Video Frame Interpolation and Extrapolation

Zibo Gong
Stanford University
450 Serra Mall, Stanford
zibo@stanford.edu

Ziyi Yang
Stanford University
450 Serra Mall, Stanford
zy99@stanford.edu

Abstract

Deep learning has shown great potential in image generation, such as texture synthesis, style transfer and generative adversarial model. In our project, we use auto-encoder network and feature fusion model to generate interpolated and extrapolated video frames. We show that for frame generation, making networks fuse and learn from feature space given frames is of great significance. An auto-encoder network is used to extract features from given frames. Each layer generated by auto-encoder of two input frames are fused by another convolutional neural networks, in order to predict object motion and frame shift and generate predicted frame. We also use the sum of l_2 loss on each feature layer in autoencoder network of predicted and true frames. Our results surpass the performance of beyond MSE model in [9].

1. Introduction

Video frame prediction is one of the classics and most challengingcomputer vision problems. Predicting video frames from existing ones involves accurately modeling the content and dynamics of image evolution. Also generating in-between frames and subsequent frames can achieve high video frame rates and predict the motions, which is of great commercial prospect. Meanwhile, video frame interpolation is also a quite challenging problem, since it requires accurate modeling of clear motion and clear reconstruction of pixel reconstructions.

The traditional solution to this problem include estimating the relative motion or optical flow between input frames. Also with the rise of deep convolutional neural works (CNNs), researcher can directly use CNNs to synthesize RGB value of pixes in image [9, 13]. We combine strengths of these two solutions to form out own method.

In this report, we present a high-performance feature fusion model using deep convolutional neural networks. Our model includes two networks: autoencoder and prediction network. Specifically, for each input frames we use a pre-trained autoencoder network to extract features. One important step is to fuse these features by a another prediction network to learn the relative motion between those frames and use this information for new frame generation. And note that the weights in autoencoder is still trainable when training prediction network.

We build our own dataset from UCF 101 motion dataset. The model is trained via Tensorflow framework on an Nvidia 1080 GPU. But we test it on several other video datasets. Our model can be applied to any resolution. There is a interpolated video demonstration section at the end of this report.

2. Related Work

Convolutional neural networks have shown great power in image recognition in recent layers [3, 6]. Also CNN is used to directly generate image and video, for example GAN (generative adversarial networks, [4]), style transfer [5] and texture synthesis [12]. Also researchers used CNN for scene reconstruction from images in multiple viewpoints [2]. In [17], optical flow (appearance flow) layers has been learned by CNN and used to render object motions.

Video interpolation is commonly used for rendering, compression and increasing video quality [10] and is one of the most fundamental computer vision and digital image processing technologies. One conventional method for frame generation is motion estimation-based method, where motions between two consecutive frames are estimated by optical flows or moving gradient methods[8, 1]. However, these methods all require some hand-crafted adjust for late image synthesis.

Instead of motion-based method, another approach for video interpolation is phase-base method. [10] kept phase information and propagate it across pyramid levels to accommodate large motion. [15] represent motions as phase variations to manipulate small movements in videos, over an analysis in complex-valued image pyramids.

For using deep learning on video interpolation, Mathieu *et al*. [9] used Mean Squared Error (MSE) loss function and multi-scale architecture to solve blurry problems in frame

prediction. Liu *et al*. [7] addressed the problem of synthesizing new video frames by utilizing optical-flow-based network. Shi *et al*. [14] proposed a feature-space based network capable of real-time prediction on 1080p videos. Niklaus *et al*. [11] employed a convolutional neural network to estimate a spatially-adaptive convolution kernel for pixel synthesis. Meyer *et al*. [10] introduced a bounded phase shift correction method to capture the phase shift of individual pixels.

The loss function also play a important role in this problem. [9] used used Mean Squared Error (MSE) loss function. [11] used gradient loss and color loss function, where they compute the gradient of input example and then perform convolution with the estimated kernel to alleviate blurriness problem.

3. Dataset

The quality of dataset can directly impact the training output, especially for deep learning model. Here we select UCF101 - Action Recognition Data Set. The Data set includes 13320 videos from 101 action categories. The action categories can be divided into five types: 1)Human-Object Interaction 2) Body-Motion Only 3) Human-Human Interaction 4) Playing Musical Instruments 5) Sports. The size of each video frame is 240×320, and the frame rate is 25 FPS. And the length of each video is approximately 4-5 seconds, including hundreds of frames.

Our method for generating dataset is as follows: for each 5 frames, we randomly sample a patch with size $32 \times 32 \times 3$ from the frame and also sample at the same place for its following 2 frames. Thus we have three frames with size $32 \times 32 \times 3$ and this is one data points.

After having sampled datasets, we split them into training and validation set, with 10:1 ratio. The training set contain about 200, 000 examples, and the test set include about 20, 000 examples. For the test set, we use 500 examples but with the same size as original video frame size, namely $240 \times 320 \times 3$, to compare our result with other methods. Note that videos only appear in training set or test set to ensure the validation of data set. Another important step is data prepossessing. We normalize the RGB value of each pixel into [-1, 1] range.

4. Model

We explored two models, one is vanilla model without feature fusing. Another is the refined model consisting of one autoencoder feature extraction network and a second feature fusion convolutional neural network.

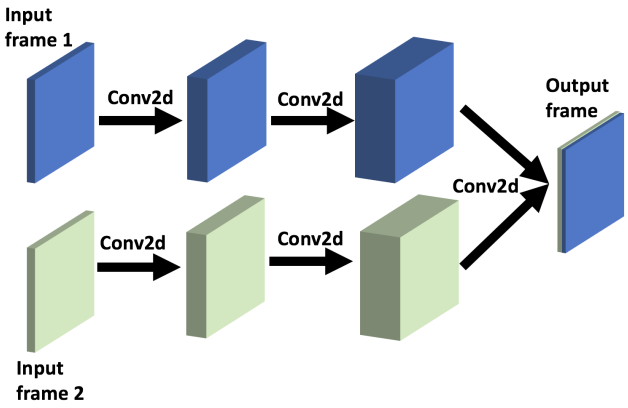

Figure 1. Architecture of vanilla model. The two input frames are first fed through two convolutional layers, with valid padding and 3×3 kernel size. After concatenating two outputs and go through another convolutional layer, a normal image size output frame is obtained.

4.1. Vanilla Model

4.1.1 Architecture

Encoder Input frames are fed into two networks with the same structure but with different weights. The structure of the network is conv-relu-conv-relu. The kernel size of convolution is 3×3. The two outputs are then concatenated on the last dimension. After the first conv-relu, the dimension of output is $32 \times 32 \times 8$. After the second conv-relu, the dimension of output is $32 \times 32 \times 16$. Concatenating two output, the dimension of the encoding of these two frames is $32 \times 32 \times 32$.

Prediction After obtaining the encoding of two input frames, we simply use a fully connected layer to generate the predicted frame. Specifically, we time the $32 \times 32 \times 32$ tensor with a matrix $W \in \mathbb{R}^{32 \times 3}$ to get a $32 \times 32 \times 3$ frame. The structure is illustrated in figure 1.

4.1.2 Primitive Result

The loss function we adopt is l_2 error between the true video frame and predicted video frame. Also we choose adam update rule to achieve faster convergence. The learning curve is shown in 2. It clearly shows fast convergence, due to the small size of the model. This also indicates it should have reached the model capacity.

One of the best primitive test results is shown in figure 3. The result looks okay, but obviously the legs and back of the horse are blurry and the background is unclear due to fast motion of frame. Despite clear flaws, the scene reconstruction by vanilla model is of reasonable quality and if playing these 3 frames in sequence, it shows smooth and

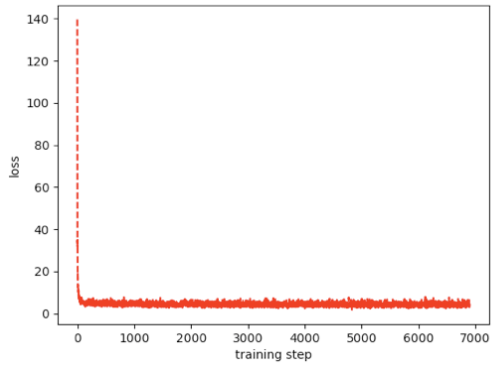

Figure 2. Learning curve of vanilla model. Learning rate 0.01

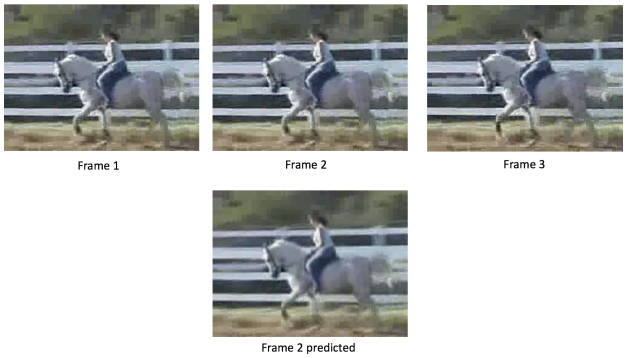

Figure 3. One of the best primitive results by vanilla model. The top three images are real three consecutive video frames. The bottom one is the predicted interpolation frame.

natural motions. However, in some extreme test cases, the result could be very blurry .

The reason why the model doesn't work well could be that it doesn't consider much about the relative motion between frame 1 and frame 3, since only the final convoluational layer might learn the object motion.

4.2. Final Model

We improve the vanilla model by adding more powerful feature extraction methods and fusing features concretely. This refined model shows much better result and surpass the result in [9].

4.2.1 Autoencoder

Instead of training feature extraction part in our model from scratch, we train an autoencoder network. Although this makes the model not end-to-end, from the results obtained, this doesn't affect much. This separate model enables the final prediction model more effective and to converge, because the encoder model will give a much better starting point.

The structure of autoencoder and layer dimension is

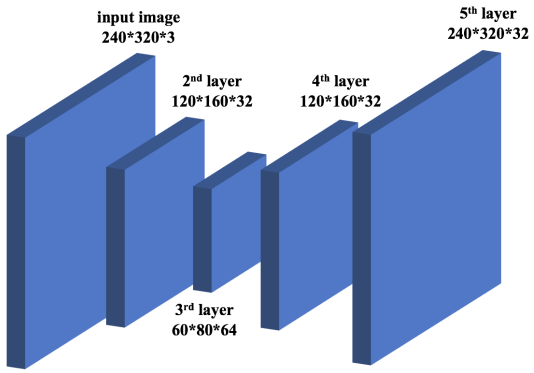

Figure 4. Structure of autoencoder network.

shown in figure 4. Starting with two convolutional and max-pooling layers with kernel size 5. The bottle neck layers is followed by two transpose convolutional layers with kernel size 3 to enlarge it back to the normal size. The first three layers of autoencoder is extracted as features for later network.

The training set of autoencoder is randomly sub-sampled from the video frames dataset pre-built. We tried other pre-trained autoendcoer and transfer learning from other datasets, but it turned out because of large size of video frames ($240 \times 320 \times 3$), training on the very same dataset works best.

4.2.2 Fusion Network

After obtaining first three-layer features from the autocoder network, we use a fusion-feature model to get the final frame. As shown in figure 5, three feature layers of two input frames are fed respectively into three different neural networks. The blue block stands for frame 1 and green block stands for frame 3. For example, the bottle neck layer in autoencoder is enlarged back to normal image height and width by two transpose convolutional layers with kernel size 5×5 kernel size. This is where comes the fusion: the feature layer of frame 1 and frame 3 are fused by two transpose convolutional layer s. And the fusion happens in other two feature layers, too.

At last, those three output layers (shown as three gray blocks) are concatenated on the last dimension and put into a final convolutional layer with 3 kernel with size $5 \times 5 \times 128$. The output is our final predicted frame.

4.2.3 Loss Function

Rather than simply calculating l_2 loss function of the final prediction layer and true frame as in our vanilla model, here we feed the predicted frame and true frame back into the autoencoder, calculate the sum l_2 loss on each layer. The

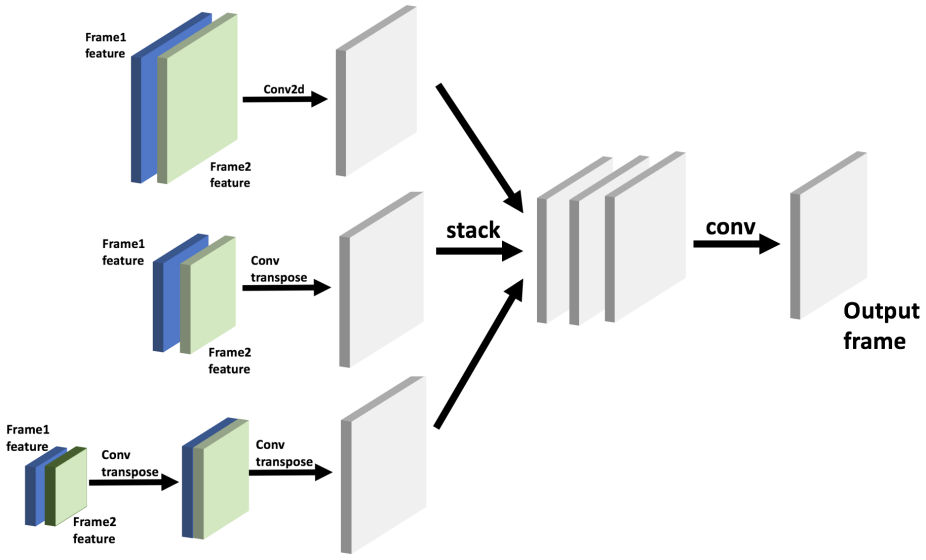

Figure 5. Structure of prediction network.

loss function given by:

$$L_{x_p} = \sum_i ||f_i(x) - f_i(x_p)||_2 \quad (1)$$

Here $f_i()$ is ith feature layer in autoencoder, x is the true frame and x_p is the predicted frame. And $||\cdot||_2$ is the l_2 norm.

5. Experiments

In this section we will present our experiments, including the training process, evaluation standard, illustration of final predicted frame and a interpolated video demo.

5.1. Network Training

Network training and parameter has a key impact on final performance. In our model, hyper-parameters include learning rate, hidden size in both autoencoder and final prediction network, updating rule. Here we presented we explored parameter space and try to find the optimal combination.

The training time takes about 30 minuets per epoch. And usually it will converge after 2 or three epochs, and the loss could be as small as 0.1.

5.2. Final Results

5.2.1 Interpolation

Some examples of predicted interpolation frame are shown in 6. A we can see, our predictions are of good quality. First, scene reconstruction is almost perfect and reconstructed images look totally reasonable. Second, predicted images generally are not blurry, which indicates that our feature fusion model works and the networks have learned the motion of frame. Still when the motion of object is too fast, the predicted image can be blurry. This could be because that the ground truth frame is blurry since the video quality of dataset is not very high.

A contrast of predicted frame with ground truth frame and frame predicted by beyond MSE method is shown in figure7. Our result is clearly better than the Beyond MSE's. As one can see in the image, regions of rider's head and horse's legs are really blurry in frame by beyond MSE. In contrast, the background in our frame is clear which is issue in the vanilla model. Also regions where object is in motion is barely blurry.

5.2.2 Extrapolation

For extrapolation, the input images just simply change to the first and second frame. One prediction result is shown in 8 and it shows good quality. This proves that our model also works for extrapolation and the robustness of our model.

5.3. Evaluation

We use two metrics for evaluation: Peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [16].

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

Where MSE is the mean squared error. MAX_I is the maximum possible pixel value of the image.

SSIM is defined as:

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

Figure 6. Interpolation prediction results. The first row is the first input frames, the second row is the predicted frames and the third row is the second input frame.

Figure 7. An example of predicted interpolation frames. The frame 1 and frame 3 is input frames. True frame 2 is the ground-truth middle frame. Beyond MSE is the frame predicted by model in [9]. The result of our model is the fourth one.

SSIM and PSNR are two standards quantity metrics widely used in frame generation quality evaluation. Our score calculated on average of 1000 predicted frames with original ones. The evaluations are shown in table 2. Our model is slightly better than beyond MSE [9] on SSNR and SSIM. But there is still a gap between the state-of-art [7].

5.4. Video Interpolation Demonstration

Here we make a demonstration of interpolated video. The demo video is available here. Please note that the shiny green dots in the video is due to decoding issue of Youtube website, not the video itself. First we sample from the video every two frame, i.e. deleting the frame in the middle. The video on the right is result of interpolating the video by copying the frame before. The video on the left is interpolated by our model. Clearly, the video played on the right is really rigid. And our interpolated video looks much more smooth, clear and natural. This again indicates good performance of our model.

Method	PSNR	SSMI
Vanilla model	26.2	0.84
Final model	29.8	0.92
Beyond MSE	28.8	0.90
Deep Voxel Flow	30.9	0.94

Table 1. Evaluations of the interpolation frame generation. Our final model performs better than the Beyond MSE model. However, it dosen't beat the state-of-art model.

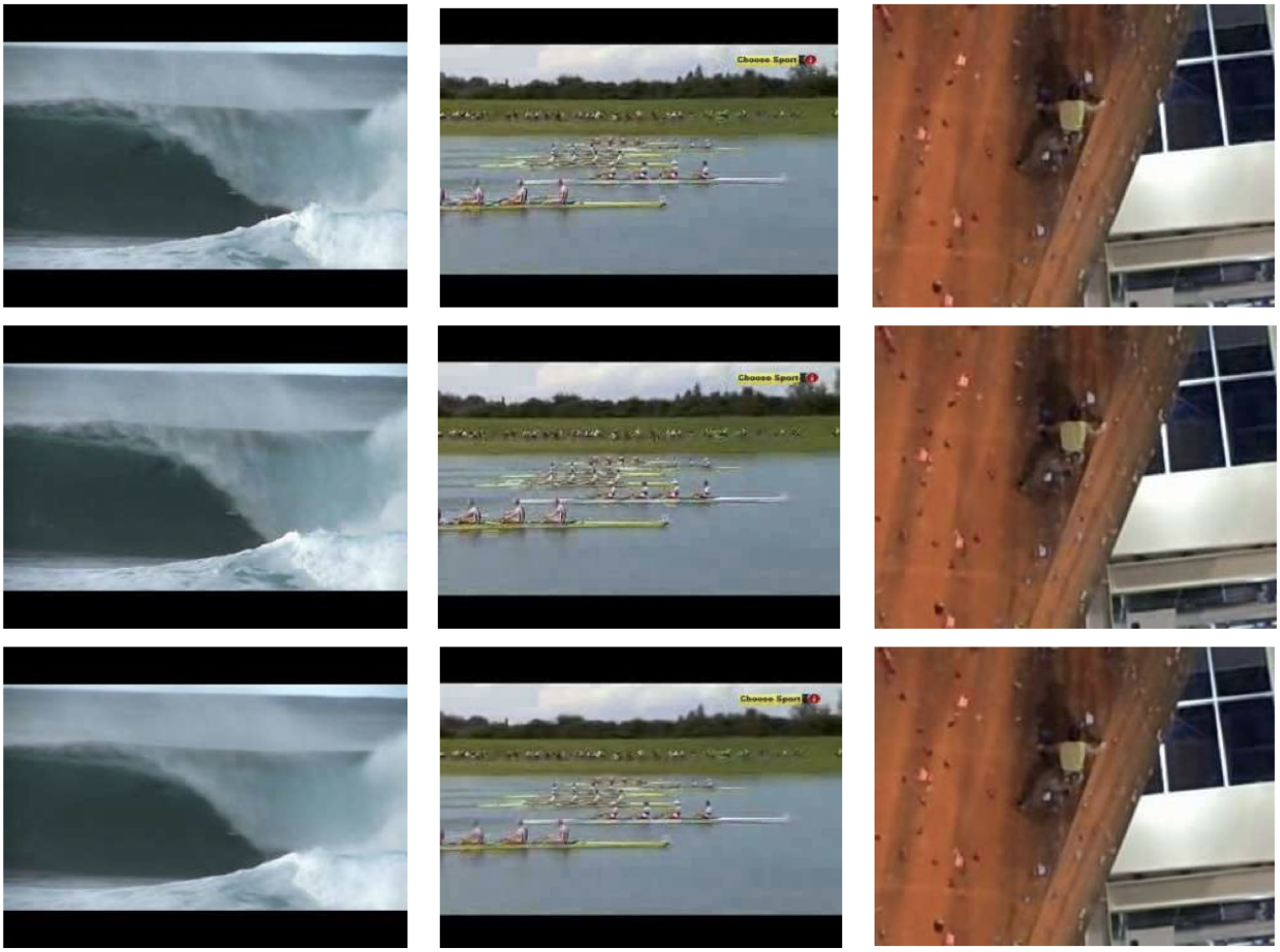

Figure 8. Extrapolation prediction results. The first two rows of images are ground truth images. The third row of images are extrapolation frames.

Method	PSNR	SSMI
Vanilla model	25.9	0.86
Final model	28.2	0.90
Beyond MSE	28.2	0.89
Deep Voxel Flow	29.6	0.92

Table 2. valuation of the extrapolation frame generation.

6. Conclusion

We use two model to generate interpolated and extrapolated video frames. The vanilla model combines the features extracted from two input frames to generate the new frame. As the receptive field for this model is relatively small(7×7), it can't catch the obvious motion. So most of the generated frames are blurry. To make up for this deficiency, we try to utilize multi-scale features of the two input frames. So we train a auto-encoder to extract those multi-scale features. We hope to use these features to learn the latent "flow-based" features, to achieve the same result as Deep Voxel Flow model. However, there is still a gap between our final model and the state-of-art. We may modify the structure of the neural network, build a new dataset or utilize the flow features to achieve a better result in the future.

7. Future Work

There are still some triplets of frames in our dataset, whose differences are to small to be detected. That means we should find a good threshold to filter the promising triplets for training and build a new training set. Whats more, we should also modify our nerual network to extact better features. Also, we would try to add the flow information to each feature map of the auto-encoder, and combine them to generate the new frame.

8. Acknowledgement

We want to thank all CS231n staffs for giving such an amazing course! And Shayne for mentoring our project! Also we appreciate Google Inc. for generously providing computing resources.

References

[1] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[2] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2016.

[3] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

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

[5] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Z. Liu, R. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. *arXiv preprint arXiv:1702.02463*, 2017.

[8] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Transactions on Graphics (TOG)*, 28(3):42, 2009.

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

[10] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1418, 2015.

[11] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. *arXiv preprint arXiv:1703.07514*, 2017.

[12] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[14] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[15] N. Wadhwa, M. Rubinstein, F. Durand, and W. T. Freeman. Phase-based video motion processing. *ACM Transactions on Graphics (TOG)*, 32(4):80, 2013.

[16] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[17] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European Conference on Computer Vision*, pages 286–301. Springer, 2016.