Convolutional LSTMs for Temporal Video Prediction and Generation

Usman Ashraf
Georgia Institute of Technology
Atlata, GA
uashraf3@gatech.edu

Lane Dalan
Georgia Institute of Technology
Atlanta, GA
lane.dalan@gatech.edu

Tevon Walker
Georgia Institute of Technology
Atlanta, GA
twalker81@gatech.edu

Abstract

In this work, we explore techniques for video generation using LSTM models. Generative modelling techniques such as generative adversarial networks (GAN) and variational autoencoders (VAE) have shown promising results for image generation. However, we are interested in applying similar techniques for the generation and completion of sequences of images i.e. videos. The Variational Autoencoder as well as an unsupervised frame prediction pipeline are evaluated. The variational autoencoder attempts to model a distribution of the video dataset while the frame predictor learned to generate the next frame in a video given the previous frames.

1. Introduction

Convolutional Neural Networks have proven to be a very reliable model for Object Classification [1], Object Detection [2] and Semantic Segmentation [2]. More recently, image captioning has been done by encoding the image into a vector using a CNN and then decoding it using a recurrent neural network (RNN) which preserves the temporal coherence in a sentence [3]. However, in all of this Deep Learning work, the emphasis has been on the analysis of the input whether it be text, image or video. In our project, we explore Variational Auto-Encoders (VAEs) to first reconstruct videos and then create new videos of its own. VAEs brings along a lot of imagination and creativity to the networks available. Moreover, it has many use cases to solve problems in a wide variety of domains. For example, VAEs can make data transfer over the internet very efficient. By encoding a video into a vector and then decoding it at the host device, we can reconstruct the video back while reducing our need for bandwidth. VAEs can also be used to produce new videos which can be used as synthetic data for Machine Learning applications. In addition to the VAE model, we also explored an unsupervised technique in which a convolutional LSTM network predicts the next frame. As we have access to entire video sequences, and we wish to predict one frame given the previous one, the labels in the problem are provided for free in a self-supervised manner. Given one frame of a video sequence, we ask the network to produce the next one. Both of these models are built using PyTorch.

2. Approach

2.1. dataset

We used the UCF101 dataset [4] as our input to the model. This dataset has 13320 videos from 101 action categories. Each category has a varied length of play time and number of videos. To feed the frames into our model, we first had to convert each video into a set of frames. For a given model, the number of frames had to be consistent across all videos to ensure data could be fed in as batch into the network. As most videos had many frames, we reduced the number of frames by taking segments of the video and skipping frames. This also allowed us to have a consistent number of frames in each batch supplied for training. At the end of the data pre-processing, we had a collection of videos each representing a batch with 's' number of frames in it.

2.2. Variational Autoencoder

2.2.1 Summary

A series of images taken from a video are submitted to a VAE in the form of a tensor. The tensor that was fed into the model had the shape: (B, S, C, H, W) where B is the batch size, S is the sequence length, C is the number of channels,

H is height of the image and W is the width of the image. To fully take advantage of the GPU, we put all video image frames from one batch together, changing the shape to (B * S, C, H, W). By placing all video image frames in a batch, as in regular classification training, we avoid having to process video frames separately and achieve a higher speed. After the convolutional features have been calculated, they are reshaped into their individual sequences, (B, S, LSTM$_{in}$), and processed by the LSTM. After the encoding-LSTM has processed the sequence, the output of the encoder is sent to the decoder where the reverse process happens. The z vector is supplied the decoder-LSTM where a vector for each generated image is created, (B, S, LSTM$_{out}$). This is then reshaped so all the deconvolutions can happen in parallel, (B * S, C', H', W') where the ' indicates the dimension in its reduced form. From this reduced form, the video is deconvolved into the original input picture: C, H W and reshaped a final time to end as (B, S, C, H, W). This output video can then be used to create a loss against the input video.

2.2.2 Encoder

Our VAE for image sequence reconstruction was built using a stacked approach where each frame in training set first went through a pre-trained CNN (VGGNet16) [5]. Then the feature vectors we get from the CNN (each having 25088 elements) is fed into the LSTM one by one. The LSTM learns temporal structure in the image sequence and passes what is has learned to each of its cells. After passing each CNN-produced vector through an LSTM, we get a vector of means and standard deviations. These 2 sets of vectors are later sampled from to reconstruct our image by the decoder.

2.2.3 Decoder

The decoder receives as input the sampled latent variable from the learned distributions and passes this through a series of LSTMs. After that, we reshape and upsample LSTM's using bilinear interpolation and then deconvolve it to generate the images in the video sequence. The deconvolution operation blows up the vector sequence into our images. With each image coming out from its Z distribution formed from sampling μ and σ, we are able to stitch them together to form a video.

2.2.4 Loss Function and Training

This sampling of distribution brings the variational part in the VAE. We used the Mean Squared Error (MSE) difference between the reconstructed and the original image as our loss to find the weights of the encoder and decoder.

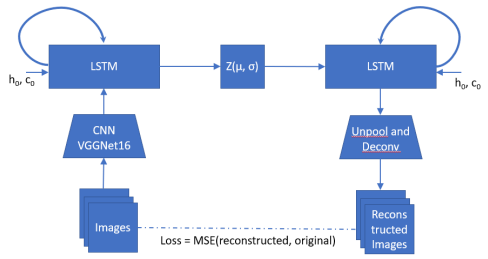

Figure 1. Block diagram of our variational autoencoder architecture

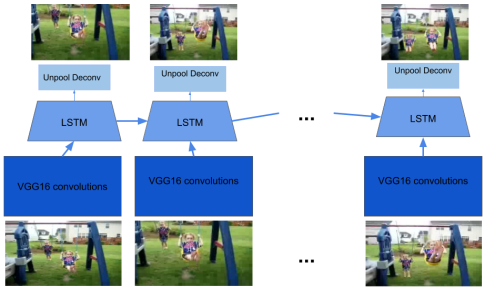

Figure 2. Block diagram of our next frame prediction architecture

2.3. Next Frame Prediction

2.3.1 Architecture

The architecture of this model is similar to that of the autoencoder. To process each frame, we use a pre-trained VGG16 network to compute visual features and pass these sequentially through an LSTM as video frames come in. At each time step, the LSTM produces an output which is reshaped, upscaled, and deconvolved to for a reconstructed image just as is done in the decoder in the VAE. The sequence of reconstructed images forms a sequence of predicted video frames based on the input video.

2.3.2 Loss Function and Training

Just as in the VAE, we use mean squared error to train this next frame predictor. We are able to use a relatively aggressive learning rate of 0.01 and train for 5000 epochs with an Adam optimizer. Every 1000 epochs, we anneal the learning rate by a factor of 10 to avoid loss oscillations. As said before, this training technique does not require hand annotated labels, as the labels are inherent in the data itself.

3. Results

3.1. Variational Autoencoder

The VAE worked well when trained on single videos for short segments (about 6 frames), being able to reconstruct

Figure 3. Here, the variational autoencoder ends up creating videos which are fused iamge sequences of videos from different classes. Perhaps in the latent space variable, the sequential and visual features are blended.

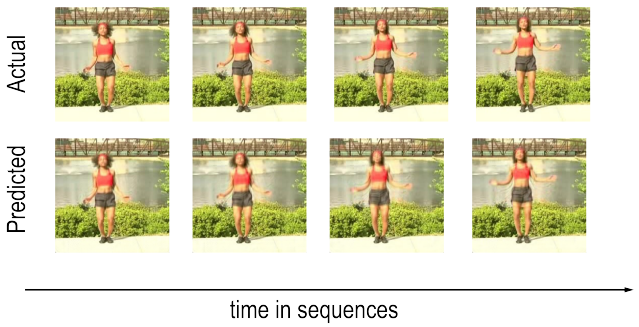

Figure 4. A predicted image sequence shown with the groundtruth image sequence of length 4. Notice the slight blurring artifacts around the woman in the predicted sequence.

reasonable video frames. However as more frames were introduced the results degraded and became more of an average of the input images. This results was exacerbated when multiple videos were fed into the model as the output frames were essentially just the averge of the different videos input frames. Work that was done to try and correct this will be expanded on in the discussion section.

3.2. Next Frame Prediction

After training, some evaluation images were shown to the next frame prediction network. Some example predictions are given in figures 4, 5, and 6.

4. Discussion and Conclusion

4.1. Variational Autoencoder

It is believed that the apparent success of the VAE is largely due to an overfitting of the decoder. For example the decoder architecture was very easily able to take a vector of 1s and reconstruct very good representations of the training images. Specifically the deconvolution kernel size was quiet large when compared to the picture (image was 224 x 224 and kernel was 190 x 190) and the kernel operated

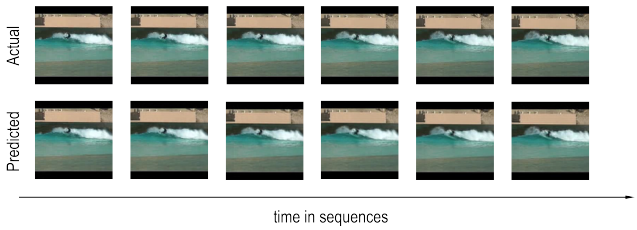

Figure 5. A predicted image sequence shown with the groundtruth image sequence of length 6. The man on the surf board is slightly blurring; however, the sequential information is still preserved in the image sequence.

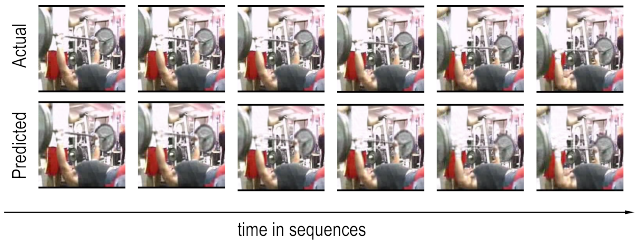

Figure 6. Another predicted image sequence shown with the groundtruth image sequence.

on may layers, 16. This is believed to give it enough expressive power to reconstruct an image well without relying on the LSTM outputs. This issue was elaborated on as attempts to reconstruct multiple videos did not go as smoothly as anticipated. It is believed that so much of the expressive power of the model was contained in the decoder that when an entirely different video produced a different LSTM output, not enough expressiveness was included in this output to be able reconstruct images with significant variation.

Another observed issue is the the reconstruction did a poor job at depicting fine changes, for example a brush moving while applying makeup. Given the large kernel of the deconvoltion this is logical. It would be difficult to get fine detail when summing the contribution of that large of a kernel

To correct both of these issues an alternative model structure was enacted with far more descriptive power allocated to the LSTM components. Also the LSTM output was split into 3 distinct vectors, one for a large scale kernel (as previously used), one as a medium scale kernel, and one as a small scale kernel (for finer detail). This model showed promise however it was not able to produce results as it was too resource intensive for the machine it was run on and ran out of memory. This tells of one of the problems with trying to do this with video; video has a large amount of information in it and is difficult to express all that information without exploding in complexity.

This is believed the direction for future work, a more descriptive LSTM with dedicated information path for high

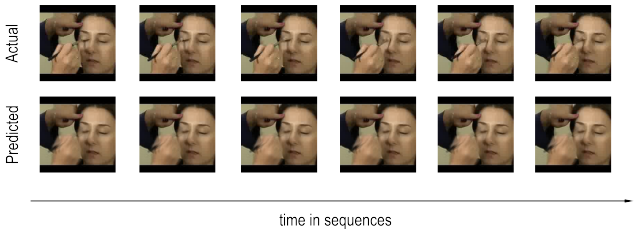

Figure 7. This figure shows a fail case of the next frame predictor. All images in the sequence are identical. This occurs when we ask the network to predict more than 7 frames. Here the 8th - 15th predicted frames are shown.

resolution detail.

4.2. Next Frame Prediction

The next frame prediction network trained well with short sequences. The network reconstructed images with some blur, but the sequence was still present in the images. The LSTM struggled with sequences longer than 6. When trained on these sequences, the network would simply reconstruct the same image at all time steps. We describe this as sequence collapse. We experimented with various hyperparameter settings and selected the best model. We did not consider regular recurrent neural networks here, as we assumed the long term memory dependence would not be supported by this simpler architecture. We experimented with different numbers of skipped frames to mitigate the sequence collapse; however, the model did not seem to learn sequence greater than 6 or 7. An example of the sequence collapse issue is shown in figure 7.

4.3. Future Work and Conclusion

This has been our project on generative modelling of videos. We demonstrated two models that had different methodologies and results. We saw limited success in video generation under constrained sequence lengths. Perhaps we could have tried a discriminator/generator architecture to mimick the GAN style of generative modelling. Another addition which could have helped the images is deep feature perceptual loss. This is where instead of running mse on the input and recontructed images, you compare intermediate feature activations of the input and reconstructed images with a pretrained network.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region & semantic segmentation-aware CNN model. *CoRR*, abs/1505.01749, 2015.

[3] Jiuxiang Gu, Gang Wang, Jianfei Cai, and Tsuhan Chen. An empirical study of language cnn for image captioning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1222–1231, 2017.

[4] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.