Programming Assignment 4: Human Pose Estimation
Hayleigh Sanders, RIN# 661195735

Introduction

The following project is an implementation of a CNN+LSTM architecture to predict human joint positions from a series of sequential image frames taken from the Youtube pose dataset. The model was able to predict joint positions relatively accurately, within 10 pixels of the ground truth coordinates.

Theory and Setups

CNN and RNN

This model is composed of a convolutional neural network and a recurrent neural network. The CNN provides spatial information to the model about the location of the joint positions through convolution and performs a feature extraction step to determine the joint positions. The LSTM (long short term memory) is a variant of a recurrent neural network (RNN) which provides temporal information about the prediction of features across time steps.

LSTM cell gate equations

LSTM cells were used for this implementation due to their ability to retain useful information about the weight gradients over a large number of iterations. This is achieved through the addition of a memory gate between cells along with the input, output and hidden state gates. A single cell is shown below. In a RNN with LSTM cells, multiple cells are strung together with the outputs of the previous cell gates fed to the next corresponding cell gates.

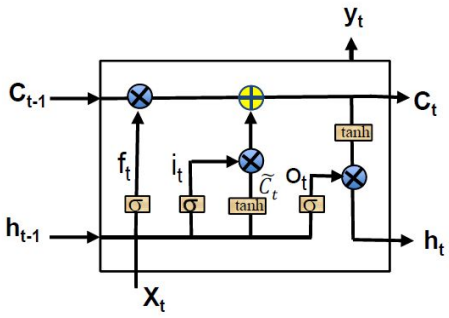

Figure 1: LSTM cell (image credit: “Recurrent Neural Networks”, Qiang Ji)

The relationship between these gates is summarized by the following:

Forget gate: $f_t = \sigma\left(W^{hf} h_{t-1} + W^{xf} x_t + W^{f}{}_0 \right)$ (1)

Memory gate: $i_t = \sigma\left(W^{hi} h_{t-1} + W^{xi} X_t + W^{i}{}_0 \right)$ (2)

Output gate: $o_t = \sigma\left(W^{ho} h_{t-1} + W^{xo} X_t + W^{o}{}_0 \right)$ (3)

Where sigma is the activation function.

LSTM Back propagation

Using initial inputs to the LSTM gate equations described above, an initial estimate $\hat{y}$ can be found through forward propagation by chaining each output to its respective input until a result is found at the output layer. The purpose of back propagation is to then feed this output $\hat{y}$ backwards through the derivatives of these gate equations to produce weight gradients for ∇y_t, ∇h_t and ∇C_t. The weight gradients are aggregated over time and then used to update the weights through gradient descent with the following:

$$W(k) = W(k-1) - \eta \nabla W \quad (4)$$

Model Architecture and Experimental Setting

The training dataset consisted of 5000 sequences of 10 images depicting a human action. Each RGB image in the sequence formed a 64x64x3 pixel ndarray of type uint8. The training labels for each image consisted of seven 2D joint position coordinates: Head, Right Wrist, Left Wrist, Right Elbow, Left Elbow, Right Shoulder, Left Shoulder. Each sequence label had the shape (10, 7, 2) corresponding to a set of ground truth coordinates for each image in the sequence.

Each image was normalized by subtracting the mean and dividing by the standard deviation. Next, the training dataset was shuffled randomly and for each training iteration, a batch of 10 sequences was fed into the CNN. A convolutional filter of size [3,3,3,16] was applied to each sequence with stride 1 and zero padding, followed by a ReLu activation layer. The result was then flattened, and the resulting 10x10x65536 tensor was fed into the RNN. This tensor corresponds to 10 feature maps of size 65536 for 10 image sequences. For each sequence, the 10 feature maps are fed into the 10 LSTM cell inputs of the RNN. A weight matrix of shape (100, 14) and bias matrix of shape (14) were initialized to small random values and trained by the forward/backward propagation process described above for a RNN over the training iterations. The shape of the weights correspond to 100 hidden nodes in the model and 14 unique x and y coordinates.

Because the estimation of these x,y positions is a regression problem, the cost was defined as the mean squared loss given by,

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N} \left(y - \hat{y}_i \right)^2 \quad (5)$$

This cost value was computed from the difference between the predicted x,y positions $\hat{y}$ and their ground truths y. This cost value was then minimized with AdamOptimizer.

Hyper-Parameters

The batch size for training was chosen as 10 because that was the maximum amount of images that could be processed by the CNN and RNN at one time without slowing down the training process significantly. The learning rate was determined through trial and error to be .01 by observing how fast the loss converged and whether or not it oscillated by a significant amount once convergence was reached. Epochs were chosen to be 50 by testing which values caused the model to minimize the pixel distance error between the predicted values and ground truth to below 10 in a reasonable amount of time. The training data subset was chosen to be 5000 out of 7000 total sequences so that enough samples could be used in the testing dataset. The number of hidden nodes for the RNN was chosen to be 100 by trial and error. The filter size for the CNN was chosen to be [3,3,3,16] as opposed to a larger 5x5 filter due to the relatively small size of the training images. Likewise, the stride of 1 and zero padding was chosen for the same reason. ReLu was set as the activation function for the activation layer after convolution to add some nonlinearities.

Experimental Results

The average pixel distance error vs iterations for training and testing data is plotted below.

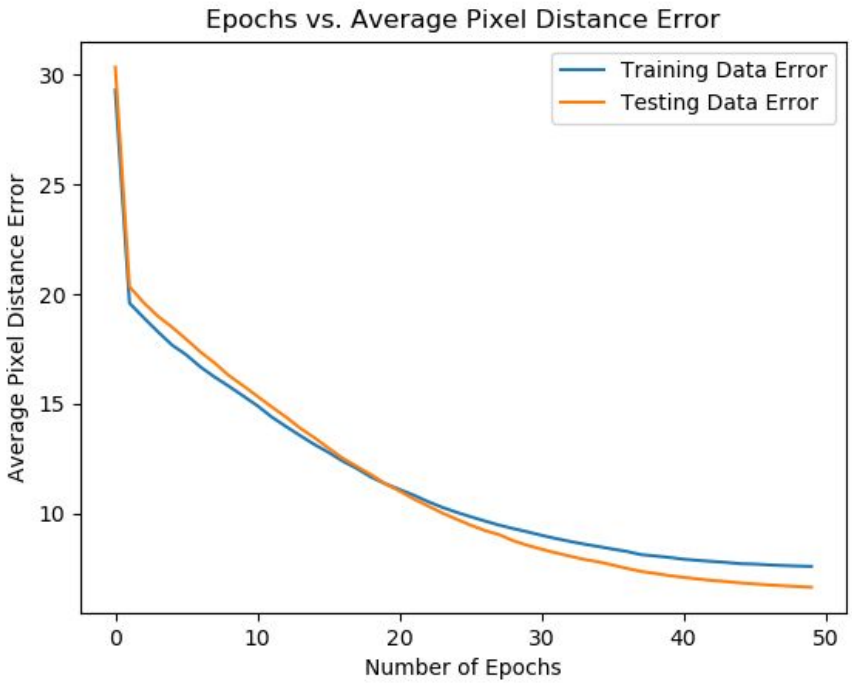

Figure 2: Average pixel distance error vs training epochs for training and testing data

The average pixel distance error for each joint was calculated based on the final model. Each class from 0 to 6 corresponds to Head, Right Wrist, Left Wrist, Right Elbow, Left Elbow, Right Shoulder, Left Shoulder.

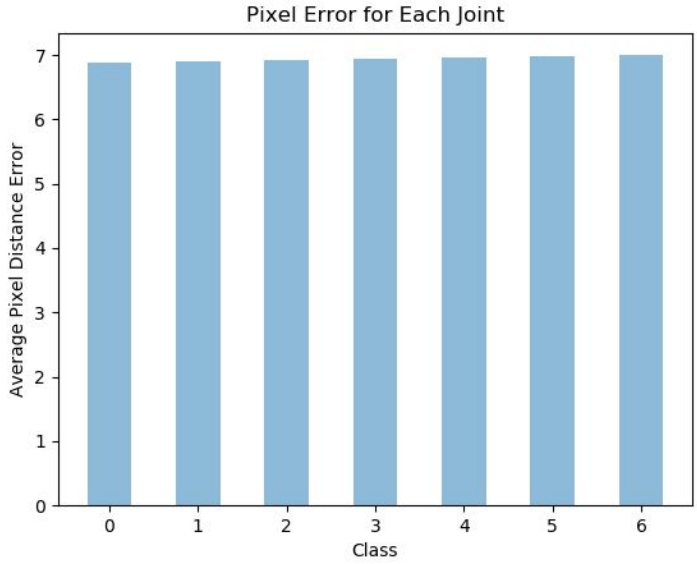

Figure 3: Average pixel distance error by class for the final model

The percentage of predicted coordinates within varying pixel distances of the ground truth values is shown below:

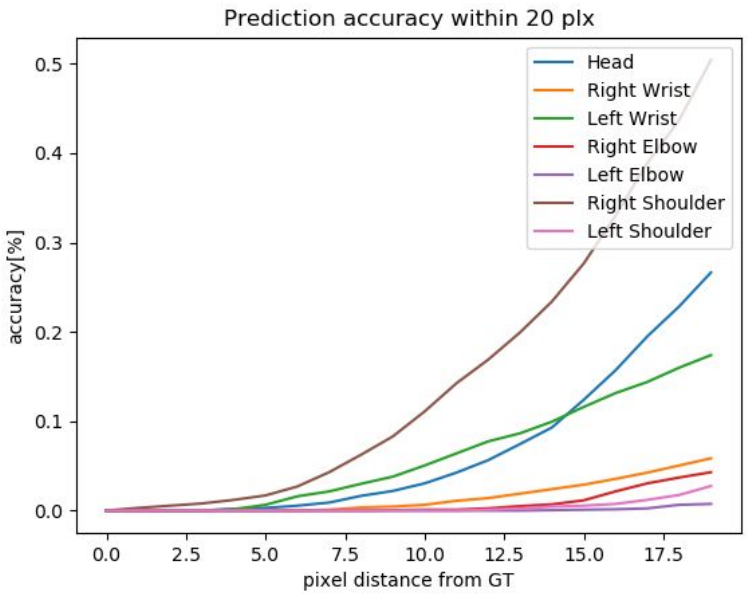

Figure 4: Prediction Accuracy

The predicted joint positions are plotted below in red, with the ground truth locations plotted in blue, for three randomly selected images.

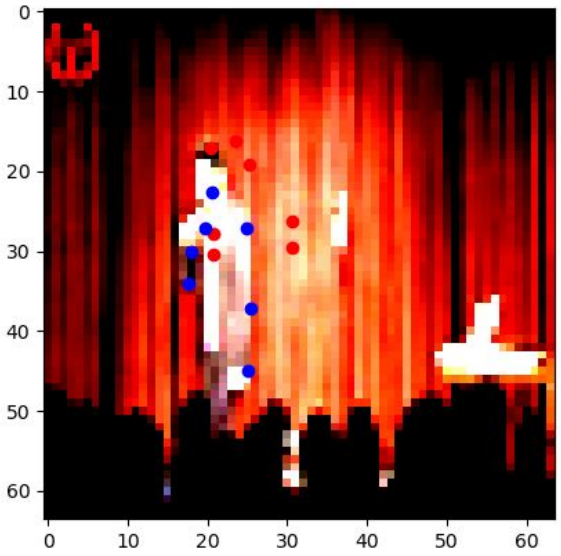

Figure 5: Predicted positions (red) vs ground truth (blue)

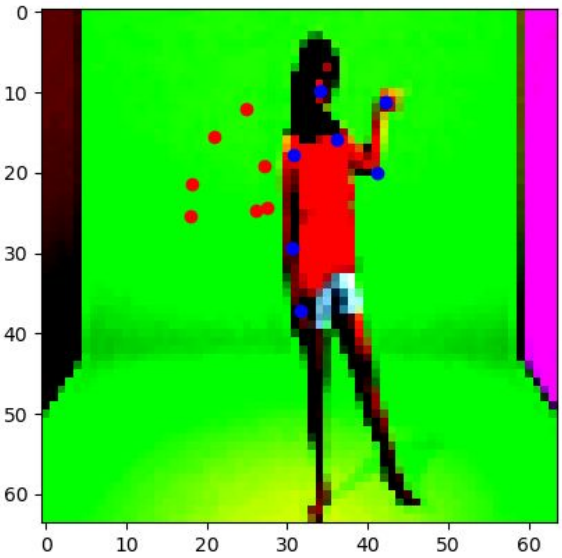

Figure 6: Predicted positions (red) vs ground truth (blue)

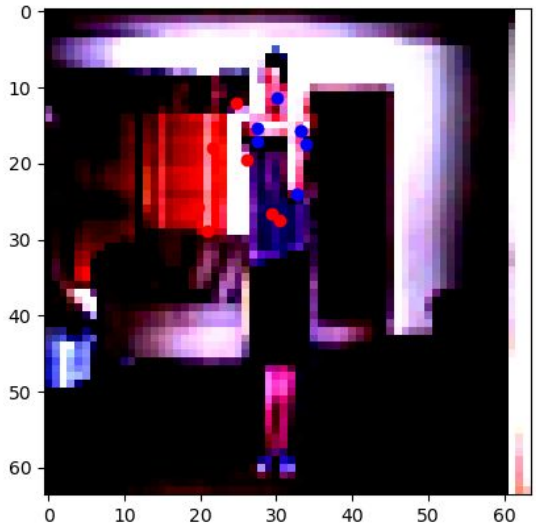

Figure 7: Predicted positions (red) vs ground truth (blue)

Conclusion

This project demonstrated the potential of a recurrent neural network with LSTM cells and a CNN front end for feature extraction to predict data with both a spatial and temporal component. By the end of the training iterations, the average pixel distance error between the predicted coordinates and the ground truth was approximately 6 for testing data and 5.5 for training data, so the model was relatively successful in predicting the joint positions.