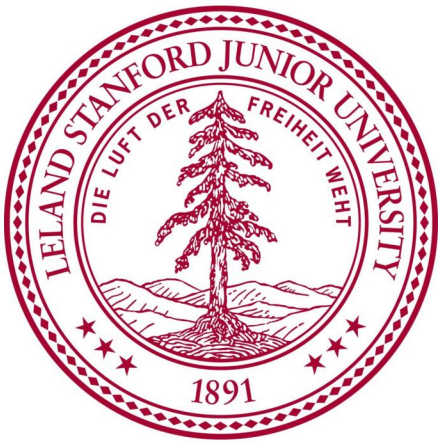

Stanford University

CS 230

Deep Learning

Video Interpolation of Human Motion

Authors:
Max Evans, Sizhu Cheng, Rishabh A. Kothari

June 11, 2018

Contents

Abstract

Video prediction of human motion has many applications namely, autonomous vehicles and predictive surveillance systems to prevent theft. This is a computationally heavy task which requires a lot of data in different background settings. To simplify this, we use video interpolation which also has an additional application in video compression, which can be used for video calling in poor connectivity zones. Previously, optical flow estimation has been used for video processing however, it is computationally expensive and less accurate. The application of deep learning to frame interpolation has been a relatively recent topic. In our final model we show that a Encoder-Decoder CNN+RNN architecture concomitant with the powers of Deep Learning is able to accurately interpolate human motion images and can be run in real time on the test data.

1. Introduction

While traditional machine learning models have not been able to satisfactory tackle video interpolation problems deep learning methods have shown promise and rekindled interest in recent years due to the expressive power of deep models. [2].

Our goal is to do a project which involves the intersection of CNN and RNN, enabling us to interpolate the in between frames of a human motion video. This could be used in reducing the frames needed to successfully store a video, could be used in video compression, in improving resolution and in operational video gathering technology such as satellite imagery. [2].

This is a challenging problem because in reality video interpolation deals with motion and as humans we use both the current location of the object as well as the direction and velocity of movement to make our predictions. For simplicity in our problem we simplify the challenge by using video in which the background is stationary. Moreover we can use the same architecture for video prediction by extrapolating as opposed to interpolating. In fact, video prediction in its full form has been a suggestion for a measure of general intelligence due to the integration of multiple elements and context into a decision. Our first simplification is to make it an interpolation rather than a pure video prediction algorithm.

2. Related Work

Some previous paper has suggested that recurrent convolution neural network (rCNN) works for interpolating frames conditioning the preceding and proceeding frames. Ranzato et al.(2014) mentioned that people can unroll the rCNN on the frame that was used at test time, take the most likely predicted atom as the input and iterate [10]. Mathieu et al. (2016) also provides some insights about generating less video frames by modifying the loss function [8].

Hinton et. al. used an auto-encoder architecture, where the two images are downsampled by a convolution block and upscaled to the original size by a deconvolution. block. [5] Another really interesting piece of work was done by He et. al. in ResNet wherein additional inputs from the deconvolution block was fed into the convolution block and is slightly more robust than the work proposed by Srivastava et. al. [11] specially for imaging purposes [4]

We will implement our own methods based on these existing algorithms, starting with frame interpolation.

3. Approach

3.1 GRU

Layers	Filters	Kernel	Strides	Activation
Conv2d	32	4	2	Relu
Conv2d	32	3	2	Relu
Conv2d	64	2	1	Relu
Conv2d	64	2	1	Relu
GRU taking 9 sequences for 1024x4 features				
Conv2dT	16	2	1	Tanh
Conv2dT	32	4	2	Relu
Conv2dT	32	3	2	Tanh
Conv2dT	16	2	1	Tanh
Conv2dT	1	2	1	Tanh

For our recurrent neural net we simply set a GRU layer in the middle of our encoder decode architecture that takes in 9 encoded images and outputs a single image to be decoded. For our hyper parameter search we changed the number of layers in both encoder and decoder sections, in particular for early testing on CPU. Similarly with e-7 < learning rates < e-3 the model would not learn in the first couple hundred iterations and we stopped the model. For values

e-4 and e-5 the model would learn and we picked e-5 for our final example.

3.2 LSTM

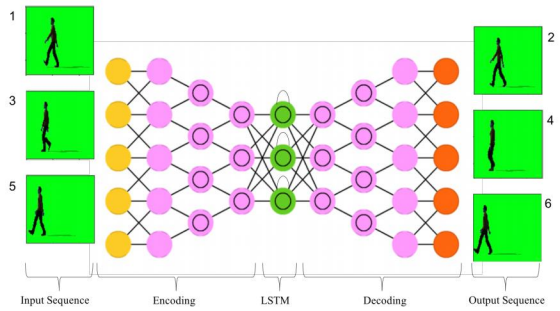

Figure 1: Encoder-LSTM-Decoder Sequence Network

Layers	Filters	Kernel	Strides	Activation
Conv2d	32	4	2	Relu
Conv2d	32	3	2	Relu
Conv2d	64	2	1	Relu
Conv2d	16	2	1	Relu
LSTM taking 9 sequences for 1024x4 features				
Conv2dT	16	2	1	Tanh
Conv2dT	32	4	2	Relu
Conv2dT	32	3	2	Tanh
Conv2dT	16	2	1	Tanh
Conv2dT	1	2	1	Tanh

Human interpolation is based on a state st it has current image related to the state It, we can determine the next state st+1 and the image generated to that state It+1. Hence, we used RNNs as an essential component of the architecture. Since RNNs inherently are fully connected layers and in image processing this would be computationally expensive, we reduce (encode) the image down to a smaller size (batch size, feature size) in order to reduce the number of parameters which must be trained. The LSTM layer is instered instead of the GRU layer as the hyperparameter search has already been done for the encoder-decoder section to prevent overfitting.

This added namely three steps to the architecture:

1. Encoding the image to a smaller size (batch size, feature size)
2. Using LSTM to predict the next state of the flattened smaller image
3. Decoding this flattened image back to the original size

3.3 Encoder-Decoder using Conv3D layers

Instead of just using a single frame respectively before and after the predicted frame, we considered to use various number of frames before and after, and stack them together to feed into a 3D convolutional neural network with similar architecture as the 2D "Encoder" and "Decoder". Specifically, we created two versions of this 3D ConvNet. One model is to use an exact number of "before" and "after" frames, and feed them into a 3D "Encoder" to generate 1 feature for "before" frames and "after" frames respectively, following by using the same 2D Decoder to generate the interpolated image. Another model is to use a group of pairs of features generated by the same 2D "Encoder", and pass them into a 3D "Decoder" and using all of these features to generate the interpolated images. We pass different number of frames into these two models and visualize the performance.

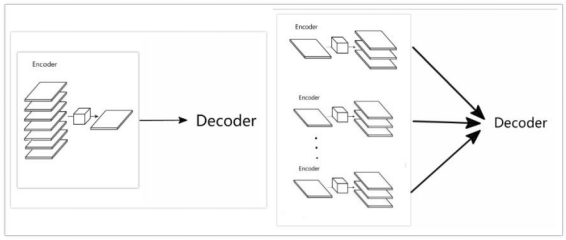

Figure 2: Two Conv3D model's architecture

4. Experiments

4.1 Dataset

4.1.1 Greyscale Videos

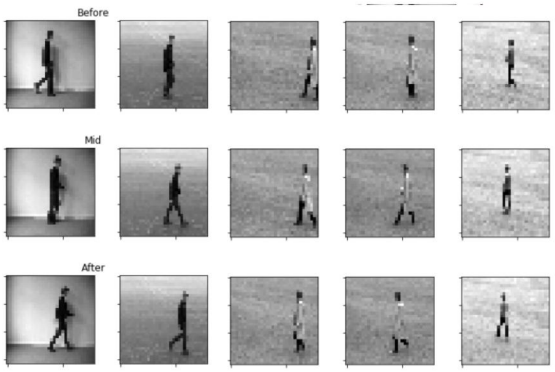

Figure 3: Greyscale sample data

The KTH dataset, [6] of human actions, namely, walking, jogging, running, boxing, hand waving and hand clapping, performed several times by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors. The database contains 2391 sequences. All sequences are over homogeneous backgrounds with a static camera with 25fps frame rate. The sequences were down sampled to the spatial resolution of 160x120 pixels and have a length of four seconds in average. Each four second video was transformed into a frame per second so that a single video become 100 images making 2391 sequences, 239,100 images.

Further, down sampled to a spatial resolution of 32x32 pixels for computing efficiency and normalized into a grey scale of a single channel and normalize from 0-1 by dividing by 255 changing the data type into a float32.

Train data to test data ratio of 90:10 was used in order to strike a balance between computational efficiency and lack of data.

4.1.2 Greenscreen Videos

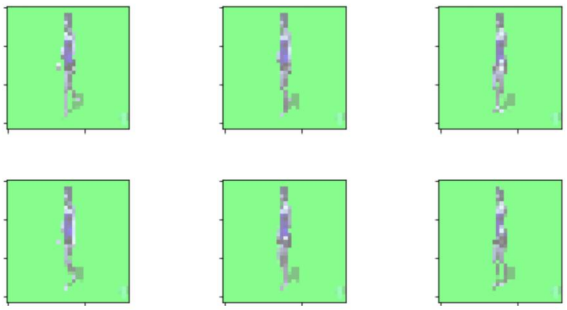

Figure 4: Greenscreen Sample Data

Downloaded 5 sequences from Youtube [3] of animated humans walking on a green screen. The procedure mentioned in 4.1.1 was followed and to get 7,600 images. Spatial resolution of 32x32x3 pixels was used. All images fed into the network were between [0,1]. Test to train ratio is as mentioned in 4.1.1

4.2 Data Augmentation

All walking videos were augmented by mirroring over the horizontal axis doubling the data. Cropping the image defeats the purpose of the interpolation model and color changing is unwarranted as the data set is in grey scale.

4.3 Metrics

4.3.1 L1 vs L2 loss

When interpolating frames, the L2 loss tends to average the features information, thus produces a more blurry image, as suggested by Mathieu et al. 2016 [8]. We tried both L1 and L2 losses functions in training and visualize the generated images. However, we realized that in our case L1 provides a more blurry image than L2, shown in **Figure 5**. So we stick to L2 loss in our experiments. We normalised the L2 loss to give the Mean Square Error (MSE).

$$MSE(y, \hat{y}) = \frac{L2}{H * W * C} \tag{1}$$

where, H, W and C is the height, width and depth of the input frames

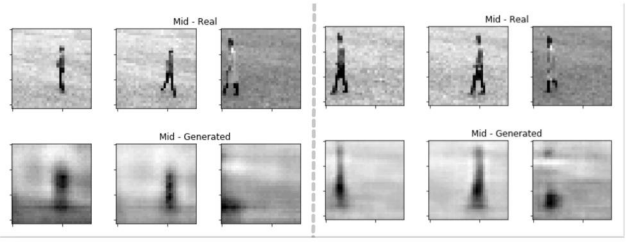

Figure 5: Left:figure generated when using L1 loss function. **Right**: Figure generated when using L2 loss function when using 2 frames-Conv3D

4.3.2 Peak-to-Noise Ratio (PSNR)

Peak signal-to-noise ratio is the log-scale ratio between the maximum possible power, which is 255 in our case, of a signal and the power of corrupting noise that affects the effectiveness of the representation. [9]

$$PSNR(y, \hat{y}) = 10 * log_{10} \frac{255^2}{MSE(y, \hat{y})} \tag{2}$$

4.3.3 Structural Similarity Index (SSIM)

The structural similarity (SSIM) index is a method for predicting the perceived quality of digital images and videos. [12] This value is a closer representation to what humans perceive.

$$SSIM(y, \hat{y}) = \frac{(2\mu_y\mu_{\hat{y}} + c_1)(2\sigma_{y\hat{y}} + c_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + c_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + c_2)} \tag{3}$$

where, μ, σ is the mean and variance of the pixels of the frame and c_1, c_2 are constants to stabilize the weak denominator

4.4 Baseline

The baseline algorithm we used is two CNN models that seek to encode our image, the premise is based on the paper about interpolating the box movement by Li, He and Deng [7]. The basic idea is expressed in **Figure 6**.

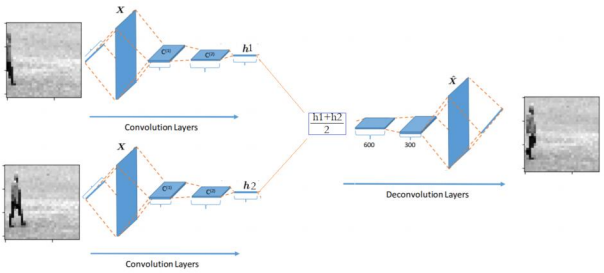

Figure 6: Baseline Architecture

4.5 Results

4.5.1 Conv3D

The frames interpolated and the real images using 3D Encoder and 3D Decoder model are shown in **Figure 7** and **Figure 8** respectively.

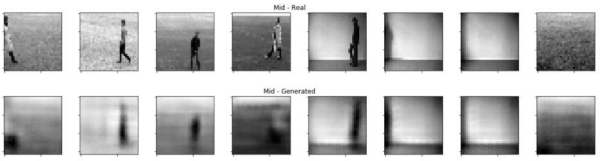

Figure 7: figure generated when convolving 2 "before and after" frames into 3D Encoder

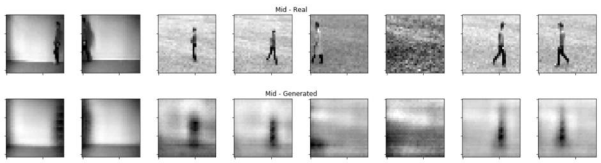

Figure 8: figure generated when convolving 2 "before and after" frames into 3D Decoder

4.5.2 LSTM

The figure generated by LSTM model for grayscale and greenscreen images are shown in **Figure 9**

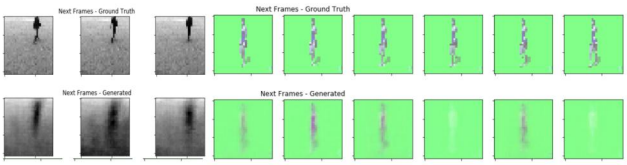

Figure 9: figure generated using LSTM

The result metrics have been compared in the table below

Model	MSE	PNSR	SSIM
Baseline	34.00	-8.60	0.062
Conv3D-3D Encoder	63.99	12.01	0.839
Conv3D-3D Decoder	58.89	12.73	0.872
LSTM-Greyscale	26.24	11.99	0.935
LSTM-Greenscreen	60.76	12.46	0.943

4.6 Hyperparameter Tuning

4.6.1 Learning Rate

For our first hyperparameter tuning we used a log scale to test value between 2e-1 and 2e-5. What we notice is that if the learning rate is too large >2e-2 then the model will not learn and if the learning rate is too small <2e-4 then the model will learn slowly. In the end it suggests a bound for which we can test learning rates and change our model as we babysit it along

Figure 10: Learning Rate Tuning

4.6.2 Number of Layers

During out model testing we changed the number of layers in the encoding the firts number, the number of sequences into our RNN, and the number of layers in our decoding layer. We have tried different number of layers and found that for small changes in the number and layers and sequences we did not

find a substancial change in the result in the way we saw in the learning rate. Because of this we picked a simple structure and focused our energy on other hyper-parameters and finding new data.

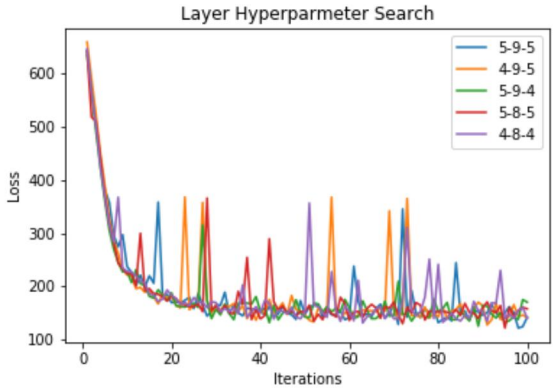

Figure 11: Learning Rate Tuning

4.6.3 Number of frames to convolve into 3D

We have tried different numbers of "before" frames and "after" frames to pass into both Conv3D models. It turns out that two frames are optimal to generate clear mid-frames. The more frames that you pass, the more blurry the generated images are. **Figure 12** shows the frames comparison when using 3D Decoder.

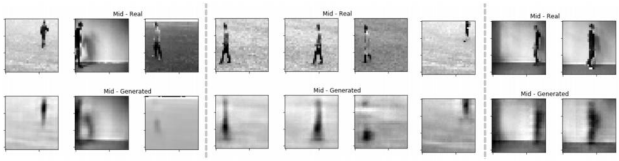

Figure 12: **Left**:figure generated when convolving 1 frame before and after respectively into Decoder Conv3D. **Mid**: when convolving 2 frames before and after respectively. **Right**: when convolving 3 frames before and after respectively

5. Discussion

Our initial motivation was to explore the intersection of RNN and CNN to learn the challenges and methods for tackling time series of images. We found video interpolation and extrapolation as a wonderful application and focused our project on the comparison between a Conv3D where time is stacked as a third dimension vs a mixture of CNN with RNN structures, in this case LSTM and GRU. We found that Conv3D tended to perform better in computational efficiency and time. However, all our models performed better than the baseline.

1. In our Conv3D vs RCNN we have found that while the results are relatively similar with a slight improvement on Conv3D the biggest take away is the reduction of parameters and computational efficiency of Conv3D

2. Against some literature that L1 loss performed better than L2 loss which may be due to the fact that the training set is small and the hyperparameters are not optimal

3. A problem of interpolation is the blurry image and while selecting the appropriate loss function can help, a GAN architecture might allow us to pick and individual element from the distribution rather than an average of the future distribution which ends with a blurry image

6. Conclusion

Through our experiments, we are able to demonstrate that by concomitantly using a CNN and RNN model we are able to generate images in a missing sequence in real time. On training on the greyscale images, the LSTM model gave above baseline results with slight background noise and a little blurriness, however, on using greenscreen images a clearer image was obtained. Additionally, in our model the background is given importance which is necessary when predicting human motion images for autonomous vehicles and surveillance systems. Our present model, may not be viable for the above mentioned applications, however, has potential application for video compression and video generation.

7. Future Work

We would work on the following in the given order:

1. Try a more robust general loss-function namely, the Welsch/Leclerc and the Charbonnier loss functions [1]

2. Use bi-LSTM as gathering information of the next action would definitely help in sharper video interpolation

3. Devise a model for video prediction

Contribution

Sizhu Cheng, Max Evans, Rishabh A. Kothari - We sat together and worked on this project which involved reading work done for similar problems, searching algorithms online, creating the data pipeline for our walking data, optimizing the GRU, LSTM and Conv3D model. We also reasoned out which approach worked and what would be our next steps as we strongly intend to take this project further.

References

[1] Jonathan T. Barron. "A More General Robust Loss Function". In: *CoRR* abs/1701.03077 (2017). arXiv: 1701.03077. URL: http://arxiv.org/abs/1701.03077.

[2] Victor Ge. "A Survey on Deep Video Prediction". In: *learning* 59 (), p. 68.

[3] *Greenscreen walking videos.* https://www.youtube.com/results?search_query=green+screen+walking+videos.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[6] Ivan Laptev. *Recognition of human actions.* http://www.nada.kth.se/cvap/actions/. 2005.

[7] Li, He, and Deng. "CNN-based Encoder-Decoder for Frame Interpolation". In: (2017).

[8] Michael Mathieu, Camille Couprie, and Yann LeCun. "Deep multi-scale video prediction beyond mean square error". In: *arXiv:1511.05440* (2016).

[9] *Peak signal-to-noise Ratio.* https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio. Referenced June 2018.

[10] Ranzato et al. "Video (language) modeling: a baseline for generative models of natural videos". In: *CoRR, abs/1412.6604* (2014).

[11] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks". In: *CoRR* abs/1505.00387 (2015). arXiv: 1505.00387. URL: http://arxiv.org/abs/1505.00387.

[12] *Structural Similarity.* https://en.wikipedia.org/wiki/Structural_similarity. Referenced June 2018.