(Re)live Photos: Generating Videos with GANs

Michael Chen
mvc@cs.stanford.edu

Sang-Goo Kang
sanggook@stanford.edu

Sam Kim
samhykim@stanford.edu

Abstract

Along with the iPhone 6s, Apple announced a new feature, dubbed 'Live Photos': along with every picture, the iPhone would take a short video in the surrounding time frame, which when combined with the photo, could make photos appear to 'come to life'. Our goal is to also bring photos to life without requiring the original video data. We implement several architectures and analyze their tradeoffs in producing these videos. We first highlight WGAN, which improves stability when training GANs, and describe our various models using WGANs. We then delve into our LSTM network for future frame prediction. Although there is no standard metric to measure video generation quantitatively, our qualitative measurements show our models are able to produce some realistic videos.

1. Introduction

With the abundance of unlabeled video data available today and even more photos, we plan to develop a system to learn to bring photos to life by leveraging this data. Still photos can provide a substantial amount of information to a human eye, and one could often imagine the sequence of events surrounding the photo. However, computers still struggle with this level of 'imagination'. Given a photo, there are unlimited possibilities of how the dynamics and motion of the scene can change. In addition, video data is often noisy or unstable, making it more difficult to track the object in motion.

The goal of this project is to explore different methods of generating videos given photos. Our primary approach involves a Generative Adversarial Network trained to imagine the dynamics surrounding a photo.

2. Related Work

Using GANs for video generation was pioneered by Vondrick *et al*. [24]. In their paper, they propose a generative adversarial network [7] using 3D convolutional neural networks to first predict scene's foreground and background on different streams. Another network discriminates these fake outputs from real videos, and passes the gradients back to the generator network. A third network acts as an encoder for the generator, which uses 2D convolutions to map a photo to an interpretable space by the generator. The idea behind generative adversarial networks is to train two networks: the generator, which produces the videos, and the discriminator, which detects whether the video is 'fake' or 'real'. These two networks work against each other in a minimax adversarial fashion. This network is trained from Flickr videos, where the generator network would learn to predict future frames of a video based on a single early frame in the video.

Much of the previous research with generative adversarial networks has been been for image modeling. Radford *et al*. [18] use generative adversarial nets with deep convolutional neural networks for the traditional task of unsupervised representational learning of images. However, there has been little work that has been done on sequences of images or videos. Most notably, in 2016, Mathieu *et al*. [13] also use adversarial networks to predict future images from a video sequence. However, they measured the accuracy for only a few predicted frames due to limitations in extrapolating further.

Since then, much work has been done to improve the stability of training GANs such as WGAN [2], CycleGAN, [30], SeqGAN [28], that we leverage. Most of these improvements tackle the generator differentiation problem that occurs in traditional GANs.

We also leverage the work of Mathieu *et al*. to extend video generation to longer sequences. In addition, we will built on top of Vondrick *et al*. in using a conditional GAN to generate videos from single static images (without having to input the entire video sequence).

Xue *et al*. [27] also develop a system to synthesize future frames from a single image. They accomplish this by combining frames with motion encoders, which a decoder then combines to create differences, which in turn produces new frames.

Walker *et al*. [25] approach future prediction by predicting the optical flow using a deep convolutional architecture and conclude that their networks vastly outperform classical methods for optical flow prediction. However, they noted that their predictions would often only remain accurate for

a few time-steps.

3. Problem Statement

Our current setup involves starting with 64x64 photos, from which we predict either the next 31 frames, or 15 frames before the video and 16 frames after (for a total of 32 frames, or just over 1 second of video). The difficulty lies in producing videos that look realistic and 'alive'. A naive approach would be just repeating the input image 32 times, which would be a valid (static) video, but would not appear alive to humans. On the opposite end of the spectrum, we could alternate between the picture and black frames, but this would not represent a realistic video.

We first implement a simple baseline that uses a CNN encoder-decoder architecture to predict the next frame by minimizing the error between the reconstructed frame and the next frame and concatenate the frames together.

We later experiment multiple generator architectures that accept single frames to learn to generate these videos directly, and a discriminator to tell which videos are real and which are fake.

4. Methods

We implemented several methods for video generation and highlight each architecture.

We begin our discussion by describing the Wasserstein GAN, an improvement of traditional GANs. We then describe our various generator models as well as our conditional model trained under this WGAN setup. Finally, we describe our a new model that uses recurrent neural networks, which should help encode information of motion dynamics between frames. Here, the network has to learn how to predict the next frame instead of having to generate a whole video all at once.

4.1. Wasserstein GAN

Following Vondrick *et al*., we use fractionally strided convolutions for our generator and regular strided convolutions for the discriminator. However, unlike Vondrick, which uses a normal DCGAN [18] other than replacing the 2D convolutions with 3D convolutions, we explore the use of the Wasserstein GAN. Typically, GANs are models that try to learn the distribution of real data by minimizing f-divergences. Goodfellow *et al*. [7], the original developers of the generative adversarial network, they optimize the Jensen Shannon divergence. However, as Arjovsky *et al*. [2] suggests, a major shortcoming of GANs is that training is very delicate and highly unstable due to the distribution the GAN tries to model not being continuous.

Because WGAN has been shown empirically to converge in a more stable manner and produce more realistic outputs, we use WGAN for our main architecture. WGAN changes the discriminator to maximize:

$$\max_{w \in W} \left[\mathbb{E}_{x \sim \mathbb{P}}[D_w(x)] - \mathbb{E}_{z \in p(z)}[D_w(g_\theta(z))] \right] \quad (1)$$

The generator loss is updated from the log probability as used in DCGAN to directly minimize:

$$\min_{\theta} -\mathbb{E}_{z \sim p(z)}[D_w(g_\theta(z))] \quad (2)$$

where z is the latent code input to the generator and x is the real video from data.

As a further improvement, we impose a gradient penalty $\lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]$ on the discriminator loss as described in Gulrajani *et al*. [8] to reduce overfitting and help prevent exploding gradients. This avoids having to clip our gradients while training. We sample $\hat{x}$ by sampling from the training data and the corresponding generated video and taking a weighted sum.

4.2. Generator Network

The generator network is responsible for mapping a latent code to a full generated video. The latent code size is a 100-dimensional vector, sampled from a Gaussian distribution. We use a fully connected layer and reshape the latent code to a 2x4x4x512 vector, whose dimensions work well to produce a 32x64x64x3 video. Because we want the video to be invariant to translations with respect to space and time, we feed it through a series of deconvolution operations. Following Vondrick *et al*., we explore the one-stream and two-stream models. After each layer, we apply batch normalization to the activations and ReLU except the last layer. A tanh is applied to output values between [-1,1] at the final layer.

4.2.1 Single-Stream

In the single-stream architecture, we use five three-dimensional deconvolutions to upsample to a 32 frame video. Each layer uses a 4x4x4 kernel with a stride of 2. Because the input vector has a size of 2, the first layer uses a 2x4x4 kernel. We use this as a baseline for our GAN implementation. One big disadvantage with the single-stream model is that it directly models the whole world, assuming nothing is stationary. We mitigate this issue with the model below.

4.2.2 Two-Stream

We make a fundamental assumption that in most cases, video involves a moving foreground combined with a stationary background. With the help from Jun-Young, [1], we

[1] Jun-Young Gwak, our TA for CS231A, provided us with some advice on how to put together our GAN

follow Vondrick *et al*. in developing a two-stream architecture for our GAN by using a series of 2D convolutions to build the background and 3D convolutions to build a foreground 1. The last step of the foreground stream also outputs a mask, which contains a 3D volume of values between 0 and 1 (by using a sigmoid function) that tell the network how to combine the foreground and background at every time-step. The generator combines both streams as a weighted sum of the foreground, background, and mask network:

$$G(z) = m(z) \odot f(z) + (1 - m(z)) \odot b(z) \qquad (3)$$

4.3. Discriminator Network

The job of the discriminator is to identify whether a video is real or fake; therefore, the output of the discriminator is a single value indicating the discriminator's confidence on whether the video is real. Our discriminator accepts 32 frames of 64x64 and uses a series of 5 transposed 3D convolutions, reducing all dimensions by a factor of two at every step except the last layer, which reduces the dimensions down to 1 value. Because the gradients of the rest of the network must flow through the discriminator, we use Leaky ReLUs as the activations in the discriminator (so we don't end up with many 0 gradients early on in backpropagation).

4.4. Conditional WGAN

By sampling a new latent code vector and running it forward through the generator network, we can hallucinate videos based on the videos it has seen while training. To condition the network to predict the surrounding frames given a single photo, we add an additional encoder network to reduce the photo to a 2x4x4x512 vector, the same size as the input to the generator. This allows us to keep the original two-stream model's weights and continue training the network end-to-end. We use the 16^{th} frame as the selected photo when training. To condition the discriminator, the generated video and the photo were concatenated along the first dimension as input [1]. In addition, we take the mean squared error between the middle frame of the generated video and the selected photo and add it to the loss to drive the generator and encoder to map photos to related videos.

4.4.1 InceptionV4 Conditional GAN

Following the methodology of Doersch *et al*. [3], we initialize many of the earlier layers in our generator with learned weights from InceptionV4 (the original paper used AlexNet). Generally, the number of parameters present in the network should be roughly proportional to the amount of data present during training. One problem with 3D convolutional networks is the size of the weights due to the curse of dimensionality.

We mitigate this problem by integrating InceptionV4 as the encoder for our network [23], except we start at the size 8 stage (from the size of the Inception output). In the discriminator, we concatenate the conditional input from InceptionV4 at the 8x8 step along the channels dimension, following an architecture similar to [6]. This way, the generator is required to generate videos that a realistic conditioned on the provided photo, and not just generate any realistic video.

The intuition behind this architecture is that networks trained on ImageNet often generalize well to other tasks; in this case, the Inception network preserves spatial information along with the corresponding information about classes of objects, which would help the generator distinguish between objects in various parts of the frame and animate them differently.

4.4.2 Semantic Segmentation

Vondrick *et al*. train for 3 days on a distributed cluster sharing 9TB of data over NFS; because we did not have the same level of computational resources at our disposal, we leverage transfer learning to assist our network. The input to the encoder includes both the photo as well as a semantic segmentation (class of every pixel in the image, such as 'person' or 'background'). We obtain the semantic segmentation by running the image through a VGG16 network [21] and then passing this through a fully convolutional network designed for semantic segmentation [19]. We use a network with size 8 strides and output of 23 classes, including background, person, and vehicle. The goal here is to provide the network with information about what various objects in the image already look like to save number of training examples the network needs to reproduce this.

4.5. Semantic Segmentation Prediction

We further extend this work in leveraging semantic segmentation by developing a network that works solely off of these semantic segmentations; this way, the network doesn't need to have complete understanding of underlying photos; rather, the network directly learns how to move segmented masks through time, thus generalizing the training between videos with objects of the same class with similar shape. Once again, we send photos through VGG16 and FCN 8s, but only store the output semantic segmentation. We also feed frames of videos through VGG16 and FCN 8s and pass the outputs to the discriminator. Essentially, we replace the video and photo in the GAN architecture above with their corresponding semantic segmentations.

After predicting a segmentation, we inpaint the back-

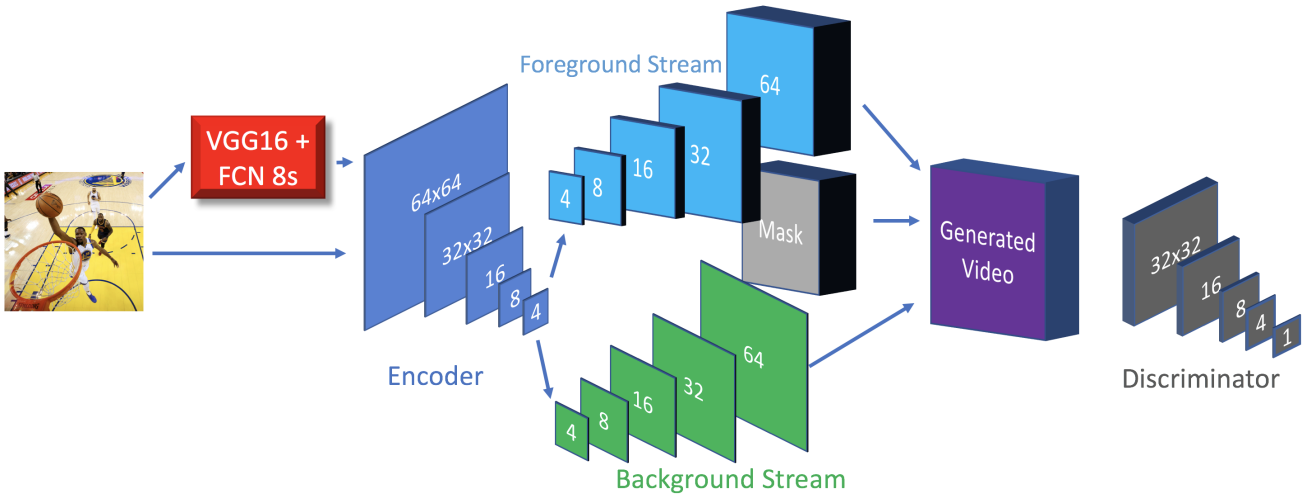

Figure 1. Two-Stream GAN with Pre-Trained Semantic Segmentation Network

ground. This follows the work of Pathak *et al*. [17], who construct an adversarial network to learn inpainting from surroundings. We did not complete mapping the foreground into the output video because of problems with the semantic segmentation outputs, which caused the network to produce incorrect video masks. More precisely, because the masks predicted by the FCN-8s network were not sharp enough for images in our training set, our predicted videos became extremely noisy, preventing us from doing any further generations with this technique. As seen in 2, predicted masks often don't include the entire entity, and are not sharp enough to accurately segment the foreground from the background. We attempted to rectify this my adding clustering on top of this predicted semantic segmentation, but many photos still did not have sharp enough masks for use in our network. This also showcases the difficulty of the problem at hand, because even state-of-the-art classification networks struggle with predicting images containing motion.

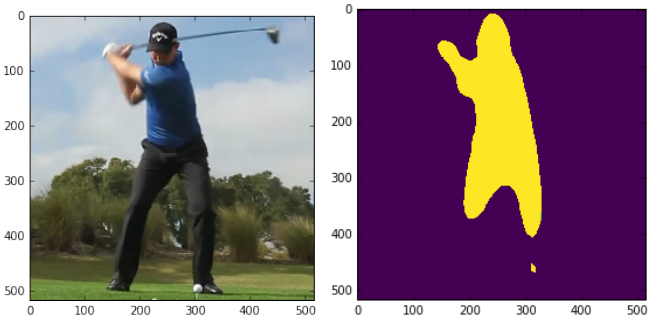

Figure 2. Outputs from VGG16/FCN 8s

4.6. Frame Prediction Using Recurrent Networks

Previous work has shown that a recurrent network is very powerful in encoding information in a video [4] [14][16], especially when using LSTM cells, which help mitigate many of the problems of RNNs such as vanishing gradients [10]. We constructed a model that attempted frame-by-frame prediction using Recurrent Networks. Specifically, an LSTM network was used to encode information from previous frames and attempt to generate the next frame based on it. The model can be seen in Figure 3. The input frames are encoded into a 1024 dimension vector through a CNN that is then fed into the LSTM network. The neural net includes multiple batch normalization layers to help speed up training as well. The output is fed through a deconvolutional network in order to generate the next frame. For training purposes, each frame of the video is fed to the network, in order to train the model to be able to output next frames given the correct previous frames. Additionally, dropout was added in between layers and in between frames in order to prevent overfitting and provide ample regularization, as well as resisting the co-adaptation of features [22]. In this case, the loss is computed as the L2 loss between the generated video and the actual video. During testing, the deconvolved output of each cell is input back into the network at the next timestep in order to get the network to generate the whole video only given the first frame.

4.6.1 Convolutional LSTMs

We also experiment with using a convolutional LSTM (convLSTM) cell for our recurrent network. ConvLSTMs have been shown to capture spatio-temporal correlations better, which is well-suited for our task. As described in [20],

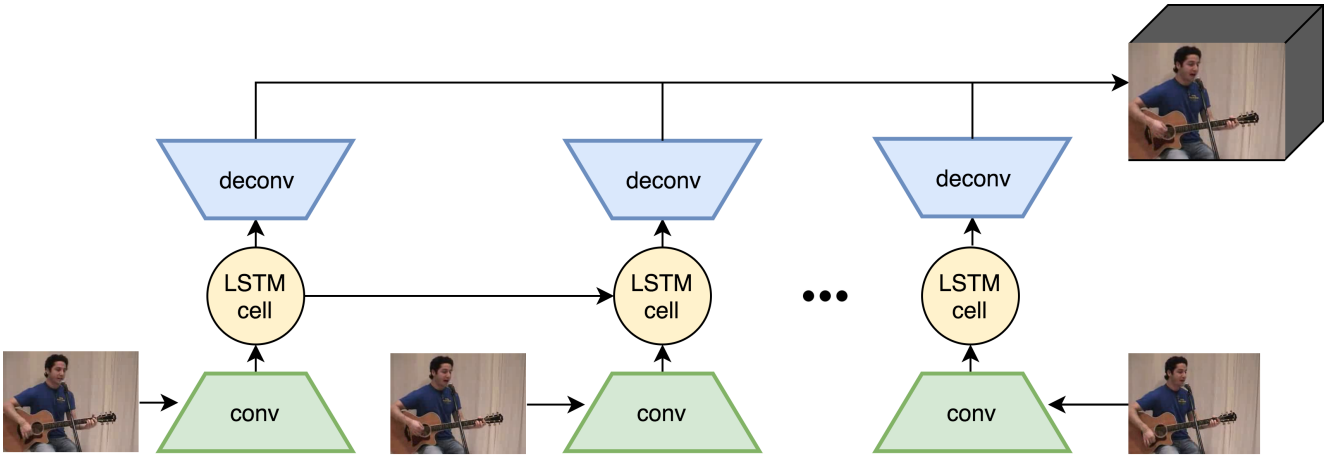

Figure 3. LSTM Single Frame Predictor

the key differences between traditional LSTMs and convLSTMs is that Hadamard products '$\odot$' are replaced with 2D convolutions, denoted as '$*$', with the weights of the cell. The input vectors X_t and hidden states H_t are now 3D tensors (first two dimensions are spatial and the third is for channels) rather than 1D vectors. This allows us to avoid having to flatten or reshape the input and output vectors, retaining the spatial information in our images. The equations are described in Eq. 4.

$$
\begin{aligned}
&i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + b_i) \\
&f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f) \\
&o_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + b_o) \\
&C_t = f_t \odot C_{t-1} + i_t \odot tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
&H_t = o_t \odot tanh(C_t) \\
&\text{Where } * \text{ denotes the convolution operation}
\end{aligned} \tag{4}
$$

4.7. Learning Parameters

We train with the Adam Optimizer using a momentum term of 0.5 and a learning rate of 0.001. In each iteration, we train the discriminator 5 times while training the generator only once, because the generator is constrained to only generating as realistic of videos as the discriminator is powerful. We also use a batch size of 32. Once the generated videos are fairly reasonable, for the conditional WGAN, we add the L2 loss between the input photo and a generated frame. All of our models were trained from scratch except the semantic segmentation network and InceptionV4 step, which are initialized with weights from their corresponding networks.

The LSTM network is trained using the L2 loss of the actual next frame and the predicted frame at each time step.

5. Experiments

We integrated several datasets and selected subsets of this data depending on the architecture used.

5.1. Datasets

To get large amounts of unlabeled video data, we primarily use two datasets: Aslan and UCF-101. Aslan contains $\sim$4000 action samples from 1571 unique Youtube videos, while UCF-101 contains 13320 videos from 101 action categories, which mostly comprise of sports and outdoor activities.

We also download a a dataset provided by Vondrick that includes over 10,000 videos of a particular scene from Flickr: outdoor 'golfing' scenes; not all of these scenes actually involve golfing, but most are outdoors and have somewhat green backgrounds. These micro-videos represent a high level of diversity [15] to prevent our models from overfitting to a particular camera position or scene setup. We hope that training on a particular scene limits the possibilities of potential scene dynamics and improve results to be more realistic.

5.2. Evaluation

One difficulty with generative models is evaluating the performance of the network because there is currently no widely-accepted accuracy metric for generated content. We could measure L2 distance on a validation set, but this could also incorrectly punish a network that produces a different plausible video with the same image. Vondrick *et al*. leveraged Amazon Mechanical Turk to evaluate the quality of the produced video. However, they also constrained their problem to producing videos of golf courses, beaches, and trains. We evaluate our results by manual inspection, and provide examples of our results online for the reader). Upon manual inspection, most of these generated videos look realistic from far away, but are clearly not real on closer inspec-

tion. This achieved similar results to Vondrick *et al*., who evaluated in the same manner on Amazon Mechanical Turk and found that generated videos could still be differentiated from real videos. Our videos are available on Firebase.

6. Results

We compare the outputs of each model qualitatively by observing the generated videos produced by each network.

6.1. Baselines

We first built a simple next-frame prediction model, which is a CNN-based autoencoder-like architecture trained on frames for outputting the following frame. This follows a similar setup to Finn *et al*. [5] and temporal transformations [29], but does not include CDNA layers. We used two approaches in the above: directly learning the proper output, or learning the difference/residual (mirroring the approach used in ResNets [9]). The first approach has issues with lighting changes in video overtaking the transitions, as shown in 4. The second approach suffers from the video darkening over time and washing out the foreground, as seen in 5.

Figure 4. Next-frame prediction

Figure 5. Residual next-frame prediction

Both of these above approaches struggle to learn motion, and also suffer from a larger problem in that small amounts of noise/error accumulate over time, so the subsequent frames would only compound this error.

6.2. WGAN

We built our main GAN model in TensorFlow and ran it on the aforementioned datasets on Nvidia Tesla K80s on Google Cloud Platform. We outline the results from the single-stream, two-stream, and conditional networks.

6.2.1 Single-Stream

We noticed that our network occasionally recognizes which parts of the image should involve motion/activity, but doesn't know what to do with these components. Instead, the network produces a blur of pixels at those locations and often forms in unrealistic blobs surrounding the object in motion.

6.2.2 Two-Stream

In the two-stream model, we notice the moving object in the foreground is not completely segmented out of the background. However, we acknowledge there are many edge cases in the dataset where the background may not be completely stationary or that the main object such as a person in the foreground is only moving a part his/her body. We follow the methodology of Wang *et al*. [26] in combining two convolutional streams. However, the majority of motion is captured in the learned mask as seen in 6.

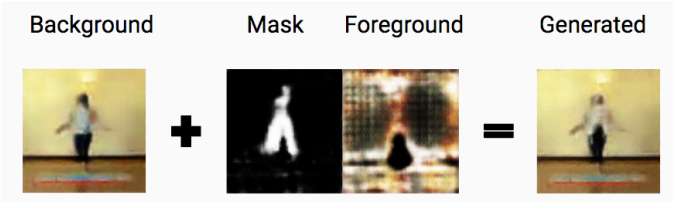

Figure 6. Visualization of the mask, and the foreground and background streams.

Figure 7. Videos that were hallucinated by the two-stream network trained on the golf dataset. A latent code was sampled from a guassian distribution was fed forward through the generator. We see many examples containing artifacts in the frames.

6.2.3 Conditional WGAN

We use the two-stream generator for our conditional network and train end-to-end. The network quickly learns to mimic the majority of the background quickly whereas it takes longer to properly learn the foreground stream. In situations where the subject moves a substantial amount, artifacts can be seen surrounding the object.

6.2.4 Inception V4 GAN

Conditioning both the generator and discriminator also led to a significant increase in runtime for our GAN, even keeping the weights in InceptionV4 fixed. Including InceptionV4 also forced us to lower our batch size in order to

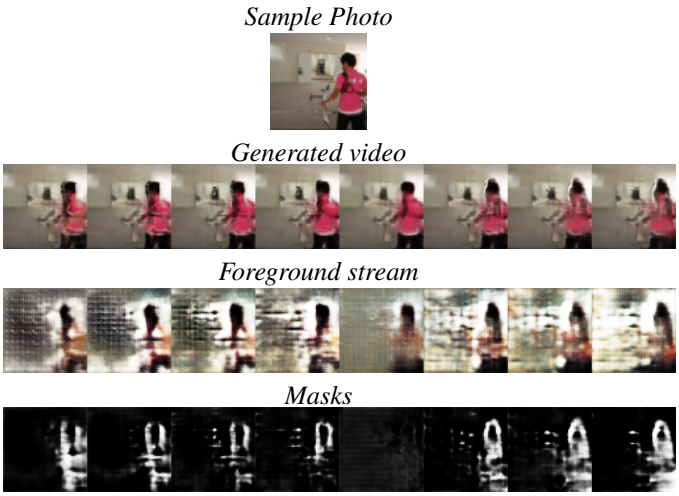

Figure 8. Outputs from two-stream WGAN Conditioned on a Static Photo. The image seen is a man about to shoot an arrow.

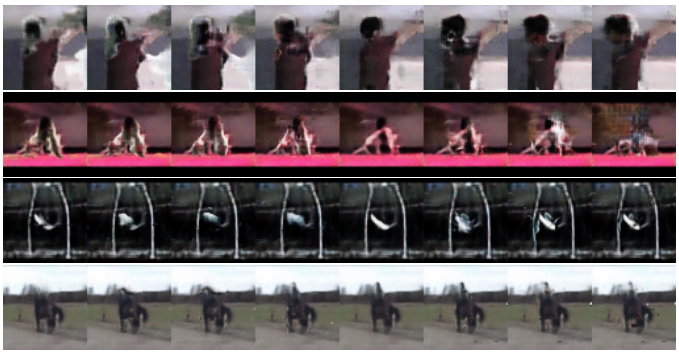

Figure 9. More Example Outputs from the Conditional WGAN.

make room for the InceptionV4 weights in GPU memory. We found that this network had some ability to predict motion, but still left the resulting videos to be blurry. Furthermore, this network tended to produce discolored people (mostly blue blobs), which suggest that the discriminator did not adequately learn that people should not be that color. As seen in 10, the network learns to move the person's arm, but not in a convincing and non-blurry way. We suspect that this network would do better after more training, but we were only able to train this network for about 3 epochs as each epoch took around 12 hours.

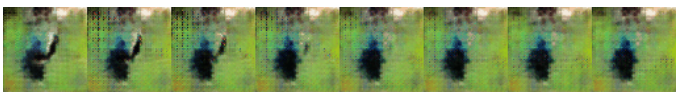

Figure 10. Inception GAN Output

6.3. Convolutional LSTM Network

We also evaluate our LSTM network model, in which frames are fed in order to predict the next frame given all of the previous frames. To generate videos, similar to the conditional GAN, we can input the first frame into the LSTM and predict future frames. The LSTM network model proved to be very effective in predicting the corresponding next frame, and does extremely well capturing movement in the foreground. However, the predicted frames often appear blurry when compared to the ground truth frames.

When training, the network was quickly able to learn the background. Because the LSTM captures temporal information between frames, it performs much better than our autoencoder baseline for subsequent frames.

To completely generate a video from a single photo, we can recursively feed in the output of the previous cell as input to the current frame. As seen in figure 11, the network was able to produce realistic frames at the beginning but towards the end, the frames begin to dissolve beyond recognition. This has also been shown as a shortcoming in [12] for video prediction using LSTMs. The error becomes compounded as the LSTM is unrolled.

Figure 11. Frame Prediction using Convolution LSTMs.

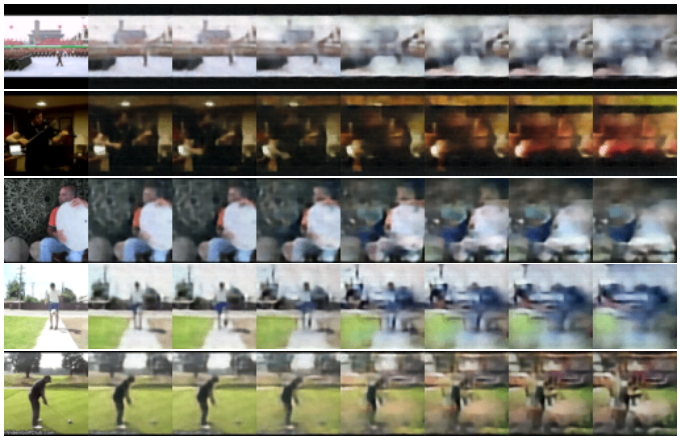

Figure 12. Recursive Frame Prediction using Convolution LSTMs. The previous output is fed into the input for the next cell.

6.4. Comparison of Architectures

We evaluate quantitatively by running our conditional models on videos excluded from the training set. We input one frame from the video, and then measure the mean-squared error between the predicted video and actual video. This metric may unnecessarily punish models that produce realistic futures that differ from the ground truth video. Furthermore, mean-squared error doesn't accurately reflect how humans perceive realism in a video. However, we found here that the numerical results somewhat reflect our own perception of the generated videos, and thus report them below:

Network Architecture	MSE
Conditional One-Stream	0.33
Conditional Two-Stream	0.24
Semantic Segmentation Network	0.19
Inception v4	0.22
LSTM Network*	0.037
LSTM Recursive	0.24
Convolutional LSTM*	0.025
Convolutional LSTM Recursive	0.15

Table 1. Average mean-squared error for each network.
*These architectures have relatively mean-squared errors due to the fact they are only doing single-frame predictions.

7. Conclusion

Overall, the two-stream WGAN architecture led to the most consistently realistic results, while the LSTM models led to more interpretable results. Generative adversarial networks have proven to be effective in this area, extending from photo generation to video. However, most of the results are still not comparable to actual real videos and many artifacts still remain in our generated video that would immediately discount it from being realistic.

Nonetheless, GANs and LSTMs have proven to be a promising architectures for video generation. A viable avenue to explore in the future would be to combine both methods by creating an LSTM network for both the generator and discriminator and optimizing the GAN loss.

In addition, to clean up our videos, we can apply pre- and post-processing steps such as stabilization and denoising as well as take advantage of other vision techniques such as optical flow, which captures how pixels move between frames.

One future method to explore is having a different network for each action class [11], and have a separate network that selects the action class present in input photo.

View our videos at relive-photos.firebaseapp.com

References

[1] G. Antipov, M. Baccouche, and J. Dugelay. Face aging with conditional generative adversarial networks. *CoRR*, abs/1702.01983, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, Jan. 2017.

[3] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[4] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2014.

[5] C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[6] A. Ghosh, B. Bhattacharya, and S. B. R. Chowdhury. Handwriting profiling using generative adversarial networks. *CoRR*, abs/1611.08789, 2016.

[7] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

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

[11] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. *Activity Forecasting*, pages 201–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[12] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104, 2016.

[13] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015.

[14] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015.

[15] P. X. Nguyen, G. Rogez, C. C. Fowlkes, and D. Ramanan. The open world of micro-videos. *CoRR*, abs/1603.09439, 2016.

[16] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. P. Singh. Action-conditional video prediction using deep networks in atari games. *CoRR*, abs/1507.08750, 2015.

[17] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[19] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, Apr. 2017.

[20] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[23] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[24] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 613–621. Curran Associates, Inc., 2016.

[25] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. *CoRR*, abs/1505.00295, 2015.

[26] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *CoRR*, abs/1507.02159, 2015.

[27] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

[28] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *CoRR*, abs/1609.05473, 2016.

[29] Y. Zhou and T. L. Berg. Learning temporal transformations from time-lapse videos. *CoRR*, abs/1608.07724, 2016.

[30] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

2

[2]Code references:

- Homework 3 code (GAN)
- https://github.com/jiamings/wgan
- https://github.com/warmspringwinds/tf-image-segmentation
- https://github.com/tensorflow/models/tree/master/inception/inception/slim
- https://github.com/raghakot/keras-resnet/blob/master/resnet.py
- https://coxlab.github.io/prednet/
- https://github.com/loliverhennigh/Convolutional-LSTM-in-Tensorflow