Hierarchical Model for Long-Term Video Prediction
Reimplementation Project

Zhongxia Yan
zxyan@berkeley.edu

Jeffrey Zhang
jeffzhang1996@berkeley.edu

Abstract

Video prediction has been an active topic of research in the past few years. Many algorithms focus on pixel-level predictions, which generates results that blur and disintegrate within a few frames. In this project, we use a hierarchical approach for long-term video prediction. We aim at estimating high-level structure in the input frame first, then predict how that structure grows in the future. Finally, we use an image analogy network to recover a realistic image from the predicted structure. Our method is largely adopted from the work by Villegas et al.[10], using a combination of LSTMs and convolutional auto-encoder networks. Additionally, in order to generate more realistic frame predictions, we also adopt adversarial loss. We evaluate our method on the Penn Action dataset, and demonstrate good results on high-level long-term structure prediction.

1. Introduction

Learning to predict the future is an important research topic in computer vision and artificial intelligence.[2] Humans make predictions and inferences about the real-world all the time. These predictions focus on the semantics of the action (i.e. where a ball will land, expected overall shape of a figure, etc.), rather than the exact location or pixel intensity. In addition, the rough semantics generally provides more value to the prediction than raw pixel values would anyways.

To better reflect how humans make predictions, we seek to develop a hierarchical approach for video prediction that first predicts some higher-level features of future frames, then regenerate realistic video from the predicted higher-level features. A recent work by Villegas et al.[10] predicts future poses of humans and then uses image analogies to map the high frequency details of input images to the predicted poses. We decided to first adopt and implement the methods within this paper, then explore potential improvements. Villegas does not provide any code or hyperparameter, so we've implemented the entire model as described in their paper and experimented with hyperparameters and improvements ourselves.

We evaluate our method on the Penn Action dataset [12], and demonstrate good results on high-level long-term structure prediction.

The rest of the paper is organized as follows: A review of the related work is presented in Section 2. The LSTM and deep convolutional models that we use are presented in Section 3. The experimental results and analysis are shown in Section 4 and 5. Finally we conclude our work with discussion of future work in Section 7.

2. Related Work

There has been numerous recent work on pixel-wise video prediction. Mathieu et al. (2015) [5] uses a multi-scale convolutional generative adversarial network (GAN) approach to recursively generate the next frame given the past few frames. Such pixel-based generations are usually reasonable for the first few predicted frames, but the quality degrades quickly. Vondrick et al. (2016) [11] separately generates a moving foreground representation of a video and a static background and combines the two. Similar approaches include [7], [4] and [9]. Their results, while not photo-realistic, captures the general motion of the foreground object rather well.

The approaches mentioned above use pixel-wise prediction on video frames, but we hope that a hierarchical approach that breaks down prediction of the semantics and predictions of the pixels would produce consistent video for a longer duration. Our approach obtains the current pose of the human in the video, predict future poses, and predict realistic video corresponding to the future poses. As a follow up in the future, we will explore replacing pose estimations with motion segmentation, which could generalize better to both humans and to nonhuman objects.

3. Approach

3.1. Overview

Our model is a three step process: 1) pose estimation, 2) pose prediction, and 3) image analogy generation. First, given the input frame x_t, our model first get the corre-

sponding pose heatmap p_t via pose estimation. Second, our pose prediction network predicts the pose heatmap for future frames. Finally, our image analogy network recovers a realistic image from the corresponding pose heatmap and input frame.

3.2. Pose Estimation

We represent each pose as a list of joint locations. There is work by Zhe Cao et al. [1] and Newell et al. [6] as well as other groups to extract joint locations for different poses. For the purposes of focusing on pose prediction and image analogy, we use the poses annotated by the Penn Action Dataset for our pose predictions.

3.3. Pose Prediction

We create a sequence-to-sequence LSTM network, illustrated in Fig. 1, to predict the future joint locations given a sequence of input joint locations.

$$(h_t, c_t) = LSTM(p_t, h_{t-1}, c_{t-1}),\ 1 \leq t \leq k \quad (1)$$

where $p_t \in \mathbb{R}^{2L}$ is a vector containing the coordinates of all the joint locations at time t, c_t is the LSTM memory cell vector at time t, and h_t is the latent output of the LSTM at time t. Initially, we feed k frames of pose estimation (p_1 to p_k) into the LSTM to give the LSTM context on the video to be predicted.

We generate the future latent outputs h_t for T frames after the k frames of initial input by feeding in $\mathbf{0}$ as the input into the LSTM at each time step.

$$(h_t, c_t) = LSTM(\mathbf{0}, h_{t-1}, c_{t-1}),\ k+1 \leq t \leq k+T \quad (2)$$

We then predict the pose in these T frames by learning a two-layer neural network mapping from the latent output of the LSTM to pose joint locations.

$$\hat{p_t} = \sigma(W_2\sigma(W_1 h_t + b_1) + b_2) \quad (3)$$

where σ is the sigmoid activation function. Notice that we normalized the x and y coordinates in p to be between 0 and 1. Also, we do not feed in the predicted poses $\hat{p}_t$ as input into the LSTM, so that the errors in the predictions does not propagate further into future frames. Instead, this encourages the LSTM model to predict all the future frames only given the original frames.

Our loss function for training the LSTM aims to enforce accuracy of the predictions equally for all T future time points.

$$\mathcal{L}_{LSTM} = \sum_{t=k+1}^{k+T} \|\hat{p}_t - p_t\|^2 \quad (4)$$

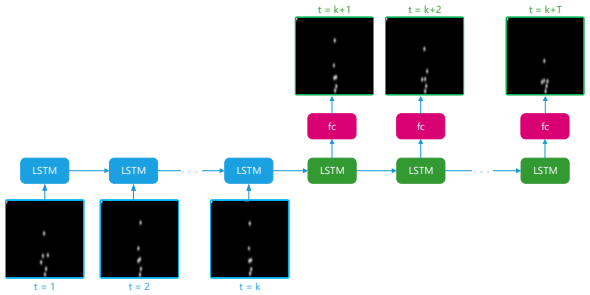


Figure 1. Sequence-to-sequence LSTM model for predicting a sequence of T future poses from the given k poses.

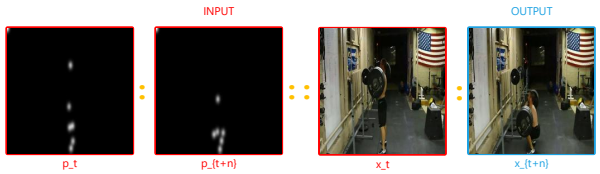


Figure 2. Synthesizing future frame by making analogies between pose structure and image features.

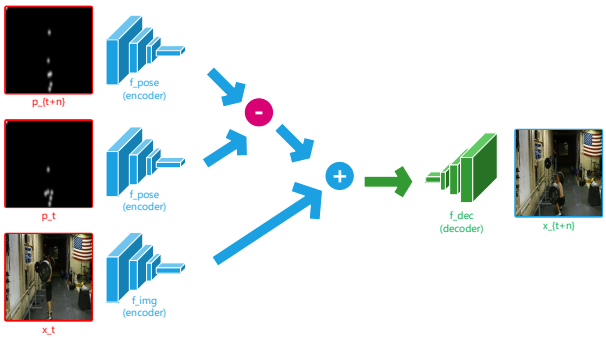


Figure 3. Illustration of our image generator. Our image generator observes an input image, its corresponding human pose, and the human pose of the future image. Through image analogy, our network generates the next frame.

3.4. Image Analogy

We use Image Analogy network [8] to synthesize the future frame from its pose structure. As is illustrated in Fig. 2, the relationship between p_t and p_{t+n} is the same as the relationship between x_t and x_{t+n}. More specifically, the future frame x_{t+n} can be generated by transferring the structure transformation from p_t to p_{t+n} to the observed frame x_t. To promote more realistic generations, we adopt a generative adversarial network [3] for our image analogy network.

3.4.1 Generator

As is shown in Fig. 3, our image generator uses a pose encoder f_{pose}, an image encoder f_{img} and an image decoder f_{dec}. Specifically, f_{pose} is a convolutional encoder that specializes on identifying key pose features from the pose input that reflects high-level human structure. f_{img} is also a convolutional encoder that acts on an image input by mapping the observed appearance into a feature space where the pose feature transformations can be easily imposed to synthesize the future frame using the deconvolutional decoder f_{dec}. The image analogy is then performed by

$$\hat{x}_{t+n} = f_{dec}(f_{pose}(g(\hat{p}_{t+n})) - f_{pose}(g(p_t)) + f_{img}(x_t))$$

where $\hat{x}_{t+n}$ and $\hat{p}_{t+n}$ are the generated image and corresponding predicted pose at time $t+n$, x_t and p_t are the input image and corresponding estimated pose at time t, $g(:)$ is a function that maps a set of $(x,\ y)$ joint coordinates to a stack of L heatmaps, and n is the difference in time between the frame to be generated and the input frame.

For the network architecture, our f_{img} encoder is a VGG-16, f_{pose} is a simplified VGG with many layers removed, while f_{dec} is a reversed VGG-16 where all convolutions are replaced by deconvolutions. Our reasoning for removing some VGG-16 hidden layers in f_{pose} is that the pose heatmaps are relatively simple and do not need as much representative power (and also our GPUs are dying).

The loss of generator is written as follows:

$$\mathcal{L}_{generator} = \lambda_{img}\mathcal{L}_{img} + \lambda_{adv}\mathcal{L}_{adv}$$

where $\mathcal{L}_{img} = \|x_{t+n} - \hat{x}_{t+n}\|_2^2$ is the ℓ-2 distance between ground-truth and predicted in image space and $\mathcal{L}_{adv} = -\log D([p_{t+n}, \hat{x}_{t+n}])$ is the adversarial loss. Finally, λ_{img} and λ_{adv} are hyperparameters denoting how much each loss function will contribute to our overall loss function.

3.4.2 Discriminator

We input pose structure and video pairs to the discriminator, which outputs 1 if pose and video are real and from the same time slice and outputs 0 otherwise. The network architecture of the discriminator is also VGG-16. Our discriminator loss is defined as follows:

$$\begin{aligned}\mathcal{L}_{disc} = &-\log D([p_{t+n}, x_{t+n}]) \\ &- 0.5\log(1 - D([p_{t+n}, \hat{x}_{t+n}])) \\ &- 0.5\log(1 - D([p_{t+n}, x_t]))\end{aligned}$$

Note that the first term incentivizes the discriminator to predict "real" given a real image and pose pair, the second term incentivizes the discriminator the predict "fake" given a real pose and fake image, and the third term incentivizes the discriminator to predict "fake" given an image at time t and a pose at a different time (so the discriminator can become sensitive to pose differences).

Table 1. Information of Penn Action Dataset

Actions		Annotated Joints	
1	baseball pitch	1	head
2	clean and jerk	2	left shoulder
3	pull ups	3	right shoulder
4	strumming guitar	4	left elbow
5	baseball swing	5	right elbow
6	golf swing	6	left wrist
7	push up	7	right wrist
8	tennis forehand	8	left hip
9	bunch press	9	right hip
10	jumping jacks	10	left knee
11	sit ups	11	right knee
12	tennis serve	12	left ankle
13	bowling	13	right ankle
14	jump rope		
15	squats		

4. Implementation Details

4.1. Dataset

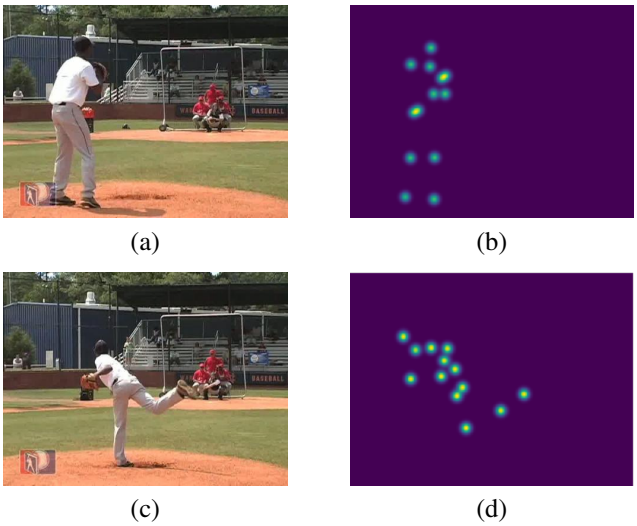
We use the Penn Action Dataset [12] for training and testing. The dataset consists of 2326 videos of human actions. The 15 actions in the dataset are baseball pitches, baseball swings, bench press, and some others. Each image has a 13 annotated joint locations stored as (x, y) coordinates denoting joint coordinate position in image. The joints are head, shoulders, elbows, wrists, hips, knees, and ankles. Most videos are 480x270 and consist around 50-150 frames. The details of the dataset is shown in Table 1.

4.2. Pose Estimation

Normally this would be generated from the input video frames with a pose estimation network. To focus our efforts on pose and video prediction, we used the existing annotated joint locations from the Penn Action Dataset. We create a corresponding heatmap with a Gaussian distribution with $\sigma = 5$ centered at the coordinate of each joint (illustrated in Fig. 4). This heatmap will represent our pose estimation input into the image analogy network.

4.3. Pose Prediction

We apply a 2-layer neural network function to the output of the LSTM architecture described above. The network consists of a 1024-node hidden layer followed by a 100-node hidden layer, both with sigmoid activation function. We use $k = 15$ input images to encode our LSTM network and decode $T = 45$ prediction images. We trained the network on 42 squats video and ran tests on 36 videos. Unless otherwise mentioned, our input image and poses are cropped versions of the original video with the human centered.



(a) (b) (c) (d)

Figure 4. Creating heatmaps from annotated labels from Penn Dataset

The choice of hyperparameter does not affect the quality of output noticeably, and we were able to get good prediction of human poses with 4000 training iterations with a batch size of 1 and a learning rate of 5e-4.

4.4. Image Analogy

For the image analogy network, we trained the network on videos of all actions (unless otherwise stated). We resize all videos to 224x224 and don't convert to grayscale unless otherwise mentioned. Our network parameters are $\lambda_{img} = 1e-2$ and $\lambda_{adv} = 1$. We define each training step to be 4 image analogy pairs and train on 200000 steps. The learning rate for the generator network is 1e-4 and the learning rate for the discriminator is 5e-5. We used ADAM optimizer for both the generator and the discriminator.

We did the majority of the training on several GTX 1080 Ti, and the 200000 steps took around 48 hours to train. We used a stochastic gradient descent optimizer with a mini-batch size of 4. Because the network is so large (several VGGs), 4 was the maximum mini-batch size we could work with.

4.5. Framework

We used Tensorflow and Numpy for all the neural network implementation and data processing. We managed to abstract Tensorflow into higher level helper functions to help organize and had no trouble with Tensorflow. We wrote our own infrastructure to easily create experiments, save outputs, and visualize results from each individual experiments. Our implementation and results can be found on Github.

5. Replicated Results

5.1. Pose Prediction

Our results show that our implementation of the LSTM network is able to capture the movement of squats accurately. There is high variation in the form and speed of how people do squats. The LSTM model is able to capture an "average" prediction model based on the trained videos. Thus, though the predicted joint locations show a convincing alternative to the ground truth. We show results of pose prediction on squat videos in Fig. 5 and 6. The squat motions are very similar, albeit at different paces.

Our initial test with a simple sequence-to-sequence network with 128 hidden units and no fully connect output layer did not work well in predicting poses for test videos. The simple sequence-to-sequence network was able to overfit to training data, but was unable to make meaningful predictions for test images. By increasing the number of hidden units in the LSTM block and adding two fully connected layers to the output of the LSTM, we were able to create a working prediction model.

5.2. Image Analogy

From the results in Fig. 7, we can see that the outputs of the analogy network captures the semantics of the human figure well. The generation is able to generate a human figure that fits the locations of the joints at time $t + n$, with an almost perfectly unaltered background. Both the figure and background are not blurry. However, we found that the discriminator is not able to train the generator to perfectly reconstruct facial and other fine details. In addition, the discriminator not able to constrain the true human figures in time t and the generated figure in time $t + n$ to look the same, i.e. often human figures in time $t + n$ wears different colored clothes compared to the true human figure in time t. Additionally, instruments and tools used by humans (e.g., weights, bats, rackets, etc.) had difficulty transferring over into the generated frames. Finally, although the generator is able to erase most of the figure from its location at time t, there is sometimes a shadow of leftover human figure, and interpolation at the location of the human at time t sees limited success. In summary, the analogy network captures the semantics of the human figure well, but sometimes fails at generating fine details realistically.

6. Result Extensions

Since our LSTM worked satisfactorily, we mostly explored extension to improve the image analogy network. Some samples of our best results are show in Fig. 7 Visit Github for more comparison results and comparisons.

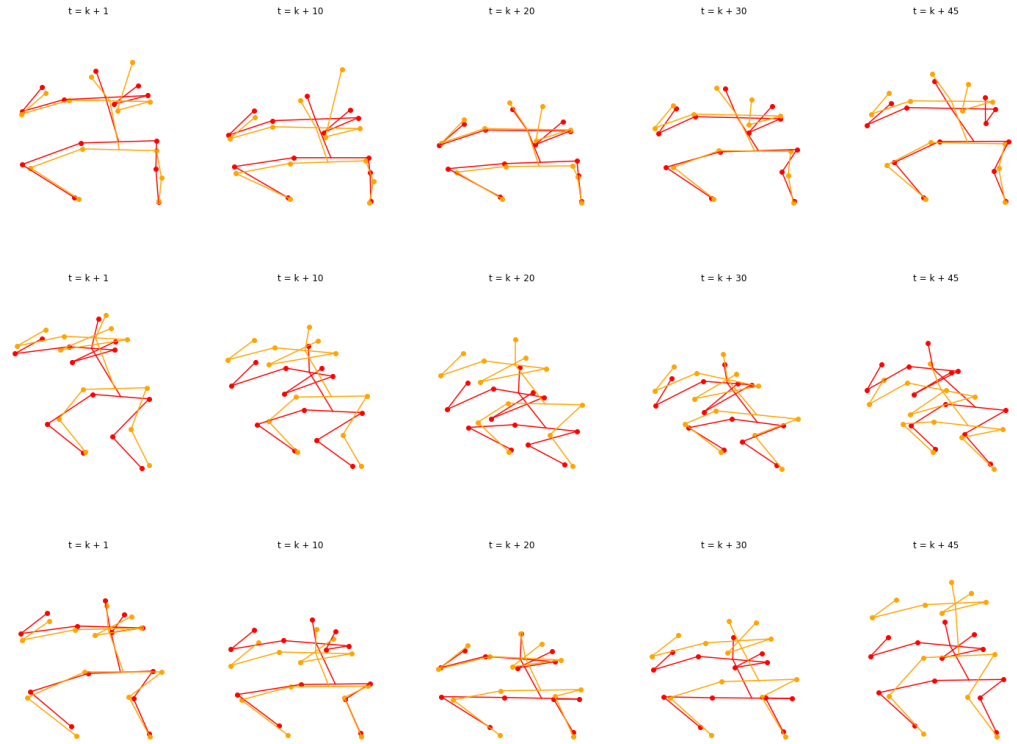


Figure 5. LSTM prediction on fully unoccluded set of joints. In our model k = 15 is the number of input frames. The predictions (**red**) do not fully match the ground truth (**orange**) but captures the motion quite well.

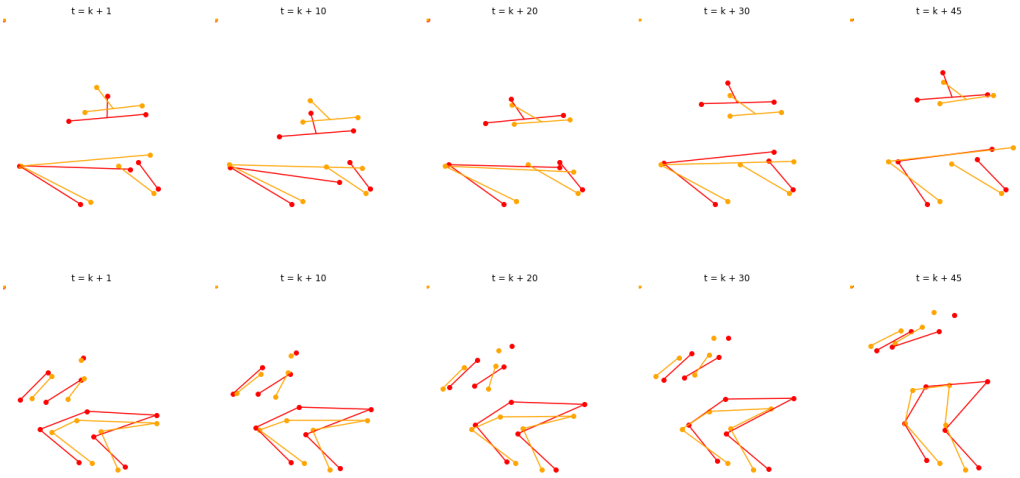


Figure 6. LSTM prediction on cases where some joints are occluded, in which case (x, y) = (0, 0) for those joints. The rest of the joints are predicted well.

6.1. Comparison with Grayscale

We originally trained the analogy network to predict on grayscale video frames, in hopes that this would be an easier task for the generator. We saw no significant improvement over RGB frames. We show this result in Fig. 8

6.2. Comparison with Full Image

Because background images in a video frame tend to be static, we wanted to focus the attention of the network on the human in video. As a result, we ran experiments on feeding in a cropped bounding box around the human and

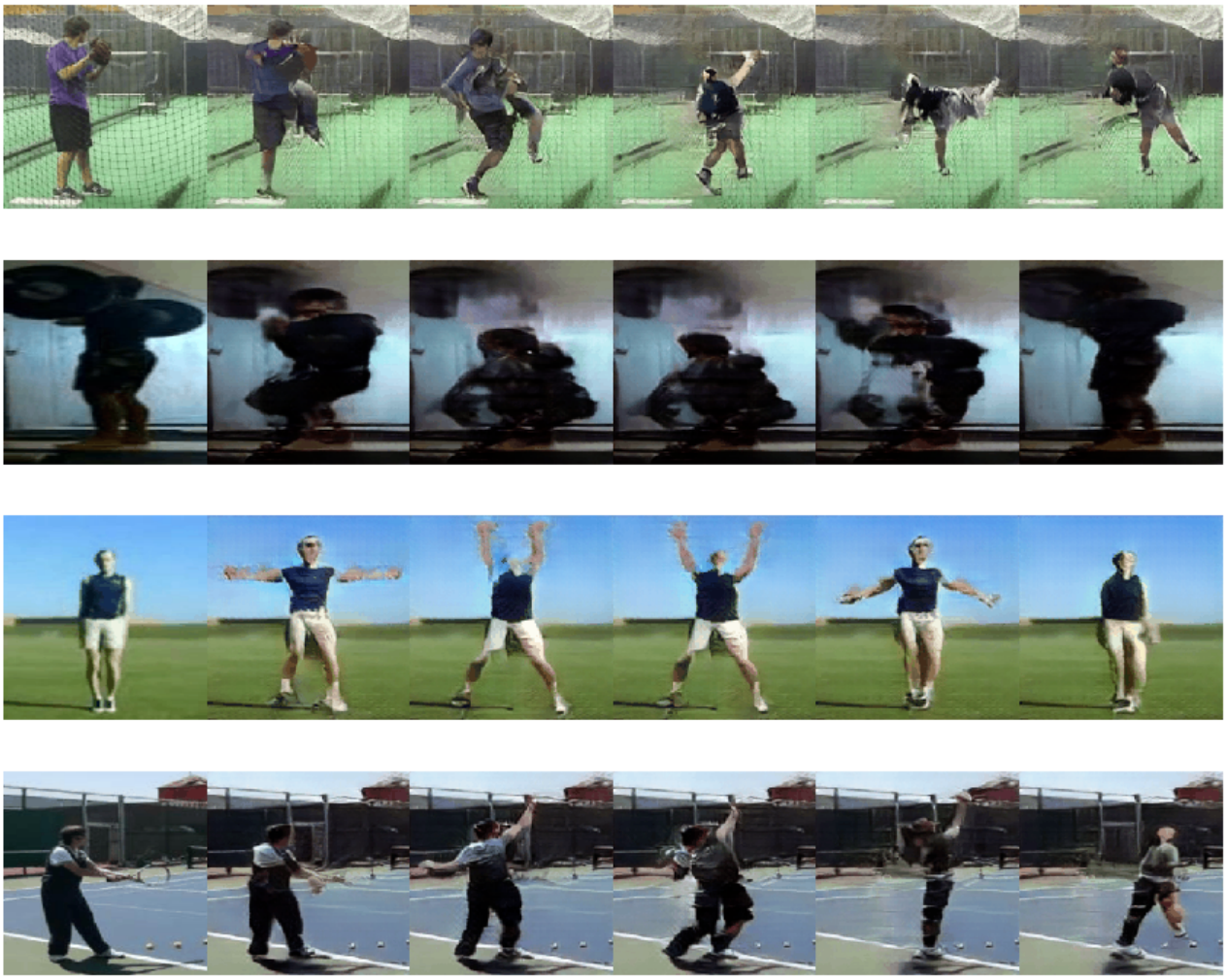


Figure 7. Sample output frames of baseball pitch, squat, jumping jack, and tennis serve actions.

Figure 8. Grayscale prediction of baseball pitch.

feeding in a full image frame into our models. The boundaries for the cropped images used the provided bounding box annotations provided in the Penn Action Dataset.

Our experiments reveal two interesting findings between full image frame prediction and cropped image frame prediction. First, cropped frame prediction produces more detailed results. This is expected as the network now focuses more of it's attention to prediction body parts of the human, rather than learning to transfer background pixels. Second, full image frame prediction captures the pose of the images better. In Fig.9, we can see that a full image frame prediction was able to accurately associate the weights of the squats with the pose compare to the same squat image in Fig.7. This is likely due to the receptive field of the convo-

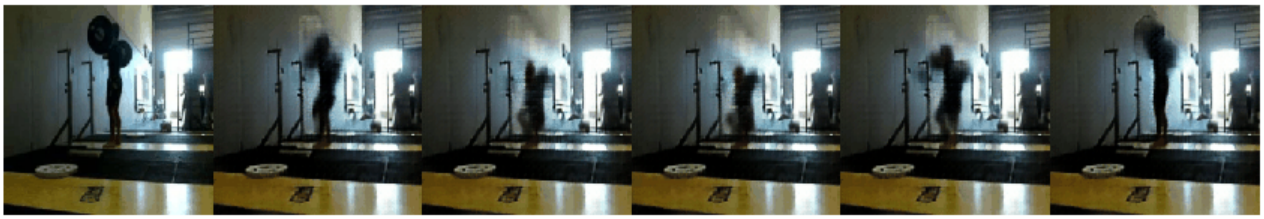


Figure 9. Prediction on the original video frames (uncropped)

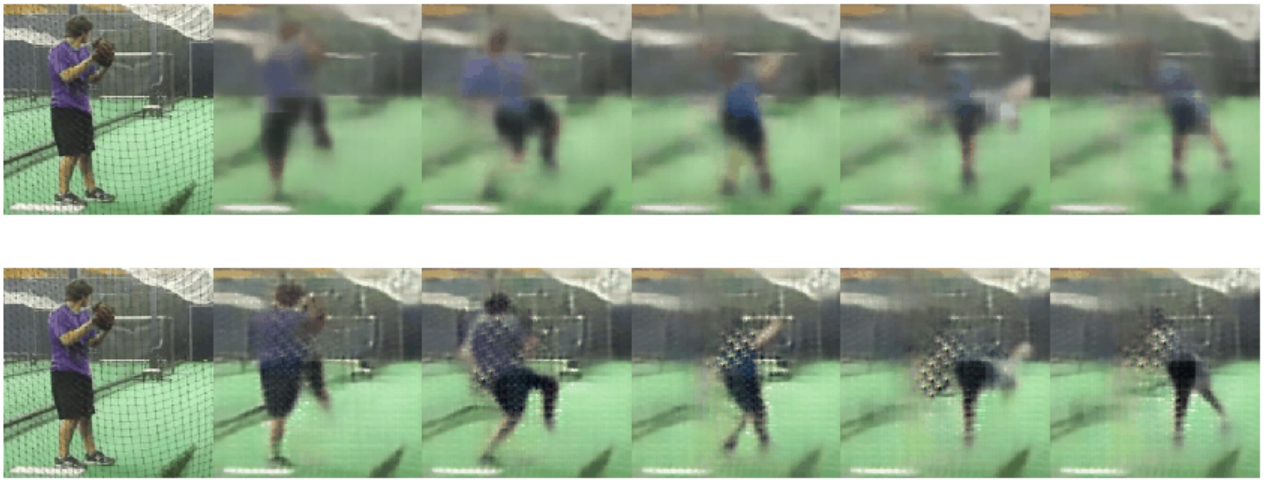


Figure 10. Top: Prediction with ℓ-2 loss only (no adversarial loss). Bottom: Prediction with a weak discriminator (trained with GradientDescentOptimizer instead of AdamOptimizer)

lutions being large enough to better capture features of the pose as the human in the video occupies less space in the video.

6.3. Comparison with Weaker Discriminator

Originally, we trained the discriminator with GradientDescentOptimizer instead of AdamOptimizer. This resulted in a weak discriminator that fails to produce sharp images. As a baseline ℓ-2 loss comparison, we trained and obtained results from the analogy network without the discriminator. We show these results in Fig. 10. One can see that although the discriminator is weak when trained with GradientDescentOptimizer, it still is able to constrain the generator to sharper losses than ℓ-2 loss alone is.

6.4. Comparison with Convolution Depths

To explore the effect of the depth of neural networks for image prediction, we ran experiments on varying depths for our networks used in the image analogy network. The original implementation adopted the VGG-16 architecture up to the pool3 layer. We experimented adopting VGG-16 architecture at pool2, pool3, pool4, and pool5. We show some comparison results in Fig. 11. Note that all experimentation was done using a weaker discriminator (as explained in Results section). Notice that as we use deeper networks, the predicted pose is more accurately represented, at the expense of a blurrier image. Pool 4 and pool 5 shows significant reduction in the ghosting affect (remnant features from the reference image) that occurs in pool 2 and pool 3.

6.5. Upsampling

Originally, we trained the discriminator with GradientDescentOptimizer instead of AdamOptimizer. This resulted in a weak discriminator that fails to produce sharp images. As a baseline ℓ-2 loss comparison, we trained and obtained results from the analogy network without the discriminator. We show these results in Fig. 10

7. Conclusion and Future Work

In this paper, we present a hierarchical model of pixel-level video prediction. Using human action video dataset as benchmark, we demonstrate our hierarchical prediction approach is able to predict pose heatmaps and effectively generate a future video frame based on the predicted pose heatmap.

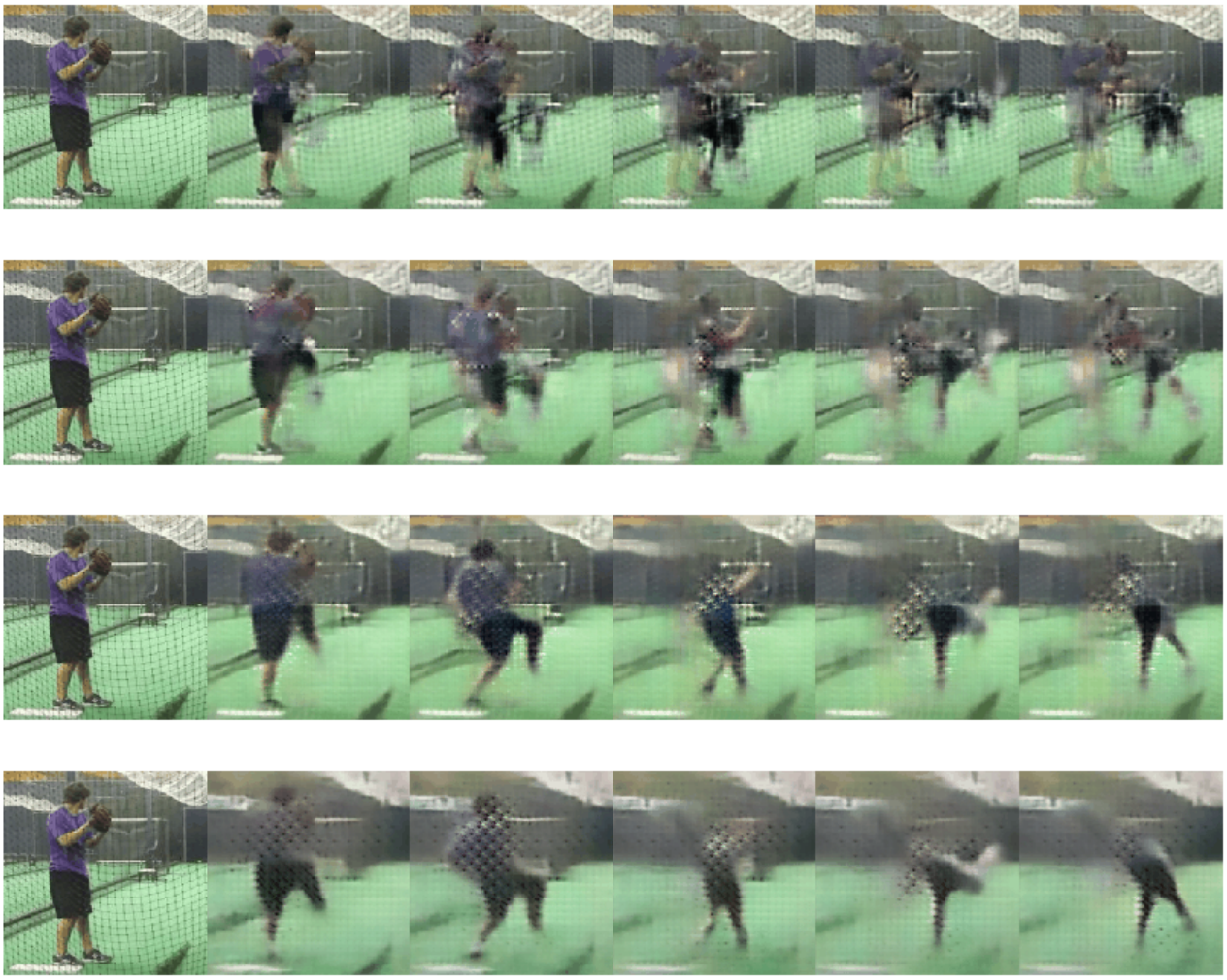


Figure 11. From top to bottom: Generations from a generator with 2-, 3-, 4-, and 5-stages of VGG convolutions (note that each stage consists of 2 or 3 sequential convolutions). All comparisons in this figure are generated with the weak discriminator (trained with GradientDescentOptimizer instead of AdamOptimizer).

The success of our work is that the model can predict long-term pose structure from video frames and supports the success of hierarchical model for video frame prediction. We have provided a working code reimplementation with an easy framework for training, testing, and experimentation. The code is on Github at the following link.

For future work and extension to the project, we can introduce another discriminator to generate images that accurately transfer features (e.g. clothes' colors, textures, etc.) from the given reference image to the predicted frame. This discriminator would be able to ensure that the human figure in the generate image matches the human figure in the input image, while ignoring the background (since the background is the same in both images). Additionally, we can experiment training stronger associations between the pose and the instruments, equipments, and tools used by the humans in each action. We can continue experimenting with varying depths and tuning discriminator parameters to better fit each depth for more comprehensive experimentation. Finally, we hope to generalize pose prediction to segmentation predictions to work for arbitrary objects (and not just human poses). Segmentations are easier to generate, potentially easier to label, and more generalizable than pose predictions.

References

[1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050*, 2016.

[2] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances*

in Neural Information Processing Systems, pages 658–666, 2016.

[3] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[4] R. Goroshin, M. F. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, pages 1234–1242, 2015.

[5] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[6] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[7] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

[8] S. E. Reed, Y. Zhang, Y. Zhang, and H. Lee. Deep visual analogy-making. In *Advances in Neural Information Processing Systems*, pages 1252–1260, 2015.

[9] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[10] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *arXiv preprint arXiv:1704.05831*, 2017.

[11] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pages 613–621, 2016.

[12] W. Zhang, M. Zhu, and K. G. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2248–2255, 2013.