Some notes on Attention-based Generative Models

by
Ali Sharifi Boroujerdi

DRAW: (Deep **R**ecurrent **A**ttentive **W**riter that uses a ***2D differentiable attention mechanism*** to read and write selectively)
(Attention-based Generative Model)

It is a method for *automatic image generation* using *generative neural networks* in which parts of the scene are created independently from others.

The core of the DRAW architecture is a pair of recurrent neural networks (RNNs):

1. An ***encoder*** network that compresses the real image presented during training. It determines a distribution over latent codes that captures salient information about the input data.

2. A ***decoder*** network reconstitutes images after receiving codes. Actually, it receives samples from the code distribution and uses them to condition its own distribution over images.

It therefore belongs to the family of ***variational auto-encoders (deep learning + variational inference)***.

Rather than generating 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 because "*visual structure can be better captured by a sequence of partial glimpses (**foveations**), than by a single sweep through the entire image".*

The main challenge: Learning where to look.

(can be addressed by reinforcement learning techniques *e.g. policy gradients*)

Note: This *sequential attention model* is *fully differentiable* in such a way that it is possible to train with standard backpropagation.

DRAW main properties:

1. Both encoder and decoder are RNNs, so they can exchange a sequence of code samples between each other. Moreover, encoder is aware of decoder's previous output.
2. The decoder's outputs are successively added to the distribution that will ultimately generate the data (that is stored in the *cumulative canvas matrix* c_T).
3. A dynamically updated *attention mechanism* is used to restrict both the input region observed by the encoder (where to read) and the output region modified by the decoder (where to write). The network also decides "what to write".

DRAW architecture elements:

RNN^{enc} : Encoder Network

h_t^{enc} :output of RNN^{enc} at time *t* that is used to parametrize a distribution *Q(* Z_t | h_t^{enc}) over the *latent vector* z_t .

The ***latent distribution*** which a diagonal Gaussian:
$\mathcal{N}(Z_t|\mu_t, \sigma_t)$:

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

z_t : is a sample drawn from the *latent distribution* at each time-step which is passed as input to the *decoder.*

RNN^{dec} : Decoder Network

h_t^{dec} :output of RNN^{dec} at time *t* that is added (via a *write* operation) to a cumulative canvas matrix c_t which is ultimately used to reconstruct the image.

RNN^{enc} inputs at each time step t are:

1. Image *x*
2. *Previous decoder output (hidden vector)* h_{t-1}^{dec}

(The precise form of the encoder input depends on a *read* operation)

The total number of the time-steps *T consumed by the network before performing the reconstruction is a free parameter that must be specified in advance.*

c_0 , h_0^{enc} and h_0^{dec} should be initialized.

Network computing equations are:

Logistic Sigmoid Function

Error image
$$\hat{x}_t = x - \sigma(c_{t-1})$$

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

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

Latent distribution sample
$$z_t \sim Q(Z_t | h_t^{enc})$$

Vector concatonation operator

depend on

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

z1:t-1 : previous latent samples

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

DRAW Loss Function:

input data   Final canvas matrix

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

Standard Gaussian Distribution

Bernoulli distribution

Latent Loss
(summed Kullback-Leibler divergence)

$$\mathcal{L}^z = \sum_{t=1}^{T} KL\big(\underbrace{Q(Z_t|h_t^{enc})}_{\text{sequence of latent distributions (diagonal Gaussian)}}||P(Z_t)\big)$$

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

Latent distribution Mean and SD

Total Loss $\mathcal{L} = \langle \mathcal{L}^x + \mathcal{L}^z \rangle_{z\sim Q}$

Draw Image generation process:

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 $D(X|\tilde{c}_T)$:

latent samples $\longrightarrow \tilde{z}_t \sim P(Z_t)$

last decoder output which determines what to write $\longrightarrow \tilde{h}_t^{dec} = RNN^{dec}(\tilde{h}_{t-1}^{dec}, \tilde{z}_t)$

updated canvas matrix $\longrightarrow \tilde{c}_t = \tilde{c}_{t-1} + write(\tilde{h}_t^{dec}) \longleftarrow$ *last writing operation*

generated image $\longrightarrow \tilde{x} \sim D(X|\tilde{c_T}) \longleftarrow$ *the model of the input data*
which will be a sample of the input data model

Note that the encoder is not involved in image generation.

DRAW ***read*** and ***write*** operations:

1. Without attention mechanism:
   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.

2. With ***2D differentiable attention mechanism***:
   It is an explicitly 2D 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.
   The *“stride”* controls the *zoom* of the patch, that is the large the stride the large an area of the original image will be visible in the attention patch, but the lower the effective resolution of the patch will be:

Mean Location of the filter Grid Center Patch row & column numbers

Stride

$$\mu_X^i = g_X + (i - N/2 - 0.5)\,\delta$$
$$\mu_Y^j = g_Y + (j - N/2 - 0.5)\,\delta$$

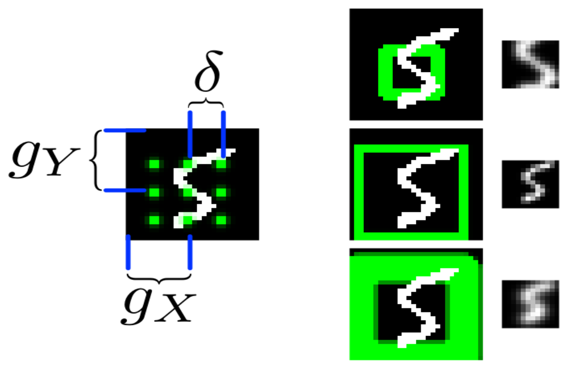


Figure 3. **Left:** A 3×3 grid of filters superimposed on an image. The stride (δ) and centre location (g_X, g_Y) are indicated. **Right:** Three $N \times N$ patches extracted from the image ($N = 12$). The green rectangles on the left indicate the boundary and precision (σ) of the patches, while the 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:

1. Isotropic variance σ^2 of the Gaussian filter
2. A scalar intensity γ that multiplies the filter response and determined by the previous output of the decoder h^{dec}_{t-1} .

Five attention parameters are dynamically determined at each time step via a linear transformation of the decoder output $h^{dec}_{\square}$, where the variance, stride and intensity are emitted in the log-scale to ensure positivity. The scaling of gX , gY and δ is chosen to ensure that the initial patch (with a randomly initialised network) roughly covers the whole input image:

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

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

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

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

Where input image dimension is A x B.

Horizontal and Vertical *filterbank matrices*:

Fx (N x A):

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

Fy (N x B):

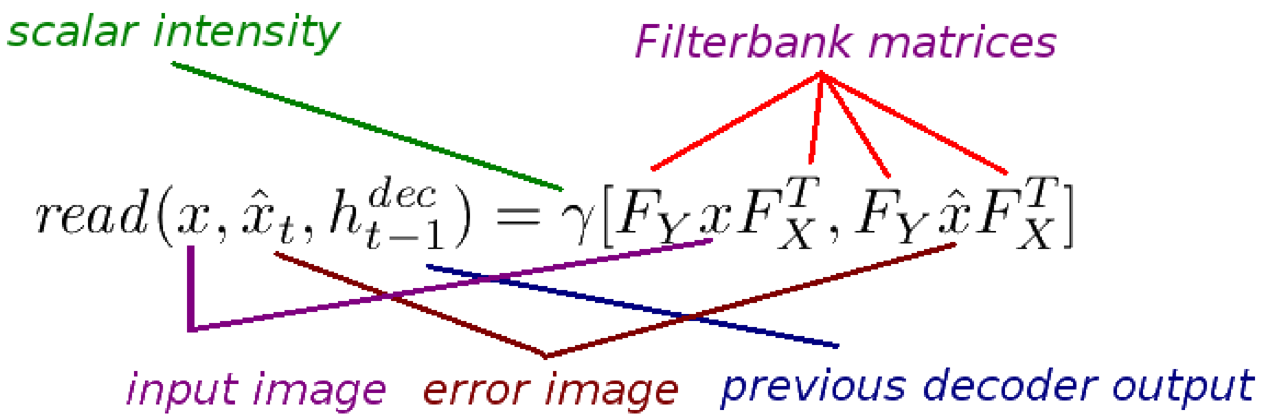
$$F_Y[j,b] = \frac{1}{Z_Y}\exp\left(-\frac{(b-\mu_Y^j)^2}{2\sigma^2}\right)$$

(i , j): A point in the attention patch.
(a , b): A point in the input image.
(zx , zy): Normalization constants to ensure that:

$\sum_a$ Fx[i,a]=1 and $\sum_b$ Fy[j,b]=1

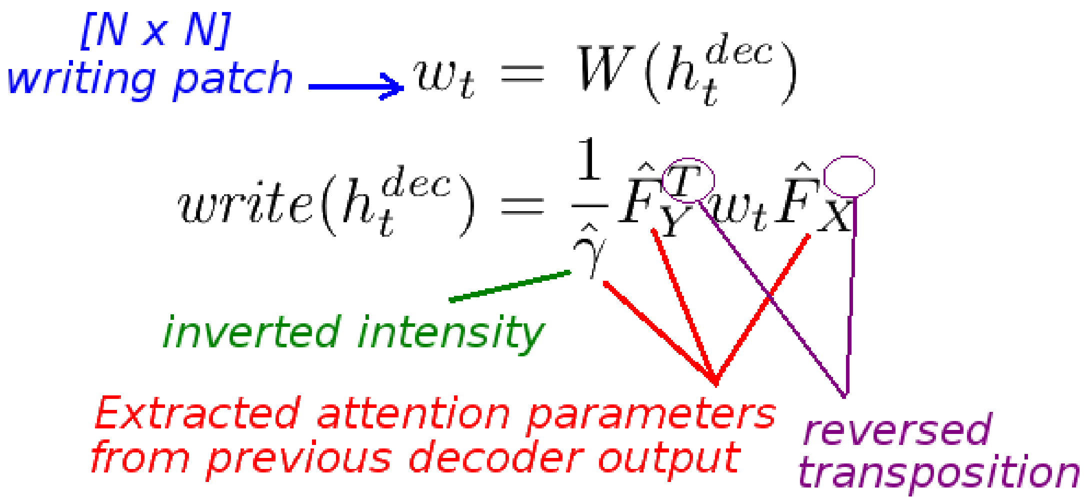
Reading and writing with attention:

Read operation returns the concatenation of two N × N patches from the image and error image:



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

For the ***write operation***, a distinct set of attention parameters γ̂, F̂X and F̂Y are extracted from h_{t-1}^{dec} , the order of transposition is reversed, and the intensity is inverted:



$$w_t = W(h_t^{dec})$$

$$write(h_t^{dec}) = \frac{1}{\hat{\gamma}} \hat{F}_Y^T w_t \hat{F}_X$$

For color 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.