**Topic:**

One of the motivations for this project was to choose an action that a Deep Neural Network could detect from a video. The action that was assigned to me was Detecting a “Stretching Body”. So, the task is to design a classifier using Deep Neural Networks architecture to detect and classify a stretching body.

**Motivation:**

In our daily lives we stretch before we hit the gym or start our morning schedule. One of the motivations for this topic could be that if we consider a camera fitted in a smart home where it could detect if a person is stretching or not, by this we could be able to record the number of times a person performed stretching in a day. With this we can set up a health monitor that could detect if the person performed stretching enough number of times or not.

**Problem Statement:**

The challenge here is to detect the activity (Stretching) through the videos. So, I would start with using different type of Neural Network architectures (Including CNN, LSTM, 3D CNN) to classify whether the "Stretching" is being performed in the video or not. The problem would be a binary class classification model where there would be two labels, "Stretching" and "No Stretching".

**Data Generation and Gathering:**

The dataset for stretching body was not available online. Thus, I collected some videos of people performing different type of stretches from YouTube. To generalize the data well enough, I looked for videos of same stretching manoeuvres performed at different camera angles and from different people. Also, I tried to include videos where a person is dancing or lying on the floor just to make model learn that any manoeuvre that seems close to stretching but is not actually a stretching manoeuvre.

I downloaded about 40 different videos which had a many stretching manoeuvres performed in them at different times. Mostly videos were about 10 to 25 min long. So, I used "Moviemaker Tool" to cut long videos into short videos to prepare the training dataset. For example, I had a video in which a person performed 10 stretches and I cut the whole video into 10 different shorter videos of stretching body. Any part of the video in between those 10 stretches had served as candidate videos for non-stretching data. Thus, in total I had about 230 videos of Stretching and Non-Stretching body, out of which 130 are the videos falling in “Stretching Class” and 100 fall in “Non-Stretching Class”.

Most of the videos were pretty short ranging from 5s to 15s, and any single video contained just one stretching manoeuvre or didn’t contain it at all.

**Train and Test Split:**

I used random shuffling of videos to split them into train and test dataset. So out of total 230 video 170 are reserved as training dataset which would be used to train the model. Whereas there are 60 videos which are reserved as test set and have 30 videos in each class.

I have used binary labelling, where stretching in the video is labelled as "1" and no stretching as "0".

**Input and Output Data Structure:**

Most of the videos I had in the training and test set were of length ranging from 5 to 15s. So, as in “Stretching” the body landmarks (leg, elbow, head etc) moves through different frames and considering the fact that I was more inclined to using less complex model architectures, I extracted videos at 1 frame per second. From each video I extracted frames at 1fps, dropped the colour channels, converted them into greyscale images, resized them to (250,250) and saved them in an array. So, the input tensor that would be fed to model would look like (Samples, 10, 250, 250, 1). Whereas, due to binary class classification the output tensor shape was (Samples,1).

**Model Architectures:**

I have about 60 videos in the test set, out of which 30 belong to “Stretching” and the rest of the 30 videos belong to the “No-Stretching” class. If I were to randomly guess, I could achieve an accuracy of 50% without using any sophisticated model. So, 50% is the baseline that we have to beat.

For this first phase of the project my idea was to try different architectures available with keeping the subsequent model complexity to low-normal range, just to see which architecture performs better than the other.

**CNN and LSTM Architecture:**

Convolutional Neural Networks are a powerful tool for extracting features from images and are widely used in image processing. If we look at the videos, they are just a set of multiple images recording at different time steps. Thus, a video has both spatial and temporal dimensions. Thus, use of convolutional layers only might not be suffice in dealing with video processing, especially when the task is to recognise the activity in video, i.e. "Stretching Body" in our case.

Thus, I researched about some of the architectures available in Keras, and I came across the "Time Distributed Layer" module in Keras. So, what it does is that it applies a layer into "N" time dimensions given. For example, if we have 10 frames in a video. And we use a Convolution Layer to extract features from images, "Time Distributed Layer" would consider 10 frames as 10 timesteps and apply the convolutional layer onto those 10 frames keeping the temporal aspect attached.

Bottom line is that Time Distributed Layer applied with Convolutional layer would learn different aspects temporally, e.g. it would detect a person/body in the first frame and then would try to learn how the orientation of the body changes in the succeeding frames.

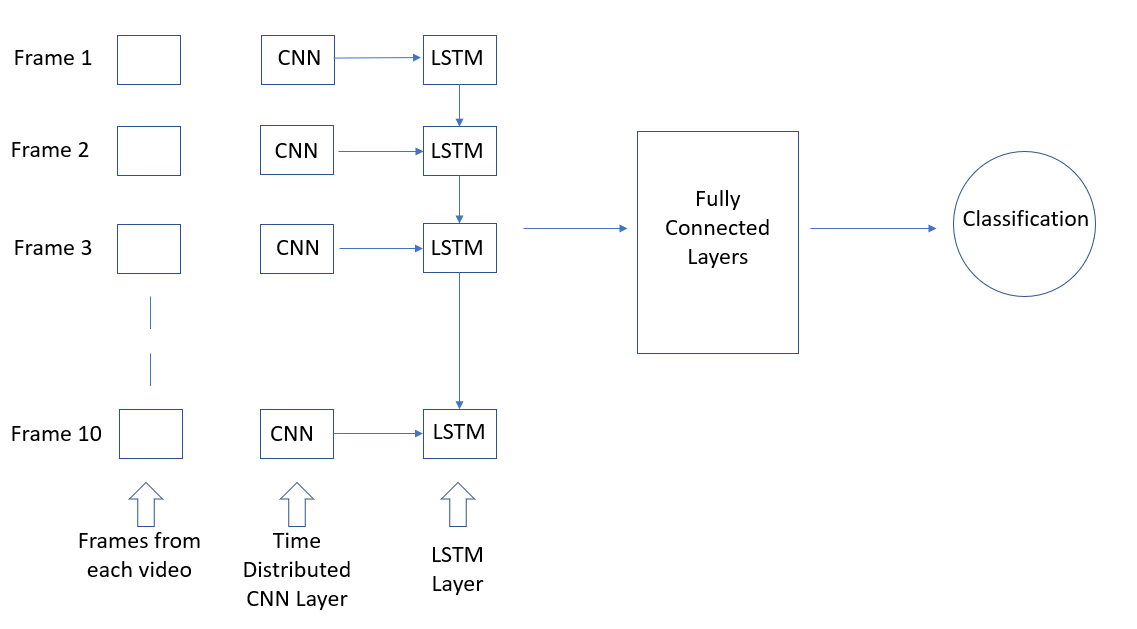


Figure 1: Time Distributed CNN and LSTM Architecture

**Results:**

So, I trained the above explained architecture with two configurations, one pretty simple, while other a bit more complex. Model Architectures are defined in the Github File shared.

* **First Model:**

First model contained 4 Time Distributed Convolutional Layers, 2 Max Pooling Layers followed by an LSTM Layer for Sequence Learning and 3 Fully Connected Dense Layers with 3rd layer as the Output Layer.

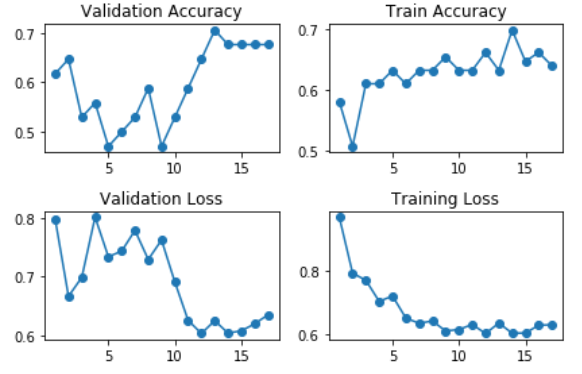
I achieved a test accuracy of 73% from this architecture. Trained the model for 20 epochs, with the best batch size of 10, and with Keras Callbacks to stop model from further training if the performance does not increase after 4th consecutive epoch. Model ran for 9 epochs and here are the graphs explaining train, validation loss and accuracy.



* **Second Model:**

Second model was a bit more complex than the first one as it contained a time distributed convolutional layers with 4 max pooling layers in between, followed by an LSTM layer with 64 units and 4 fully connected dense layers with last layer as the output layer.

I achieved a test accuracy of 66% from this architecture. Trained the model for 20 epochs, with the best batch size of 10, and with Keras Callbacks to stop model from further training if the performance does not increase after 4th consecutive epoch. Model ran for 17 epochs and here are the graphs explaining train, validation loss and accuracy.



**3D CNN Architecture:**

3-D CNN are a type of convolutional neural network that are extensively used in action detection through a moving array of images or videos. 3-D CNN applies the convolution both spatially and temporally at the same time. The previous architecture of Time Distributed CNN applies convolution operation separately on different frames of a video, such that the weights are learned separately but in a temporal fashion, while in 3-D CNN, the weights are learned in a combined way as the convolution is applied on all the frames together.

3-D CNN are computationally very expensive on top of that they are complex as well, so a lot of room to overfit until we have lots of data. Thus, I would start with some basic 3-D CNN and see how it works.

In the model architecture, I just defined two convolutional layers in the model and it has about 30 million parameters to learn.

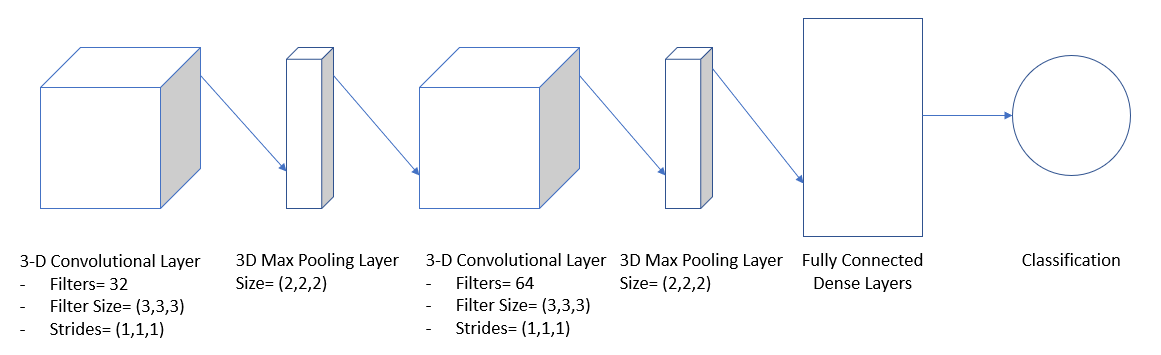
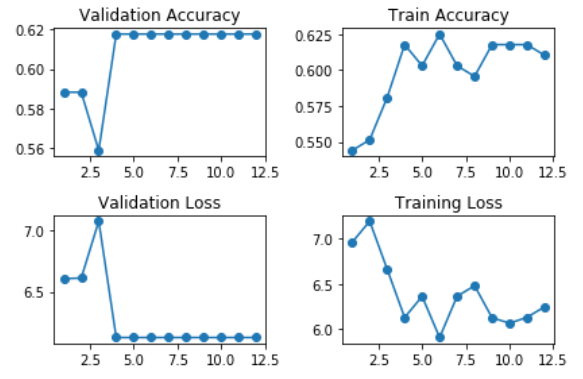


Figure 2: 3D CNN Architecture

**Results:**

I trained the above model architecture using 20 epochs, best batch size=10, and the best optimizer of “Adagrad”. Model has Keras Callbacks defined where it stopped at 12th epoch and validation accuracy has hit the bottleneck at 61%. The test accuracy achieved is 65%, which seems fine. But not as good as we got from the CNN+LSTM architecture. Thus, it seems like Time Distributed CNN+LSTM architecture has outer performed this 3D CNN architecture.



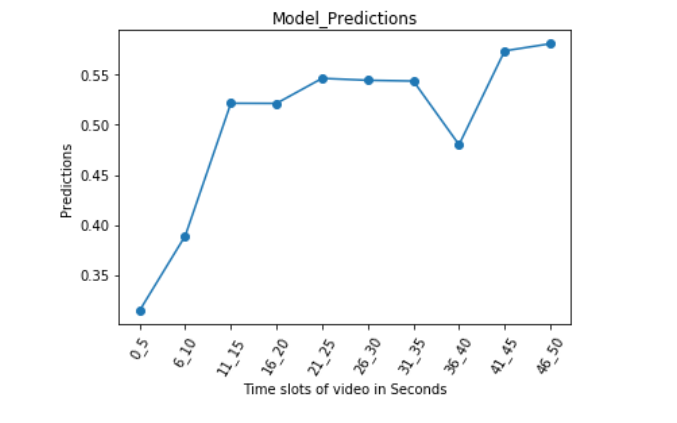
**Real Time Prediction on Videos:**

To see how the model performs on one video where there is stretching in some part of it and then there is no stretching in the other parts, I have created some functions that would help us do the real time predictions on those long videos. I have made these functions keeping in mind that each video is at least 50 seconds long. So, if you would want to test the model on your own videos, please make sure that they are at least 50s long.

For Real Time Prediction, I have created two functions that would first help with breaking long video into short clips of 5 seconds each and then make predictions on all those 5 sub-clips of a clip. Then I have plotted the predictions for each video along with time frames on the x-axis and predictions on y-axis.

The explanation of the functions and their parameters have been added on the Github along with the code itself. I have also created a demo video on how to run the model and test it on these 5 videos. I used the best two models I had trained earlier and doing a weighed average of their predictions for each of the video to make predictions in this case.

Here I would just show a glimpse of how the model make predictions on a video of length 50 seconds. In the plot below x-axis contains the time frame, e.g. “0-5s” means the prediction made for the time frame 0-5 seconds. Where, y-axis contains the predictions. Below is the graph of the predictions for 1st Test Video.



So, we can see that for first ten seconds the predictions are less than 0.5 indicating there is no stretching, whereas after that till 35s there is stretching in the video, then for a short interval from 36-40s there is no stretching and then the probability goes up again showing that there is stretching in the last 10 seconds.

**Findings:**

There are a few problems with the prediction accuracy of the model, as the model already gives high probability when it detects a human and when they start moving. For the first 3 test videos, model is working with almost 100% accuracy, such that it is detecting where the body is stretching and where it is not. But for the last 2 videos it is confusing random movement with stretching. About 65% of the times model is predicting the stretching correctly. To my understanding this is because I need to feed model some more data, such that it can distinguish between simple movement (Random Movement) and Stretching. In Future I plan to add more data and train the same models to see if it improves the accuracy.

**Future Work and Steps:**

The best model with this work has given me 73% accuracy. I plan to first use some predefined complex neural network architecture bases like “VGG” and “Inception”, connect them with the architectures defined above and try to see if the performance improves.

I plan to expand my dataset by 100 more videos and see if more data would help increase the accuracy. Finally, I have a plan to experiment with using “OpenPose” to generate body landmarks and see if those body landmarks used with basic architectures defined above can help improve the accuracy of the model.

**Demo Video File:**

I have included the demo video file, the “.ipynb” file containing my code and explanations. Also, I have provided the link to my Github repository where I have put all the material related to this project. The video file named as “Demo\_Video\_File” shows all the methods on how to run the model and make real time predictions. I have also included the zip file containing the 5 videos I have tested my model on for real time predictions.