Video Interpolation of Human Motion

Sizhu Cheng {scheng72} , Rishabh A. Kothari {rishabh6} , Max Evans {mcevans}

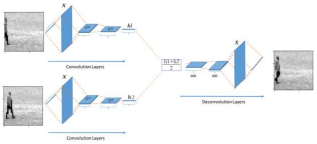
Abstract

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. 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.

Models

Baseline

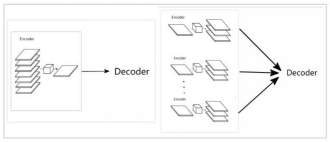
- Perform standardization of image pixels and data augmentation
- Implement Encoder
- Average both images
- Pass encoded vector to Decoder



Conv3D

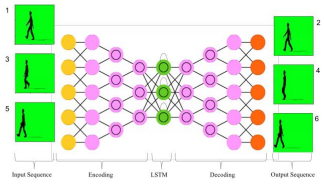
We perform the following procedure on the data [3]:

- Implement 3D Encoder and 3D Decoder
- Convolve frames into 3D Encoder and pass the outputs to 2D Decoder, or
- Convolve frames into 2D Encoder separately and pass all outputs into 3D Decoder

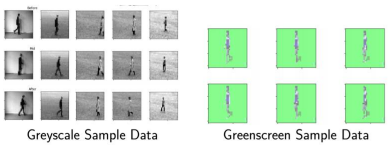


LSTM

- Standardize pixels and create RGB channels.
- Implement an encoder network of 4 layers
- LSTM cell of 9 sequences and 1024x4 features
- Implement a decoder network of 5 layers



Data



Greyscale Sample Data

Greenscreen Sample Data

- Each walking video was augmented by mirroring over the horizontal axis doubling the data
- A four second video was transformed into a frame per second so that a single video become on average 100 images
- Train data to test data ratio of 90:10 was to optimally use our limited dataset

Greyscale Videos: The KTH dataset, [2] of human actions, namely, walking, jogging, running, boxing, hand waving and hand clapping, performed several times by 25 subjects in four different scenarios. 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 a spatial resolution of 32x32

Greenscreen Videos: Downloaded 5 videos from Youtube of animated humans walking on a green screen. Preprocessing gave 76 sequences. All images fed into the network were between [0,1]

Metrics

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

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

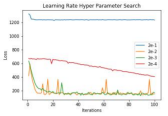
$$SSIM(y, \hat{y}) = \frac{(2\mu_y\mu_{\hat{y}} + c_1)(2\sigma_{y\hat{y}} + c_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + c_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + c_2)}$$

H, W and C is height, width and depth of the frame

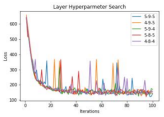
μ, σ is the mean and variance of the pixels of the frame

c_1, c_2 are constants to stabilize the denominator

Hyperparameter Tuning



Learning Rate



Number of layers

Learning Rate: Learning rate >2e-2 the model does not learn and <2e-4 it learns extremely slow. This bound is used to test learning rates as we babysit our model

Number of Layers: Varied the number of layers in the encoding and decoding section and the number of sequences in our RNN. Noticed no substantial variation in the losses. We therefore picked a simple structure to gain computational efficiency

Results

Model	MSE	PSNR	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

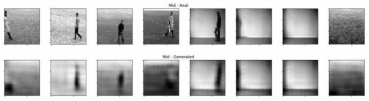


Figure: Convolving 2 "before and after" frames into 3D Encoder

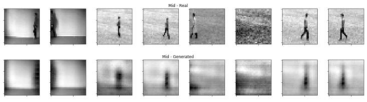


Figure: Convolving 2 "before and after" frames into 3D Decoder

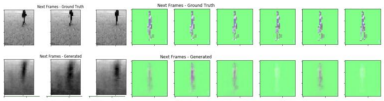


Figure: LSTM : "Left 3 images" Greyscale Data, "Right 6 images" Greenscreen Data

Discussion

- 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
- Much against intuition L2 loss performed better than L1 loss which may be due to the fact that L2 always minimizes to a stable single solution
- 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

Future Work

- Try a more robust general loss-function namely, the Welsch/Leclerc and the Charbonnier loss functions [1] or include the SSIM metric into the loss function
- Use a bi-LSTM as gathering information of the next action should help get sharper images
- Devise a model for video prediction

References

[1] Jonathan T. Barron. "A More General Robust Loss Function". In: *CoRR* (2017). eprint: 1701.03077.

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

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