DRAW: A Recurrent Neural Network For Image Generation

1 Citation

Gregor, Karol, et al. "Draw: A recurrent neural network for image generation." arXiv preprint arXiv:1502.04623 (2015).

`https://arxiv.org/pdf/1502.04623.pdf`

2 Abstract

The Deep Recurrent Attentive Writer (DRAW) combines variational autoencoders (VAE) with a spatial attention mechanism to generate realistic looking MNIST and Street View House Number (SVHN) images.

3 Introduction

Artists iteratively update their canvas drawings, starting with an outline and ending with the final drawing. We aim to capture this intuition in DRAW, which consists of encoder and decoder recurrent neural networks (RNN) that optimize a variational lower bound (from VAEs). The decoder iteratively updates the canvas. We use a spatial attention mechanism so DRAW can focus on different parts of the image.

4 The DRAW Network

Like a typical VAE, we have an encoder and decoder network. However, in DRAW, (1) both encoder/decoder are RNNs (2) encoder is conditioned in decoder's previous output, (3) decoder's output is added to its cumulative output, (4) spatial attention is used to identify focus areas for encoder's reading and decoder's writing.

Letting $W(x)$ represent a parameterized affine transform, our equations are:

$$\hat{x}_t = x - \boldsymbol{\sigma}(c_{t-1}) \tag{1}$$
$$r_t = read(x, \hat{x}_t, h^{dec}_{t-1}) \tag{2}$$
$$h^{enc}_t = RNN^{enc}(h^{enc}_{t-1}, [r_t, h^{dec}_{t-1}]) \tag{3}$$
$$z_t \sim Q(Z_t|h^{enc}_t) = \mathcal{N}(Z_t|W(h^{enc}_t), \exp(W(h^{enc}_t))) \tag{4}$$
$$h^{dec}_t = RNN^{dec}(h^{dec}_{t-1}, z_t) \tag{5}$$
$$c_t = c_{t-1} + write(h^{dec}_t) \tag{6}$$

where c_t is the canvas, x is the input image, and the hidden states and canvas are initialized with learned biases. For the sampling of z_t, we use the reparameterization trick (i.e. sample Gaussian noise and feed it as network input and make Q a deterministic function of the noise) to avoid doing sampling in the middle of the network (we can't backpropagate through stochastic units).

We let $\boldsymbol{\sigma}(c_t)$ be the parameters of the Bernoulli distribution $D(X|c_t)$ that determines the output image X. The number of steps that we run the RNNs for is a hyperparameter we call T. Our reconstruction loss is $\mathcal{L}^x = -\log D(x|c_T)$ and our latent loss is $\mathcal{L}^z = \sum_{t=1}^{T} KL(Q(Z_t|h_t^{enc})||P(Z_t)))$, where $P(Z_t)$ is a zero-mean unit-variance Gaussian. Our total loss is thus $\mathcal{L} = (\mathcal{L}^x + \mathcal{L}^z)_{z\sim Q}$, where we consider one sample per training example and we sample according to the VAE reparameterization trick.

At test time, we throw away the encoder and do this:

$$\tilde{z}_t \sim P(Z_t) \tag{7}$$
$$\tilde{h}_t^{dec} = RNN^{dec}(\tilde{h}_{t-1}^{dec}, \tilde{z}_t) \tag{8}$$
$$\tilde{c}_t = \tilde{c}_{t-1} + write(\tilde{h}_t^{dec}) \tag{9}$$
$$\tilde{x} \sim D(X|c_t) \tag{10}$$

5 Read and Write Operations

It's possible to make *read* and *write* simply return the entire image, or canvas, in which case $read(x, \hat{x}_t, h_{t-1}^{dec}) = [x, \hat{x}_t]$ and $write(h_t^{dec}) = W(h_t^{dec})$. We prefer to use a spatial attention mechanism because it gives better results.

For our attention mechanism, we consider a grid of $N \times N$ (we pick $N = 12$) Gaussian filters (i.e. Gaussian blur). The grid is centered at the learned coordinates (g_X, g_Y). The position of the (i, j) (for $1 \le i, j \le N$) Gaussian filter is

$$\mu_X = g_X + (i - N/2 - 1/2)\delta \tag{11}$$
$$\mu_Y = g_Y + (j - N/2 - 1/2)\delta \tag{12}$$

where δ is a learned parameter called stride that controls the size of the grid of Gaussians. Each Gaussian has a variance of σ^2 for learned σ. The learned parameters are computed as follows (where the input image has size $A \times B$ and γ is an intensity parameter that we will use later):

$$(\tilde{g}_X, \tilde{g}_Y, \log\sigma^2, \log\tilde{\delta}, \log\gamma) = W(h^{dec}) \tag{13}$$
$$g_X = \frac{A+1}{2}(\tilde{g}_X + 1) \tag{14}$$
$$g_Y = \frac{B+1}{2}(\tilde{g}_Y + 1) \tag{15}$$
$$\delta = \frac{\max(A, B) - 1}{N - 1}\tilde{\delta} \tag{16}$$

We can then compute filterbank matrices F_X (size $N \times A$) and F_Y (size $N \times B$) for point (i, j) in attention patch and point (a, b) in the input image as follows (Z_x and Z_y are normalization constants to ensure $\sum_a F_X[i, a] = 1$ and $\sum_b F_Y[j, b] = 1$)

$$F_X[i,a] = \frac{1}{Z_X} \exp -\frac{(a-\mu_X^i)^2}{2\sigma^2} \tag{17}$$

$$F_Y[j,b] = \frac{1}{Z_Y} \exp -\frac{(b-\mu_Y^j)^2}{2\sigma^2} \tag{18}$$

We can compute one set of filterbanks for reading and and one for writing (the writing ones will be denoted with a hat $\hat{\cdot}$). Our read/write operations are:

$$read(x, \hat{x}_t, h_{t-1}^{dec}) = \gamma[F_Y x F_X^T, F_Y \hat{x}_t F_X^T] \tag{19}$$

$$write(h_t^{dec}) = \frac{1}{\hat{\gamma}} \hat{F}_Y^T W(h_t^{dec}) \hat{F}_X \tag{20}$$

6 Experimental Results

We consider the MNIST, SVHN, and CIFAR-10 datasets. For MNIST and SVHN, we create images indistinguishable from real images. For CIFAR-10, our generated images are blurry. We optimize with Adam. MNIST uses black-white pixels while SVHN and CIFAR-10 use RGB pixels. Our encoder/decoder networks are Long Short-Term Memory (LSTM) Networks.

We can apply our network to the Cluttered MNIST classification task and it does quite well.

For generating MNIST digits, our model builds up the digits incrementally (starts with outline and then darkens them). We also made our network generate two digits and they look realistic.

Our generated SVHN images look very realistic.

Our generated CIFAR-10 images are blurry, but generally capture the size, shape, and color of blurred objects in natural images.

7 Conclusion

DRAW combines a VAE with spatial attention and generates realistic looking MNIST and SVHN images.