Variational Recurrent Neural Networks for Long-Term Future Frame Prediction

Muhammad Haziq Bin Razali

Advisor: Beril Besbinar

Abstract

Recurrent Neural Networks currently dominate the state of the art in a variety of tasks that involve a sequence of inputs. The inclusion of latent random variables into these models are a recent innovation that has shown to further enhance performance. In this project, we investigate whether these models are also effective for tasks involving videos. We evaluate the variational recurrent neural network on the moving mnist and moving shapes datasets and show that a recurrent network augmented with stochasticity can outdo its deterministic counterpart in long-term future frame prediction and are especially effective at recovering from occlusions.

Acknowledgements

Foremost, I would like to express my sincere gratitude to Beril Besbinar for her continuous support during this project. Her patience and knowledge have helped steer me in the right direction. I could not have imagined having a better advisor during this first leg of my journey at EPFL.

My sincere thanks also goes to Professor Pascal Frossard from the LTS4 laboratory for allowing me the opportunity to work on this project.

Contents

Chapter 1
Introduction

The past few years have seen deep neural networks emerge as one of the most powerful machine learning methods, forming the backbone of nearly every successful machine learning pipeline. Currently, recurrent neural networks, specifically a variant with Long Short Term Memory (LSTM) cells [1] , hold the state-of-the-art results in a wide range of sequence based tasks. At a high level, they belong to the family of autoregressive models where the predicted element is conditioned on the history of inputs received thus far. From video analysis [2] to speech recognition [3], text generation [4], machine translation [5] and image captioning [6], the versatility of recurrent networks has proven to be an indispensable tool for machine learning practitioners.

More recently, a number of works have presented architectures that combine a recurrent network with the variational autoencoder to form the variational recurrent neural network [7, 8, 9]. Core to their approach is the use of latent random variables to propagate uncertainty through a recurrent network. While these models have shown promising results on speech, handwriting and music generation, their performance on videos have yet to be seen. Thus in this paper, we experiment with an existing model and show that the inclusion of latent random variables into recurrent networks can significantly enhance the performance of generative models. In summary, our main contributions in this paper are as follows:

- We present experimental results on the moving mnist and moving shape datasets and show that recurrent networks fitted with stochasticity qualitatively outperforms the standard recurrent network for long-term future frame prediction.
- We show through experiments, a minor weakness inherent to these models and reason the cause behind it.

- We outline venues for future research and pinpoint potential improvements that can be made to the model

The remainder of this paper is organized as follows: we begin in chapter 2 by covering some background for the work presented in this paper before introducing the variational recurrent neural network in chapter 3. We then specify architecture and training details in 4 before presenting the results of our experiments in 5. Lastly, we conclude and suggest some possible ways to increase performance in chapter 6. The source code and results presented here are available at `https://github.com/HaziqRazali/VRNN-Long-Term-Future-Frame-Prediction`

Chapter 2
Background

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are architectures designed to handle sequential data of arbitrary length. Given an input sequence $(x_1, x_2 \dots x_t)$, a RNN computes a sequence of outputs $(y_1, y_2 \dots y_{t'})$ through the following recurrence equations:

$$h_t = g(W_h \cdot [x_t, h_{t-1}] + b_h)$$
$$y_t = g(W_y \cdot h_t + b_y)$$

where g is an element-wise non-linearity, such as a sigmoid or a tanh, (.) represents the dot product, and W and b denote parameters trainable via the backpropagation through time [10] algorithm. When used with videos, RNNs are often augmented with a convolutional encoder and decoder network [2] . While principally simple, vanilla RNNs, which refer to recursive tanh units, suffer from the problem of *vanishing* or *exploding* gradients [11, 12] when backpropagating across many timesteps. If the weights in the matrices are small, repeated matrix multiplications can lead to a situation where the gradient gets vanishingly small, preventing the weights of the recurrent network from updating its value. Conversely, if the weights are large, the gradient signal can grow to the extent that it causes learning to diverge. All in all, these limitations have prevented the network from learning long range dependencies.

LSTM [1] units combat this issue by incorporating memory cells and forget gates. Their operation is completely parameterized by the following equations:

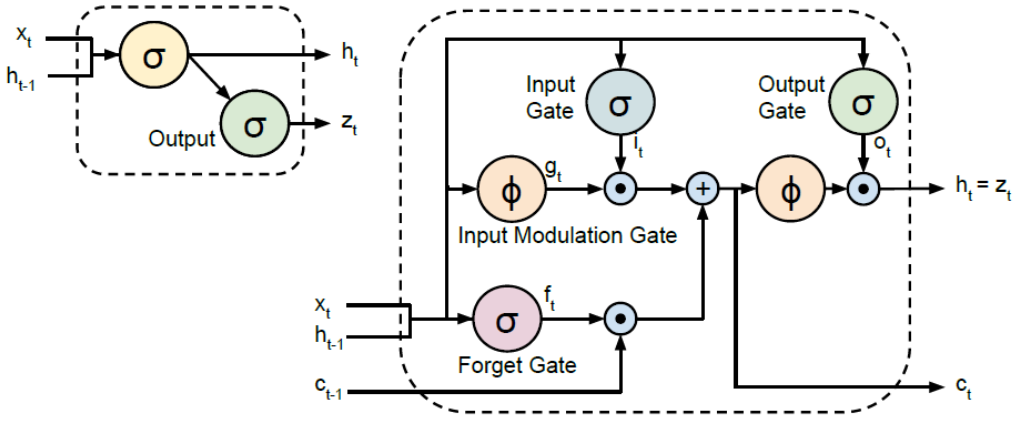


Figure 2.1: A diagram of a basic RNN cell and an LSTM memory cell [2].

$$
\begin{aligned}
\text{input gate } i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\
\text{forget gate } f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\
\text{output gate } o_t &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\
\text{input modulation gate } g_t &= tanh(W_g \cdot [x_t, h_{t-1}] + b_g) \\
\text{memory unit } c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
\text{hidden unit } h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

Intuitively, the sigmoids at the gates control the flow of information through the cell. As long as any of the gates are closed, its respective information is either trapped in the cell or prevented from entering / escaping. These gates, and the additive feature of the memory unit is what makes it possible to train the LSTM so robustly, as their design allows gradients on the memory cells to flow backwards through time without vanishing or exploding.

2.2 Variational Autoencoders

Autoencoders combine an encoder that embeds the data x into a latent space of lower dimension, and a decoder that maps this latent representation back to $\hat{x}$, such that the loss between x and $\hat{x}$ is minimized. Variational Autoencoders (VAEs) [13] modify the conventional autoencoder framework by assuming that the samples x are influenced by a set of latent variables according to the following equation: $p(x) = \int_z p(x|z)p(z)$. These latent variables reflect a lower dimensional representation of x

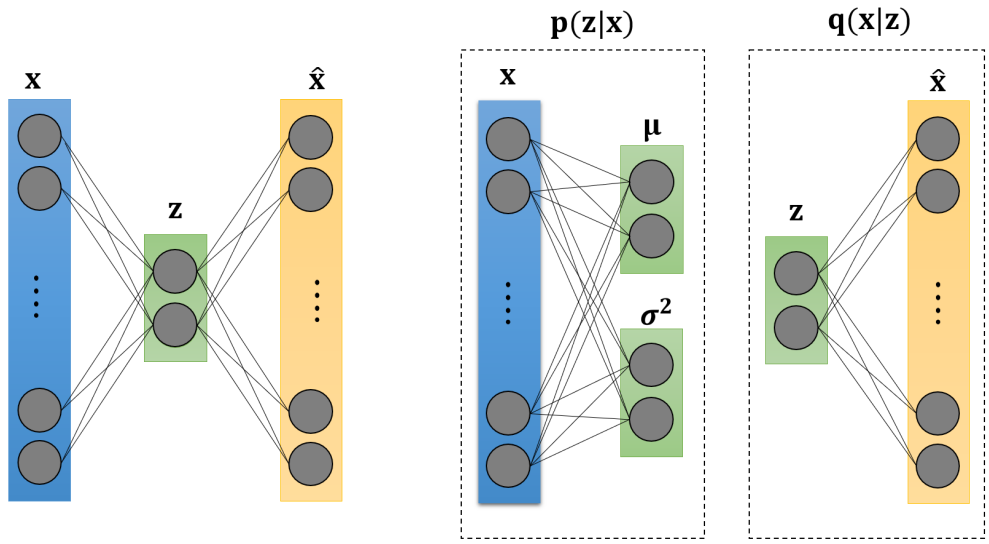
that characterize its global structure. The task is then to learn the model parameters that maximize the likelihood of the samples. However, as the marginal over z makes the problem intractable, the expression is instead reformulated as a lower bound of the log likelihood.

$$\begin{aligned}\log\ p(x) &= p(X|z) - D(q(z|X)||p(z)) + D(q(z|X)||p(z|X)) \\ &\geq p(X|z) - D(q(z|X)||p(z))\end{aligned}$$

The expression is essentially a sum of two losses - a reconstruction loss $p(X|z)$ that encourages the decoder to reconstruct data that are consistent with the training samples, and a latent loss $D(q(z|X)||p(z))$ that measures how well the posterior $q(z \mid x)$ approximates the designated prior $p(z)$. This prior is generally represented by a reparameterizable distribution such as a Gaussian with zero mean and unit variance as it allows us to compute the KL term between the posterior and prior in closed form. The whole model can then be trained using backpropagation and the reparameterization trick [13].

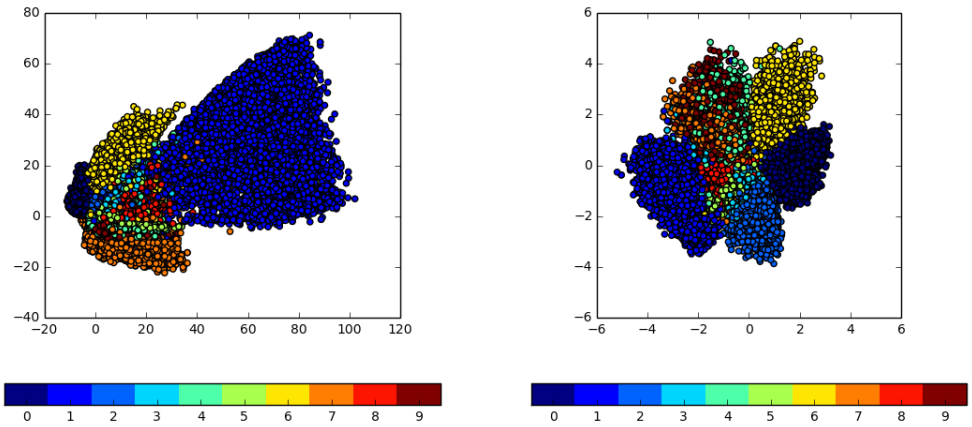
The connection with the classical autoencoder becomes clear when viewing the architectures in figure 2.2 and the resulting latent space in figure 2.3. Here, we generate figure 2.3 by feeding the mnist test set through a trained VAE and an AE, both with latent vectors of dimension 2. Each point in the figure represents the latent encoding of each image, generated by the encoder. It would seem that the encodings of the digits by the VAE are more evenly distributed compared to the AE. From this, generating novel samples from the VAE becomes trivial as it is a matter of sampling from the learned latent space.

One limitation of the VAE described above is that it does not provide any control over the class of the image being generated. Conditional VAEs [14, 15] extend this by modelling the latent variables and data, conditioned on the class labels. Consequently, the variational lower bound introduced earlier is now of the following form: $\log p(x) \geq p(X|z,c) - D(q_\lambda(z|X,c)||p(z|c))$ where c denotes the class label, typically a one-hot vector.



(a) Autoencoder (b) Variational Autoencoder

Figure 2.2: An architecture of (a) an Autoencoder and (b) a Variational Autoencoder. In the AE, the encoder and decoder network are directly linked by the latent vector z. In the VAE, an additional layer that generates the parameters of the gaussian distribution connects the two networks.



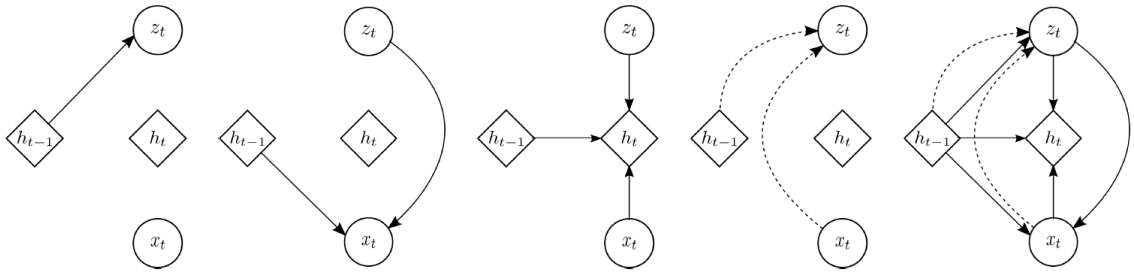
(a) AE encoded digits (b) VAE encoded digits

Figure 2.3: Latent encodings for the (a) Autoencoder and (b) Variational Autoencoder.

Chapter 3
Variational Recurrent Neural Network

The RNN presented in section 2.1 has been shown in [7] to be insufficient at modelling natural sequences that are highly structured. Furthermore, its deterministic nature inhibits its use in a generative setting as it can only deliver a single output for any fixed input. For these reasons, recent efforts have begun to incorporate the VAE into a recurrent network and have conclusively shown that augmenting RNNs with stochasticity enable them to capture the large amount of variability often present in speech and music. Recurrent networks built this way vary in the way they propagate uncertainty across time. Here, we briefly summarize our model of choice, the Variational Recurrent Neural Network (VRNN) [8] and would encourage the reader to refer to [7, 8, 9] for the other variants.

The schematic in Figure 3.1 summarizes the architecture of the VRNN. Simply put, it contains a VAE at every timestep whose mean and variance are conditioned on the hidden unit h_t of an RNN. Additionally, these latent variables are used to condition the recurrent network as well as the output. As we will see in chapter 5, these latent variables can be intuited as a regularizer that limits the representation fed through the decoder. The forward pass of the VRNN is completely described by the recurrence equations as follows:



(a) Prior (b) Generation (c) Recurrence (d) Inference (e) Overall

Figure 3.1: Operations of the Variational Recurrent Neural Network [8].

$$
\begin{aligned}
\text{input gate } i_t &= sigm(W_i \cdot [\text{Encoder}(x_t), h_{t-1}, \text{F}_\text{z}(z_{t-1})] + b_i) \\
\text{forget gate } f_t &= sigm(W_f \cdot [\text{Encoder}(x_t), h_{t-1}, \text{F}_\text{z}(\text{z}_\text{t-1})] + b_f) \\
\text{output gate } o_t &= sigm(W_o \cdot [\text{Encoder}(x_t), h_{t-1}, \text{F}_\text{z}(\text{z}_\text{t-1})] + b_o) \\
\text{input modulation gate } g_t &= tanh(W_g \cdot [\text{Encoder}(x_t), h_{t-1}, \text{F}_\text{z}(z_{t-1})] + b_g) \\
\text{memory unit } c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
\text{hidden unit } h_t &= o_t \circ tanh(c_t)
\end{aligned}
$$

$$
\begin{aligned}
\text{Prior } p(z_t|x_{<t}, z_{<t}) &\sim \mathcal{N}(\mu_t,\, \sigma_t^2), \text{ where } [\mu_t,\, \sigma_t^2] = \text{F}_\text{prior}(h_{t-1}) \\
\text{Image } x_t &= \text{Decoder}(\text{F}_\text{decoder}([z_t, h_{t-1}])
\end{aligned}
$$

where the commas denote a concatenation, the components F_decoder, F_prior, and F_z all denote fully connected layers and the Encoder and Decoder are convolutional and transpose convolutional networks respectively. The parameters μ_t, σ_t^2 are optimized using an inference network that approximates the posterior distribution.

$$
\text{Posterior } q(z_t|x_{\leq t}, z_{<t}) \sim \mathcal{N}(\mu_t,\, \sigma_t^2), \text{ where } [\mu_t,\, \sigma_t^2] = \text{F}_\text{encoder}([\text{Encoder}(x_t), h_{t-1}])
$$

With that, the objective function becomes a timestep-wise variational lower bound as given in equation 3.1. As in the VAE, the whole model can then be trained using backpropagation and the reparameterization trick.

$$
\sum_{t=1}^{T} \log p(x_t|z_{\leq t}, x_{<t}) - \text{D}(q(z_t|x_{\leq t}, z_{<t})||p(z_t|x_{<t}, z_{<t})) \tag{3.1}
$$

Chapter 4
Experimental Setup

4.1 Architecture

The models used in our experiments are summarized in table 4.1. The encoder and decoder are both built alternating between between strided convolutions (transpose convolutions for the decoder) and Rectified Linear Units (ReLu). We base the design of our "F" layers on the implementation of the VRNN [8], using the same number of units at each fully connected layer. Note that the derived model for the RNN is built analogously but not shown in the table to avoid clutter. Specifically, the RNN does not contain all components with the prefix F and the decoder of the RNN takes its input directly from the LSTM.

4.2 Datasets

Moving mnist: The moving mnist dataset [16] consists of two digits (0 to 9) of size 28 x 28 moving inside a 64 x 64 patch. The digits are chosen randomly from the mnist training set and placed at random locations inside the patch. Each digit was assigned a velocity whose direction was chosen uniformly randomly on a unit circle and whose magnitude was also chosen uniformly at random over a fixed range. The digits bounce off the edges of the 64 x 64 frame and overlapped as they move past each other. The training and validation set contains 10000 and 1000 sequences respectively, each 20 frames long.

Moving shapes: The moving shapes dataset consists of two shapes chosen among a square, circle and triangle, each of size 28 x 28, moving inside a 64 x 64 patch. The colors were sampled from a uniform distribution. Its behavior is similar to that of the moving mnist dataset. The training set contains 10000 sequences, each 20 frames long.

Encoder(Image)	$\mathbf{F_{prior}}$**(LSTM)**	$\mathbf{F_{posterior}}$**(LSTM, Encoder)**
7x7x3-16 / 7x7x1-16 conv + ReLu	256 / 128	256 / 128
5x5x16-32 conv + ReLu	256 / 128	256 / 128
5x5x32-48 conv + ReLu	256,256 / 128,128	128,128 / 128,128
3x3x48-64 conv + ReLu		
Decoder($\mathbf{F_{Decoder}}$)	$\mathbf{F_Z}$ **– Sampled from** $\mathbf{F_{prior}}$	$\mathbf{F_{Decoder}}$**(LSTM,** $\mathbf{F_Z}$**)**
3x3x64-48 transpose-conv + ReLu	256 / 128	1024
5x5x48-32 transpose-conv + ReLu	256 / 128	1024
5x5x32-16 transpose-conv + ReLu		
7x7x16-3 / 7x7x16-1 transpose-conv + Sigm		

Table 4.1: Hyperparameters for the moving shapes in red and the moving mnist in blue. The entry in the bracket indicates the component that it receives its input from, with commas denoting a concatenation. The F_{prior} and $F_{posterior}$ both contain 2 separate layers to output the hyperparameters μ and σ^2 of the gaussian distribution. Note that all architectures carry the LSTM cell with 1024 units.

4.3 Training

We minimize the negative of the variational lower bound of equation 3.1, using the cross entropy loss as the metric for the reconstruction error. We initialize all parameters using the glorot uniform initializer [17]. Optimization was performed using stochastic gradient descent with a batch size of 50. We use the learning scheme of RMSProp [18] with a base learning rate of 0.001, decay of 0.9 and with no momentum. We unroll the LSTM for 20 time-steps and train the model for 100 epochs. The experiments are run in a machine with a NVIDIA GTX 780M and takes approximately 7 hours of training time. We use the Tensorflow library [19] to develop the model. Lastly, no pre-processing was applied to the images besides linearly scaling to the range of the sigmoid activation function [0, 1].

The training and validation loss plots, averaged over the timesteps, are summarized in Figure 4.1. It can be seen that the KL divergence for both models show an increase before decreasing. This behavior indicates that the objective function is initially dominated by the reconstruction loss. Thus in the early iterations, the model ignores the latent term and focuses on getting good reconstructions out. As the reconstruction loss stabilizes, the model begins to work on the KL divergence term. Although the divergence term for the moving mnist showed signs of overfitting, we noted in our experiments that this variant, amongst others, produced the best results.

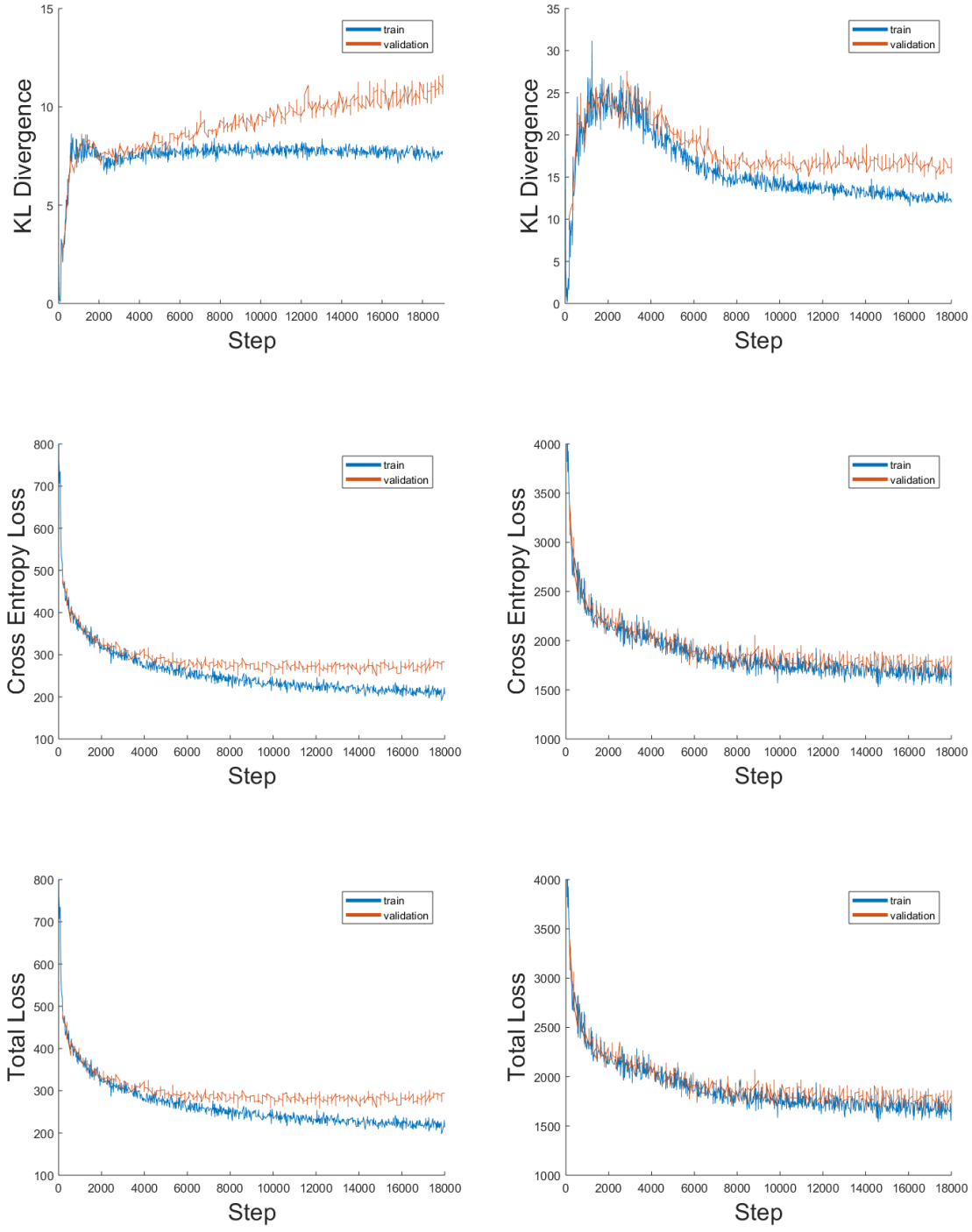


Figure 4.1: Training and validation loss plots for the (left column) moving mnist and (right column) moving shapes datasets.

Chapter 5
Results

5.1 Long Term Prediction

In the first experiment, we study the quality of the image predictions resulting from the different tested architectures. Here, we feed the ground truth as input over the first 20 timesteps and let the model generate its predictions for 100 timesteps. Visual results are presented in figure 5.1.

On both datasets, it can be observed that generations from the RNN get progressively worse without the ground truth as input, especially when there are two objects moving past and occluding each other. For the moving mnist, although the model is able to separate the superimposed digits, reconstructions get worse to the extent that it hardly resembles the digits from the ground truth. Although it does a better job on the moving shapes, we can see that the shapes are also not very well formed. This phenomenon is commonly observed in long-term video prediction because small errors in pixel space - especially during occlusions in this case - exacerbate as predictions are made further into the future.

The VRNN on the other hand, perform very well at recovering from occlusions. We can see that the generated images for both sequences do not exhibit visible deterioriation in quality over time, even towards the end. The digits in the mnist sequence remain interpretable and the shapes on the moving shapes dataset are more well defined compared to the RNN. These findings suggest that conditioning on the stochastic latent variables reduces or even eliminates the impact pixel-level errors have on reconstructions by forcing them to look a certain way. With regards to the changing of the digits / shapes over time, we tried conditioning on the labels by concatenating the class labels at the fully connected layers but the results did not show a definitive improvement. If we take a closer look at reconstructions for the

moving shapes over the first 20 timesteps, we can observe that unlike the RNN, the shapes generated by the VRNN contain a mixture of colors, even when receiving the ground truth as input. This is due principally to the fact that information coming from the latent space acts as noise, which interferes with reconstruction. Lastly, we also noted that the velocity of the objects tend to decrease after the models stopped receiving the ground truth as input and that all models are able to correctly predict the motion after bouncing off the walls.

5.2 Generation Diversity

In the second experiment, we test the VRNN's ability to produce sequences that are novel during repeated runs. Figure 5.2 shows some samples when running the VRNN 5 times on both datasets. Each row indicates a separate run and each column, the generated image in timesteps of 5. Notice how they are diverse in terms of position and shape, and additionally for the moving shapes, color. Additionally, we saw during the runs on the moving mnist that it was impossible, for this specific sequence, to generate a digit that contains the number 4. Lastly, note that a deterministic model can be obtained by sending the mean through the decoder instead of sampling from the latent space.

5.3 Out of domain inputs

In the third experiment, we test the models' ability to deal with out-of-domain inputs by feeding them sequences containing one and three moving objects. Figures 5.3 and 5.4 show the results for moving shapes and moving mnist respectively. For one input object, we see that the VRNN generates a second object close to the first one, even when receiving the ground truth as input. We speculate that this is because the LSTM hidden units and latent encodings of these images are close to the ones that are under occlusion. Thus when conditioned on this, the VRNN generates a sequence imitative of 2 objects occluding and moving past each other. For three objects, it would seem that the VRNN ignores the input images, generating images with only 2 objects. These findings reveal that the additional information coming from the latent space has a strong influence on the decoder, highlighting a minor drawback in recurrent networks that are built this way. Lastly, for both cases, we see the RNN performing better only when receiving the ground truth as input. Without the ground truth, the model reverts to generating images containing 2 objects.

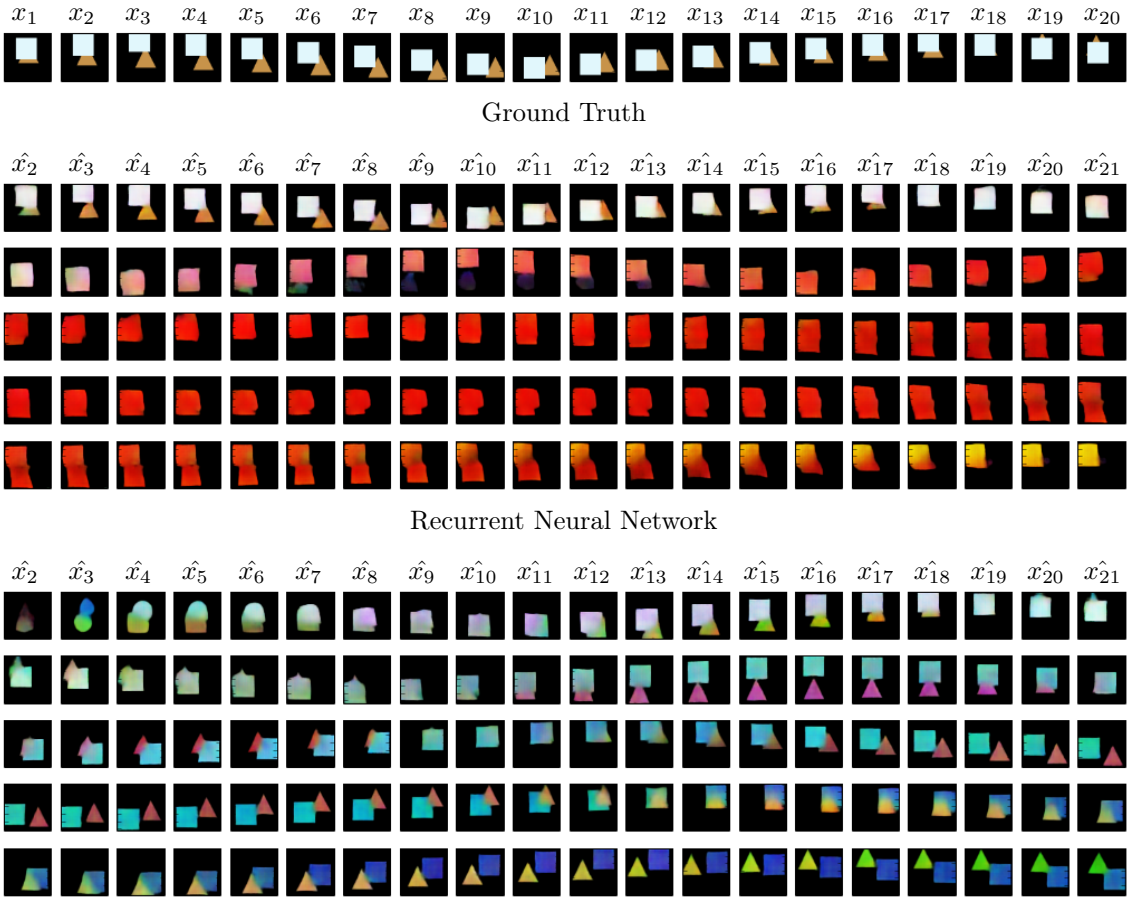


Figure 5.1: Reconstruction results on the (top) moving shapes dataset and (bottom) moving mnist dataset. The models received the ground truth, denoted as x, as input for the first 20 timesteps. Reconstructions from $\hat{x_{22}}$ onwards are thus conditioned on the models' own output.

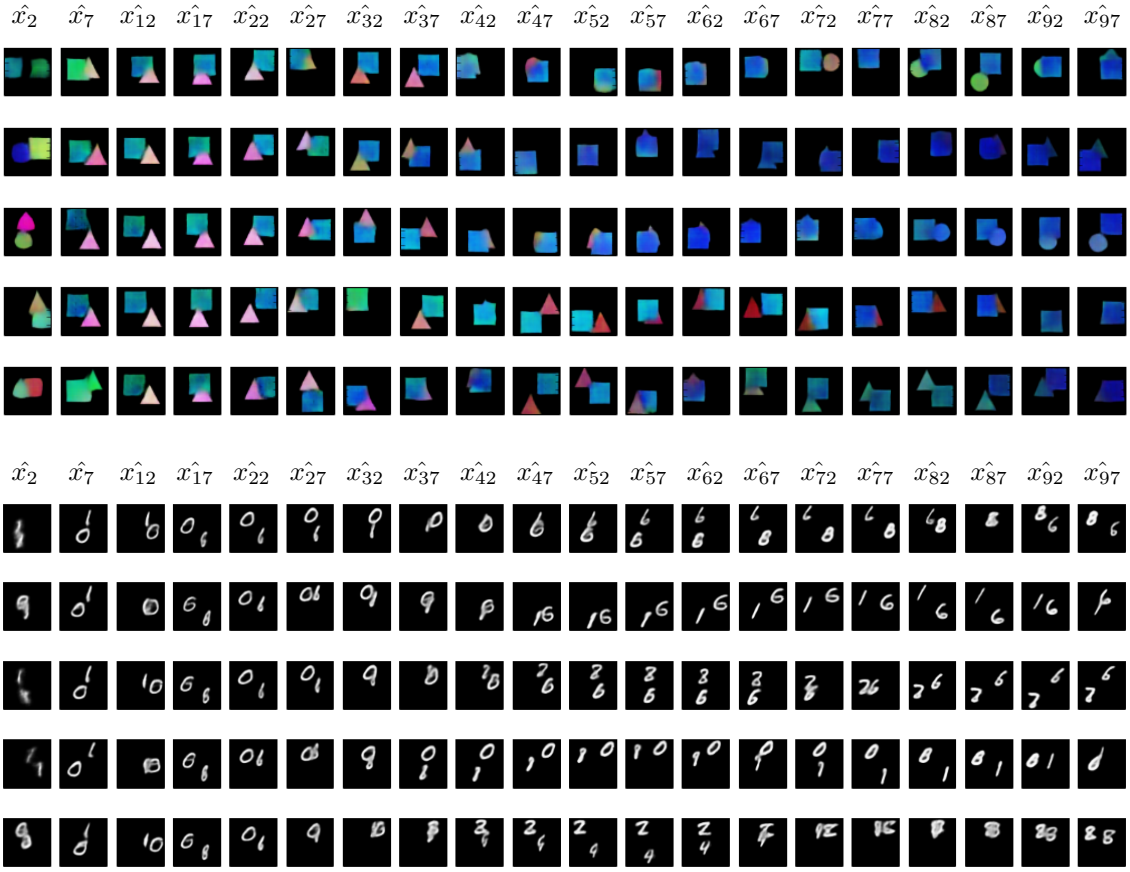


Figure 5.2: Reconstruction results on the (top) moving shapes dataset and (bottom) moving mnist dataset on 5 separate runs. Each row indicates a separate run and each column, the generated image in timesteps of 5. Similarly as before, the models received the ground truth as input (not shown here) for the first 20 timesteps. Reconstructions from $\hat{x_{22}}$ onwards are thus conditioned on the models' own output.

Figure 5.3: Out of domain runs for (top) 1 shape and (bottom) 3 shapes on the moving shapes dataset. The model received the ground truth, denoted as x, as input for the first 20 timesteps. Reconstructions from $\hat{x_{22}}$ onwards are thus conditioned on the model's own output.

Figure 5.4: Out of domain runs for (top) 1 digit and (bottom) 3 digits on the moving mnist dataset. The model received the ground truth, denoted as x, as input for the first 20 timesteps. Reconstructions from $\hat{x_{22}}$ onwards are thus conditioned on the model's own output.

5.4 Latent Space Analysis

We now turn our attention to the latent space to study the effect it has on image reconstructions. For this, we sent in a batch of images to the VRNN then linearly interpolate between the latent encodings of random pairs for a fixed timestep according to the following equation: $v = v_1 + (v_2 - v_1) \times \alpha$, where v_1 and v_2 denote the encoding of image 1 and 2 respectively. At each step of the interpolation, we pass this information through the decoder network to generate an image. We noted in our experiments that there were no noticeable changes in the output when varying α between the range $[0, 1]$. Unlike in the VAE where the decoder receives only the latent information as input (figure 2.2), there exists a non-linear interaction between the latent information and the LSTM hidden units in the VRNN as they get concatenated and passed through the fully connected layers $\mathrm{F}_{\mathrm{decoder}}$. Changes in the latent variable therefore, might not directly affect the output. This makes it difficult to study the effect the latent space has on reconstructions and might also suggest that the LSTM cells have a dominant effect on reconstructions. We experimented with many other combinations, varying the number of units in $\mathrm{F}_{\mathrm{prior}}$ and F_{z} but did not see a noticeable change in reconstructions. Thus the results shown in figure 5.5 are generated by incrementing α from 0 to 10 in steps of 1.

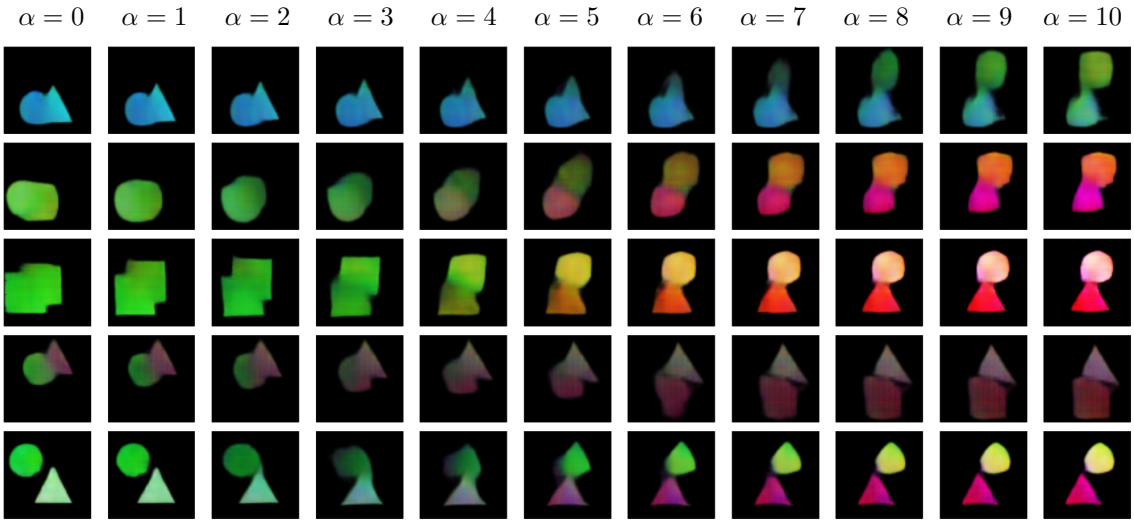


Figure 5.5: Interpolating between 2 points at the output of F_z according to $v = v_1 + (v_2 - v_1) \times \alpha$. We did not show the images corresponding to v_2 since they all look nothing like the image corresponding to $\alpha = 1$.

Chapter 6
Conclusion

We have presented a model for future frame prediction. Its core components are recurrent network injected with stochasticity. Experimental results on two synthetic datasets have shown that the use of latent variables enable the model to persistently generate predictions well beyond the time steps it was trained for. The results also reveal a minor limitation intrinsic to models built this way: that information coming from the latent space can negatively influence image reconstructions, as seen on the moving shapes, and that the latent information restricts the model's ability in dealing with out of domain inputs. This then raises the question on whether recurrent networks built this way will be capable of handling real world datasets since for any given context there exists a large variety of background scenes, objects of interest and view points. This might thus suggest the necessity of object detectors to isolate moving entities from the background then learn its activity over time.

We believe that there are many improvements to be made that will further enhance the performance of the model developed in this paper. Here we list some that will be left as future work. (1) LSTMs, although proven powerful at handling temporal information, cannot maintain structural locality due to the matrix multiplications involved. We thus suggest the use of convolutional LSTMs [20] which replaces said operations of the LSTM with convolutions, retaining 2D spatial information in both the cell state and output. (2) In order to retain local information returned by the convolutional LSTMs, we suggest eliminating all fully connected layers, replacing them instead with the spatial bilinear pooling operation described in [21].

Bibliography

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[2] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *Computer Vision and Pattern Recognition*, 2014.

[3] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," *Proceedings of Machine Learning Research*, 2014.

[4] I. Sutskever, J. Martens, and G. Hinton, "Generating text with recurrent neural networks," *International Conference on Machine Learning*, 2011.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Neural Information Processing Systems*, 2014.

[6] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *Computer Vision and Pattern Recognition*, 2015.

[7] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," *Neural Information Processing Systems*, 2016.

[8] J. Chung, K. Kastner, L. Dinh, K. Goel, and Y. B. Aaron Courville and, "A recurrent latent variable model for sequential data," *Neural Information Processing Systems*, 2015.

[9] M. Fraccaro, S. K. Sonderbyz, U. Paquet, and O. Winther, "Sequential neural models with stochastic layers," *Neural Information Processing Systems*, 2016.

[10] P. J. Werbos, "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE*, 1990.

[11] Y. Bengio and P. F. Patrice Simard and, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, 1994.

[12] S. Hochreiter, Y. Bengio, and J. S. Paolo Frasconi and, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies." *A field guide to dynamical recurrent neural networks*, 2001.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *International Conference on Learning Representations*, 2013.

[14] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *Neural Information Processing Systems*, 2014.

[15] H. L. Kihyuk Sohn and X. Yan, "Learning structured output representation using deep conditional generative models," *Neural Information Processing Systems*, 2015.

[16] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," *International Conference on Machine Learning*, 2015.

[17] Y. B. Xavier Glorot and, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of Machine Learning Research*, 2010.

[18] T. Tieleman and G. E. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, 2012.

[19] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *LREC*, 2010.

[20] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Neural Information Processing Systems*, 2015.

[21] H. Kwak and B.-T. Zhang, "Ways of conditioning generative adversarial networks," *NIPS Workshop on Adversarial Training*, 2016.