DRAW: A Recurrent Neural Network For Image Generation

Karol Gregor KAROLG@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Alex Graves GRAVESA@GOOGLE.COM
Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

Abstract

This paper introduces the *Deep Recurrent Attentive Writer* (DRAW) neural network architecture for image generation. DRAW networks combine a novel spatial attention mechanism that mimics the foveation of the human eye, with a sequential variational auto-encoding framework that allows for the iterative construction of complex images. The system substantially improves on the state of the art for generative models on MNIST, and, when trained on the Street View House Numbers dataset, it generates images that cannot be distinguished from real data with the naked eye.

1. Introduction

A person asked to draw, paint or otherwise recreate a visual scene will naturally do so in a sequential, iterative fashion, reassessing their handiwork after each modification. Rough outlines are gradually replaced by precise forms, lines are sharpened, darkened or erased, shapes are altered, and the final picture emerges. Most approaches to automatic image generation, however, aim to generate entire scenes at once. In the context of generative neural networks, this typically means that all the pixels are conditioned on a single latent distribution (Dayan et al., 1995; Hinton & Salakhutdinov, 2006; Larochelle & Murray, 2011). As well as precluding the possibility of iterative self-correction, the "one shot" approach is fundamentally difficult to scale to large images. The *Deep Recurrent Attentive Writer* (DRAW) architecture represents a shift towards a more natural form of image construction, in which parts of a scene are created independently from others, and approximate sketches are successively refined.

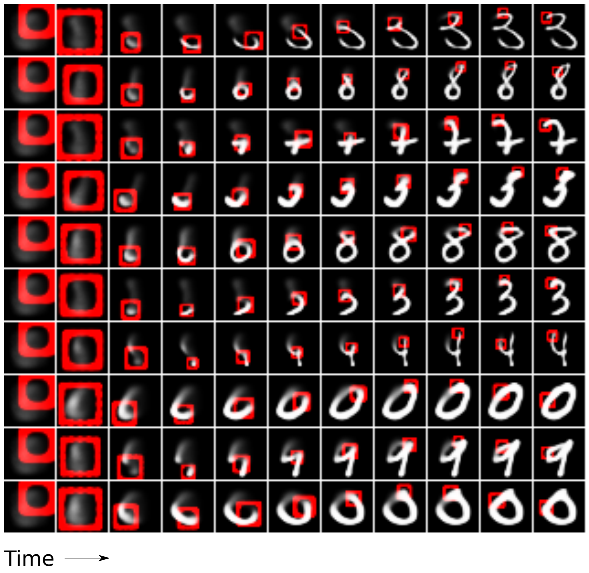

Figure 1. **A trained DRAW network generating MNIST digits.** Each row shows successive stages in the generation of a single digit. Note how the lines composing the digits appear to be "drawn" by the network. The red rectangle delimits the area attended to by the network at each time-step, with the focal precision indicated by the width of the rectangle border.

The core of the DRAW architecture is a pair of recurrent neural networks: an *encoder* network that compresses the real images presented during training, and a *decoder* that reconstitutes images after receiving codes. The combined system is trained end-to-end with stochastic gradient descent, where the loss function is a variational upper bound on the log-likelihood of the data. It therefore belongs to the family of *variational auto-encoders*, a recently emerged hybrid of deep learning and variational inference that has led to significant advances in generative modelling (Gregor et al., 2014; Kingma & Welling, 2014; Rezende et al., 2014; Mnih & Gregor, 2014; Salimans et al., 2014). Where DRAW differs from its siblings is that, rather than generat-

Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

ing images in a single pass, it iteratively constructs scenes through an accumulation of modifications emitted by the decoder, each of which is observed by the encoder.

An obvious correlate of generating images step by step is the ability to selectively attend to parts of the scene while ignoring others. A wealth of results in the past few years suggest that visual structure can be better captured by a sequence of partial glimpses, or foveations, than by a single sweep through the entire image (Larochelle & Hinton, 2010; Denil et al., 2012; Tang et al., 2013; Ranzato, 2014; Zheng et al., 2014; Mnih et al., 2014; Ba et al., 2014; Sermanet et al., 2014). The main challenge faced by sequential attention models is learning where to look, which can be addressed with reinforcement learning techniques such as policy gradients (Mnih et al., 2014). The attention model in DRAW, however, is fully differentiable, making it possible to train with standard backpropagation. In this sense it resembles the selective read and write operations developed for the Neural Turing Machine (Graves et al., 2014).

The following section defines the DRAW architecture, along with the loss function used for training and the procedure for image generation. Section 3 presents the selective attention model and shows how it is applied to reading and modifying images. Section 4 provides experimental results on the MNIST, Street View House Numbers and CIFAR-10 datasets, with examples of generated images; and concluding remarks are given in Section 5. Lastly, we would like to direct the reader to the video accompanying this paper (https://www.youtube.com/watch?v=Zt-7MI9eKEo) which contains examples of DRAW networks reading and generating images.

2. The DRAW Network

The basic structure of a DRAW network is similar to that of other variational auto-encoders: an encoder network determines a distribution over latent codes that capture salient information about the input data; a decoder network receives samples from the code distribuion and uses them to condition its own distribution over images. However there are three key differences. Firstly, both the encoder and decoder are recurrent networks in DRAW, so that a *sequence* of code samples is exchanged between them; moreover the encoder is privy to the decoder's previous outputs, allowing it to tailor the codes it sends according to the decoder's behaviour so far. Secondly, the decoder's outputs are successively added to the distribution that will ultimately generate the data, as opposed to emitting this distribution in a single step. And thirdly, a dynamically updated attention mechanism is used to restrict both the input region observed by the encoder, and the output region modified by the decoder. In simple terms, the network decides at each time-step "where to read" and "where to write" as well as "what to write". The architecture is sketched in Fig. 2, alongside a feedforward variational auto-encoder.

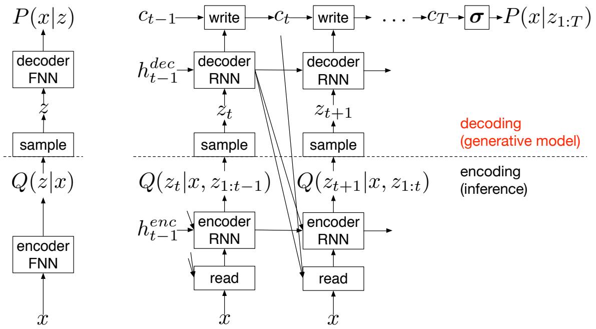

Figure 2. **Left: Conventional Variational Auto-Encoder**. During generation, a sample z is drawn from a prior $P(z)$ and passed through the feedforward decoder network to compute the probability of the input $P(x|z)$ given the sample. During inference the input x is passed to the encoder network, producing an approximate posterior $Q(z|x)$ over latent variables. During training, z is sampled from $Q(z|x)$ and then used to compute the total description length $KL\big(Q(Z|x)||P(Z)\big) - \log(P(x|z))$, which is minimised with stochastic gradient descent. **Right: DRAW Network**. At each time-step a sample z_t from the prior $P(z_t)$ is passed to the recurrent decoder network, which then modifies part of the canvas matrix. The final canvas matrix c_T is used to compute $P(x|z_{1:T})$. During inference the input is read at every time-step and the result is passed to the encoder RNN. The RNNs at the previous time-step specify where to read. The output of the encoder RNN is used to compute the approximate posterior over the latent variables at that time-step.

2.1. Network Architecture

Let RNN^{enc} be the function enacted by the encoder network at a single time-step. The output of RNN^{enc} at time t is the encoder hidden vector h^{enc}_t. Similarly the output of the decoder RNN^{dec} at t is the hidden vector h^{dec}_t. In general the encoder and decoder may be implemented by any recurrent neural network. In our experiments we use the *Long Short-Term Memory* architecture (LSTM; Hochreiter & Schmidhuber (1997)) for both, in the extended form with *forget gates* (Gers et al., 2000). We favour LSTM due to its proven track record for handling long-range dependencies in real sequential data (Graves, 2013; Sutskever et al., 2014). Throughout the paper, we use the notation $b = W(a)$ to denote a linear weight matrix with bias from the vector a to the vector b.

At each time-step t, the encoder receives input from both the image x and from the previous decoder hidden vector h^{dec}_{t-1}. The precise form of the encoder input depends on a *read* operation, which will be defined in the next section. The output h^{enc}_t of the encoder is used to parameterise a distribution $Q(Z_t|h^{enc}_t)$ over the latent vector z_t. In our experiments the latent distribution is a diagonal Gaussian $\mathcal{N}(Z_t|\mu_t, \sigma_t)$:

$$\mu_t = W(h_t^{enc}) \tag{1}$$

$$\sigma_t = \exp\left(W(h_t^{enc})\right) \tag{2}$$

Bernoulli distributions are more common than Gaussians for latent variables in auto-encoders (Dayan et al., 1995; Gregor et al., 2014); however a great advantage of Gaussian latents is that the gradient of a function of the samples with respect to the distribution parameters can be easily obtained using the so-called *reparameterization trick* (Kingma & Welling, 2014; Rezende et al., 2014). This makes it straightforward to back-propagate unbiased, low variance stochastic gradients of the loss function through the latent distribution.

At each time-step a sample $z_t \sim Q(Z_t|h_t^{enc})$ drawn from the latent distribution is passed as input to the decoder. The output h_t^{dec} of the decoder is added (via a *write* operation, defined in the sequel) to a cumulative *canvas* matrix c_t, which is ultimately used to reconstruct the image. The total number of time-steps T consumed by the network before performing the reconstruction is a free parameter that must be specified in advance.

For each image x presented to the network, $c_0, h_0^{enc}, h_0^{dec}$ are initialised to learned biases, and the DRAW network iteratively computes the following equations for $t = 1 \ldots, T$:

$$\hat{x}_t = x - \boldsymbol{\sigma}(c_{t-1}) \tag{3}$$

$$r_t = read(x_t, \hat{x}_t, h_{t-1}^{dec}) \tag{4}$$

$$h_t^{enc} = RNN^{enc}(h_{t-1}^{enc}, [r_t, h_{t-1}^{dec}]) \tag{5}$$

$$z_t \sim Q(Z_t|h_t^{enc}) \tag{6}$$

$$h_t^{dec} = RNN^{dec}(h_{t-1}^{dec}, z_t) \tag{7}$$

$$c_t = c_{t-1} + write(h_t^{dec}) \tag{8}$$

where $\hat{x}_t$ is the *error image*, $[v, w]$ is the concatenation of vectors v and w into a single vector, and $\boldsymbol{\sigma}$ denotes the logistic sigmoid function: $\boldsymbol{\sigma}(x) = \frac{1}{1+\exp(-x)}$. Note that h_t^{enc}, and hence $Q(Z_t|h_t^{enc})$, depends on both x and the history $z_{1:t-1}$ of previous latent samples. We will sometimes make this dependency explicit by writing $Q(Z_t|x, z_{1:t-1})$, as shown in Fig. 2. h^{enc} can also be passed as input to the $read$ operation; however we did not find that this helped performance and therefore omitted it.

2.2. Loss Function

The final canvas matrix c_T is used to parameterise a model $D(X|c_T)$ of the input data. If the input is binary, the natural choice for D is a Bernoulli distribution with means given by $\boldsymbol{\sigma}(c_T)$. The *reconstruction loss* $\mathcal{L}^x$ is defined as the negative log probability of x under D:

$$\mathcal{L}^x = -\log D(x|c_T) \tag{9}$$

The *latent loss* $\mathcal{L}^z$ for a sequence of latent distributions $Q(Z_t|h_t^{enc})$ is defined as the summed Kullback-Leibler divergence of some latent prior $P(Z_t)$ from $Q(Z_t|h_t^{enc})$:

$$\mathcal{L}^z = \sum_{t=1}^{T} KL\big(Q(Z_t|h_t^{enc})||P(Z_t)\big) \tag{10}$$

Note that this loss depends upon the latent samples z_t drawn from $Q(Z_t|h_t^{enc})$, which depend in turn on the input x. If the latent distribution is a diagonal Gaussian with μ_t, σ_t as defined in Eqs 1 and 2, a simple choice for $P(Z_t)$ is a standard Gaussian with mean zero and standard deviation one, in which case Eq. 10 becomes

$$\mathcal{L}^z = \frac{1}{2}\left(\sum_{t=1}^{T} \mu_t^2 + \sigma_t^2 - \log \sigma_t^2\right) - T/2 \tag{11}$$

The total loss $\mathcal{L}$ for the network is the expectation of the sum of the reconstruction and latent losses:

$$\mathcal{L} = \langle \mathcal{L}^x + \mathcal{L}^z \rangle_{z \sim Q} \tag{12}$$

which we optimise using a single sample of z for each stochastic gradient descent step.

$\mathcal{L}^z$ can be interpreted as the number of nats required to transmit the latent sample sequence $z_{1:T}$ to the decoder from the prior, and (if x is discrete) $\mathcal{L}^x$ is the number of nats required for the decoder to reconstruct x given $z_{1:T}$. The total loss is therefore equivalent to the expected compression of the data by the decoder and prior.

2.3. Stochastic Data Generation

An image $\tilde{x}$ can be generated by a DRAW network by iteratively picking latent samples $\tilde{z}_t$ from the prior P, then running the decoder to update the canvas matrix $\tilde{c}_t$. After T repetitions of this process the generated image is a sample from $D(X|\tilde{c}_T)$:

$$\tilde{z}_t \sim P(Z_t) \tag{13}$$

$$\tilde{h}_t^{dec} = RNN^{dec}(\tilde{h}_{t-1}^{dec}, \tilde{z}_t) \tag{14}$$

$$\tilde{c}_t = \tilde{c}_{t-1} + write(\tilde{h}_t^{dec}) \tag{15}$$

$$\tilde{x} \sim D(X|\tilde{c_T}) \tag{16}$$

Note that the encoder is not involved in image generation.

3. Read and Write Operations

The DRAW network described in the previous section is not complete until the *read* and *write* operations in Eqs. 4 and 8 have been defined. This section describes two ways to do so, one with selective attention and one without.

3.1. Reading and Writing Without Attention

In the simplest instantiation of DRAW the entire input image is passed to the encoder at every time-step, and the decoder modifies the entire canvas matrix at every time-step. In this case the *read* and *write* operations reduce to

$$read(x, \hat{x}_t, h^{dec}_{t-1}) = [x, \hat{x}_t] \tag{17}$$

$$write(h^{dec}_t) = W(h^{dec}_t) \tag{18}$$

However this approach does not allow the encoder to focus on only part of the input when creating the latent distribution; nor does it allow the decoder to modify only a part of the canvas vector. In other words it does not provide the network with an explicit selective attention mechanism, which we believe to be crucial to large scale image generation. We refer to the above configuration as "DRAW without attention".

3.2. Selective Attention Model

To endow the network with selective attention without sacrificing the benefits of gradient descent training, we take inspiration from the differentiable attention mechanisms recently used in handwriting synthesis (Graves, 2013) and Neural Turing Machines (Graves et al., 2014). Unlike the aforementioned works, we consider an explicitly two-dimensional form of attention, where an array of 2D Gaussian filters is applied to the image, yielding an image 'patch' of smoothly varying location and zoom. This configuration, which we refer to simply as "DRAW", somewhat resembles the affine transformations used in computer graphics-based autoencoders (Tieleman, 2014).

As illustrated in Fig. 3, the $N \times N$ grid of Gaussian filters is positioned on the image by specifying the co-ordinates of the grid centre and the stride distance between adjacent filters. The stride controls the 'zoom' of the patch; that is, the larger the stride, the larger an area of the original image will be visible in the attention patch, but the lower the effective resolution of the patch will be. The grid centre (g_X, g_Y) and stride δ (both of which are real-valued) determine the mean location μ^i_X, μ^j_Y of the filter at row i, column j in the patch as follows:

$$\mu^i_X = g_X + (i - N/2 - 0.5)\,\delta \tag{19}$$

$$\mu^j_Y = g_Y + (j - N/2 - 0.5)\,\delta \tag{20}$$

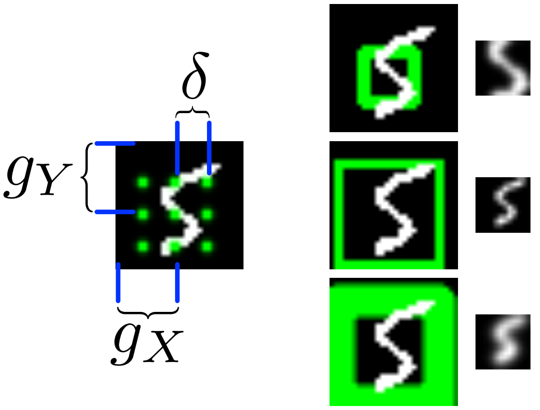

Figure 3. **Left:** A 3×3 grid of filters superimposed on an image. The stride (δ) and centre location (g_X, g_Y) are indicated. **Right:** Three $N \times N$ patches extracted from the image ($N = 12$). The green rectangles on the left indicate the boundary and precision (σ) of the patches, while the patches themselves are shown to the right. The top patch has a small δ and high σ, giving a zoomed-in but blurry view of the centre of the digit; the middle patch has large δ and low σ, effectively downsampling the whole image; and the bottom patch has high δ and σ.

Two more parameters are required to fully specify the attention model: the isotropic variance σ^2 of the Gaussian filters, and a scalar intensity γ that multiplies the filter response. Given an $A \times B$ input image x, all five attention parameters are dynamically determined at each time step via a linear transformation of the decoder output h^{dec}:

$$(\tilde{g}_X, \tilde{g}_Y, \log \sigma^2, \log \tilde{\delta}, \log \gamma) = W(h^{dec}) \tag{21}$$

$$g_X = \frac{A+1}{2}(\tilde{g}_X + 1) \tag{22}$$

$$g_Y = \frac{B+1}{2}(\tilde{g}_Y + 1) \tag{23}$$

$$\delta = \frac{\max(A, B) - 1}{N - 1}\tilde{\delta} \tag{24}$$

where the variance, stride and intensity are emitted in the log-scale to ensure positivity. The scaling of g_X, g_Y and δ is chosen to ensure that the initial patch (with a randomly initialised network) roughly covers the whole input image.

Given the attention parameters emitted by the decoder, the horizontal and vertical filterbank matrices F_X and F_Y (dimensions $N \times A$ and $N \times B$ respectively) are defined as follows:

$$F_X[i, a] = \frac{1}{Z_X} \exp\left(-\frac{(a - \mu^i_X)^2}{2\sigma^2}\right) \tag{25}$$

$$F_Y[j, b] = \frac{1}{Z_Y} \exp\left(-\frac{(b - \mu^j_Y)^2}{2\sigma^2}\right) \tag{26}$$

where (i, j) is a point in the attention patch, (a, b) is a point in the input image, and Z_x, Z_y are normalisation constants that ensure that $\sum_a F_X[i, a] = 1$ and $\sum_b F_Y[j, b] = 1$.

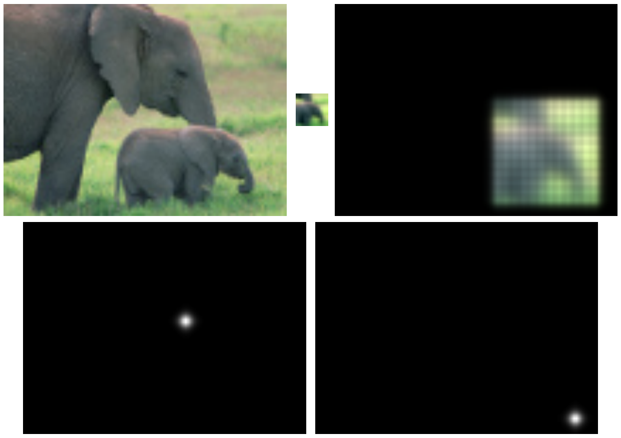

Figure 4. **Zooming. Top Left:** The original 100×75 image. **Top Middle:** A 12×12 patch extracted with 144 2D Gaussian filters. **Top Right:** The reconstructed image when applying transposed filters on the patch. **Bottom:** Only two 2D Gaussian filters are displayed. The first one is used to produce the top-left patch feature. The last filter is used to produce the bottom-right patch feature. By using different filter weights, the attention can be moved to a different location.

3.3. Reading and Writing With Attention

Given F_X, F_Y and intensity γ determined by h^{dec}_{t-1}, along with an input image x and error image $\hat{x}_t$, the *read* operation returns the concatenation of two $N \times N$ patches from the image and error image:

$$read(x, \hat{x}_t, h^{dec}_{t-1}) = \gamma[F_Y x F_X^T, F_Y \hat{x} F_X^T] \tag{27}$$

Note that the same filterbanks are used for both the image and error image. For the write operation, a distinct set of attention parameters $\hat{\gamma}$, $\hat{F}_X$ and $\hat{F}_Y$ are extracted from h^{dec}_t, the order of transposition is reversed, and the intensity is inverted:

$$w_t = W(h^{dec}_t) \tag{28}$$

$$write(h^{dec}_t) = \frac{1}{\hat{\gamma}} \hat{F}_Y^T w_t \hat{F}_X \tag{29}$$

where w_t is the $N \times N$ *writing patch* emitted by h^{dec}_t. For colour images each point in the input and error image (and hence in the reading and writing patches) is an RGB triple. In this case the same reading and writing filters are used for all three channels.

4. Experimental Results

We assess the ability of DRAW to generate realistic-looking images by training on three datasets of progressively increasing visual complexity: MNIST (LeCun et al., 1998), Street View House Numbers (SVHN) (Netzer et al., 2011) and CIFAR-10 (Krizhevsky, 2009). The images generated by the network are always novel (not simply copies of training examples), and are virtually indistinguishable from real data for MNIST and SVHN; the generated CIFAR images are somewhat blurry, but still contain recognisable structure from natural scenes. The binarized MNIST results substantially improve on the state of the art. As a preliminary exercise, we also evaluate the 2D attention module of the DRAW network on cluttered MNIST classification.

For all experiments, the model $D(X|c_T)$ of the input data was a Bernoulli distribution with means given by $\boldsymbol{\sigma}(c_T)$. For the MNIST experiments, the reconstruction loss from Eq 9 was the usual binary cross-entropy term. For the SVHN and CIFAR-10 experiments, the red, green and blue pixel intensities were represented as numbers between 0 and 1, which were then interpreted as independent colour emission probabilities. The reconstruction loss was therefore the cross-entropy between the pixel intensities and the model probabilities. Although this approach worked well in practice, it means that the training loss did not correspond to the true compression cost of RGB images.

Network hyper-parameters for all the experiments are presented in Table 3. The Adam optimisation algorithm (Kingma & Ba, 2014) was used throughout. Examples of generation sequences for MNIST and SVHN are provided in the accompanying video (`https://www.youtube.com/watch?v=Zt-7MI9eKEo`).

4.1. Cluttered MNIST Classification

To test the classification efficacy of the DRAW attention mechanism (as opposed to its ability to aid in image generation), we evaluate its performance on the 100×100 cluttered translated MNIST task (Mnih et al., 2014). Each image in cluttered MNIST contains many digit-like fragments of visual clutter that the network must distinguish from the true digit to be classified. As illustrated in Fig. 5, having an iterative attention model allows the network to progressively zoom in on the relevant region of the image, and ignore the clutter outside it.

Our model consists of an LSTM recurrent network that receives a 12×12 'glimpse' from the input image at each time-step, using the selective *read* operation defined in Section 3.2. After a fixed number of glimpses the network uses a softmax layer to classify the MNIST digit. The network is similar to the recently introduced Recurrent Attention Model (RAM) (Mnih et al., 2014), except that our attention method is differentiable; we therefore refer to it as "Differentiable RAM".

The results in Table 1 demonstrate a significant improvement in test error over the original RAM network. Moreover our model had only a single attention patch at each

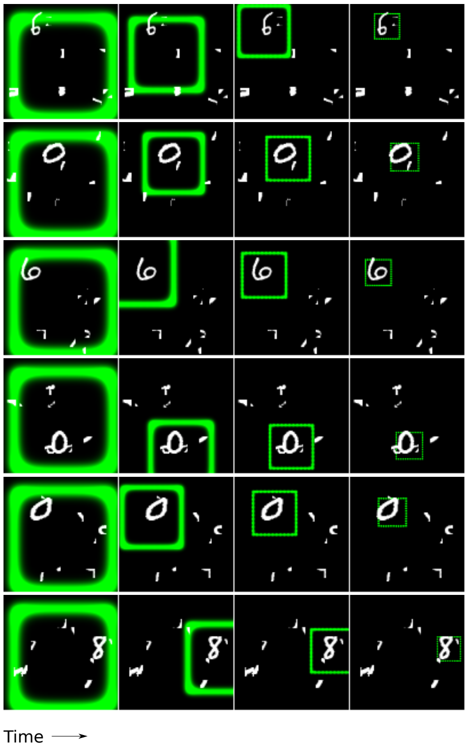

Figure 5. **Cluttered MNIST classification with attention.** Each sequence shows a succession of four glimpses taken by the network while classifying cluttered translated MNIST. The green rectangle indicates the size and location of the attention patch, while the line width represents the variance of the filters.

Table 1. Classification test error on 100 × 100 Cluttered Translated MNIST.

Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12 × 12, 4 scales	9.41%
RAM, 8 glimpses, 12 × 12, 4 scales	8.11%
Differentiable RAM, 4 glimpses, 12 × 12	4.18%
Differentiable RAM, 8 glimpses, 12 × 12	**3.36%**

time-step, whereas RAM used four, at different zooms.

4.2. MNIST Generation

We trained the full DRAW network as a generative model on the binarized MNIST dataset (Salakhutdinov & Murray, 2008). This dataset has been widely studied in the literature, allowing us to compare the numerical performance (measured in average nats per image on the test set) of DRAW with existing methods. Table 2 shows that DRAW without selective attention performs comparably to other recent generative models such as DARN, NADE and DBMs, and that DRAW with attention considerably improves on the state of the art.

Table 2. Negative log-likelihood (in nats) per test-set example on the binarised MNIST data set. The right hand column, where present, gives an upper bound (Eq. 12) on the negative log-likelihood. The previous results are from [1] (Salakhutdinov & Hinton, 2009), [2] (Murray & Salakhutdinov, 2009), [3] (Uria et al., 2014), [4] (Raiko et al., 2014), [5] (Rezende et al., 2014), [6] (Salimans et al., 2014), [7] (Gregor et al., 2014).

Model	$-\log p$	$\leq$
DBM 2hl [1]	≈ 84.62	
DBN 2hl [2]	≈ 84.55	
NADE [3]	88.33	
EoNADE 2hl (128 orderings) [3]	85.10	
EoNADE-5 2hl (128 orderings) [4]	84.68	
DLGM [5]	≈ 86.60	
DLGM 8 leapfrog steps [6]	≈ 85.51	88.30
DARN 1hl [7]	≈ 84.13	88.30
DARN 12hl [7]	-	87.72
DRAW without attention	-	87.40
DRAW	-	**80.97**

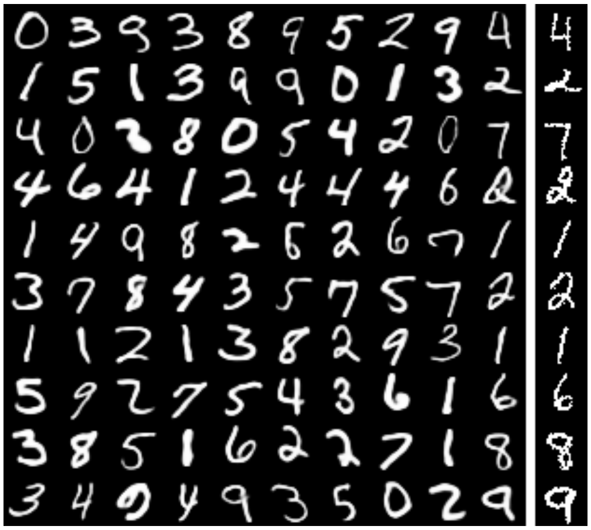

Figure 6. **Generated MNIST images.** All digits were generated by DRAW except those in the rightmost column, which shows the training set images closest to those in the column second to the right (pixelwise L^2 is the distance measure). Note that the network was trained on binary samples, while the generated images are mean probabilities.

Once the DRAW network was trained, we generated MNIST digits following the method in Section 2.3, examples of which are presented in Fig. 6. Fig. 7 illustrates the image generation sequence for a DRAW network without selective attention (see Section 3.1). It is interesting to compare this with the generation sequence for DRAW with attention, as depicted in Fig. 1. Whereas without attention it progressively sharpens a blurred image in a global way,

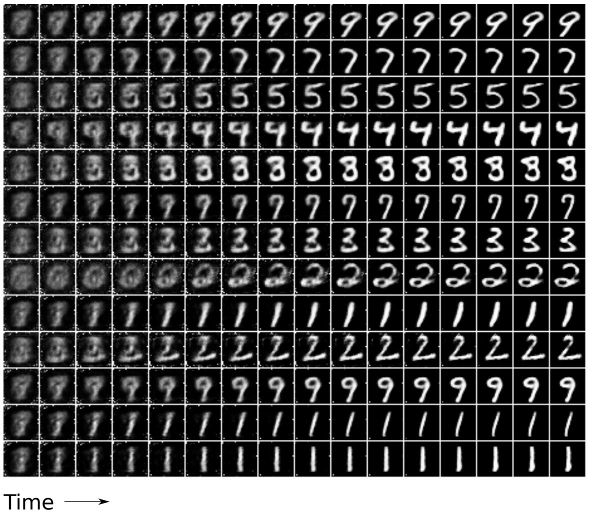

Figure 7. **MNIST generation sequences for DRAW without attention.** Notice how the network first generates a very blurry image that is subsequently refined.

with attention it constructs the digit by tracing the lines—much like a person with a pen.

4.3. MNIST Generation with Two Digits

The main motivation for using an attention-based generative model is that large images can be built up iteratively, by adding to a small part of the image at a time. To test this capability in a controlled fashion, we trained DRAW to generate images with two 28 × 28 MNIST images chosen at random and placed at random locations in a 60 × 60 black background. In cases where the two digits overlap, the pixel intensities were added together at each point and clipped to be no greater than one. Examples of generated data are shown in Fig. 8. The network typically generates one digit and then the other, suggesting an ability to recreate composite scenes from simple pieces.

4.4. Street View House Number Generation

MNIST digits are very simplistic in terms of visual structure, and we were keen to see how well DRAW performed on natural images. Our first natural image generation experiment used the multi-digit Street View House Numbers dataset (Netzer et al., 2011). We used the same preprocessing as (Goodfellow et al., 2013), yielding a 64 × 64 house number image for each training example. The network was then trained using 54 × 54 patches extracted at random locations from the preprocessed images. The SVHN training set contains 231,053 images, and the validation set contains 4,701 images.

The house number images generated by the network are highly realistic, as shown in Figs. 9 and 10. Fig. 11 reveals that, despite the long training time, the DRAW network underfit the SVHN training data.

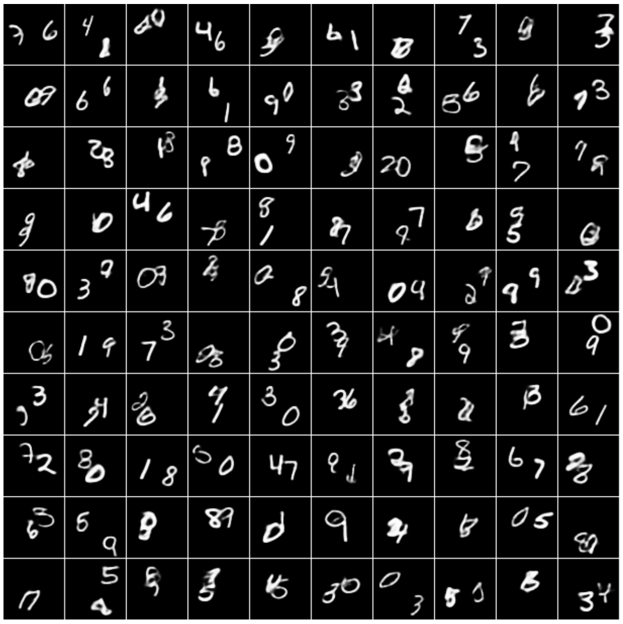

Figure 8. **Generated MNIST images with two digits.**

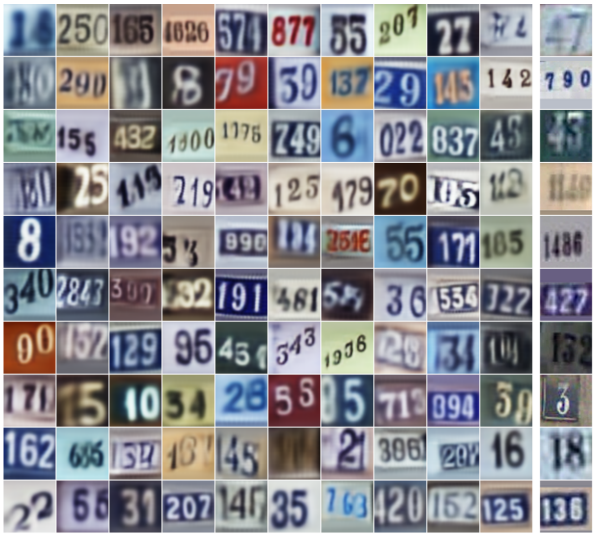

Figure 9. **Generated SVHN images.** The rightmost column shows the training images closest (in L^2 distance) to the generated images beside them. Note that the two columns are visually similar, but the numbers are generally different.

4.5. Generating CIFAR Images

The most challenging dataset we applied DRAW to was the CIFAR-10 collection of natural images (Krizhevsky,

Table 3. Experimental Hyper-Parameters.

Task	#glimpses	LSTM #h	#z	Read Size	Write Size
100 × 100 MNIST Classification	8	256	-	12 × 12	-
MNIST Model	64	256	100	2 × 2	5 × 5
SVHN Model	32	800	100	12 × 12	12 × 12
CIFAR Model	64	400	200	5 × 5	5 × 5

s

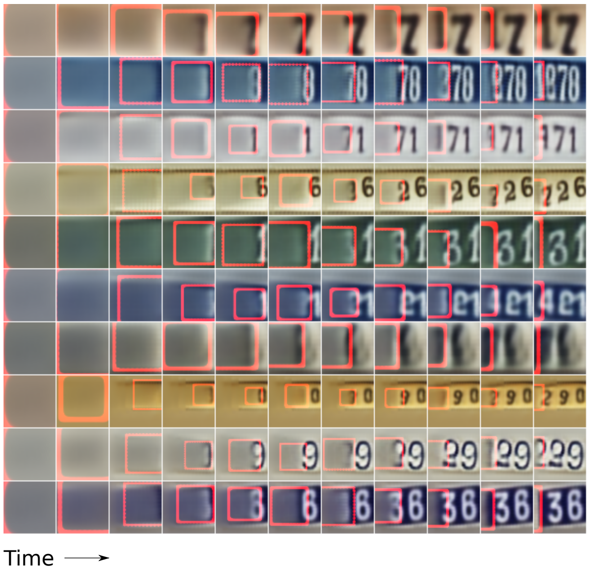

Figure 10. **SVHN Generation Sequences.** The red rectangle indicates the attention patch. Notice how the network draws the digits one at a time, and how it moves and scales the writing patch to produce numbers with different slopes and sizes.

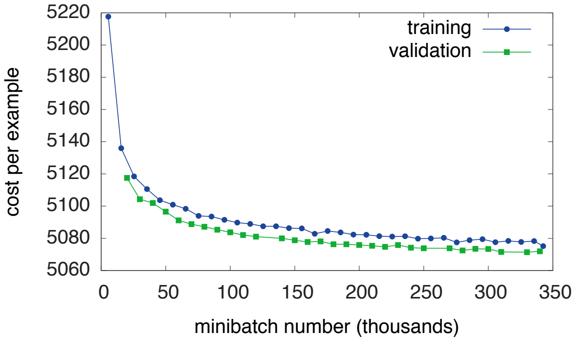

Figure 11. **Training and validation cost on SVHN.** The validation cost is consistently lower because the validation set patches were extracted from the image centre (rather than from random locations, as in the training set). The network was never able to overfit on the training data.

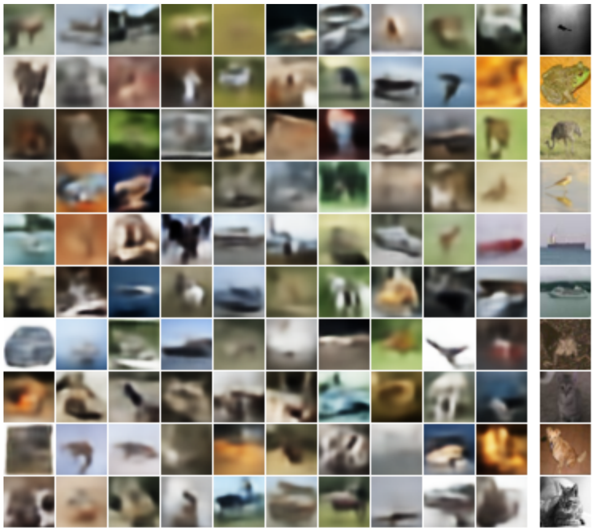

Figure 12. **Generated CIFAR images.** The rightmost column shows the nearest training examples to the column beside it.

2009). CIFAR-10 is very diverse, and with only 50,000 training examples it is very difficult to generate realistic-looking objects without overfitting (in other words, without copying from the training set). Nonetheless the images in Fig. 12 demonstrate that DRAW is able to capture much of the shape, colour and composition of real photographs.

5. Conclusion

This paper introduced the Deep Recurrent Attentive Writer (DRAW) neural network architecture, and demonstrated its ability to generate highly realistic natural images such as photographs of house numbers, as well as improving on the best known results for binarized MNIST generation. We also established that the two-dimensional differentiable attention mechanism embedded in DRAW is beneficial not only to image generation, but also to image classification.

Acknowledgments

Of the many who assisted in creating this paper, we are especially thankful to Koray Kavukcuoglu, Volodymyr Mnih, Jimmy Ba, Yaroslav Bulatov, Greg Wayne, Andrei Rusu and Shakir Mohamed.

References

Ba, Jimmy, Mnih, Volodymyr, and Kavukcuoglu, Koray. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

Dayan, Peter, Hinton, Geoffrey E, Neal, Radford M, and Zemel, Richard S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Denil, Misha, Bazzani, Loris, Larochelle, Hugo, and de Freitas, Nando. Learning where to attend with deep architectures for image tracking. *Neural computation*, 24(8):2151–2184, 2012.

Gers, Felix A, Schmidhuber, Jürgen, and Cummins, Fred. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Goodfellow, Ian J, Bulatov, Yaroslav, Ibarz, Julian, Arnoud, Sacha, and Shet, Vinay. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

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

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Gregor, Karol, Danihelka, Ivo, Mnih, Andriy, Blundell, Charles, and Wierstra, Daan. Deep autoregressive networks. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. 2009.

Larochelle, Hugo and Hinton, Geoffrey E. Learning to combine foveal glimpses with a third-order boltzmann machine. In *Advances in Neural Information Processing Systems*, pp. 1243–1251. 2010.

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

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.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

Murray, Iain and Salakhutdinov, Ruslan. Evaluating probabilities under high-dimensional latent variable models. In *Advances in neural information processing systems*, pp. 1137–1144, 2009.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. 2011.

Raiko, Tapani, Li, Yao, Cho, Kyunghyun, and Bengio, Yoshua. Iterative neural autoregressive distribution estimator nade-k. In *Advances in Neural Information Processing Systems*, pp. 325–333. 2014.

Ranzato, Marc'Aurelio. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014.

Rezende, Danilo J, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

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 Annual International Conference on Machine Learning*, pp. 872–879. Omnipress, 2008.

Salimans, Tim, Kingma, Diederik P, and Welling, Max. Markov chain monte carlo and variational inference: Bridging the gap. *arXiv preprint arXiv:1410.6460*, 2014.

Sermanet, Pierre, Frome, Andrea, and Real, Esteban. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

Tang, Yichuan, Srivastava, Nitish, and Salakhutdinov, Ruslan. Learning generative models with visual attention. *arXiv preprint arXiv:1312.6110*, 2013.

Tieleman, Tijmen. *Optimizing Neural Networks that Generate Images*. PhD thesis, University of Toronto, 2014.

Uria, Benigno, Murray, Iain, and Larochelle, Hugo. A deep and tractable density estimator. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 467–475, 2014.

Zheng, Yin, Zemel, Richard S, Zhang, Yu-Jin, and Larochelle, Hugo. A neural autoregressive approach to attention-based recognition. *International Journal of Computer Vision*, pp. 1–13, 2014.