CS231N Project Final Report
CNN-based Encoder-Decoder for Frame Interpolation

Yue Li
yulelee@stanford.edu

Sijun He
sijunhe@stanford.edu

Li Deng
dengl11@stanford.edu

Abstract

The ability to accurately interpolate intermediate frames with two given frames has many applications, including video compression and frame rate up-sampling. Traditional video processing techniques has always dominated the field and tackles the challenge with optical flow estimation. However, optical flow estimation has a trade-off between speed and accuracy and estimating optical flow robustly is difficult in real time. The application of Deep Learning have been introduced to Frame Interpolation recently. Our final model extends an CNN-based encoder-decoder approach to Frame Interpolation. While the performance of our model is sub-par to the current state-of-the-art techniques, it outperforms the baseline of Linear Interpolation and can be run in real time.

1. Introduction

Frame Interpolation is a computer vision task where interpolation methods are applied to generate intermediate frames between the previous and subsequent frames. The main application of Frame Interpolation is video compression. Given that video sequence naturally contains temporal redundancy, frame interpolation can be used to reduce this redundancy by storing only the key elements and infers the rest. Frame interpolation could also be used to enhance video quality by up-sampling the frame rate (HDTV) [1], or even generate short video clips with a limited number of frames.

The underlying challenge of Frame Interpolation is to understand the motion between frames. Current state-of-the-art algorithms are mostly traditional video processing techniques, which model the motion between frames by estimating the optical flow. One of the key advantages of optical flow based methods is that they are generalized methods and delivers similar performance regardless of the context of the video. However, there are plenty of use cases (sports games, security camera footages) where the background and the content of the video are consistent so that a more specific approach could deliver superior performance.

Deep learning has been the main driving force behind the breakthroughs in Computer Vision, but its application in Frame Interpolation has only been introduced recently. Convolutional Neural Network (CNN) are specifically suitable for extracting features and identifying contextual relationships from images. Our project extends an approach using a CNN-based encoder-decoder and compares its performance against traditional video processing techniques.

2. Related Work

2.1. Motion-Compensated Interpolation

Motion-Compensated Interpolation (MCI) is a popular optical flow based approach to Frame Interpolation and is implemented on the hardware of many HDTVs [1]. A key part of MCI is Motion Estimation (ME), which estimates the optical flow. Optical flow is used to compute the motion of the pixels of an image sequence and provides a dense pixel-level correspondence [2]. The search strategy for Motion Estimation is generally broken into pixel-based methods and block-based methods.[3] Pixel-based methods are rather exhaustive and seek to determine motion vectors for every pixel in the image and delivers superior performance, but is unsuitable for real time implementation due to excessively large computation time[3]. The alternative and faster methods are the block-based methods, which divide candidate frames into non-overlapping blocks and compute a single motion vector for each block[3]. Block-based methods can be implemented for real-time estimation and thus is the standard method for Motion Estimation. After the Motion Estimation step, the intermediate frame is generated with the estimated optical flow with another step called Motion Compensation. The work of Guo and Lu [4] provides a detailed background and recent development of MFI. In our work, we use the implementation of MFI in ***FFmpeg*** as a benchmark to our own method.

2.2. Convolutional Neural Network

Given the MCI approach identifies contextual relationships between images by estimating the optical flow, it is natural to train a Neural Network architecture capable of learning to identify the contextual relationships. There has been some very recent development of applying CNN to Frame Interpolation. A recent bachelor's Thesis by Haitam Ben Yahia [5] and a recent paper by Long et al. [6] both adopted CNN based architectures and demonstrated that CNN is a viable approach to Frame Interpolation. The architecture is similar to the auto-encoder architecture by Hinton et al. [7], where the two images are encoded by a series of downsampling convolutional block and decoded by a series of upsampling deconvolutional block. Long et al. and Yahia shared "shortcut" technique allows the outputs of convolutional block to be additional input to the deconvolutional blocks and to skip part of the architecture. Similar idea is also used in ResNet by He et al. [8] and Highway Networks by Srivastava et al. [9]. Long et al. also carried over techniques popular in optical flow based methods into its work and adopted Charbonnier loss $(x) = \sqrt{x^2 + \epsilon^2}$, which is a differentiable and more robust variant of the *L1* loss [10].

3. Approach

3.1. Linear Interpolation

The naive approach to Frame Interpolation is to average the pixel values of the two input frames I and I'.

$$\hat{I} = \frac{I + I'}{2}$$

This approach is simplistic and treats raw pixel values as features for interpolation. The results are often a blurry overlay of the two input frames.

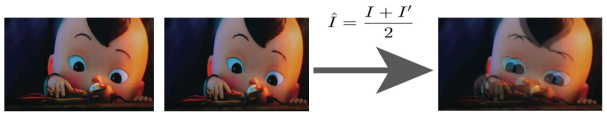

Figure 1. Blurry Overlay of Linear Interpolation

3.2. CNN Based Encoder-Decoder

A logical step after using raw pixels as features is to extract features from the image for interpolation. Inspired by the work of Yahia and Long et al, we adopt a CNN based encoder-decoder architecture (Figure 2). Different from Yahia and Long et al, who stack the input frames to 6 channel as the input to the encoder, we encode the two input frames independently into latent representations, and then decodes the concatenated latent presentations of the input frames to generate the intermediate frames.

As shown in Table 1, our model is fully convolutional. We used a series of convolutional layer as the encoder, a single convolutional layer to combine the concatenated features, and a series of convolution transpose layers as the decoder. The intuition behind the varying kernel sizes and strides was that we felt the need to first capture the large-scale features and then small-scale features with the encoder. The same intuition also applies to the decoder, where larger-scale structure are generated first with larger kernel and strides and details/textures are generated later with small kernels. The exact architecture is fine-tuned for each dataset.

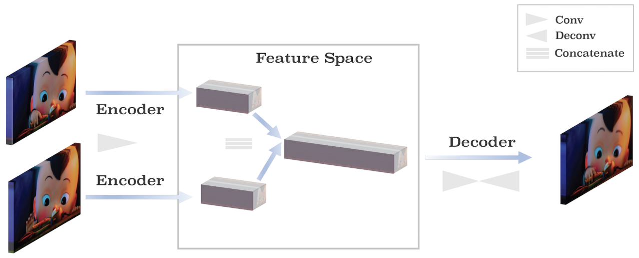

Figure 2. CNN Based Encoder-Decoder Architecture

3.3. Sequence-to-Sequence

One of the shortcomings of the CNN-based Encoder-Decoder model is that it only takes into account the *local* information of two input frames. In many cases, long-range patterns are only observable if a more *global* view of the motion is given. A good example would be a circular motion. If using the previous model, the motion would be piecewise linear as opposed to the desired circular motion.

This idea led us to the Sequence-to-Sequence model (Figure 3), where the input is a sequence of frames (Frame 1, 3, 5, 7, 9...) and the output is the sequence of subsequent frames (Frame 2, 4, 6, 8...). As 3.3 shows, a combined loss of the sequence images generated is computed and back-propagated to each step, so that global sequence information can be encoded.

To drive the point home, we designed a circular motion pattern in our toy moving box dataset. For this non-linear motion, local piece-wise prediction can only predict linear movement, and we expected the Sequence-to-Sequence model model to capture the global circular motion, and present more accurate prediction. However, due to time constraints, we weren't able to fully explore this model on other datasets. We also take inspirations from [11], where *LSTM* is used to explicitly encode motions from a sequence of frames.

4. Experiment

4.1. Datasets

Our datasets include the following parts:

Table 1. CNN Based Encoder-Decoder for Tennis Dataset

Layer	Kernel Size	Strides	Filters	Activation
Encoder				
conv	8	2	48	*ReLu*
conv	5	2	48	*ReLu*
conv	4	2	48	*ReLu*
conv	3	1	48	*ReLu*
conv	2	1	48	*ReLu*
conv	2	1	128	*Tanh*
Decoder				
conv	3	1	96	*ReLu*
conv_transpose	5	2	48	*ReLu*
conv_transpose	7	2	48	*ReLu*
conv_transpose	4	2	48	*ReLu*
conv_transpose	3	1	48	*ReLu*
conv_transpose	2	1	24	*ReLu*
conv_transpose	2	1	3	*Tanh*

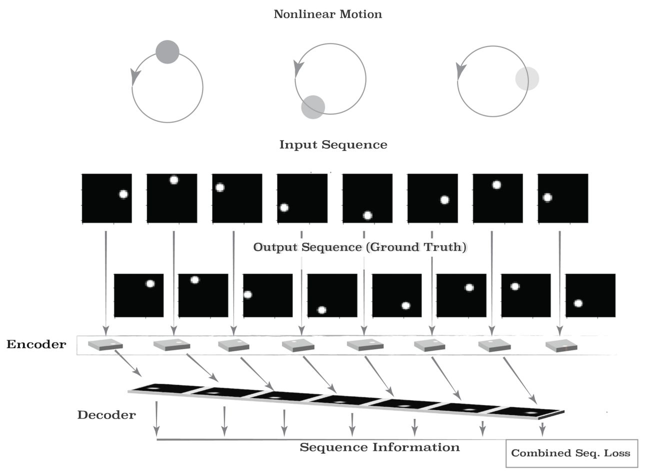

Figure 3. Sequence-to-Sequence Model for Encoder-Decoder

- **Moving Shapes**
  We hand-crafted a series of videos of moving shapes in various colors, shapes, speeds and paths (as shown in Figure 4). We captured the frames with *FFmpeg* and have more than 500 triplets of data (before, ground-truth middle and subsequent frames). This dataset was used as a simplified toy example and it designed to be overfitted. We experimented our network architecture based on this dataset in order to get a better understanding of how the motion feature is being captured and how each layer affects the result.

- **Tennis**
  In addition to our toy dataset. We also experimented our model on a small-scale real-life dataset (Figure 5). We used a short video depicting a female player playing tennis on a tennis court[12], within which the tennis court background is fairly stable and the moving objects (including the player, racket, or even the tennis ball) have clear movements, and therefore is suitable to serve as probe for testing our model's motion capturing ability.

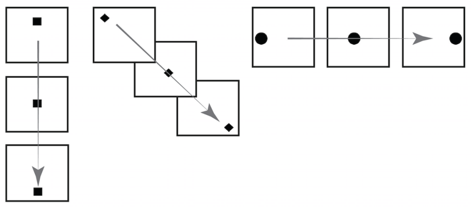

Figure 4. *Moving Box* Dataset

Figure 5. Tennis Dataset

4.2. Data Preprocessing

- **Data Augmentation**
  All moving box frames generated from *keynote* videos are all white background and black objects, and they were augmented by reversing the colors, or reversing the moving directions. In this way, we were able to obtain more than 3000 frames from the original 500 frames directly generated.

- **Centering**
  Original pixel values range between [0, 255]. To feed the network with balanced dataset, all data points are first transformed by function $f(x) = 2 \cdot x/255 - 1$. Thus all images fed into the network have a range between [-1.0, 1.0].

4.3. Metrics

- **Mean Squared Error (*MSE*)**
  The primary metric we use to assess the performance of our model is the Mean Squared Error (*MSE*) between the generated frames and the ground-truth images. We also use the *MSE* as the loss function when training our model.

$$MSE(y, \hat{y}) = \frac{||y - \hat{y}||_2^2}{H * W * C}$$

  where H, W, C are the height, width and depth of the input frames y and $\hat{y}$.

- **Peak-to-Noise Ratio (*PSNR*)**
  Peak-to-Noise Ratio (*PSNR*) is the log-scale ratio between the maximum possible power of a signal and the power of corrupting noise [13]. It is commonly used to measure the quality of reconstruction of lossy compression and is a function of the *MSE*.

$$\begin{aligned} PSNR(y, \hat{y}) &= 10 * \log_{10} \frac{\text{max power}^2}{MSE(y, \hat{y})} \\ &= 10 * \log_{10} \frac{255^2}{MSE(y, \hat{y})} \end{aligned}$$

- **Structural Similarity Index (SSIM)**
  The structural similarity (*SSIM*) index is a method for predicting the perceived quality of images and videos.[14] It is a qualitative metric and a good supplement to *PSNR* or *MSE*, which are robust quantitative measurements but is inconsistent with human visual perception.

4.4. Benchmark

As described in Section 2.1, Motion-Compensated Interpolation (*MCI*) is one of the state-of-the-art Frame Interpolation techniques at the moment. **FFmpeg** provides an implementation of *MCI* through a filter called **minterpolate**, which is used as a benchmark to our model. The technical configurations of the *MCI* adopted is the following:

- **Motion compensation mode**: overlapped block motion compensation
- **Motion estimation mode**: bilateral motion estimation
- **Motion estimation algorithm**: uneven multi-hexagon search algorithm

4.5. Results

- **Moving Box**
  The toy dataset of moving-box serves as our exploration and experiments to understand the problem. With the initial feed-forward convolutional neural network, we were able to generate relatively accurate movement predictions, as Figure 6 shows. With our encoder-decoder network, much more clear and accurate predictions can be made, while nuance shapes and edges can still not be accurately captured, as 7 shows. The model is fairly robust to various types of background colors, moving paths, or motion pattens.

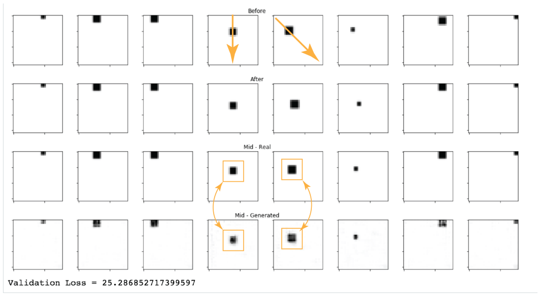

Figure 6. *Moving Box* 32x32 by Feed-Forward Network

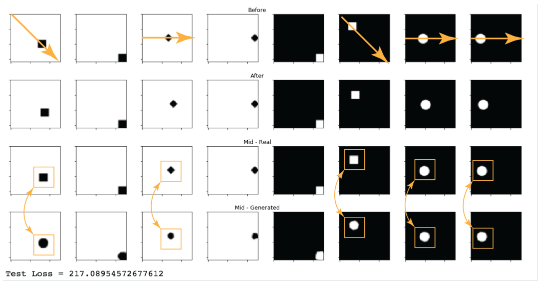

Figure 7. *Moving Box* 64x64 by Encoder-Decoder

- **Tennis**
  The encoder-decoder model is then applied to the tennis dataset. During training time, the model can fairly quickly output images that match with the ground-truth in most cases, while may be vulnerable to some failure cases, as Figure 9 shows.

  For the test data, the model is able to generate approximate configurations of the intermediate frames, but is unable to generate sharp edges, as Figure 8 shows.

  Table 2 shows a comparisons of the metrics evaluated on the tennis dataset. Our model delivers a inferior performance compared with the state-of-the-art MCI technique. It is worth noting that the comparison of our model against Linear Interpolation is consistent,

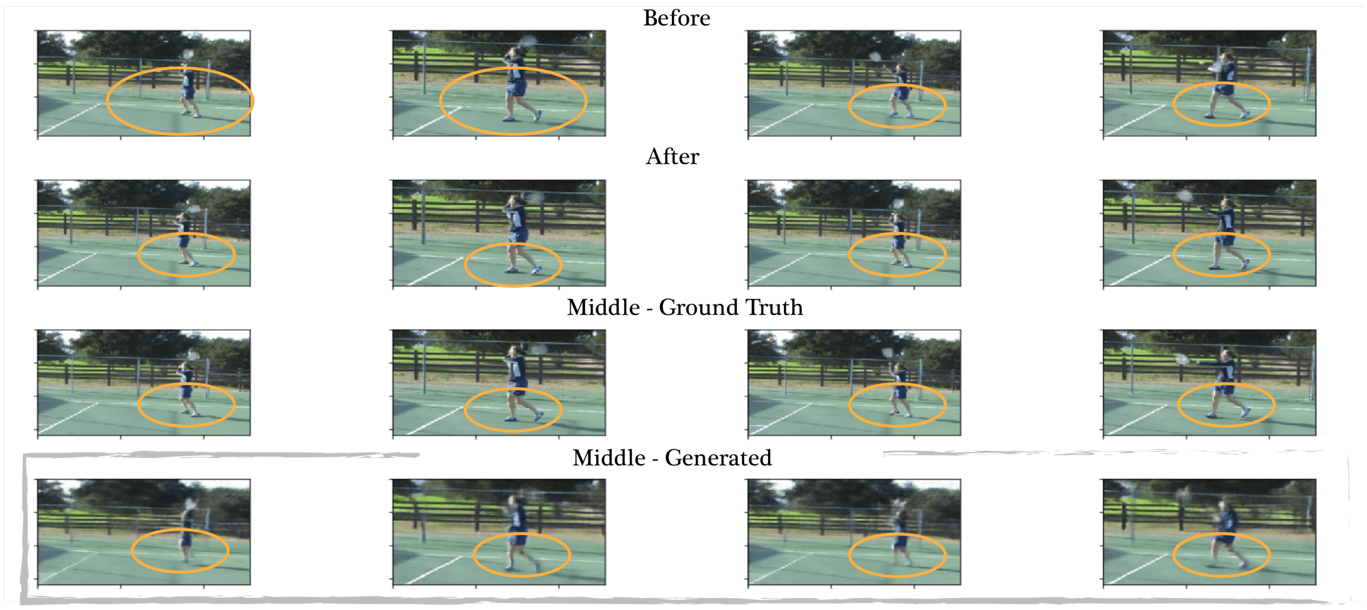

Figure 8. Tennis on Test Dataset

as our model outperforms in terms of the qualitative metrics SSIM but under-performs in terms of the quantitative metric of *MSE* and *PSNR*. This is in line with our own observation. The result of our model is less blurry on the moving object, which results in a better SSIM. But the model also introduces noise in the background, which lead to the under-performance in *MSE* and *PSNR*.

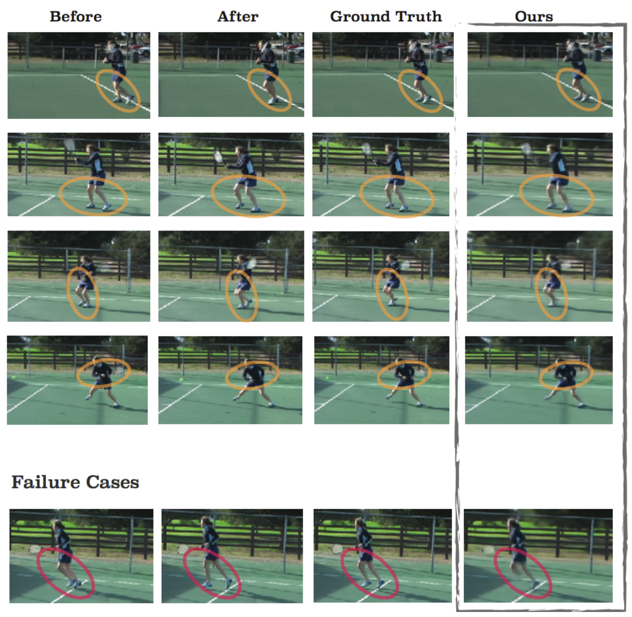

Figure 9. Tennis on Training Dataset

Table 2. Metrics Comparisons for Tennis Dataset

Interpolation Technique	MSE	PSNR	SSIM
Linear Interpolation	36.09	32.68	0.8319
MCI	29.43	33.53	0.9102
CNN-Based Interpolation	52.21	31.06	0.8721

5. Other Ideas Explored

5.1. Sequence Encoding Enhanced with LSTM

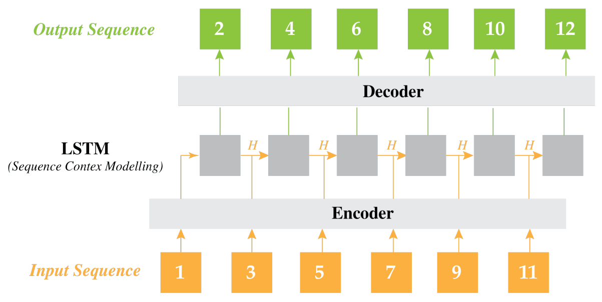

Figure 10. Architecture of LSTM for Sequence Prediction

Our Sequence-to-Sequence Approach described in Section 3.3 was originally conceived as a *LSTM* approach (Figure 10). The idea is to encoder a sequence of input frames individually into the feature space and to use the features as the input to a *LSTM/Bi-LSTM* layer to model the contextual relationship between the frame sequences. The decoder decodes the output of the *LSTM/Bi-LSTM* layer into intermediate frames. However, during implementation we find

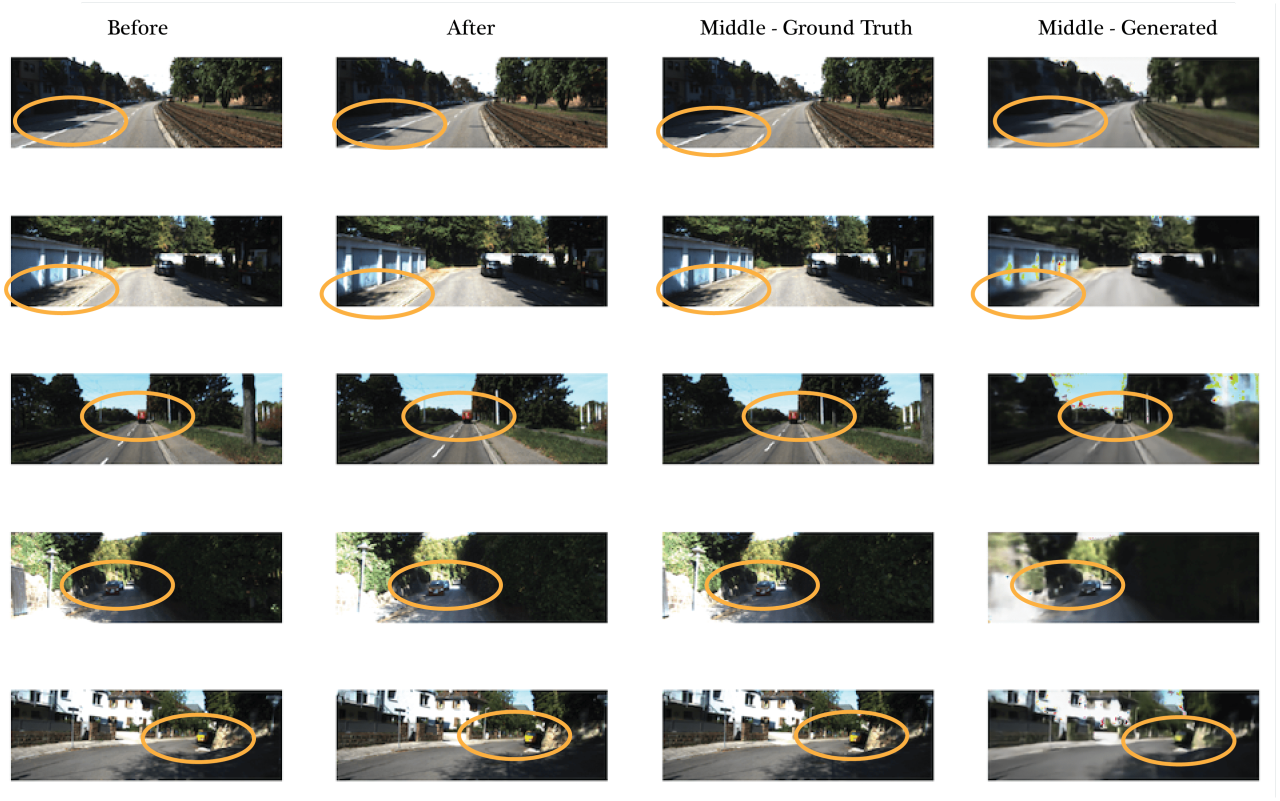

Figure 11. Result of reproduction of the work of Long et al.[6]

that the $O(n^2)$ space complexity of *LSTM* is a memory bottleneck of model and severely limits the size of the encoded feature space. We were not able to overcome this issue and didn't get a good result.

5.2. Reproduction of the work of Long et al.[6]

One of initial our approaches is to reproduce the work of Long et al.[6] in hope of further improvement. But we underestimated the difficulty of reproducing a state-of-the-art Deep CNN model. Long et al.'s model was trained for 5 days on a multi-GPU cluster and we were not able match with their resources or expertises. We experimented a few iterations with the model on the *kitti* dataset [15] and was able to get some promising results (Figure 11). However, iterating on Long et al.'s model was too costly in time and resources so we instead experimented with our hand-crafted *Moving-Box* data, and came up with our own encoder-decoder based models.

6. Conclusion & Future Work

Through our experiments, we are able to demonstrate that the CNN-based encoder-decoder approach is capable of learning the contextual relationship between two frames and capturing the latent features of motions. The model is also able to reconstruct the output frame from the concatenated latent features. From our experiments on the toy *Moving-Box* dataset, we explored a variety of models and ideas. When later trained on the tennis dataset, the model is able to approximate and generate the intermediate frames, with imperfections like noisy background or loss of sharp edges within the output images. Our model may find potential application in video compression and video generation, where the model can be trained for a certain context and generate frames for inference. The sequence-to-sequence approach which takes in a sequence of input images, and generates a series of intermediate frames corresponding to the input sequence, is also promising for enhancing the prediction quality of stronger sequential pattern encoding.

For the next step, we plan to develop alternative ways of utilizing *LSTM* in our sequence-to-sequence model, as our current design was restricted by memory bottleneck. Furthermore, we want to further experiment with our current CNN-based model with *Charbonnier* loss and the "shortcut" technique mentioned in Section 2.2.

The reader is welcome to visit our web demo at `https://dengl11.github.io/CS231N-Project/`.

References

[1] Wikipedia. Motion interpolation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Motion_interpolation`, 2017. [Online; accessed 18-May-2017].

[2] Aleix M Martinez. Lecture slides: Optical flow. [Online; accessed 7-June-2017].

[3] Metkar and Talbar. Performance evaluation of block matching algorithms for video coding. In *Motion Estimation Techniques for Digital Video Coding*, pages pp 13–31. Springer.

[4] Dan Guo and Zhihong Lu. Motion-compensated frame interpolation with weighted motion estimation and hierarchical vector refinement. *Neurocomput.*, 181:76–85.

[5] Haitam Ben Yahia. Frame interpolation using convolutional neural networks on 2d animation. *Bachelor's Thesis from University of Amsterdam*.

[6] Gucan Long, Laurent Kneip, Jose M. Alvarez, and Hongdong Li. Learning image matching by simply watching video. *European Conference on Computer Vision(ECCV)*, 2016.

[7] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[8] S Ren K He, X Zhang and J Sun. Deep residual learning for image recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[9] K Greff R.K. Srivastava and J Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[10] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2439. IEEE, June 2010.

[11] Xunyu Lin, Victor Campos, Xavier Giro-i Nieto, Jordi Torres, and Cristian Canton Ferrer. Disentangling motion, foreground and background features in videos. 2017.

[12] Tennis dataset. `https://lmb.informatik.uni-freiburg.de/resources/datasets/sequences.en.html`.

[13] Wikipedia. Peak signal-to-noise ratio — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio`, 2017. [Online; accessed 8-June-2017].

[14] Wikipedia. Structural similarity — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Structural_similarity`, 2017. [Online; accessed 8-June-2017].

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.