DRAW: A Recurrent Neural Network For Image Generation

MEC 634 Final Project

Prerna Kothari

December 9, 2018

This work is an implementation of a generative neural network architectures for image generation called **D**eep **R**ecurrent **A**ttentive **W**riter (DRAW). DRAW network, developed by Google Deepmind team, combines 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.

1 Introduction

This approach mimics human intuition of image creation. For example, when 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. The network presented comes under the family of variational auto-encoder networks.

1.1 Variational Auto-Encoder Problem Statement

Given a dataset, capture the probability distribution of data and generate new data samples from the estimated probability distribution. Also discover the salient features and efficiently internalize the essence of the data in order to generate it.

Let $\{x\}_i^N$ be the dataset consisting of N data points, where x is a d dimensional vector given by,

$$x = \{x_1, x_2, ..., x_d\} \tag{1}$$

Let us assume that the observed data $\{x\}_i^N$ is generated by a model with θ parameters based on hidden variables $\{z\}_i^N$, where z are l dimensional vectors and are distributed according to probability distribution P. We only see x, but we would like to infer the characteristics of z. In other words, wed like to compute $p(z|x)$.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2}$$

Unfortunately finding (x) is quite difficult.

$$p(x) = \int p(x|z)p(z)dz \tag{3}$$

Usually, it turns out to be an intractable distribution. We try to approximate $p(z|x)$ with a neural network, which we call an encoder neural network. Output of the encoder neural network is $q(z|x)$ and we want it to be very close the actual $p(z|x)$. Thus, we minimize the KL divergence between $p(z|x)$ and $q(z|x)$ given by,

$$L_z = KL(q(z|x)||p(z|x)) \tag{4}$$

Also, we want accurate generation of the new data samples given z, which is generated by a neural network called Generator or Decoder. The reconstruction loss in generation could be given either by least squared loss between input and generated input or the negative of log likelihood if the input is binary.

$$L_x = (\hat{x} - x)^2 \text{ or } -\sum(xlog(\hat{x}) + (1-x)log(1-\hat{x})) \tag{5}$$

Objective is the find set of parameters $\theta_{enc}, \theta_{dec}$ such that sum of mean reconstruction loss and KL divergence loss is minimum. In other words,

$$\arg\min_{\theta_{enc},\theta_{dec}}(L_x(\theta_{enc} + \theta_{dec}) + L_z(\theta_{enc})), \tag{6}$$

Where, θ_{enc} are parameters of encoder network and θ_{dec} are parameters of decoder network.

2 DRAW Network

The network presented in [1] is a variational auto-encoder neural network with three main differences.

1. 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 decoders previous outputs, allowing it to tailor the codes it sends according to the decoders behavior so far.

2. Secondly, the decoders outputs are successively added to the distribution that will ultimately generate the data, as opposed to emitting this distribution in a single step.
3. Dynamically updated attention mechanism is used to restrict both the input region observed by the encoder, and the output region modified by the decoder.

2.1 Network Architecture

At each time-step t, the encoder receives input from both the image x and from the previous decoder hidden vector h^{dec}_{t1}. 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 $N(Z_t|\mu_t, \sigma_t)$.

At each time-step a sample $z_t Q(Z_t|h^{enc}_t)$ drawn from the latent distribution is passed as input to the decoder. The output h^{dec}_t of the decoder is added (via a write operation, defined in the sequel) to a cumulative canvas matrix ct, 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, $c0$, h^{enc}_0 , h^{dec}_0 are initialised to learned biases, and the DRAW network iteratively computes the following equations for $t = 1...,T$:

$$\begin{aligned}
\hat{x}_t &= xSigmoid(ct1) && (7)\\
r_t &= read(x_t, \hat{x}_t, h^{dec}_{(t1)} && (8)\\
h^{enc}_t &= RNN_{enc}(h^{enc}_{t1}, [rt, h^{dec}_{t1}]) && (9)\\
z_t &\sim Q(Z_t|h^{enc}_t) && (10)\\
h^{dec}_t &= RNN_{dec}(h^{dec}_{t1}, z_t) && (11)\\
c_t &= c_{t1} + write(h^{dec}_t) && (12)
\end{aligned}$$

The loss functions for DRAW network as same as that of traditional variation auto-encoder. The final image drawn at canvas is used for calculating reconstruction loss given in Eq. 5. The KL divergence loss for Gaussian Distribution of $P(z)$ would be given by,

$$L_z = \frac{1}{2}(\sum_{t=1}^{T} {\mu^2}_t + {\sigma^2}_t - \log {\sigma^2}_t - T) \quad (13)$$

2.2 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

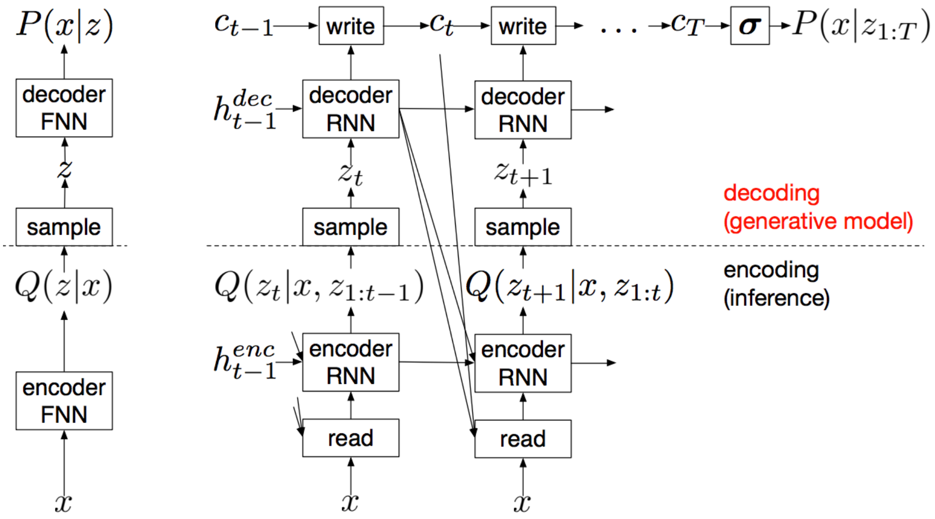


Figure 1: **Left: Conventional Variational Auto-Encoder**. During generation, a sample z is drawn from a prior $P(z)$ and passed through the feedforward decoder network to compute the probability of the input $P(x|z)$ given the sample. During inference the input x is passed to the encoder network, producing 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 loss, 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 $P(x|z1 : T)$. During inference the input is read at every timestep 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

canvas matrix $\tilde{c}_t$. After T repetitions of this process the generated image is a sample from $D(X|\tilde{c}_T)$:

$$
\begin{aligned}
\tilde{z}_t &\sim P(Zt) && (14)\\
\tilde{h}_t^{dec} &= RNN_{dec}(\tilde{h}_{t1}^{dec}, \tilde{z}_t) && (15)\\
\tilde{c}_t &= \tilde{c}_{t1} + write(\tilde{h}_t^{dec}) && (16)\\
\tilde{x} &\sim D(X|\tilde{c}_T) && (17)
\end{aligned}
$$

3 Read and Write Operations

The DRAW network described in the previous section is not complete until the read and write operations in Eqs. 8 and 12 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,

$$
\begin{aligned}
read(x, \tilde{x}_t, h_{t1}^{dec}) &= [x, \tilde{x}_t] && (18)\\
write(h_t^{dec}) &= W(h_t^{dec}) && (19)
\end{aligned}
$$

Where W is a linear weight matrix with weights and biases.

3.2 Reading and Writing with Attention

In order to allow model to have control over where to read, what to read and what to write, five parameters based on linear multiplication of decoder output are obtained. These five parameters are used to compute the patch origin (g_x, g_y), path size (δ), isotropic Gaussian variance σ_{var} and scalar intensity γ.

$$
\begin{aligned}
(\tilde{g_x}, \tilde{g_y}, \log\sigma^2, \log\tilde{}, \log\gamma) &= W(h^{dec}) && (20)\\
g_x &= \frac{A+1}{2}(\tilde{g_x}+1) && (21)\\
g_y &= \frac{B+1}{2}(\tilde{g_y}+1) && (22)\\
\delta &= \frac{max(A,B)-1}{N-1}\tilde{\delta} && (23)
\end{aligned}
$$

Table 1: DRAW Network Hyper Parameters

Learning Rate	0.001
Batch Size	100
Total Time Steps (T)	10
Encoder Hidden Units	128
Decoder Hidden Units	128
Latent Dimension	10

Using the Selective Attention machinery, we formulate read operation as follows,

$$read(x, \tilde{x}_t, h_{t1}^{dec}) = \gamma[F_Y x F_X^T, F^Y \hat{x} F_X^T] \tag{24}$$

Where, F_X and F_Y are filterbank matrices given by,

$$F_X[i,a] = \frac{1}{ZX} exp(-\frac{(a-\mu_X^i)^2}{2\sigma_{var}^2}) \tag{25}$$

$$F_X[j,b] = \frac{1}{ZY} exp(-\frac{(b-\mu_Y^j)^2}{2\sigma_{var}^2}) \tag{26}$$

Where, 1) a and b are indices from 1 to height and 1 to width respectively; 2) i and j are indices from 1 to $\tilde{\delta}$ respectively. Also, the patch matrix μ is calculated is below:

For i is a row and j is a column of the patch,

$$\mu_X^i = g_x + (i - N/2 - 0.5)\delta \tag{27}$$

$$\mu_Y^j = g_y + (j - N/2 - 0.5)\delta \tag{28}$$

$$\tag{29}$$

4 Results

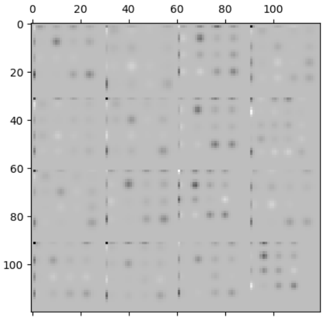
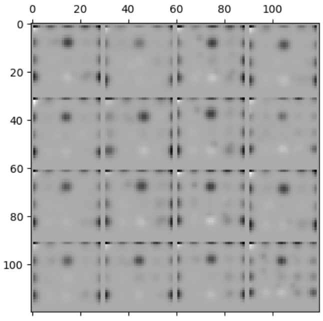
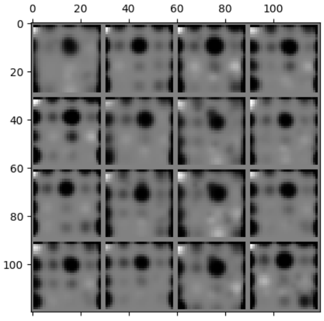
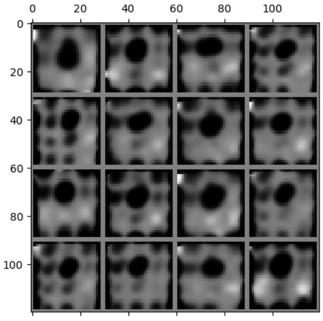
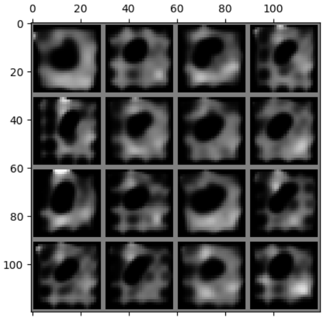
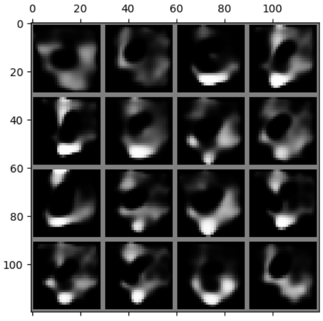
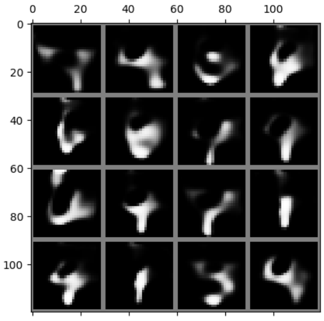
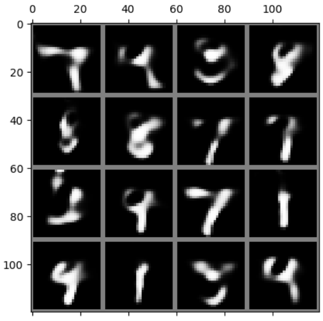
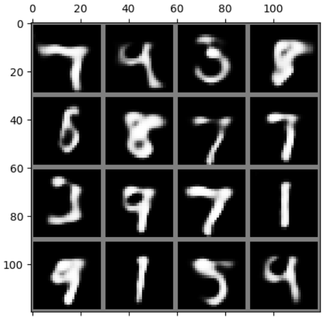
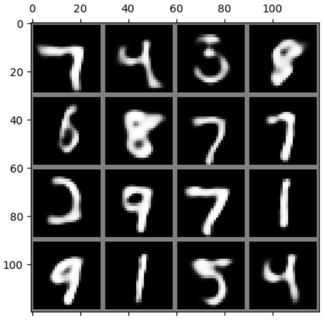
The model is trained on MNIST dataset. Hyper parameters used are tabulated in Table 1. Figure 2 depicts the training losses with respect to number of training batches, where each batch consists of 100 images.

The figures in Table 2 depict the step-wise reconstruction of 16 MNIST images in 10 steps. It can be seen that the images are improved iterative fashion over ten steps. Thus it can be concluded that implementation is successful for iterative image generation with attention.

References

[1] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D., 2015, “Draw: A recurrent neural network for image generation”, arXiv preprint arXiv:1502.04623.

Table 2: Step Wise Reconstruction

 t_0	 t_1
 t_2	 t_3
 t_4	 t_5
 t_6	 t_7
 t_8	 t_9

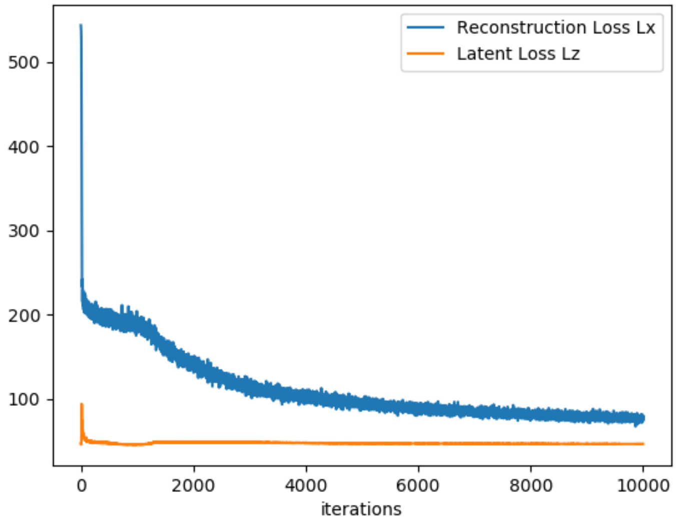


Figure 2: Training of DRAW Network. Two losses (Reconstruction and KL Divergence) compete with each other to find an equilibrium.