Action Recognition in Video Using LSTMs and SVM

Feature extraction:-

- As mentioned the shapes of the images are different to that of the input for VGG architecture and hence we used cropping technique where we take 5 cropped images from the single images.
- Then we take the features of these 5 images from the VGG architecture's linear layer which gives 4096 dim features for each of the image.
- We then the average of these images to convert these feature size to a sing 4096 dim.
- That means for single image we get 4096 and there are 25 such images for single video.
- So, we get 25*4096 feature matrix for a single video.
- These feature matrix is then stores into .mat files using io.savemat from scipy library.

Time taken for feature extraction and saving them into ".mat" files took **90m 38s. (Done in CPU)**

Train and Test data:-

- Once we save the features files in the .mat format as explained above.
- We read these files for train and test splitting them according to the "videos_labels_subsets.txt"
- Now we have the train and test data split into two different parts and we only consider first 15 classes for training and testing the models.
- There are 1442 train videos and 568 test videos and each of them is converted into batch format where each batch contains 5 videos.

LSTM:-

- LSTM, Long short term memory is mainly used for data that can be represented in a sequential manner.
- And since the problem given to us a series of images that happen to be a video and we know that video is nothing but sequential representation of images.
- And using LSTM we can learn what the sequence of images represent through which we can predict the event happening in the video, like applying makeup etc.

Training the LSTM:-

- I use the following hyper parameters for building the LSTM architecture:-
    - hidden_size
    - Num_layers
    - num_epochs
    - Learning_rate
- I have kept the batch size as 5 images for training.
- I am using Cross Entropy as a loss function.
- And AdamOptmizer as optimizer for learning

- I have kept batch size as 5 for all my experiments.
- sequence_length = 4096, input_size = 25 are always constant.

Experiments for LSTM (running in GCP+GPU):-
GPU:- NVIDIA Tesla P100

For all the below experiments the following parameters were kept constant:-

- Batch_size = 5
- Input_size = 25
- Sequence_length = 4096
- Num_classes = 15

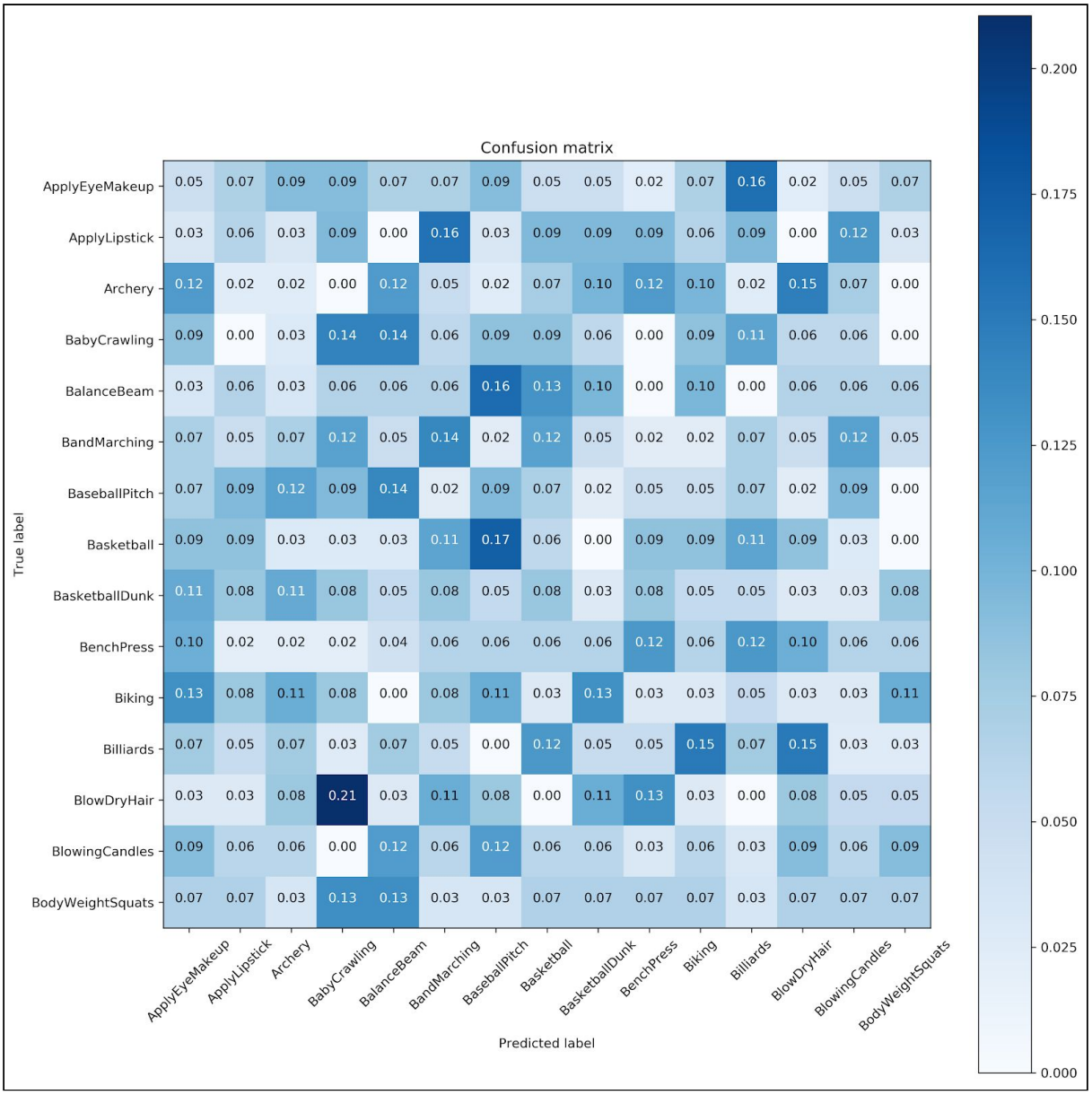
Hidden_size	num_layers	num_epochs	learning_rate	training_time	testing_time	Accuracy (%)
32	2	5	0.01	4. 48s	12s	70.2464
32	2	10	0.01	9m 38s	13s	70.5985
64	2	10	0.001	9m 35s	12s	75.8802
64	3	10	0.001	14m 23s	17s	74.1197
64	3	20	0.001	28m 56s	16s	79.2253
128	3	10	0.001	14m 50s	17s	76.4084
128	**2**	**20**	**0.001**	**21m 56s**	**11s**	**83.0985**
256	2	20	0.001	24m 2s	11s	78.1690
512	2	20	0.001	52m 35s	26s	73.7676

Training the LSTM took (for best model) :-

Best accuracy for LSTM:- **83.0985**

Confusion Matrix for LSTM:-

- As we can see from the confusion matrix for LSTM that the model's accuracy is good but the model is confused among many classes.



Training the SVM:-

- From the above mentioned train and test data we create a combined matrix each for train and test data.
- For the train data we take each file which is in the form 25*4096 shape.
- We flatten each of these matrices for both train and test data.
- We then accumulate all the train images into a train_size*flatten format.
- This accumulated matrix is fed to SVM for training.
- Similarly this is done for test data as well.
- Then prediction labels are matched with the truth labels to get the accuracy.

Training the SVM took (for best model) :- (in local)

Training time	Testing time	Accuracy (%)
1m 41s	0.222s	**95.6**

Confusion matrix for SVM:-

- We can see from the confusion matrix for SVM that all the classes were predicted almost correct.
- The model was slightly confused between Baseball and basketball Pitch.