Generative Image Models

Project Report, Group 11, Nov 10, 2016

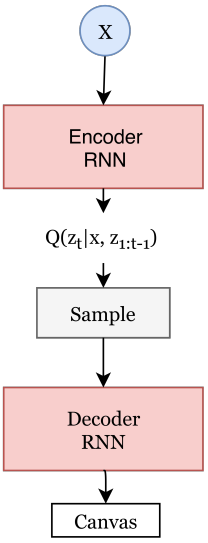
Nirbhay Modhe (13444) Vikas Jain (13788)
Department of Computer Science and Engineering, IIT Kanpur
CS698N: Recent Advances in Computer Vision, Autumn 2016, Instructor: Prof Gaurav Sharma

1 Introduction and Problem Definition

The problem of designing generative image models is important as it leads to more efficient representations of images. The aim of this project is to build generative image models which can suitably learn the representation of the images and can reproduce the new images without any conditioning. In this project, we analysed and worked on the generative image model *DRAW* proposed by Gregor *et al.* [2]. We propose new models which incorporate convolutional neural network with the *DRAW* model to enhance its performance.

2 Related Work - *DRAW* Model

In the *DRAW* model, probability distribution of the latent representation of images are learned using *Variational Autoencoders* [3]. The key features of the model include *progressive refinement* of the generated image and a *spatial attention* mechanism. The autoencoder is composed of two *Recurrent Neural Networks* and the attention mechanism is composed of *read* and *write* functions which direct the focus of the autoencoder on parts of the image rather than the whole. We used the *DRAW* model as the base model for generative image modelling.



3 Datasets Used

We have used the binarized *MNIST Dataset* [4] and Street View House Numbers (SVHN)[5], dataset for all of our experiments. The MNIST dataset has 28x28 images of handwritten digits with 60,000 images in the training set and 10,000 images in the testing set. The SVHN-cropped dataset has 32x32 RGB images of multiple digits of house numbers centered on a particular digit, with 73,257 training images and 26,032 test images.

4 Methodology

In this project we experimented with the original DRAW model by varying several hyperparameters and introducing our own modifications to certain segments of the model. The following subsections describe our experiments in detail.

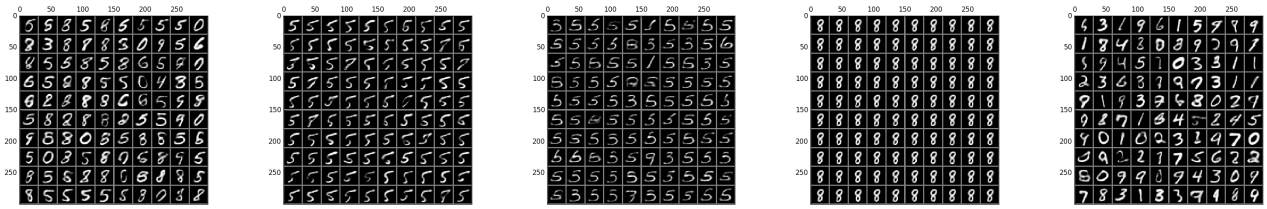
4.1 Studying the *DRAW* Model

4.1.1 On MNIST

We trained the original *DRAW* model for the parameters described in the paper [2]. For the MNIST dataset, it took nearly 10 minutes for 10000 training iterations (17 epochs) each with a batch size of 100 images to train the original *DRAW* model with an encoder and decoder size of 256 hidden units each, 10 time steps, latent z vector of size 10, and a glimpse window size of 5x5. Listed below are the experiments done with different parameters of this model.

- Training DRAW with and without attention for reading as well as writing.
- Varying the time-steps T for encoder-decoder network for $T = 1, 2, 5, 10$. The case for $T = 1$ reduces to a model similar to a variational autoencoder[3].
- Removed the decoder output from the encoder input to observe the importance of this decision by the authors.
- At each time step, the encoder network takes the original image and error image as input. We removed the original image and just kept the error image since it may already contain the necessary information about the original image.

Latent Space Analysis: Our aim was to observe the dependency of the generated image on the sampling from latent space during stochastic image generation. To this end, we studied the contribution of the sample at each time step in determining the final digit generated.



(a) First fixed (b) First 2 fixed (c) Last 9 fixed (d) First 9 fixed (e) Last fixed

Figure 1: Effect of generated images on fixing the samples at various timesteps. Each figure has 100 images where the N^{th} fixed means that the N^{th} time step sample is fixed (same) for all 100 images and randomly varying for the other time steps.

4.1.2 On SVHN

We trained the DRAW model on the SVHN cropped dataset after converting the 32x32 RBG images to grayscale. The authors of DRAW stated that convergence for SVHN was slow and their training time was very long. We did not observe convergence for 10,000 iterations (13 epochs, 30 mins), however, increasing this to 500,000 (682 epochs, 23 hours) iterations did not make any difference. With attention on read and write, the reconstruction loss and latent loss fluctuated around the same initial values throughout training. Without any attention, the latent loss converged but the reconstruction loss did not.

Figure 2: Linear interpolation in latent space for samples in all time steps, each row represents a different line.

4.2 Our Modifications

4.2.1 Model I - Each Step Convolution

The architecture of the *DRAW* model uses RNNs for encoding the output of their $read$ function as well as decoding of the *latent representation* of the image required by the $write$ function. Figure 3 shows the how convolutional layers are introduced into the encoder network. The modifications we made to the original model are as follows:

Convolution After Read: The read function (without attention) now passes the training image and error image through two convolution layers. The flattened filter maps are then used subsequently in two possible way – they are concatenated with the source image (and error image respectively) or they are returned unchanged. The results for both these cases are tabulated in the next section.

4.2.2 Model II - Supervised Encoder

We added a supervision in the form of a digit classification network to the original DRAW network as shown in Fig 4a. This supervised network has an independent classification layer which is interconnected with the encoder network in two places:

- Two convolution layers with 5x5 kernels take as input the image and produce filter response maps which are concatenated with the original encoder input at all time steps.
- The convolution layer output is fed to two fully connected layers after which a softmax cross-entropy loss function as the classification loss L_c

4.2.3 Model III - Convolution Encoder

In the *DRAW* model, input image is passed to the Encoder RNN. Instead of passing original image, we passed the cnn features of the image to the Encoder RNN, and the loss is backpropogated to the added CNN. Figure 4b shows the proposed network architecture.

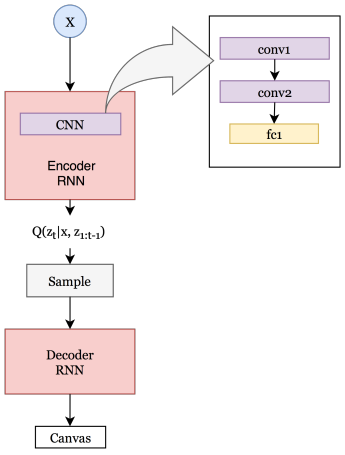
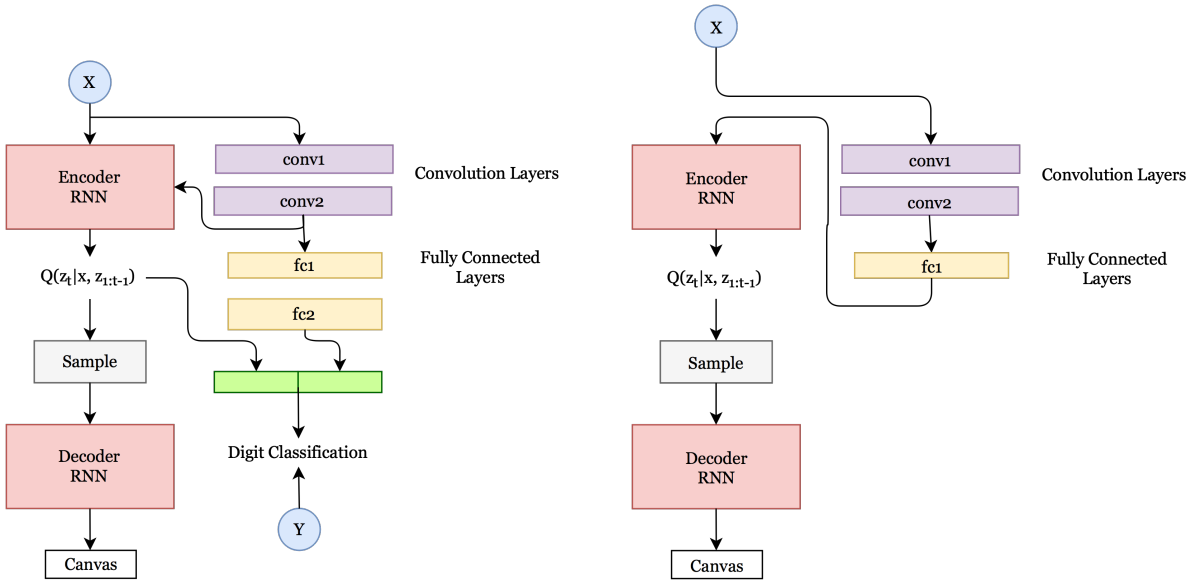


Figure 3: Model I



(a) Model II - Supervised Encoder

(b) Model III - Convolutional Encoder

Figure 4: Proposed modifications to original DRAW network.

5 Existing Code and Libraries Used

Several open source implementations of the DRAW model are available on varied platforms. We have used the existing implementation of DRAW in TensorFlow documented and licensed by Eric Jang[1]. This implementation was limited to the binarized MNIST dataset.

5.1 Our contributions on top of the existing code

- Implemented **stochastic data generation** which was missing from Eric Jang's code. This allowed for sampling of latent space at each time step without the need for any input image and encoder network.
- Added an interface for the **SVHN dataset** to be given as input to the draw network.

- Added **convolution and deconvolution** wrappers for facilitating our experiments of introducing convolutional layers in various segments of the original network for **training the proposed models I, II and III**.
- Implemented the evaluation phase to bf calculate the negative log-likelihood of the generated images using nearest neighbour of the generated image in the MNIST dataset.
- Added new sampling functionalities to **visualize the latent space** by means of fixing samples across sessions and linearly interpolating between two points in latent space.
- **Visualizing and plotting kernels** of the learned CNN.

6 Results and Discussions

6.1 Experimental Study of the *DRAW* Model

We calculated value of *Negative log-likelihood* of 1000 generated images for the original model and the modified models described in the section 4.1.

	Average Error	**Standard Deviation**	**Maximum Error** ($\leq$)
***DRAW*(Attention)**	**643**	**264**	1370
***DRAW*(No-Attention)**	644	268	**1337**
***DRAW*(T=1)**	653	290	1724
***DRAW*(T=2)**	698	276	1402
***DRAW*(T=5)**	796	302	1853
***DRAW*(no-privy[1])**	809	298	2546
***DRAW*(only error image)**	648	302	1853

It can be observed that ***DRAW* model with attention** is performing better than the other models which is the proposed model of the paper [2].

6.2 Proposed Models

We proposed three new models incorporating CNN (described in section 4.2) in the original *DRAW* model. We calculated the **negative log-likelihood** of 1000 generated images for each of the model shown in the table 6.2. Both models took 4-6 hours for 50,000 iterations (84 epochs)

	Average Error	**Standard Deviation**	**Maximum Error** ($\leq$)
DRAW	820	327	1989
Model I - Each Step Convolution	656	267	1429
Model II - Supervised Encoder	791	295	1823
Model III - Convolutional Encoder	**625**	255	1637

6.3 Discussion

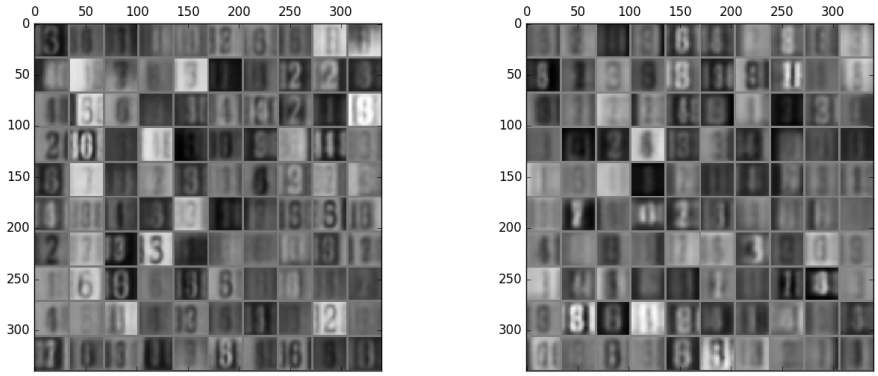
Latent Space: Smooth transitions observed by linear interpolation in latent space show that the DRAW model learns meaningful representation of images. Each image generated stochastically requires a new sample to be drawn at each time step. We showed that by fixing the first sample across 100 images restricts the subsequently generated images to a large extent, indicating that the earlier samples are more important in determining which digit will be generated than the latter samples.
Proposed Models: We proposed three convolution-based models as modifications to the DRAW network, and obtained better negative log-likelihood values for most of the models. The first model was very sensitive to initializations as it did not converge around half of the time when randomly initialized.

[1] decoder output is not made privy to the encoder

7 Comparison and distribution of work

Task	Proposed	Midterm	Endterm	Member
Reproduction of results on MNIST and SVHN	✓	✓	✓	VN
Experimenting with parameters	✓	✓	✓	VN
Evaluation using negative log likelihood	✗	✓	✓	V
Model I - Each Step Convolution	✓	✓	✓	VN
Model II - Supervised Encoder	✗	✗	✓	N
Model III - Convolutional Encoder	✗	✗	✓	V
Latent Space Analysis	✗	✗	✓	N

Table 1: Work completed before and after midterm, along with the member-wise distribution. **V** represents Vikas, **N** represents Nirbhay, **VN** represents joint contribution by both members.



(a) (b)

Figure 5: Stochastically generated images from DRAW model on SVHN dataset without 5a and with 5b attention.

References

[1] Eric jang: Understanding and implementing deepmind's draw model. `http://blog.evjang.com/2016/06/understanding-and-implementing.html`. Accessed: 2016-10-2.

[2] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

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