Video Frame Prediction with Deep Learning

Megan Backus
Stanford University
backusm@stanford.edu

Yiwen Jiang
Stanford University
yjiang98@stanford.edu

Dana Murphy
Stanford University
dmurphy7@stanford.edu

June 2, 2022

Abstract

In this work we adapt and evaluate the performance of two different architectures, a Convolutional LSTM network and a GAN model, for human motion video frame prediction. Each of these models outputs a sequence of future video frames conditioned on a sequence of past video frames. In particular, our Convolutional LSTM model has an input of four past frames and predicts one frame, while our GAN model receives six past frames and outputs six future frames. We conducted experiments on the UCF101 dataset. The predicted frames are evaluated visually and quantitatively using MSE, SSIM and PSNR. Our results show that the GAN trained on a subset of the dataset outputs plausible predictions, but was outperformed by the Convolutional LSTM model trained on the full dataset, which achieved an SSIM of 0.8743.

1 Introduction

Predicting future human motion has important applications in many fields, such as photo-to-video conversion and decision-making systems. While relatively simple for humans, human motion prediction is a challenging task for computers due to filming factors such as lighting changes, occlusions and camera position, the unpredictability of human intent, and the presence of multiple plausible outcomes. Several techniques have been proposed for frame prediction, but many have not been applied to human action. As such, this work adapts and evaluates the performance of two of these architectures, a convolutional long short-term memory (LSTM) network and a generative adversarial network (GAN) model, to predict future frames of video clips depicting human motion.

Each of these models receives a sequence of images as the input, representing the past video clips, and outputs a different sequence of images, representing the predicted video clips. The frame sequence is represented by a tensor, with dimensions (T, H, W, C) where T is the number of frames, C is the number of channels, H is the height of each frame, and W is the width of each frame. The output has all the same dimensions as the input with the exception of T, which we varied. We evaluate the predicted frames both qualitatively and quantitatively; qualitatively, we visually examine whether the generated frames are a coherent continuation of the input data; quantitatively, we use Mean Standard Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) to compare the generated frames with the future frames in the original clip. Our results show that the GAN trained on a subset of the dataset output plausible predictions, but was outperformed by the Convolutional LSTM model trained on the full dataset, which achieved an SSIM of 0.8743.

2 Related Work

Future frame prediction is a widely researched topic. Optical flow algorithms and CNN architectures have been used to predict the motion of every pixel [1, 2]. This flow vector is then used to generate new images, representing future frames. Instead of producing im-

ages based on per-pixel metrics, convolutional recurrent neural networks (RNN) were suggested for frame prediction [3, 4]. Following this baseline work, several different approaches have been taken for frame prediction. One branch of research suggests using LSTMs, whose ability to capture temporal correlation is appealing for video prediction [5, 6, 7]. Shi et. al found that a Convolutional LSTM outperforms a fully-connected LSTM for precipitation nowcasting [5]. We found this to be a particularly interesting approach and in this project experiment with its potential to perform well in human activity video frame prediction. Alternative techniques involve incorporating stochasticity into the models, which handles future frame uncertainty by averaging over a set of possible predictions [8, 9]. Another promising approach for reducing uncertainty is to use an adversarial network [10, 11, 12, 13]. By using a GAN in addition to a multi-scale architecture and modified loss function, the authors of [11] achieved sharper, but less similar predicted images compared to those predicted by [3]. [14] builds upon this idea by incorporating a progressively growing structure, as suggested in [15]. We found this work to be particularly interesting, as it outperformed three other architectures on three datasets; MovingMNIST, KTH Action and Cityscapes. Our work adds to this body of research by applying the techniques from [5] and [14] to human motion, and comparing these two very different deep learning architectures.

3 Methods

3.1 Convolutional LSTM

The first approach we implemented is a Convolutional LSTM model, which is effective in the frame prediction task due to the model's ability to capture both temporal and spatial correlations. The Convolutional LSTM model was presented by Shi et al. in 2015 for the purpose of precipitation nowcasting [5]. To start, the model follows the general structure of the traditional LSTM. LSTM is a modified version of an RNN that is designed to solve long-term dependency problem such that information can be maintained in memory for longer period of time. The LSTM network achieves this by utilizing multiple gates instead of one tanh layer in its repeating modules and cell states that get modified slightly in each cell. These added layers include a "forget gate" that decides which elements of the previous cell state to forget, an "update gate" that decides which elements of the previous cell state to update and the values to add to those elements. The final output of the cell will be based off of the updated cell state. Specifically, a sigmoid layer is used to decide the parts to output and tanh layer to force the values between -1 and 1. The below equation describes a single LSTM module:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
C_t &= f_t \circ C_{t-1} + i_t \circ tanh(W_{xc}X_t + W_{hc}H_{t-1} + b_c) \\
\sigma_t &= \sigma(W_{xo}X_t + W_{ho}H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= \sigma_t \circ tanh(C_t)
\end{aligned}
$$

The Convolutional LSTM proposed by Shi et al. modifies the original LSTM module in that convolution is used instead of matrix multiplication [5]. The updated equations are shown below:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
C_t &= f_t \circ C_{t-1} + i_t \circ tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
\sigma_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= \sigma_t \circ tanh(C_t)
\end{aligned}
$$

where i, f, o, c, h are the input gate, forget gate, output gate, cell activations and hidden states respectively.

The final network design stacks three of these convolutional LSTM layers together, each followed by a 3D batch normalization layer and with a final 2D convolutional layer that predicts a frame from the hidden state at the last time step of the last layer. We apply this modified model to our human activity video frame prediction task.

3.2 GAN

The second technique we used is a multi-scale architecture with adversarial training, as suggested in [11]

and [14]. A GAN consists of a generative model G and a discriminative model D that are trained simultaneously. In the video prediction setting, G learns to generate frames that are challenging for D to predict whether they are manufactured by G, and D learns to discriminate which frames are created by G.

One of the main challenges with frame prediction is the presence of multiple plausible outcomes; when pixel-wise loss is used, these equally-likely predictions are averaged and lead to a blurred frame [11]. However, a GAN architecture helps alleviate this problem. The input frame followed by the averaged output frame is an unlikely sequence, and so D should be able to discriminate such sequences easily. This forces G to learn to output predicted frames with an increasingly similar distribution as the real frames.

Specifically, we trained FutureGAN, the progressively growing, encoder-decoder model proposed in [14]. The loss function is a combination of the application specific loss and the adversarial loss. For the application loss we used Binary Cross Entropy, and for the adversarial loss we used Wasserstein GAN with gradient penalty (WGAN-GP) loss with epsilon-penalty. The WGAN-GP with epsilon penalty loss function is shown below

$$L_D(x, \tilde{x}, \hat{x}) = E[D(\tilde{x}] - E[D(x)] + \lambda E[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] + \epsilon E[D(x)^2]$$

, where $\tilde{x} = G(z)$, ϵ is the epsilon-penalty coefficient, and λ is the gradient-penalty coefficient.

3.2.1 Progressive Growth Structure

Particularly with image and video data like the UCF101 data we use, training GANs is a highly unstable process. As such, a progressive growth structure allows us to perform the beginning stages of the training process on lower resolution data and learn broader image features first, and then gradually increase the frame resolution [15]. FutureGAN begins by transforming the input frames to 4x4 px resolution frames. Layers are then added during training, increasing the resolution to 8x8 px, 16x16 px, 32x32 px and 64x64 px in intermediary steps, and then finally, to the desired 128x128 px. To ensure a seamless transition between two different resolution steps, layers are added to the network in a two step process, comprised of the *transition phase* and the *stabilization phase*. In the *transition phase*, the network regards the layers operating on the larger resolution as a block with weights α which increase linearly from 0 to 1, and interpolates inputs of the two different resolutions. This is followed by the *stabilization phase*, in which the network resolution is continually doubled after training for a certain number of epochs.

3.2.2 Generator Design

The role of the generator G is to receive an input frame sequence and output future frame sequences that are able to fool the discriminator D. FutureGAN's G uses an encoder-decoder architecture, where the encoder learns a latent representation of the historical frames and the decoder uses this latent representation to generate the predicted frames. As such, 3D convolutional layers are used in order to encode and decode both the spatial and temporal parts of the input sequence. Furthermore, 3D convolutional layers with asymmetric filter shape and strides are used to handle upsampling and downsampling between layers operating on different resolution frames, in the decoder and encoder, respectively. Two additional convolutional layers are used to increase the network's resolution in both the encoder and the decoder. Other layers are included to upsample and downsample between the input sequence's and predicted sequence's temporal depth. Lastly leaky ReLU with a parameter of 0.2, followed by a pixel-wise feature vector normalization layer, are used after each convolution in the hidden layers [15]. The exact generator design is shown in Figure 1.

3.2.3 Discriminator Design

The discriminator D learns to identify sequences created by the generator G. It receives a ground truth frame sequence and returns a score, ranking the provided input as real or fake. D downsamples via 3D convolutional layers with asymmetric filter shape and strides. Additionally, mini-batch standard deviation is used in one of the last layers to increase variation in G's outputs [15]. This generates a feature map of the scalar standard deviation of every feature in

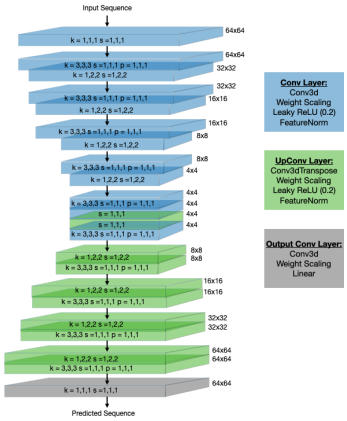

Figure 1: GAN generator architecture. This diagram depicts the generator design for the full 128x128 generator. Similar models with fewer layers are used in the 4x4, 8x8, 16x16, 32x32 and 64x64 models that are trained during progressive growth. Adapted from [14].

each spatial and temporal location in the mini-batch, averaged over all features and spatial and temporal locations. Finally, the output layer is fully connected and followed by a linear activation function in order to produce the score.

4 Dataset

We use the UCF101 dataset, which includes approximately 13,320 video clips corresponding to 101 broad activity classes [16]. The videos in each class are grouped into 25 groups, which each consist of 4-7 videos of an action. We selected this dataset as it provides large variability in both actions and filming features, such as camera motion, object appearance and pose, object scale, viewpoint, background clutter, lighting conditions. Sample video frames from this dataset are visualized in the Appendix.

For use with our Convolutional LSTM model, we implemented a custom collate function to remove the audio and label data, and keep only the video data in each batch. We then used PyTorch's UCF101 class to specify certain desired parameters for the videos, such as the number of frames per clip and steps between clips.

For the GAN, we preprocessed the data by using FFmpeg, a suite of libraries for handling multimedia data, to split each video into its component RGB frames, with a specified frame rate and image size [17].

5 Experiments

5.1 Convolutional LSTM

We conducted experiments with the Convolutional LSTM model using both a subset of and the full UCF101 dataset. Videos in this data set have a frame rate of 25 frames per second (fps). For the sake of keeping training time to a manageable level, the data in these experiments are loaded as a sequence of frames at a resolution of 64x64.

To get a baseline idea of the predictive power of the model, we started with predicting one future frame given an input sequence of 4 frames. A total of 5 video frames was chosen to load per training example (4 input frames, 1 target frame) in order to reduce the GPU memory requirement during training.

To select the learning rate, we performed a hyperparameter search over a sequence of learning rate values in the range [1e-3, 1e-6]. From this search, a learning rate of 1e-6 was selected as the optimal value to avoid the exploding gradients problem while increasing training speed as much as possible. An Adam optimizer was used to optimize over the binary cross entropy loss. A batch size of 64 was chosen as the largest batch size (which is optimal given the large training data set size) that could be used without exceeding the available GPU memory.

In order to visualize intermediate results in a more time efficient manner, we decided to first train the Convolutional LSTM model on a subset of the full UCF101 activity training dataset. This subset is comprised of 365 videos in the following four action classes: Baseball Pitch, High Jump, Volleyball Spiking and Walking with Dog. The corresponding training set consists of 151 videos from these same activity classes. The Convolutional LSTM model was trained for 20 epochs for approximately 60 hours using an NVIDIA T4 GPU.

To further improve the frame prediction results of the trained Convolutional LSTM model, we

conducted further training on the full UCF101 dataset. Because of the massive size of the full UCF101 dataset (13,320 videos), training using random weight initialization would likely require a long training period before stabilizing. To reduce training time, weights were initialized using the learned weights previously trained on the data subset. The model was then trained for 7 epochs for approximately 209 hours.

The trained model was also evaluated on its robustness to frame rate to explore how well the model, which was trained on the full UCF101 dataset, is able to predict future frames using test video data at a lower frame rate.

5.2 GAN

The FutureGAN model was trained on the same data subset mentioned above, using videos with a default frame rate of 25 fps and a size of 128x128 pixels. We passed six frames as the input sequence and predicted six future frames.

Since we are using a progressively growing model, which varies from training on 4x4 pixel images to 128x128 pixel images, we have an increasing number of parameters. As such, to optimize the trade-off between accuracy and GPU memory, we use a different batch size on each of the intermediate models; 32 for the 4x4 model; 16 for the 8x8 model; 8 for the 16x16 model; 4 for the 32x32 model; 2 for the 64x64 model; and 1 for the 128x128 model. We also used an Adam optimizer with parameters $\beta_1 = 0.0$ and $\beta_2 = 0.99$ and had a learning rate of 0.001. For the WGAN-GP loss with epsilon-penalty, we used penalty coefficients $\lambda = 10$ and $\epsilon = 0.001$ [15, 14].

We trained the model for a total of 161 epochs; 20 epochs on each of the 5 intermediary models (10 for the *transition phase* and 10 for the *stabilization phase*) and 61 on the final 128x128 stage, which took approximately 66 hours.

We then evaluated the final 128x128 model on our original test set, in which the videos have a frame rate of 25 fps, and on two modified versions of the same clips: one with a frame rate of 12 fps and the other with a frame rate of 6 fps. We selected these three values because they are each large enough to ensure that the six predicted frames correspond to timestamps that have ground truth frames in the original clips, and thus enable us to accurately characterize the network's performance, but the downsampling factor of two is significant enough to observe meaningful changes in the evaluation.

In order to more accurately compare the GAN's performance to the Convolutional LSTM network, we also evaluated the intermediate 64x64 model on the same video clips as before, but this time scaled to 64x64 pixels and with the default frame rate of 25 fps, in order to match the video parameters used in evaluating the Convolutional LSTM model.

6 Results

To evaluate the models' performance quantitatively, we use three metrics, Mean Square Error (MSE), Peak Signal-to-Noise Ration (PSNR), and Structural Similarity Index (SSIM) to compare the predicted frames to their corresponding ground truth frames.

MSE measures the average squared differences between every pixel in two images [18]. For RGB images, it is defined by the following equation, where mxn images I and K are the ground truth frame and the predicted frame, respectively, and (i, j, c) represents the particular pixel location and color channel being compared:

$$MSE = \frac{1}{3mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \sum_{c=0}^{2} [I(i,j,c) - K(i,j,c)]^2 \tag{1}$$

PSNR is defined in terms of the MSE, and represents the ratio between the maximum pixel intensity of the ground truth image divided by the MSE between the ground truth and predicted images [19]. It is defined below, where MAX_I^2 is the maximum pixel intensity of image I, and MSE is the MSE between image I and image K:

$$PSNR = 10 \log_{10}(\frac{MAX_I^2}{MSE}) \tag{2}$$

The last metric, SSIM, considers dependency between neighboring pixels [20]. It is computed by considering two NxN windows x and y, the mean and variance of pixel intensities over each of these windows $\mu_x, \mu_y, \sigma_x^2, \sigma_y^2$, respectively, the covariance of the

pixel intensities of these windows, σ_{xy}, and stabilization variables, c_1 and c_2, that depend on the dynamic range of the pixel values and a constant:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2} \quad (3)$$

As can be seen in Figure 2, the Convolutional LSTM model, trained over the entire training dataset, produces a predicted frame that is visually relatively similar to the ground truth frame. While the sample visualized result shows some blurriness in the predicted frame, the image is able to capture a difference in movement between the input frames and the new predicted frames, with the predicted frame showing the human subject positionally-lower to the ground with respect to the green line than in the four input frames.

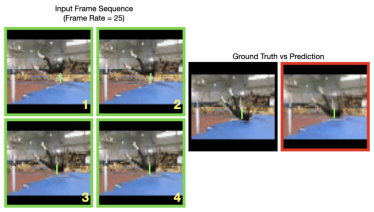

Figure 2: A sample test video prediction from the Convolutional LSTM model. The input sequence of 4 frames is highlighted in green on the left, and the predicted fifth frame is shown in red on the right next to the ground truth frame. To better visualize the falling motion of the video's subject, a green line representing the distance between the falling person and the ground in the first frame is placed at the same pixel location in each frame.

Quantitative metrics comparing the average peak signal-to-noise ratio, pixel mean squared error, and structural similarity between ground truth and predicted frames over the test dataset are listed in Tables 1, 3 and 5.

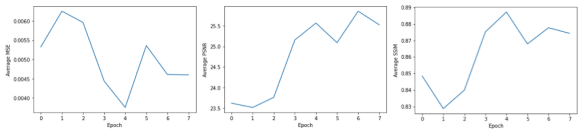

Figure 3: Average MSE, PSNR, and SSIM values evaluated at each training iteration over the full test set. At epoch 0, weights trained for 20 epochs over the data subset are used.

Figure 3 depicts the average MSE, PSNR, and SSIM values at each training iteration over the entire UCF101 dataset. As can be seen from each of these figures, there is a significant decrease in average MSE and significant increase in average PSNR and SSIM between the model trained over the data subset (shown here as epoch 0) and the model trained over the full dataset at epoch 7.

Results for a sample test video for the FutureGAN model are also visualized in Figures 4, 5, 6 for different frame rates and resolution. We can see from the visualized example that the predicted frames relatively capture the same subject components of the input frames as well as some movement of the human subject, but also display blurriness, which increases with future frames.

The quantitative evaluation metrics for FutureGAN are also displayed in the below tables.

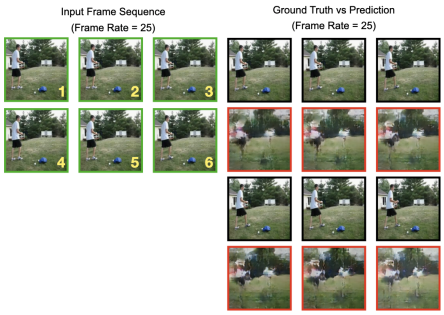

Figure 4: A sample test video prediction from the FutureGAN model with frame rate of 25. The input sequence of 6 frames is highlighted in green on the right, and the predicted 6 frames is shown in red on the left under the corresponding ground truth frame in black.

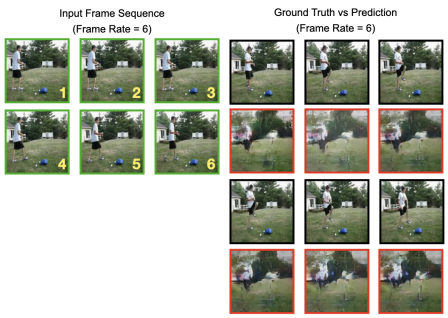

Figure 5: A sample test video prediction from the FutureGAN model with frame rate of 6. The input sequence of 6 frames is highlighted in green on the right, and the predicted 6 frames is shown in red on the left under the corresponding ground truth frame in black.

Figure 6: A sample test video prediction from the FutureGAN model with 64x64 resolution. The input sequence of 6 frames is highlighted in green on the right, and the predicted 6 frames is shown in red on the left under the corresponding ground truth frame in black.

MSE		
Frame	ConvLSTM 64x64	FutureGAN 64x64
1	0.0046	0.0706
2	N/A	0.0795
3	N/A	0.0900
4	N/A	0.0989
5	N/A	0.1062
6	N/A	0.1140
Avg.	0.0046	0.0932

Table 1: Average MSE over each frame for ConvLSTM and FutureGAN evaluated on 25 fps, 64x64 px images

MSE for 128x128 FutureGAN			
Frame	25 fps	12 fps	6 fps
1	0.0841	0.0918	0.1038
2	0.0930	0.1047	0.1192
3	0.1031	0.1177	0.1378
4	0.1119	0.1259	0.1586
5	0.1214	0.1352	0.1710
6	0.1289	0.1435	0.1806
Avg.	0.1071	0.1198	0.1452

Table 2: Average MSE over each frame for FutureGAN evaluated on 128x128 px images with different frame rates

PSNR		
Frame	ConvLSTM 64x64	FutureGAN 64x64
1	25.5266	18.0436
2	N/A	17.5727
3	N/A	17.0052
4	N/A	16.6343
5	N/A	16.3350
6	N/A	16.0261
Avg.	25.8583	16.9361

Table 3: Average PSNR over each frame for ConvLSTM and FutureGAN evaluated on 25 fps, 64x64 px images

PSNR for FutureGAN 128x128			
Frame	25 fps	12 fps	6 fps
1	17.4465	17.0446	16.3874
2	17.0350	16.4775	15.8366
3	16.5750	15.9938	15.2262
4	16.2431	15.6777	14.6320
5	15.8703	15.3676	14.3152
6	15.5948	15.0721	14.1123
Avg.	16.4608	15.9389	15.0849

Table 4: Average PSNR over each frame for FutureGAN evaluated on 128x128 px images with different frame rates

SSIM		
Frame	ConvLSTM 64x64s	FutureGAN 64x64
1	0.8743	0.3348
2	N/A	0.3086
3	N/A	0.2810
4	N/A	0.2599
5	N/A	0.2424
6	N/A	0.2226
Avg.	0.8777	0.2749

Table 5: Average SSIM over each frame for ConvLSTM and FutureGAN evaluated on 25 fps, 64x64 px images

SSIM for FutureGAN 128x128			
Frame	25 fps	12 fps	6 fps
1	0.3011	0.2846	0.2609
2	0.2866	0.2674	0.2431
3	0.2688	0.2507	0.2231
4	0.2553	0.2387	0.2038
5	0.2402	0.2258	0.1915
6	0.2284	0.2138	0.1832
Avg.	0.2634	0.2468	0.2176

Table 6: Average SSIM over each frame for FutureGAN evaluated on 128x128 px images with different frame rates

7 Discussion

To compare ConvLSTM, which receives 25 fps, 64x64 px images, to FutureGAN, we evaluated a testset of 25 fps, 64x64 px images on FutureGAN's intermediate 64x64 model checkpoint. ConvLSTM only predicts one frame, so we look at the results obtained on each model's first predicted frame. ConvLSTM outperforms FutureGAN significantly in all of the quantitative evaluation metrics presented in Tables 1, 3 and 5. Moreover, through visual examination, we see that ConvLSTM's predicted frame highly resembles the ground truth frame. Specifically, it successfully outlines the surrounding background as well as producing a reasonable prediction of the in-motion object, all of which the FutureGAN model fails to do.

This is likely a result of ConvLSTM being trained on the full UCF101 dataset, while FutureGAN was only trained on a subset of four classes, due to limitations in computational resources and time.

We also evaluated the final 128x128 px FutureGAN model on videos with different frame rates. The quantitative per-frame average results in Tables 2, 4 and 6 demonstrate that the prediction accuracy decreases as the frame rate decreases. A lower frame rate means that the time between frames is larger, meaning the sequence captures much more disjoint motion, and thus the similarity between frames is smaller. Intuitively, this implies that as the model predicts further into the future, more inference is required, and as such the distance between the predicted frame and the input frame will increase. As a result, it is expected that the deviation from the ground truth frames increases as well, which is supported by our results. The qualitative results, shown in Figures 4 and 5 further supports this claim: Figure 5, which depicts the frames for 6 fps videos, shows far more error and noise between the predicted and ground truth sequences.

The original FutureGAN paper trained the model on three datasets, MovingMNIST, KTH Action and Cityscapes [14]. The quantitative results we obtained using UCF101 are comparably worse than those from the original paper. This difference can be attributed to a few aspects. First, the datasets used in [14] are significantly larger than the subset we used. For example, MovingMNIST have 13,499 training sequences, and KTH Action have 15,156 training sequences. Furthermore, MovingMNIST and KTH Action, which have comparable level of motion to UCF101, were trained using grayscale images, while the UCF101 data was trained in RGB. Cityscapes was trained in RGB, but has much more subtle video motion. Moreover, both KTH Action and MovingMNIST have static backgrounds that are easily distinguished from the foreground action figure. In summary, using limited training data in RGB with complex motion and dynamic backgrounds contributed to the large errors between our results and [14].

8 Conclusion and Future Work

While both models produced plausible predicted frames given their training data, because ConvLSTM was trained on the full UCF101 dataset, it outperformed FutureGAN quantitatively and qualitatively on the first future frame.

Since video data is very large and the networks used are complex, time and compute resources were our largest obstacle. Even on a small subset of UCF101, training the models took several days and each teammate almost fully used their AWS credits. If these constraints were not imposed, we would explore the following next steps.

For FutureGAN, we would begin by training the network on the full UCF101 dataset with the current hyperparameters and network architecture, since as discussed in the results section, training over the full dataset significantly increased the performance of the Convolutional LSTM model. Other future steps include tuning the loss component weights to ensure that the GAN model is appropriately prioritizing adversarial loss, and performing an ablation study to determine how many intermediate layers are necessary in the progressive growing step.

We would also like to explore architectural changes to the current Convolutional LSTM model. Given more compute resources, we would modify the number of stacked Convolutional LSTM layers to find the optimal architecture for this predictive task. We would also train the model for more epochs.

For both models, we would modify the number of input and predicted frames, in order to improve their ability to perform long-term predictions. Ideally, the networks would be able to accurately predict more frames than the input sequence has. In a similar vein, we would also feed our predicted images back into the network to output even more future frames, to analyze the evolution of prediction.

Furthermore, we would experiment with slower frame rates to evaluate the models' performance on a sequence of images that are increasingly dissimilar. This is of particular importance for ConvLSTM, since we have only trained and evaluated it on videos with a frame rate of 25 fps.

9 Appendices

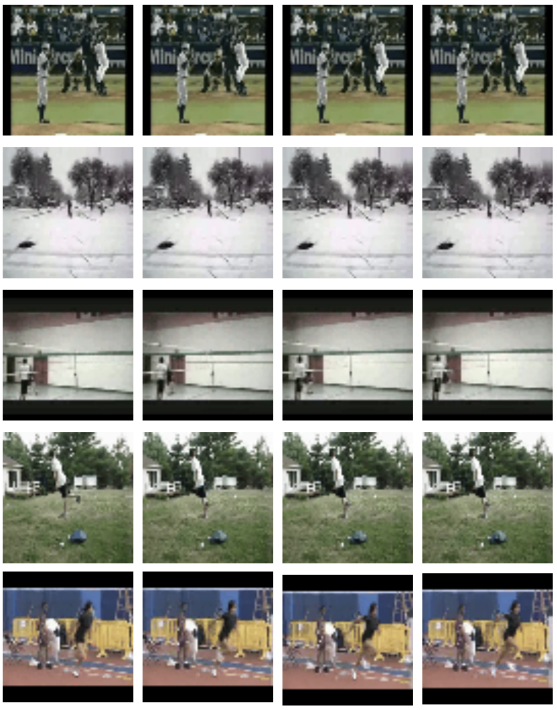

Figure 7: Sample video frames from the UCF101 dataset. The dataset includes videos of various human activities captured at a frame rate of 25 fps.

Figure 8: A sample test video prediction from the FutureGAN model with frame rate of 12. The input sequence of 6 frames is highlighted in green on the right, and the predicted 6 frames is shown in red on the left under the corresponding ground truth frame in black.

10 Contributions and Acknowledgements

Each of the project members contributed to this work equally. All three of us worked together on all data preprocessing, model code adaptation, and training and evaluation. We adapted and trained the models from [21, 22]

References

[1] J. Walker, A. Gupta, and M. Hebert, "Dense optical flow prediction from a static image," 2015. 1

[2] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," 2017. 1

[3] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: a baseline for generative models of natural videos," 2014. 2

[4] M. Oliu, J. Selva, and S. Escalera, "Folded recurrent neural networks for future video prediction," 2017. 2

[5] 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," 2015. 2

[6] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," 2015. 2

[7] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction," 2017. 2

[8] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic variational video prediction," 2017. 2

[9] E. Denton and R. Fergus, "Stochastic video generation with a learned prior," 2018. 2

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. 2

[11] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," 2015. 2, 3

[12] M. Saito, E. Matsumoto, and S. Saito, "Temporal generative adversarial nets with singular value clipping," 2016. 2

[13] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine, "Stochastic adversarial video prediction," 2018. 2

[14] S. Aigner and M. Körner, "Futuregan: Anticipating the future frames of video sequences using spatio-temporal 3d convolutions in progressively growing gans," 2018. 2, 3, 4, 5, 8

[15] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," 2017. 2, 3, 5

[16] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," 2012. 4

[17] "FFmpeg, howpublished = https://ffmpeg.org/, note = Accessed: 2021-05-15." 4

[18] "Peak signal-to-noise ratio, howpublished = https://en.wikipedia.org/wiki/mean_squared_error, note = Accessed: 2021-05-30." 5

[19] "Mean squared error, howpublished = https://en.wikipedia.org/wiki/peak_signal-to-noise_ratio, note = Accessed: 2021-05-30." 5

[20] "Structural similarity, howpublished = https://en.wikipedia.org/wiki/structural_similarity, note = Accessed: 2021-05-30." 5

[21] R. Panda, "Convlstm." https://github.com/sladewinter/ConvLSTM, 2021. 9

[22] L. Liebel and S. Aigner, "Futuregan." https://github.com/TUM-LMF/FutureGAN, 2018. 9

[23] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros, "A review on deep learning techniques for video prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 2806–2826, jun 2022.

[24] "PyTorch, howpublished = https://pytorch.org/, note = Accessed: 2021-05-10."

[25] "NumPy, howpublished = https://numpy.org/, note = Accessed: 2021-05-10."

[26] "Pillow, howpublished = https://pypi.org/project/pillow/, note = Accessed: 2021-05-15."

[27] "SciPy, howpublished = https://scipy.org/, note = Accessed: 2021-05-15."

[28] "scikit-image, howpublished = https://scikit-image.org/, note = Accessed: 2021-05-25."