}^{n_x}$ are weight vector parameters and $b_j^{(H|X)}$ and $b_j^{(X|H)}$ are the scalar biases. Whilst in principle, Eq. 6 could be made autoregressive, we shall typically choose not to do so, as this can have significant computational advantages, as we shall see later in Section 2.3.

2.1. Deeper Architectures

The simple model presented so far is already a universal distribution approximator — it can approximate any (reasonable) distribution given sufficient capacity. As adding extra hidden layers to models such as deep belief networks strictly improves their representational power (Le Roux & Bengio, 2008), we could ask whether that is also the case for DARN. Although every distribution on H may be written as Eq. 1, not every factorisation can be parameterised as Eq. 2. Thus we propose boosting DARN's representational power in three ways: by adding stochastic hidden layers, by adding deterministic hidden layers, and by using alternate kinds of autoregressivity. We now consider each approach in turn.

Additional stochastic hidden layers. We consider an autoencoder with hidden stochastic layers $H^{(1)}, \ldots, H^{(n_{\text{layers}})}$ each with $n_h^{(1)}, \ldots, n_h^{(n_{\text{layers}})}$ units, respectively. For convenience we denote the input layer by $H^{(0)} = X$ and let $H^{(n_{\text{layers}}+1)} = \emptyset$. The decoder and encoder probability distributions become

$$p(H^{(l)}|H^{(l+1)}) = \prod_{j=1}^{n_h^{(l)}} p(H_j^{(l)}|H_{1:j-1}^{(l)}, H^{(l+1)}), \quad (7)$$

$$q(H^{(k)}|H^{(k-1)}) = \prod_{j=1}^{n_h^{(k)}} q(H_j^{(k)}|H_{1:j-1}^{(k)}, H^{(k-1)}) \quad (8)$$

for $l = 0, \ldots, n_{\text{layers}}$ and $k = 1, \ldots, n_{\text{layers}}$.

Additional deterministic hidden layers. The second way of adding complexity is to insert more complicated deterministic functions between the stochastic layers. This applies both to the encoder and the decoder. If we wished to add just one deterministic hidden layer, we could use a simple multi-layer perceptron such as:

$$d^{(l)} = \tanh(Uh^{(l+1)}) \quad (9)$$

$$\begin{aligned} p(H_j^{(l)} &= 1|h_{1:j-1}^{(l)}, h^{(l+1)}) \\ &= \sigma(W_j^{(H)} \cdot (h_{1:j-1}^{(l)}, d^{(l)}) + b_j^{(H)}). \end{aligned} \quad (10)$$

where $U \in \mathbb{R}^{n_d \times n_h^{(l+1)}}$ is a weight matrix, n_d is the number of deterministic hidden units, $W_j^{(H)} \in \mathbb{R}^{j-1+n_d}$ is a weight vector, and $b_j^{(H)}$ is a scalar bias.

Alternate kinds of autoregressivity. Finally, we can increase representational power by using more flexible autoregressive models, such as NADE (Larochelle & Murray, 2011) and EoNADE (Uria et al., 2013), instead of the simple linear autoregressivity we proposed in the previous section.

The amount of information that can be stored in the representation H is upper bounded by the number of *stochastic* hidden units. Additional *deterministic* hidden units *do not* introduce any extra random variables and so cannot increase the capacity of the representation, whilst additional *stochastic* hidden units *can*.

2.2. Local connectivity

DARN can be made to scale to high-dimensional (image) data by restricting connectivity, both between adjacent layers, and autoregressively, within layers. This is particularly useful for modelling larger images. Local connectivity can be either fully convolutional (LeCun et al., 1998) or use less weight sharing. In this paper we use the periodic local connectivity of Gregor & LeCun (2010) which uses less weight sharing than full convolutional networks.

2.3. Sampling

Sampling in DARN is simple and efficient as it is just ancestral sampling. We start with the top-most layer, sample the first hidden unit $h_1 \sim p(H_1^{(n_{\text{layers}})})$ and then for each i in turn, we sample $h_i \sim p(H_i^{(n_{\text{layers}})}|h_{1:i-1}^{(n_{\text{layers}})})$. We repeat this procedure for each successive layer until we reach the observation. Sampling from the encoder works in exactly the same way but in the opposite direction, sampling each hidden unit from $q(H_i^{(l)}|h_{1:i-1}^{(l)}, h^{(l-1)})$ successively.

A DARN without a stochastic hidden layer but with an autoregressive visible layer is a fully visible sigmoid belief network (FVSBN; Frey, 1998). Thus FVSBN sampling scales as $O(n_x^2)$. In NADE (Larochelle & Murray, 2011; Gregor & LeCun, 2011), autoregression is present in the visibles, but only deterministic hidden units are used. Sampling then scales as $O(n_x n_d)$ where n_x is the number of visibles and n_d is the number of deterministic hidden units. The complexity of sampling from a fully autoregressive single stochastic hidden layer DARN is $O((n_h + n_x)^2)$. If we omit the autoregressivity on the observations, we obtain a time complexity of $O(n_h(n_x + n_h))$. Furthermore, if the stochastic hidden layer is sparse, such that at n_s units are active on average, we obtain an expected time complexity of $O(n_s(n_x + n_h))$. We call this sparse, fast version fDARN. As more stochastic or deterministic hidden layers are added to DARN, the advantage of DARN becomes greater as each part of the decoder need only be computed once for DARN per datum, wheras deeper NADE-like models (Uria et al., 2013) require re-computation of large parts of the model for each unit.

3. Minimum Description Length and Autoencoders

Autoencoders have previously been trained by an MDL principle derived from bits-back coding (Hinton & Van Camp, 1993). This yields generative autoencoders trained to minimise the Helmholtz variational free energy (Hinton & Zemel, 1994). Here we extend this work to deeper, autoregressive models trained by a stochastic approximation to backpropagation as opposed to expectation maximisation.

According to the MDL principle, we shall train this autoencoder by finding parameters that try to maximally compress the training data. Suppose a sender wishes to communicate a binary sequence of n_x elements, x, to a receiver. We shall first sample a representation of n_h binary elements, h, to communicate and then send the residual of x relative to this representation. The idea is that the representation has a more concise code than the original datum x and so can be compressed effectively: for example, by arithmetic coding (MacKay, 2003).

The description length of a random variable taking a particular value indicates how many bits must be used to communicate that particular value. Shannon's source coding theorem shows that the description length is equal to the negative logarithm of the probability of the random variable taking that particular value (MacKay, 2003). Hence, when communicating a datum x, having already communicated its representation h, the description length would be

$$L(x|h) = -\log_2 p(x|h). \tag{11}$$

We wish for the parameters of the autoencoder to compress the data well on average, and so we shall minimise the expected description length,

$$L(x) = \sum_h q(h|x)(L(h) + L(x|h)), \tag{12}$$

where $L(h)$ denotes the description length of the representation h, and $q(h|x)$ is the encoder probability of the representation h. As we are using bits-back coding, the description length of the representation h is

$$L(h) = -\log_2 p(h) + \log_2 q(h|x). \tag{13}$$

Substituting Eq. 11 and Eq. 13 into Eq. 12 we recover the Helmholtz variational free energy:

$$L(x) = -\sum_h q(h|x)(\log_2 p(x,h) - \log_2 q(h|x)). \tag{14}$$

Picking the parameters of $q(H|X)$ and $p(X,H)$ to minimise the description length in Eq. 14 yields a coding scheme that requires the fewest expected number of bits to communicate a datum x and its representation h.

As Eq. 14 is the variational free energy, the encoder $q(H|X)$ that minimises Eq. 14 is the posterior $p(H|X)$. Variational learning methods sometimes refer to the negative expected description length $-L(x)$ as the expected lower bound as it serves as a lower bound upon $\log_2 p(x)$. Note here that we shall be interested in optimising the parameters of $q(H|X)$ and $p(X,H)$ simultaneously, whereas variational learning often only optimises the parameters of $q(H|X)$ and $p(X,H)$ by co-ordinate descent.

4. Learning

Learning in DARN amounts to jointly training weights and biases θ of both the encoder and the decoder, simultaneously, to minimise Eq. 12. The procedure is based on gradient descent by backpropagation and is based upon a number of approximations to the gradient of Eq. 12.

We write the expected description length in Eq. 12 as:

$$L(x) = \sum_{h_1=0}^{1} q(h_1|x) \cdots \sum_{h_{n_h}=0}^{1} q(h_{n_h}|h_{1:n_h-1}, x) \times (\log_2 q(h|x) - \log_2 p(x,h)) \tag{15}$$

Calculating Eq. 15 exactly is intractable. Hence we shall use a Monte carlo approximation.

Learning proceeds as follows:

1. Given an observation x, sample a representation $h \sim q(H|x)$ (see Section 2.3).
2. Calculate $q(h|x)$ (Eq. 4), $p(x|h)$ (Eq. 3) and $p(h)$ (Eq. 1) for the sampled representation h and given observation x.
3. Calculate the gradient of Eq. 15.
4. Update the parameters of the autoencoder by following the gradient $\nabla_\theta L(x)$.
5. Repeat.

We now turn to calculating the gradient of Eq. 15 which requires backpropagation of the MDL cost through the joint encoder/decoder. Unfortunately, this pass through the model includes stochastic units. Backpropagating gradients through stochastic binary units naïvely yields gradients that are highly biased, yet often work well in practice (Hinton, 2012). Whilst it is possible to derive estimators that are unbiased (Bengio et al., 2013a), their empirical performance is often unsatisfactory. In this work, we backpropagate gradients through stochastic binary units, and then re-

weight these gradients to reduce bias and variance. Details are given in Appendix A.

Finally note that when the encoder is not autoregressive, the entire system can be trained using standard matrix operations and point-wise nonlinearities. Hence it is easily implementable on graphical processing units. The decoder's autoregressive computation is expressed as a full matrix multiplication with a triangular matrix.

5. Results

We trained our models on binary UCI data sets, MNIST digits and frames from five Atari 2600 games (Bellemare et al., 2013).

The quantitative results reported here are in terms of the probability the decoder assigns to a test datum: $p(x)$. For small DARN models, we can evaluate the likelihood $p(x)$ exactly by iterating over every possible representation: $p(x) = \sum_h p(x, h)$. As the computational cost of this sum grows exponentially in the size of the representation, for DARN models with more than 16 stochastic hidden units, we use an importance sampling estimate using the encoder distribution:

$$p(x) \approx \frac{1}{S} \sum_{s=1}^{S} \frac{p(x, h^{(s)})}{q(h^{(s)}|x)}, \qquad h^{(s)} \sim q(H|x), \qquad (16)$$

where s indexes one of S samples. As this estimate can have high variance, we repeat the estimation ten times and report the 95 per cent confidence interval. In our experiments, the variance of the estimator was low. Where available, we used a validation set to choose the learning rate and certain aspects of the model architecture, such as the number of hidden units. We used the Monte Carlo approximation to expected description length in Eq. 15 of the validation set to select these.

5.1. Binary UCI data sets

We evaluated the test-set performance of DARN on eight binary data sets from the UCI repository (Bache & Lichman, 2013). In Table 1, we compare DARN to baseline models from Uria et al. (2013).

We used a DARN with two hidden layers. The first layer was deterministic, with tanh activations. The second layer was a stochastic layer with an autoregressive prior $p(H)$. The decoder conditional $p(X|H)$ included autoregressive connections.

The architecture and learning rate was selected by cross-validation on a validation set for each data set. The number of deterministic hidden units was selected from 100 to 500, in steps of 100, whilst the number of stochastic hidden units was selected from $\{8, 12, 16, 32, 64, 128, 256\}$. We used RMSprop (Graves, 2013) with momentum 0.9 and learning rates 0.00025, 0.0000675 or 10^{-5}. The network was trained with minibatches of size 100. The best results are shown in bold in Table 1. DARN achieved better test log-likelihood on four of eight data sets than the baseline models reported in Uria et al. (2013). We found that regularisation by adaptive weight noise on these small data sets (Graves, 2011) did not yield good results, but early stopping based on the performance on the validation set worked well.

5.2. Binarised MNIST data set

We evaluated the sampling and test-set performance of DARN on the binarised MNIST data set (Salakhutdinov & Murray, 2008), which consists of $50,000$ training, $10,000$ validation, and $10,000$ testing images of hand-written digits (Larochelle & Murray, 2011). Each image is 28×28 pixels.

We used two hidden layers, one deterministic, one stochastic. The results are in Table 2 with n_h denoting the number of stochastic hidden units. The deterministic layer had 100 units for architectures with 16 or fewer stochastic units per layer, and 500 units for more than 16 stochastic units. The deterministic activation function was taken to be the tanh function. We used no autoregressivity for the observation layer — the decoder conditional is a product of independent Bernoulli distributions, conditioned upon the representation. Training was done with RMSprop (Graves, 2013), momentum 0.9 and minibatches of size 100. We used a learning rate of 3×10^{-5}. Adaptive weight noise (Graves, 2011), denoted by "adaNoise" in Table 2, was used to avoid the need for early stopping.

After training, we were able to measure the exact log-likelihood for networks with 16 or fewer stochastic hidden units. For the network with 500 hidden units, we estimated the log-likelihood by importance sampling given by the above procedure. For each test example, we sampled $100,000$ latent representations from the encoder distribution. The estimate was repeated ten times; we report the estimated 95 per cent confidence intervals. The obtained log-likelihoods and confidence intervals are given in Table 2 along with those of other models. DARN performs favourably compared to the other models. For example, a DARN with just 9 stochastic hidden units obtains almost the same log-likelihood as a mixture of Bernoullis (MoBernoullis) with 500 components: $\log_2 500 \approx 9$. DARN with 500 stochastic hidden units compares favourably to state-of-the-art generative performance of deep Boltzmann machines (DBM; Salakhutdinov & Hinton, 2009) and deep belief networks (DBN; Salakhutdinov & Murray, 2008; Murray & Salakhutdinov, 2009). Notably, DARN's upper bound, the expected de-

Table 1. Log likelihood (in nats) per test-set example on the eight UCI data sets.

Model	Adult	Connect4	DNA	Mushrooms	NIPS-0-12	Ocr-letters	RCV1	Web
MoBernoullis	20.44	23.41	98.19	14.46	290.02	40.56	47.59	30.16
RBM	16.26	22.66	96.74	15.15	277.37	43.05	48.88	29.38
FVSBN	**13.17**	12.39	83.64	10.27	276.88	39.30	49.84	29.35
NADE (fixed order)	13.19	**11.99**	84.81	9.81	273.08	**27.22**	46.66	28.39
EoNADE 1hl (16 ord.)	13.19	12.58	**82.31**	**9.68**	**272.38**	27.31	**46.12**	**27.87**
DARN	13.19	**11.91**	**81.04**	**9.55**	274.68	28.17 ± 0	**46.10** $\pm$ **0**	28.83 ± 0

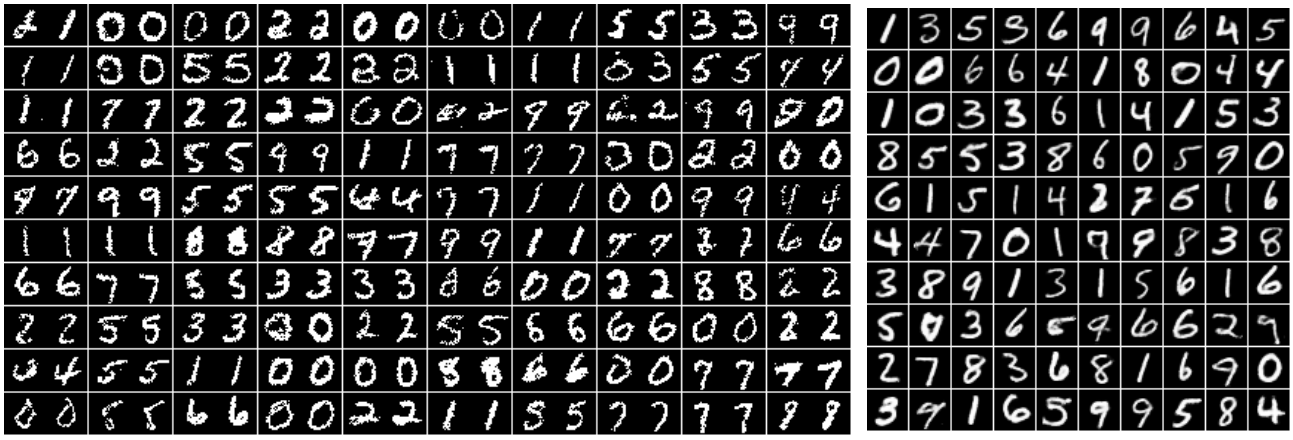

Figure 2. **Left:** Samples from DARN paired with the nearest training example from binarised MNIST. The generated samples are not simple memorisation of the training examples. **Right:** Sample probabilities from DARN trained on pixel intensities directly.

scription length given in the far right column, is lower than the likelihood of NADE whilst DARN's estimated log likelihood is lower than the log-likelihood of the best reported EoNADE (Uria et al., 2013) results.

We performed several additional experiments. First, we trained fDARN with 400 hidden units and 5% sparsity, resulting in an upper bound negative log-likelihood of 96.1 (Table 2). The estimated speed of generation was 2.4×10^4 multiplications per sample, which compares favourably to NADE's of 2.2×10^6, a nearly 100 fold speedup. While the likelihood is worse, the samples appear reasonable by ocular inspection.

Next, we trained a very deep, 12 stochastic layer DARN with 80 stochastic units in each layer, and 400 tanh units in each deterministic layer. Here we also used skip connections where each tanh layer received input from all previous stochastic layers. Due to computational constraints we only evaluated the upper bound of this architecture as reported in Table 2 — where it records the best upper bound among all DARN models, showing the value of depth in DARN.

Finally, we trained a network with one stochastic layer, 400 units, and one tanh layer (1000 units) in both encoder and decoder on the pixel intensities directly, rather than binarising the data set. We show the sample probabilities of observables in Figure 2(right).

5.3. Atari 2600 game frames

We recorded $100,000$ frames from five different Atari 2600 games (Bellemare et al., 2013) from random play, recording each frame with 1% probability. We applied an edge detector to the images yielding frames of 159×209 pixels[1]. Frames of these games generated by DARN are shown in Figure 4.

To scale DARN to these larger images, we used three hidden layers. The stack of layers contains a locally connected layer with weight-sharing (see Section 2.2), a rectified linear activation function, another locally connected layer followed by a rectified linear function and a fully connected layer followed by 300 stochastic binary units. The autoregressive prior on the representation was also fully connected. The locally connected layers had 32 filters with a period of 8. The first locally connected layer used stride 4 and kernel size 8. The second locally connected layer used stride 2 and kernel size 4. The autoregressive connections

[1]The script to generate the dataset is available at `https://github.com/fidlej/aledataset`

Table 2. Log likelihood (in nats) per test-set example on the binarised MNIST data set. The right hand column, where present, represents the expected description length (in Eq. 15) which serves as an upper bound on the log-likelihood for DARN.

Model	$-\log p$	$\leq$
MoBernoullis K=10	168.95	
MoBernoullis K=500	137.64	
RBM (500 h, 25 CD steps)	≈ 86.34	
DBM 2hl	≈ 84.62	
DBN 2hl	$\approx$ **84.55**	
NADE 1hl (fixed order)	88.33	
EoNADE 1hl (128 orderings)	87.71	
EoNADE 2hl (128 orderings)	85.10	
DARN n_h=9, adaNoise	138.84	145.36
DARN n_h=10, adaNoise	133.70	140.86
DARN n_h=16, adaNoise	122.80	130.94
DARN n_h=500	84.71 ± 0.01	90.31
DARN n_h=500, adaNoise	$\mathbf{84.13 \pm 0.01}$	88.30
fDARN $n_h = 400$	-	96.1
deep DARN	-	**87.72**

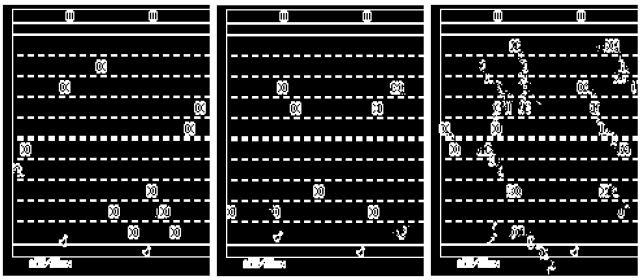

Figure 3. Samples from networks of different depths. **Left:** Frames generated from the network described in Figure 5. **Middle:** Frames generated from the network with the same sizes but without the fully-connected top layer. **Right:** Frames generated from the network with the top two layers removed.

for the visible layer used a period of 14 and kernel size 15.

The games in Figure 4 are ordered from left to right, in decreasing order of log probability. While DARN captures much of the structure in these games, such as the scores, various objects, and correlations between objects, when the game becomes less regular, as with River Raid (second to right) and Sea Quest (far right), DARN is not able to generate reasonable frames.

Figure 5 shows the representation that DARN learns for a typical frame of the game Freeway. To show the effect of deeper layers in DARN, Figure 3 shows the frames DARN generates using the representation from different depths. In the game of Freeway, several cars (the white blobs) travel along lanes, delimited by dashed white lines. Here we used a DARN with three stochastic hidden layers. When DARN was trained using a sparsity penalty on the activations, as we described in Section 2.3, it learnt a representation where the second hidden layer captures the rough outline of each car, whereas the first hidden layer filled in the details of each car. A global bias learns the background image. A third hidden layer decides where to place the cars. All layers except the very top layer are locally connected. The top layer has 100 units. The second hidden layer was locally connected with a kernel of size 21×21, whilst the first hidden layer was locally connected with a kernel of size 7×7.

6. Conclusion

In this paper we introduced deep autoregressive networks (DARN), a new deep generative architecture with autoregressive stochastic hidden units capable of capturing high-level structure in data to generate high-quality samples. The method, like the ubiquitous autoencoder framework, is comprised of not just a decoder (the generative element) but a stochastic encoder as well to allow for efficient and tractable inference. Training proceeds by backpropagating an MDL cost through the joint model, which approximately equates to minimising the Helmholtz variational free energy. This procedure necessitates backpropagation through stochastic units, as such yielding an approximate Monte Carlo method. The model samples efficiently, trains efficiently and is scalable to locally connected and/or convolutional architectures. Results include state-of-the-art performance on multiple data sets.

A. Derivation of Gradients

We derive the gradient of the objective function with respect to the inputs to a stochastic *binary* hidden unit. Let $q(h_i)$ be the probability distribution of the ith hidden unit, $f(h_i)$ be the part of the network which takes h_i as an input. The expected value of $f(h_i)$ and its gradient are:

$$\mathbb{E}\left[f(H_i)\right] = \sum_{h_i=0}^{1} q(h_i) f(h_i) \tag{17}$$

$$\nabla_\theta \mathbb{E}\left[f(H_i)\right] = \sum_{h_i=0}^{1} q(h_i) \nabla_\theta \log q(h_i) f(h_i) \tag{18}$$

where f does not depend directly on the inputs θ of the $q(h_i)$ distribution. We use a Monte Carlo approximation to estimate $\nabla_\theta \mathbb{E}\left[f(H_i)\right]$ where h_i is sampled from $q(h_i)$ (Williams, 1992; Andradóttir, 1998). Monte Carlo approximations of the gradient are unbiased but can have high variance. To combat the high variance, we introduce a baseline b, inspired by control variates (Paisley et al., 2012):

$$\nabla_\theta \mathbb{E}\left[f(H_i)\right] \approx \mathbb{E}\left[\widehat{\nabla_\theta F}\right] \tag{19}$$

$$\widehat{\nabla_\theta F} = \nabla_\theta \log q(h_i)(f(h_i) - b) \tag{20}$$

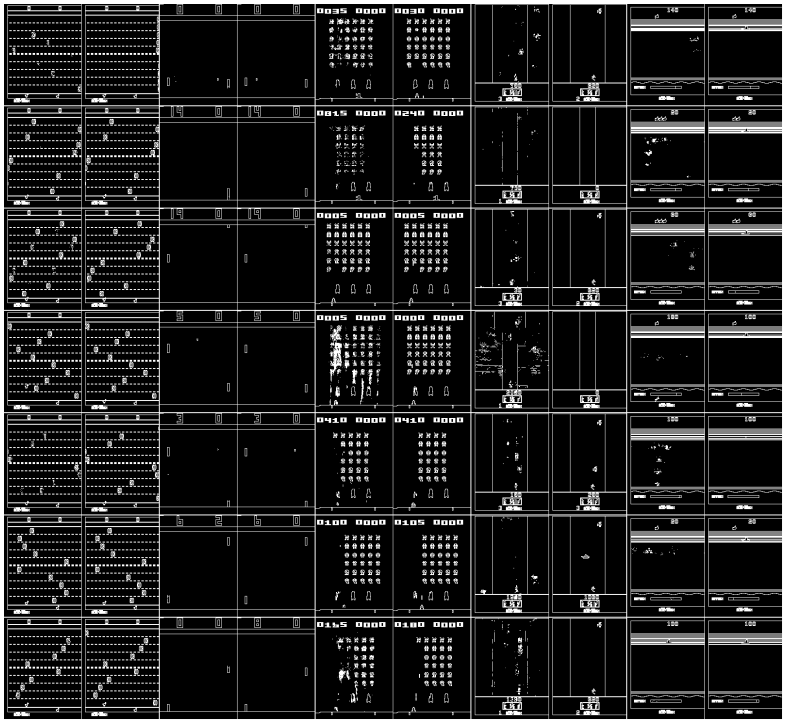

Figure 4. Samples from a locally connected DARN paired with the nearest training example. Some of the generated frames have novel combinations of objects and scores. Columns from left to right, along with upper bound negative log-likelihood on the test set: Freeway (19.9), Pong (23.7), Space Invaders (113.0), River Raid (139.4), Sea Quest (217.9).

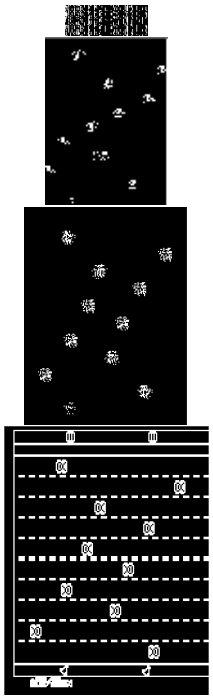

Figure 5. **Bottom:** An input frame from the game Freeway. **Lower Middle:** Activations in the encoder from the first hidden layer. **Upper Middle:** Activations in the encoder from the second hidden layer. **Top:** Each of 25 rows is an activation of the fully-connected third hidden layer with 100 units.

where $\widehat{\nabla_\theta F}$ denotes our estimator. A good baseline should be correlated with $f(h_i)$, have low variance, and also be such that the expected value of $\nabla_\theta \log q(h_i) b$ is zero to get an unbiased estimate of the gradient. We chose a *non-constant* baseline. The baseline will be a first-order Taylor approximation of f. We can get the first-order derivatives from backpropagation. The baseline is a Taylor approximation of f about h_i, evaluated at a point h_i':

$$b(h_i) = f(h_i) + \frac{df(h_i)}{dh_i}(h_i' - h_i) \tag{21}$$

To satisfy the unbiasedness requirement, we need to solve the following equation for h_i':

$$0 = \sum_{h_i=0}^{1} q(h_i) \nabla_\theta \log q(h_i) (f(h_i) + \frac{df(h_i)}{dh_i}(h_i' - h_i)) \tag{22}$$

The solution depends on the shape of f. If f is a linear function, any h_i' can be used. If f is a quadratic function, the solution is $h_i' = \frac{1}{2}$. If f_i is a cubic or higher-order function, the solution depends on the coefficients of the polynomial. We will use $h_i' = \frac{1}{2}$ and our estimator will be biased for non-quadratic functions.

By substituting the baseline into Eq. 20 we obtain the final form of our estimator of the gradient:

$$\widehat{\nabla_\theta F} = \nabla_\theta \log q(h_i) \frac{df(h_i)}{dh_i}(h_i - \frac{1}{2}) \tag{23}$$

$$= \frac{\nabla_\theta q(H_i = 1)}{2q(h_i)} \frac{df(h_i)}{dh_i} \tag{24}$$

An implementation can estimate the gradient with respect to $q(H_i = 1)$ by backpropagating with respect to h_i and scaling the gradient by $\frac{1}{2q(h_i)}$, where h_i is the sampled binary value.

References

Andradóttir, Sigrún. A review of simulation optimization techniques. In *Simulation Conference Proceedings, 1998. Winter*, volume 1, pp. 151–158. IEEE, 1998.

Bache, K. and Lichman, M. UCI ma-

chine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 06 2013.

Bengio, Yoshua, Léonard, Nicholas, and Courville, Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013a.

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. *arXiv preprint arXiv:1305.6663*, 2013b.

Frey, Brendam J. *Graphical models for machine learning and digital communication*. The MIT press, 1998.

Graves, Alex. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

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

Gregor, Karol and LeCun, Yann. Emergence of complex-like cells in a temporal product network with local receptive fields. *arXiv preprint arXiv:1006.0448*, 2010.

Gregor, Karol and LeCun, Yann. Learning representations by maximizing compression. *arXiv preprint arXiv:1108.1169*, 2011.

Hinton, Geoffrey. Neural networks for machine learning. Coursera video lectures, 2012.

Hinton, Geoffrey E and Van Camp, Drew. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

Hinton, Geoffrey E and Zemel, Richard S. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pp. 3–3, 1994.

Larochelle, Hugo and Murray, Iain. The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 15:29–37, 2011.

Le Roux, Nicolas and Bengio, Yoshua. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

MacKay, David JC. *Information theory, inference and learning algorithms, Ch 6*. Cambridge university press, 2003.

Murray, Iain and Salakhutdinov, Ruslan. Evaluating probabilities under high-dimensional latent variable models. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1137–1144. 2009.

Paisley, John, Blei, David, and Jordan, Michael. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Ranzato, Marc'Aurelio, Boureau, Y-lan, and Cun, Yann L. Sparse feature learning for deep belief networks. In *Advances in neural information processing systems*, pp. 1185–1192, 2007.

Rifai, Salah, Bengio, Yoshua, Dauphin, Yann, and Vincent, Pascal. A generative process for sampling contractive auto-encoders. 2012.

Rissanen, Jorma. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

Salakhutdinov, Ruslan and Murray, Iain. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pp. 872–879. ACM, 2008.

Uria, Benigno, Murray, Iain, and Larochelle, Hugo. A deep and tractable density estimator. *arXiv preprint arXiv:1310.1757*, 2013.

Vincent, Pascal. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7): 1661–1674, 2011.

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

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.