DeepGIFs: Using Deep Learning to Understand and Synthesize Motion

Ali Malik Michael Troute Brahm Capoor
Stanford University
{malikali, mtroute, brahm}@stanford.edu

Abstract

Recurrent neural networks (RNNs) have seen widespread use across a variety of generative tasks requiring temporality [6] [18]. In this paper we explore the effectiveness of deep recurrent networks in the task of understanding motion. In particular, we propose a modular architecture for inferring realistic motion from still 'seed images', based on the recent success of generative adversarial networks. We evaluate this architecture on a variety of synthetic datasets and show that our network is able to successfully learn motion and generate new videos from still images.

1. Introduction

How do objects move? Humans are adept at intuiting the the rules of motion from imperfect visual information; rules that determine how objects *can* move, and how they *should* move, with varying degrees of certainty. From a young age we understand, for instance, that an object in motion should remain in motion, and should continue to move in the same direction unless acted upon by an external force; similarly, we understand that when two objects moving towards one another on the same plane collide, they should reverse their directions and move apart. Crucially, we form these understandings long before we form understandings about the physical mechanisms by which they are underpinned. We understand that balls bounce before we understand gravity, or the equations that govern elastic collisions. Unfortunately, as is frequently the case in computer vision, what comes naturally to a human presents a sophisticated learning task for a computer.

Understanding these 'rules of motion', while not critical for understanding existing videos, is central to the task of *generating* realistic sequences of images over time. Each frame must plausibly follow from the frame that precedes it, and objects depicted should move, deform, and interact in ways consistent with the physical rules that govern objects in motion. Structuring the video generation task to account for the importance of consistency with regards to the rules of motion is a challenging task. Situations wherein an algorithm capable of synthesizing convincing natural motion are easily imagined. Consider, for instance, the production of animated films, which at present is a prescriptive task requiring high degrees of human intervention. The implicit learning of physical laws from visual information, further, is a task with implications that are interesting even when divorced from a practical application.

In this paper, we apply deep learning to the problem of producing realistic sequences of images from an initial seed image. Specifically, we present a recurrent adversarial architecture for frame generation, trained on two datasets to produce natural (if highly simplified) motion. We use a modified variational autoencoder (VAE) to produce smooth latent encodings of video frames, which are used as inputs to a Long Short-Term Memory (LSTM) network that predicts the next frame in the current sequence. The outputs of this combined network is enhanced by an adversarial discriminator, which is trained alongside the frame generation network.

In Section 2, we discuss previous approaches to the problem, as well as solutions to other similar tasks that influenced our work. In Section 3, we outline the datasets and preprocessing techniques we used in arriving at our results. In Section 4 we describe our architecture and training procedure in greater detail. In Section 5, we summarize the results we obtained. Finally, in Section 6, we suggest a wide range of future directions for continuing work in this vein.

2. Related work

The task of video generation seems to naturally decompose into two problems: first, finding an effective feature encoding of video frames that can easily and effective reproduce the original frames, and second, generating future frames from the feature encoding of the current frame. We explore the literature surrounding both these problems individually, and then look at work that synthesises both for the specific purpose of video prediction.

2.1. Image generation

Recent literature has seen an explosion of work related to image generation. Architectures like PixelCNN [19], variational autoencoders (VAE) [10], and generative adversarial networks (GANs) [5] have proven to be effective in learning a latent distribution from which new images can be sampled.

VAEs have been applied to a variety of image generation tasks but on their own have several limitations. In particular, experiments have shown that reconstruction from VAEs using pixel-wise difference loss generally results in blurry outputs [17]. In this regard, generative adversarial networks (GANs) have drastically outperformed other alternatives in producing sharp images. GANs work by having the generative model compete with a binary classifier known as the discriminator; the former tries to fool the latter in thinking its outputs are real. We use this idea in our architecture to produce clearer outputs by having a discriminator network on top of our generative model.

2.2. Future Prediction

A large subset of recent research has explored tasks involving some kind of sequence prediction. In particular, recurrent neural networks (RNNs) have been used in widely ranging tasks such as highway trajectory prediction, precipitation forecasting, human trajectory estimation, and video game sequence predictions [1][2][14][16]. In our model, we use a recurrent network to generate future frames from the current frame. Following the convention in literature, we use a Long Short-Term Memory (LSTM) recurrent network [7] due to its training stability and superior results.

2.3. Future Video Prediction

There has been some prior work on learning video representations that capture motion of objects, although this work has been restricted to relatively simple datasets due to the complexity of the task.

One of the earlier forays into this problem was led by Srivastava et. al [17]. Their architecture uses an encoder and decoder LSTM to learn representations of video sequences which they use to predict future frames. Although their results exhibit clear motion, the outputs are fairly blurry and seem to lose the form of the numbers.

Other notable work related to this problem has been done by Zhou and Berg [20] in generating timelapse videos and cinemagraphs from still images. Their model uses an idea similar to ours—a combination of a convolutional autoencoder and LSTM—but only uses the encoding of the initial frame to generate all future frames. Our architecture on the other hand uses the previous generated frame as input to the next timestep and uses a variational encoder to encourage a smoother latent space. Moreover, we train on unsupervised data, hoping to learn motion just from frames.

3. Data

3.1. Datasets

We use two datasets for our experiments. First, we use the Bouncing Balls dataset [12], which consists of animations of 2 balls bouncing around a square box. Collisions between the two balls are simulated fairly consistently, and momentum is conserved realistically during collisions both between balls and between balls and the bounding box of the frame.

Second, we use the synthetic Moving MNIST dataset [17]. This dataset consists of animations of two randomly chosen MNIST digits moving within and bouncing against the edges of the frame. Moving digits do not collide, passing through each other instead, and movement is generally less realistic than in the Bouncing Balls dataset. The limited number of classes allows the nation to focus on motion, while the possibility for variation within classes highlights the importance of an effective encoding mechanism.

Each dataset had a training set of size 80,000 and validation and test sets of size 10,000 each. Both datasets were generated using adaptations of open source scripts [3] [11].

3.2. Preprocessing

To preprocess and standardize our data, we downsampled all our gifs to be of size 64 x 64 and 20 frames long.

4. Methods

To successfully generate motion from still frames, we had to capture both spatial and temporal motion. Convolutional neural networks have proven extremely effective in capturing important spatial information in pictures whereas recurrent models such as LSTMs have been shown to be successful in effectively dealing with sequential tasks. Our architecture attempts to coalesce these two ideas. All our models was implemented with the PyTorch framework [15] and were trained on NVIDIA K80, P100 or V100 GPUs.

4.1. Architecture

Our model consists of four distinctive modules: an encoder network, a decoder network, an LSTM, and a discriminator (Figure 1). At a high level, our architecture encodes the seed frame into a latent vector representation which the LSTM takes as input. At each time step t, the LSTM takes the encoded representation of the last generated frame as well as the previous hidden/cell state and attempts to generate the latent representation of the next frame in the sequence. The decoder network then converts this latent representation back into an image.

The discriminator network is employed in two separate GAN architectures. Firstly, the discriminator and encoder/decoder network are pitted against each other in order to improve the output of the autoencoder. Secondly, the

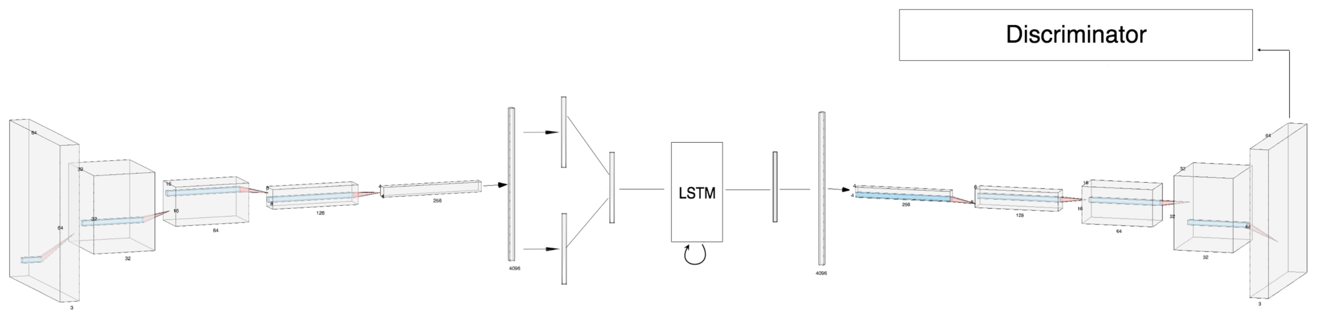

Figure 1. The generative architecture consists of a convolutional encoder network that outputs a latent vector. The LSTM operates on the latent representation and then uses the decoder network to reconstruct the image. A discriminator network tries to distinguish real and generated images.

discriminator is also fed the output at each timestep of the LSTM in order to coax the LSTM to produce output that more closely resembles the initial data.

4.1.1 Encoder/Decoder network

The encoder network takes inputs of size $64x64$ with 1 or 3 channels (depending on the dataset) and applies four convolution layers of the form:

1. 32 filters of size 4×4 with stride 2 and padding 1.
2. 64 filters of size 4×4 with stride 2 and padding 1.
3. 128 filters of size 4×4 with stride 2 and padding 1.
4. 256 filters of size 4×4 with stride 2 and padding 1.

Each of the layers is followed by batch normalisation [9] and a leaky ReLU activation function [13]. The output of the convolutions is flattened and connected via two fully connected layers that each output a vector in $\mathbb{R}_{95}$ of means μ and log variances Σ, respectively, to sample the latent vector z from. The z is sampled normally with mean and standard deviation given by the output of the encoder i.e.

$$z \sim \mathcal{N}(\mu, \Sigma).$$

The decoder network's architecture is the inverse of the encoder, with nearest neighbour upsampling followed by convolution, batch normalisation, and leaky ReLU. The final output is followed by a $\tanh$ nonlinearity.

4.1.2 LSTM

For our RNN, we use a single-layer LSTM with a hidden and cell state size of 200. It takes an input of size 95 (the same as the output of the encoder) and its output (of size 200) is passed through a fully connected layer to produce a new encoding of size 95 which is passed into the decoding network to produce the next frame of the animation.

4.1.3 Discriminator

The discriminator network, for the most part, has the same architecture as the encoder discussed in 4.1.1. However, the flattened output of this architecture is passed into a fully connected layer that outputs a vector of size 95, which then undergoes Batch Normalization and a leaky ReLU activation function. Following this, the vector is passed into another fully connected layer that outputs a scalar logit. In 4.2, we discuss how this discriminator is trained but in summary, we use Binary-Cross Entropy loss on this logit.

4.2. Objective

To evaluate the progress of our network, we used a combination of different loss functions to backpropagate into the network.

The first portion of our loss is the reconstruction loss, measured by mean squared error (MSE) between the generated frames and the ground truth frames of the sequence (G(x) refers to the output of the network):

$$\mathcal{L}_{\text{rec}} = |G(x) - x|^2$$

It is a clear sentiment in the literature that using purely a pixel-wise reconstruction loss results in blurry output [17], so we took inspiration from the recent success of generative adversarial networks in producing better quality images. In this respect, the output of the generator network is evaluated by a discriminator D that is trained to distinguish between real and generated frames. We use binary cross entropy loss to train the discriminator.

$$\mathcal{L}_{\text{gan}} = \log(D(G(x))) - \log(1 - D(G(x))$$

In addition to this, we also penalised the encoder portion of the network from straying too far from a standard distribution when encoding images in the latent space. This requirement was imposed to encourage a smooth transformation of generated outputs as the latent variable is smoothly varied. The loss term was computed using the KL divergence between the encoder's outputs and $\mathcal{N}(0, 1)$:

$$\mathcal{L}_{\text{KL}} = D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||\mathcal{N}(0, 1)).$$

Together, these three terms, weighted by some constants λ_{gan} and λ_{KL}, gave the overall loss

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{gan}} \cdot \mathcal{L}_{\text{gan}} + \lambda_{\text{KL}} \cdot \mathcal{L}_{\text{KL}}.$$

For our model, we used $\lambda_{\text{gan}} = 1e-4$ and $\lambda_{\text{KL}} = 1e-8$,

5. Experiments

In this section we outline the experiments we performed on our two datasets. Visualisations of gifs, frame by frame are provided in the paper but the medium of a pdf document does not lend itself to displaying animated figures. Animated gifs can be found on this page

5.1. Bouncing Balls

We chose to begin our experimentation on the Bouncing Balls dataset. This dataset is a good starting point as the Bouncing Ball GIFS are relatively simple, since they have only one colour channel and the balls are clearly defined and therefore easy for a network to recognise. Thus, the dataset better facilitates the network learning rules of motion and collision, without the additional burden of reproducing more complex shapes and colors.

While training, we used the Adam update with a learning rate of $1e-3$ for the VAE-LSTM as well as the discriminator and a batch size of 32.

Figure 2 shows the model's frame-by-frame output on the test set. The results are fairly encouraging for several reasons. Firstly, we see that between each pair of consecutive frames, the balls undergo motion. Moreover, local acceleration and deceleration appear fairly realistic, further indicating that the network understands rules of how balls move between frames. In addition, each ball generally retains its shape under translation rather than deforming, meaning that the network recognises that its primary task is that of translation. Finally, the balls – when they collide with each other (as in Figure 2B) or with the wall (as in Figure 2C), move away from the object the collided with, meaning that the model has learned the general mechanic of collisions.

That said, the model's output is not quite perfect. For example, while the local acceleration and deceleration of balls is realistic, it is not globally consistent and so the balls occasionally jitter back and forth in a fairly inconsistent manner (as in Figure 2C). One way to fix this is to pass in a series of seed frames rather than a single frame in order to give the network some sense of where the ball is moving. In addition, we noticed that the network was prone to occasionally producing new balls, which we hypothesised was because of the VAE's learned latent space varying smoothly and thus susceptible to small variations. We display the effects of perturbing the latent representation of a particular image in Figure 3.

5.2. Moving MNIST

The Moving MNIST dataset provides several interesting challenges for a video generation model, in addition to those presented by the Bouncing Balls dataset. Firstly, the shapes of these digits are significantly more complex than the balls, but must still be preserved under translation. In addition, the digits pass through each other instead of moving apart upon collision, and so the model must be able to disentangle the digits and move them separately regardless of their location. As a result, the model must not only learn the physics of natural motion and collision (with walls), but also be able to keep track of each moving entity individually, even if they overlap.

We used the Adam update to optimise the network. We used a learning rate of $1e-3$ for the VAE-LSTM as well as the discriminator and a batch size of 32

5.2.1 Smoothness of motion

The model's frame-by-frame output on the test set can be seen in Figure 4. Firstly, we note that the generated outputs demonstrate clear and consistent motion between frames which serves as a promising reinforcement that our network is able to understand and synthesise notions of movement. Moreover, upon seeing the animated motions between the generated GIFs for the Bouncing Ball dataset and the Moving MNIST dataset (as shown on this page), we noticed the motion in the latter was actually much smoother and more consistent. We believe this might have to do with the digits being asymmetric and thus allowing the network to use the asymmetries as points of reference. This would be an interesting avenue to investigate further.

5.2.2 Morphing digits and disentangling digits

We also noticed that the same problem from the Bouncing Ball dataset seems to appear in the generations of this dataset. In particular, the digits morph from their original shape into other digits as they move. Again, we think this is due to the way the latent space is structured and this hypothesis seems to be supported by a visualisation of viewing the outputs when the latent space is varied (Figure 5). Since the variational encoder ensures the generations trans-

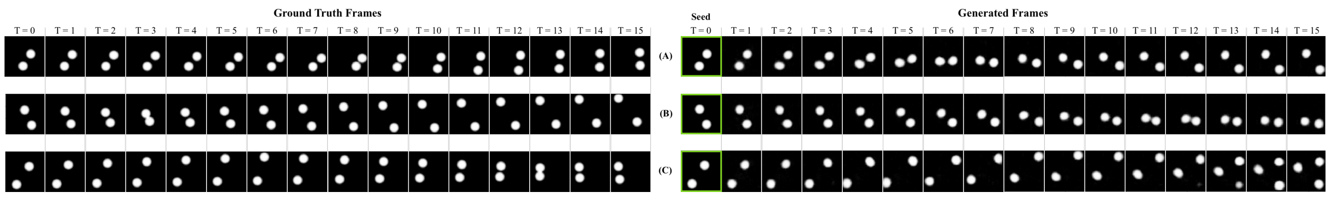

Figure 2. LSTMVAE-GAN output on test set. The left column shows the ground truth frames of three different animations (A), (B), (C) from the dataset. The right column shows the output generated by our model from each initial seed frame (highlighted in green).

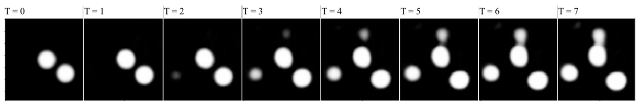

Figure 3. Outputs from the decoder network as the latent vector z is varied across an arbitrary dimension. The decoder smoothly interpolates between two pairs of positions for the ball.

form smoothly, we see the digits morph smoothly from one to another as they move.

Another thing the network noticeably struggles with is disentangling the digits when they go through each other. We hope to possibly deal with this issue in the future by conditioning on the labels of the digits in the image. We could also potentially have a classifier to detect the images present in the image and add a further adversarial loss for when the generator produces digits that were not in the original image.

6. Future work

While promising, our results are still fairly preliminary, and opportunities for future research in this space abound. Work in the immediate future should be concentrated on making models which are more resilient against the compounding of small errors over longer time frames.

This is especially true of the Moving MNIST dataset, for which predicted sequences frequently portray numbers melding together or fading altogether, billowing and dissipating like smoke. We hypothesise that this occurs because during train-time the model is only evaluated on its ability to generate single-frame predictions from seed frames; because the seed is effectively 'reset' to ground truth after each predicted frame is generated, the model is not penalised for introducing small irregularities that, when compounded over several iterations, lead to significant deviations from ground truth.

In the bouncing ball dataset, this issue is visible in the lack of conservation of the momentum of individual balls over long periods. Locally, over small time sequences, ball movement is surprisingly even, displaying smooth acceleration and deceleration. This is critical to maximising the model's training objective; over longer sequences, however, balls will frequently switch direction, meandering organically and seemingly at random (though of course the direction of movement is in fact determined by small biases in the randomly generated dataset). Modifying the training objective to penalise deviations over longer time sequences would likely improve the continuity of the balls' momentum.

Alongside modifying the training objective, several other techniques appear promising in minimising long-term deviations. Training and evaluating on longer seed frames would likely improve the consistency of the model, especially in the first few generated frames. Similarly, novel architectures that allow conditioning a model's output on a chosen direction or set of directions could improve consistency. Finally, Temporal LSTM models have also been proposed, and seem highly applicable to this problem.

Of course, our synthetic datasets do not present particularly realistic or sophisticated examples of objects in motion; more realistic datasets present a massive area for future exploration. As a first step in this diretion, we trained our VAE on the Tiny Imagenet dataset and reproduce its outputs in Figure 6. In a similar vein, datasets with more variation in the number, shape, and distance (relative to the visual point of observation) of objects in motion will allow the training of much more sophisticated models.

Critical to advancing research towards more sophisticated video generation is the curation of large-scale dataset for the problem. Work in this direction, while not immediately rewarding, will be crucial moving forward, especially for learning tasks centered around visual representations of more complex systems of physical interactions.

7. Conclusion

In this paper, we propose an architecture for generating animated GIFs from still image frames. The model combines the most effective techniques from generative networks, future sequence prediction, and adversarial training in a modular way to effectively understand and synthesise motion in images. We show that our model preforms successfully on this task, generating animations of objects that move smoothly and bounce off each other and walls when colliding.

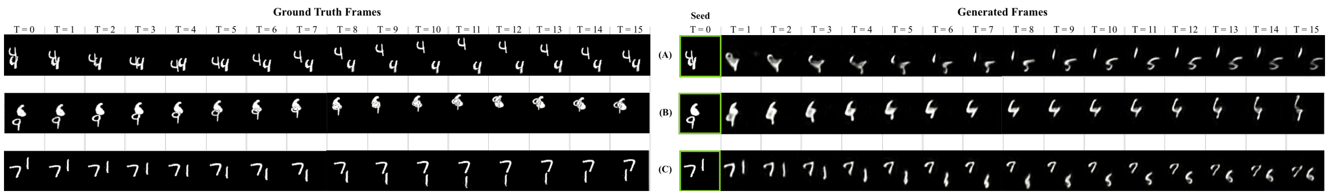

Figure 4. LSTMVAE-GAN output on test set. The left column shows the ground truth frames of three different animations (A), (B), (C) from the Moving MNIST dataset. The right column shows the output generated by our model from each initial seed frame (highlighted in green).

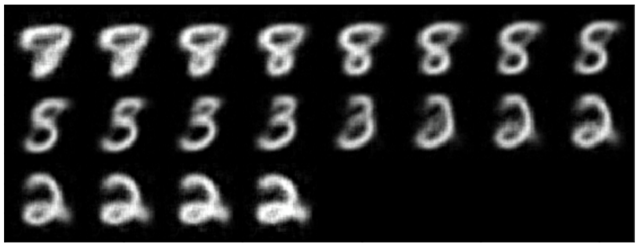

Figure 5. Varying latent space of MMNIST generator.

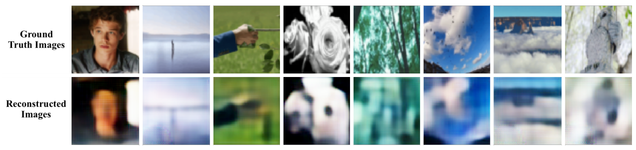

Figure 6. Reconstruction capability of the encoder/decoder network trained on Tiny Imagenet.

Nevertheless, we see a lot of avenues for improvement in our model. Although it is able to synthesise new motion, the smoothness of the motion could be better and moreover, the continuity of this motion over time could be markedly more consistent. We believe this problem mainly stems from the fact that the network only has a single seed frame from which it arbitrarily picks motion to generate. Training our network with longer seed frames or conditioning on a direction would possibly yield more robust results.

Furthermore, the network could perform better in retaining the forms of the objects as the animation progresses. The error that arises from this issue compounds as the generation progresses since the last generated frame is used to generate the next one. To this end, we hope to improve the network by possibly conditioning on the objects present in the initial frame or bolstering our discriminator with a categorical classifier that penalises the generator for producing digits that are not in the original frame. In general, quantitatively evaluating generative models is a difficult problem and work has been done in using better metrics [8]. We hope to apply some of these ideas to our model in the future.

Motion is a difficult problem for computer vision due to a variety of factors such as an added dimension of complexity, computational and memory limitations, and sparsity of datasets. We intend for this to be a small step forward in the task of understanding motion using deep networks and hope to continue our exploration on more complex settings.

8. Acknowledgements and Author Contributions

All the authors contributed equally to this paper. A. Malik, M. Troute and B. Capoor all helped design and implement the various models, tuning hyperparameters and designing infrastructure. A. Malik led the literature review, and M. Troute and B. Capoor each wrote scripts to produce one of the datasets.

The authors would like to thank Jared Quincy Davis for his invaluable suggestions of potential avenues for research as well as the CS231N course staff – past and present – for designing such an edifying and enjoyable course.

References

[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, June 2016.

[2] F. Altché and A. de La Fortelle. An LSTM network for highway trajectory prediction. *CoRR*, abs/1801.07962, 2018.

[3] Z. Gan. `https://github.com/zhegan27/TSBN_code_NIPS2015/blob/master/bouncing_balls/data/data_handler_bouncing_balls.py`.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[6] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[8] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. *CoRR*, abs/1610.00291, 2016.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[10] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, Dec. 2013.

[11] T. Lee. `https://gist.github.com/tencia/afb129122a64bde3bd0c`.

[12] W. Lotter, G. Kreiman, and D. D. Cox. Unsupervised learning of visual structure using predictive generative networks. *CoRR*, abs/1511.06380, 2015.

[13] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. 2013.

[14] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2863–2871. Curran Associates, Inc., 2015.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[16] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 802–810. Curran Associates, Inc., 2015.

[17] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015.

[18] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 1017–1024, New York, NY, USA, June 2011. ACM.

[19] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

[20] Y. Zhou, Y. Song, and T. L. Berg. Image2gif: Generating cinemagraphs using recurrent deep q-networks. In *WACV*, 2018.