Predicting the immediate future with Recurrent Neural Networks: Pre-training and Applications

Brandon Richardson

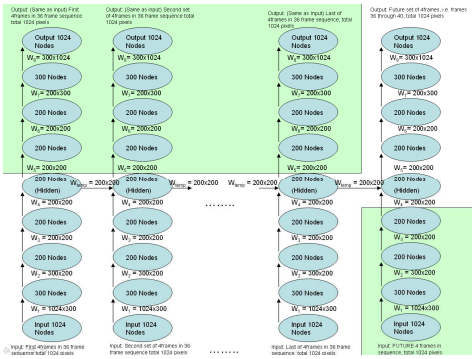
December 16, 2011

Introduction

Research preformed from the last 5 years has shown that the process of training deep networks can be improved by carrying out the learning process in smaller phases called “pre-training.” Rather than using back propagation to train the whole network, we could just train several small modules separately and then when assembled fine tune the entire network. This idea has only been applied to RNN in recent Stanford research. The current research done at Stanford has divided the training into three phases. First, the input to Hidden Unit layer connections is trained using an auto-encoding objective. Second, the input to the Hidden Unit layer connections are held fixed and the temporal connections are trained with short term memory. And third, the whole network is fine-tuned using the main objective function. The previous research shows that this new “pre-training” gives a better reconstruction error then previous methods, in some cases even out performing the best hand-engineered feature selection. This project is being done in conjunction with Quoc Le’s research here at Stanford. The main objective of this project is to apply the “pre-training” RNN algorithm to audio prediction.

The Algorithm Architecture:

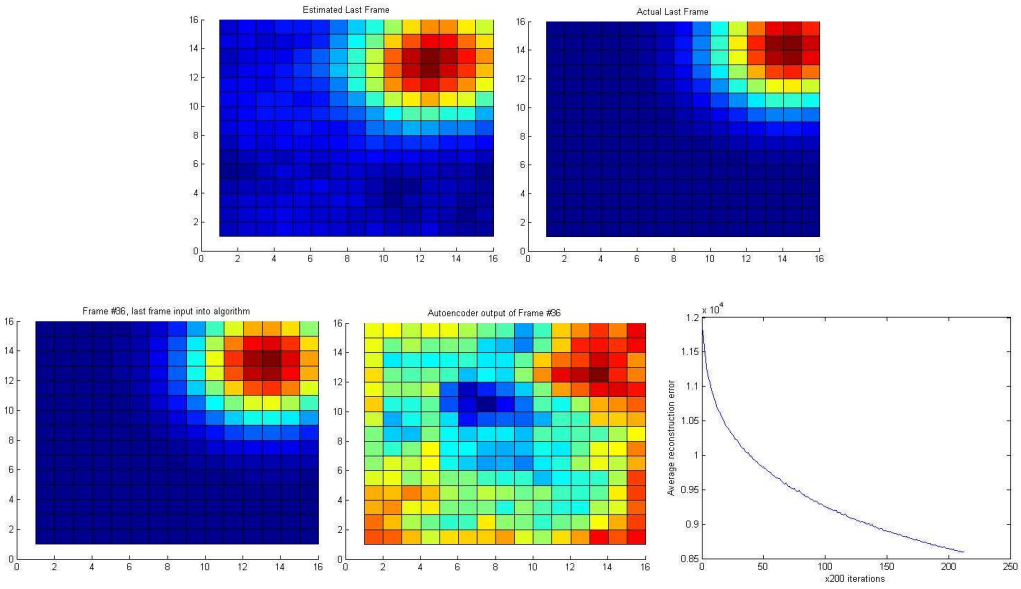
The RNN algorithm is best described by referring to the figure below. This figure shows a 4 layer network, with [300 200 200 200] nodes per layer where the input and output are composed of 1024 values. The vertical connections are autoencoders and the horizontal connections are the temporal connections. The nodes shown in green are only used during pre-training of the autoencoders.



Initial Testing:

To verify the functionality of the RNN, and reinforce my understanding of the algorithms architecture I began by training an RNN using the 36000 video training examples Quoc had previously used to obtain results for his current research. This previous research was attempting to predict the last 4 frames in a video from the previous 36 frames.

The images shown below were generated using a 4 layer RNN with each interior layer containing 400 nodes. These parameters were chosen to closely match the research that had been previous completed using this frame prediction algorithm, and training data. This model took about a week and half to process it was stopped before it fully converged. The images shown below are the predictions made by this RNN on a test set I designed. The test set is a simple ball of light moving from the bottom left corner of the image to the top right hand corner of the image. Top left, shows the RNN's prediction of the 40th frame given the previous 36 frames. Top right, shows the actual 40th frame of the video. Bottom left, shows the actual 36th frame, the last frame given as an input to the RNN. Bottom middle, shows the auto-encoder output of the 36th frame. From these images it's easy to see that the RNN's prediction of the 40th frame closely resembles 36th frame, the last frame given as an input to the RNN. Therefore, the model appears to poorly capture the temporal effect. Also interesting to note is that the auto-encoder output for the 36th frame scarcely resembles the actual 36th frame. This leads me to believe that the final objective function optimization is drastically changing the pre-trained auto-encoder parameters, such that the decode parameters properly output a prediction only when the temporal connections are used.

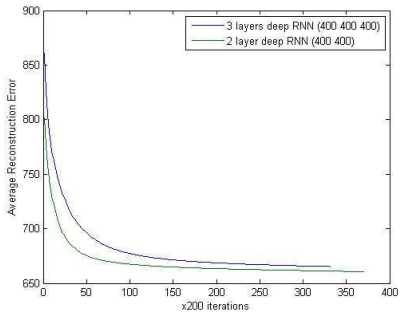


Application of RNN to Audio:

After achieving reasonable results from the video data I applied the algorithm to audio data. My goal was to predict the last second of a 10 second clip of music. The training and test data is made up of 3000 audio loops obtained from the Apple program Soundtrack. The data clips from Soundtrack where chosen since each loop in Soundtrack is on average 10 seconds in length and repeats a particular beat. Conveniently the clips are also made to be added together to create a full recording. This enabled me to created 77,000 unique training examples by adding the tracks together in random combinations. I held 1000 of the original 3000 examples aside to be used as the test set. Unfortunately, 10 seconds of data at 44100 samples/second is too much data to process efficiently with the algorithm, therefore, I compressed all the data to 1024 samples/second. This compression reduces the complexity of the data while keeping the overall beat; the difference is hardly noticeable when played back on normal speakers. Each audio loop is sampled at 16bit resolution which is then normalize for use with the algorithm.

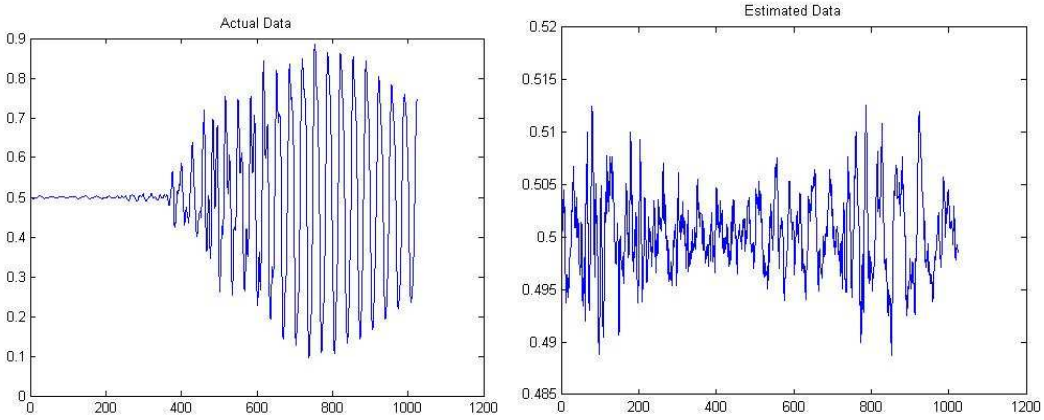
The primary difference between the audio data and the video data is that each temporal step of the video data contains 4 frames of data, where each frame of the data contains 16x16 samples from the same instance in time. Each temporal step of the audio data contains 1 second, or 1024 samples of audio, where each sample has been obtained at a different instance in time.

Initially I attempted to fit the RNN model using only the 2000 training tracks supplied by Soundtrack. The plot below shows the average reconstruction error for two different networks fit to the training data. Oddly the 2 layer, 400 nodes per layer, RNN achieved a lower reconstruction error then the 3 layer RNN.

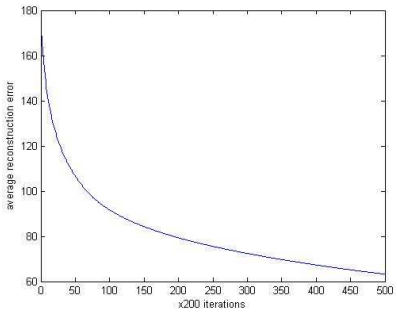


When applied to the 1000 example test set the average reconstruction error was nearly identical to the training set, implying a good fit. Unfortunately, when listening to the estimated data it is clear the model has converged to a locally optimal solution rather then the global solution causing the model to poorly reproduce the final second of audio. This occurs for both the examples in the test set and the examples in the training set. The plots below show the estimated data and the actual data for one training set example,

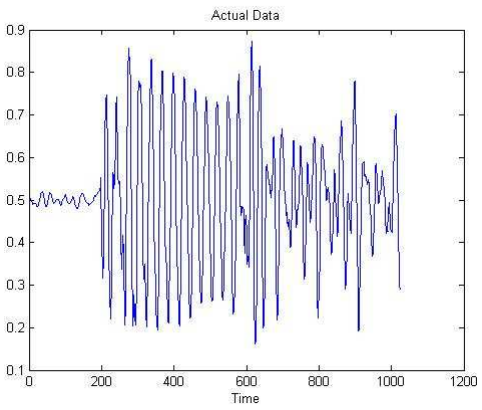
probably the best one out of the many examples I viewed by hand.

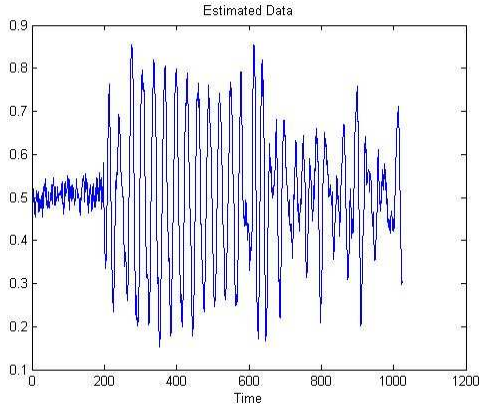


Rerunning the 3 layer, 400 nodes per layer, RNN model with the 77,000 training examples created by combining random combinations of the original 2000 training tracks supplied by Soundtrack resulted in a considerably better average reconstruction error as seen in the plot below.

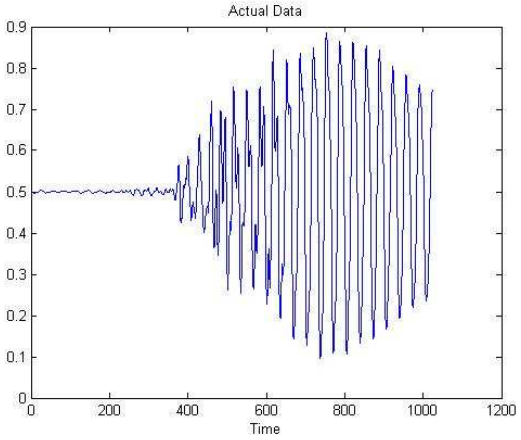


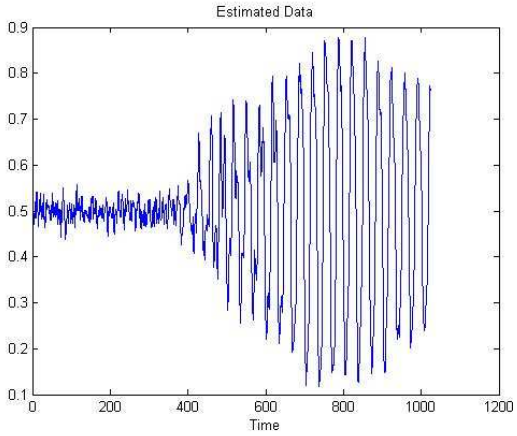
Some of the audio samples the model fails to predict entirely. However, after listening to a few of the samples that the algorithm best fits and then listening to a few of the ones that the algorithm fails to predict. It appears that the more times the beat repeats in the first 9 seconds the better it can predict the last second. The samples where the algorithm completely fails seem to correspond to samples where the beat never fully repeats in the 10 second interval. The two graphs below shows the actual last second of data, and the predicted last second of data for a “test-set” sample where the beat repeated a total of 2 times in the 10 second interval. The estimated last second is rather noisy, but appears to accurately reproduce the subtly in the beat. Filtering out the high frequency noise, and playing the clip back with the estimated last second produces a near match to the actual data.





Shown below is this models fit to the training data example that the previous model failed to fit.





Conclusion:

This project demonstrated how an RNN using pre-training could be effective in predicting audio. These results show that an RNN can successfully predict the last second of audio data from the previous 9 seconds. However, it was also discovered that a large number of training examples are necessary to prevent the algorithm from converging to local minima. Further research needs to be completed testing different numbers of layers and different node sizes to find an optimal RNN for this type of audio prediction. The loops from Soundtrack only contained audio from around 500 different instruments. Therefore, it would be interesting to try predicting vocal audio clips, or even to test the algorithms sensitivity to the different types of instruments.

References:

Quoc, Le. “Predicting the immediate future with Recurrent Neural Networks: Pretraining and Applications,” NIPS, 2011.