Utrecht University

Computer Vision

Assignment 4 Report

Antonis Thalassinos : 6073492

Gurharmeet Singh: 6140076

Ishdeep Pal Singh Bhandari : 6060927

Model Description

Extraction Functionality

The extraction functionality is defined in extract_frames.py and is divided into 5 different functions namely; allFramesTraining (Used to retrieve all frames for a video for training, but was discarded due to the python array not being able to read so many images), allFramesTesting (Used to retrieve all the frames required to test all the frames of a video), frameTraining (Used to retrieve the middle frame and the flip of that frame was used for training), frameTesting (Used to retrieve the middle and the flip of the frame to be tested), multipleFramesTraining (Used to retrieve the middle, the flip and the middle + 4th frame required for training; most used for training and to test performance for different changes). The videos used for the training were the ones in the UCF folder for the required actions. The videos required for testing were found videos online and self recorded videos.

Training

The training functionality is defined in the train.py which basically involves extracting the frames (Using one of the extraction functions but default is multipleFramesTraining) from the videos folder in the training_data folder and placing them in the individual action folders. Define the labels i.e. the actions, set the desired image size (128X128), batch size (32) and the validation set size (20%). Then, traversing them to the training_data and retrieving the extracted frames from the individual folder and resizing them into 128X128 sizes (Done in dataset.py). After that, move them to the train.py in an array which would be used in training and validating the data later.

Now, we move on to defining the cnn model which involves create 3 convolutional layers using the Relu activation function(with size 3 and number of filters 32,32,64 respectively) with max pooling, along with defining the weights and biases required. Then, creating a flattening layer so that the last convolutional layer be passed into the fully_connected layer. We calculate the predicted value using the softmax function and then using the the the cross entropy cost function with a learning rate of 0.0001 we optimise it. Finally, we train using the train() function defined in train.py by running 1500 iterations on the dataset which roughly translates to 44 epochs, wherein in each epoch the entire data is looked at once and randomly (done using a rano) 32 images (with 20% used in validation) are selected for training and validating. The progress is reported for each epoch and recorded in train_values file with the validation accuracy, training accuracy and validation loss which would be used to test the performance of the model. The model is stored as the action-classifier-model in the pythonApplication1 folder.

Testing

This happens directly after training the model using the predict() function defined in the predict.py. Here, firstly using one of the testing extraction functions (default is frameTesting) we extract the frames to the frames folder in the testing_data folder. Now, from this folder one frame is loaded and resized as long as all the frames in the folder are predicted. The trained model is retrieved every time for each frame and the prediction for the frame is printed and recorded in the predictions file in the pythonApplication1 folder.

The result is stored in the form of the name of the frame (with the count, if required) followed by the probability of similarity to the 5 different action classifications.

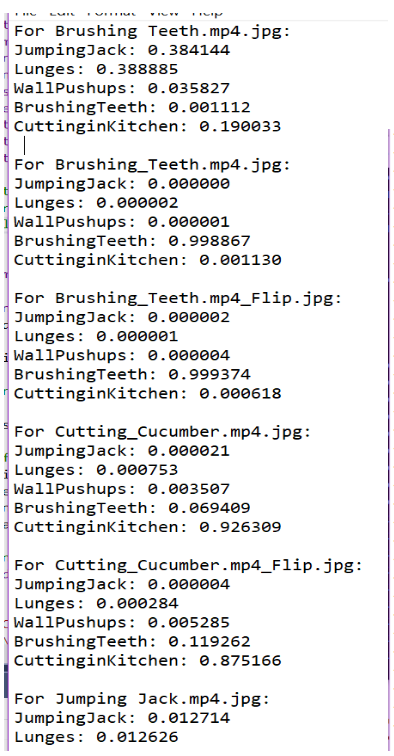
Application

To run the application run the train.py file and change paths in all the files where required.

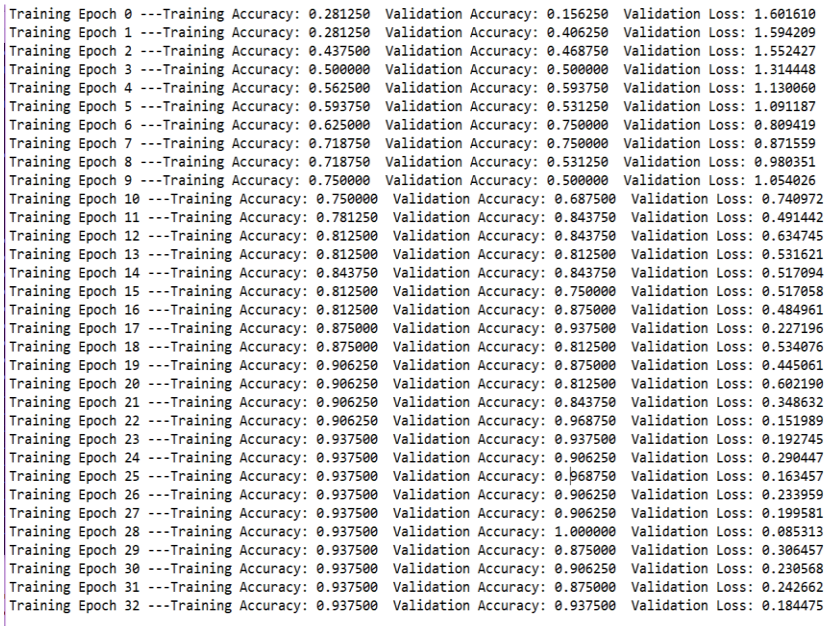
Performance Graphs and Images

The files shown here are not all the files but shown as an example of the format used. More can be found in the predictions and train folder for further clarification. All training was done using UCF videos and predictions done using own videos (as submitted in part one). In any case we have included the own videos used in the testing in data/own folder.

Different Network Layers



```
For Brushing Teeth.mp4.jpg:
JumpingJack: 0.384144
Lunges: 0.388885
WallPushups: 0.035827
BrushingTeeth: 0.001112
CuttinginKitchen: 0.190033

For Brushing_Teeth.mp4.jpg:
JumpingJack: 0.000000
Lunges: 0.000002
WallPushups: 0.000001
BrushingTeeth: 0.998867
CuttinginKitchen: 0.001130

For Brushing_Teeth.mp4_Flip.jpg:
JumpingJack: 0.000002
Lunges: 0.000001
WallPushups: 0.000004
BrushingTeeth: 0.999374
CuttinginKitchen: 0.000618

For Cutting_Cucumber.mp4.jpg:
JumpingJack: 0.000021
Lunges: 0.000753
WallPushups: 0.003507
BrushingTeeth: 0.069409
CuttinginKitchen: 0.926309

For Cutting_Cucumber.mp4_Flip.jpg:
JumpingJack: 0.000004
Lunges: 0.000284
WallPushups: 0.005285
BrushingTeeth: 0.119262
CuttinginKitchen: 0.875166

For Jumping Jack.mp4.jpg:
JumpingJack: 0.012714
Lunges: 0.012626
```



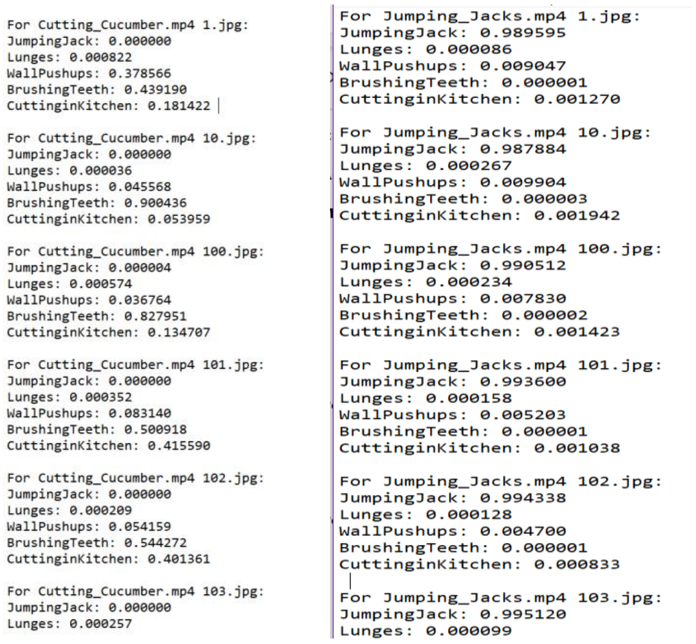
```
Training Epoch 0 ---Training Accuracy: 0.281250  Validation Accuracy: 0.156250  Validation Loss: 1.601610
Training Epoch 1 ---Training Accuracy: 0.281250  Validation Accuracy: 0.406250  Validation Loss: 1.594209
Training Epoch 2 ---Training Accuracy: 0.437500  Validation Accuracy: 0.468750  Validation Loss: 1.552427
Training Epoch 3 ---Training Accuracy: 0.500000  Validation Accuracy: 0.500000  Validation Loss: 1.314448
Training Epoch 4 ---Training Accuracy: 0.562500  Validation Accuracy: 0.593750  Validation Loss: 1.130060
Training Epoch 5 ---Training Accuracy: 0.593750  Validation Accuracy: 0.531250  Validation Loss: 1.091187
Training Epoch 6 ---Training Accuracy: 0.625000  Validation Accuracy: 0.750000  Validation Loss: 0.809419
Training Epoch 7 ---Training Accuracy: 0.718750  Validation Accuracy: 0.750000  Validation Loss: 0.871559
Training Epoch 8 ---Training Accuracy: 0.718750  Validation Accuracy: 0.531250  Validation Loss: 0.980351
Training Epoch 9 ---Training Accuracy: 0.750000  Validation Accuracy: 0.500000  Validation Loss: 1.054026
Training Epoch 10 ---Training Accuracy: 0.750000  Validation Accuracy: 0.687500  Validation Loss: 0.740972
Training Epoch 11 ---Training Accuracy: 0.781250  Validation Accuracy: 0.843750  Validation Loss: 0.491442
Training Epoch 12 ---Training Accuracy: 0.812500  Validation Accuracy: 0.843750  Validation Loss: 0.634745
Training Epoch 13 ---Training Accuracy: 0.812500  Validation Accuracy: 0.812500  Validation Loss: 0.531621
Training Epoch 14 ---Training Accuracy: 0.843750  Validation Accuracy: 0.843750  Validation Loss: 0.517094
Training Epoch 15 ---Training Accuracy: 0.812500  Validation Accuracy: 0.750000  Validation Loss: 0.517058
Training Epoch 16 ---Training Accuracy: 0.812500  Validation Accuracy: 0.875000  Validation Loss: 0.484961
Training Epoch 17 ---Training Accuracy: 0.875000  Validation Accuracy: 0.937500  Validation Loss: 0.227196
Training Epoch 18 ---Training Accuracy: 0.875000  Validation Accuracy: 0.812500  Validation Loss: 0.534076
Training Epoch 19 ---Training Accuracy: 0.906250  Validation Accuracy: 0.875000  Validation Loss: 0.445061
Training Epoch 20 ---Training Accuracy: 0.906250  Validation Accuracy: 0.812500  Validation Loss: 0.602190
Training Epoch 21 ---Training Accuracy: 0.906250  Validation Accuracy: 0.843750  Validation Loss: 0.348632
Training Epoch 22 ---Training Accuracy: 0.906250  Validation Accuracy: 0.968750  Validation Loss: 0.151989
Training Epoch 23 ---Training Accuracy: 0.937500  Validation Accuracy: 0.937500  Validation Loss: 0.192745
Training Epoch 24 ---Training Accuracy: 0.937500  Validation Accuracy: 0.906250  Validation Loss: 0.290447
Training Epoch 25 ---Training Accuracy: 0.937500  Validation Accuracy: 0.968750  Validation Loss: 0.163457
Training Epoch 26 ---Training Accuracy: 0.937500  Validation Accuracy: 0.906250  Validation Loss: 0.233959
Training Epoch 27 ---Training Accuracy: 0.937500  Validation Accuracy: 0.906250  Validation Loss: 0.199581
Training Epoch 28 ---Training Accuracy: 0.937500  Validation Accuracy: 1.000000  Validation Loss: 0.085313
Training Epoch 29 ---Training Accuracy: 0.937500  Validation Accuracy: 0.875000  Validation Loss: 0.306457
Training Epoch 30 ---Training Accuracy: 0.937500  Validation Accuracy: 0.906250  Validation Loss: 0.230568
Training Epoch 31 ---Training Accuracy: 0.937500  Validation Accuracy: 0.875000  Validation Loss: 0.242662
Training Epoch 32 ---Training Accuracy: 0.937500  Validation Accuracy: 0.937500  Validation Loss: 0.184475
```

5 Convolutional Layers Predictions and Training Values (Fig 1.)

Different Class Predictions

Class Comparisons (Fig 2.)

Predicting all frames per video

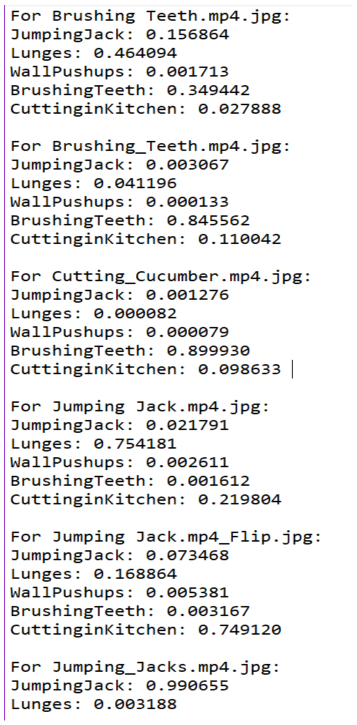


```
For Cutting_Cucumber.mp4 1.jpg:
JumpingJack: 0.000000
Lunges: 0.000822
WallPushups: 0.378566
BrushingTeeth: 0.439190
CuttinginKitchen: 0.181422

For Cutting_Cucumber.mp4 10.jpg:
JumpingJack: 0.000000
Lunges: 0.000036
WallPushups: 0.045568
BrushingTeeth: 0.900436
CuttinginKitchen: 0.053959

For Cutting_Cucumber.mp4 100.jpg:
JumpingJack: 0.000004
Lunges: 0.000574
WallPushups: 0.036764
BrushingTeeth: 0.827951
CuttinginKitchen: 0.134707

For Cutting_Cucumber.mp4 101.jpg:
JumpingJack: 0.000000
Lunges: 0.000352
WallPushups: 0.083140
BrushingTeeth: 0.500918
CuttinginKitchen: 0.415590

For Cutting_Cucumber.mp4 102.jpg:
JumpingJack: 0.000000
Lunges: 0.000209
WallPushups: 0.054159
BrushingTeeth: 0.544272
CuttinginKitchen: 0.401361

For Cutting_Cucumber.mp4 103.jpg:
JumpingJack: 0.000000
Lunges: 0.000257
```

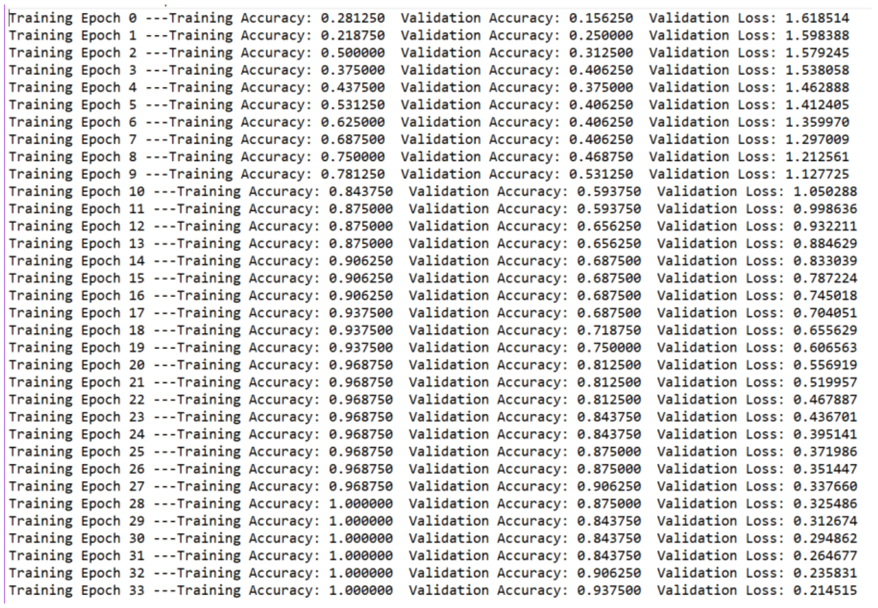
```
For Jumping_Jacks.mp4 1.jpg:
JumpingJack: 0.989595
Lunges: 0.000086
WallPushups: 0.009047
BrushingTeeth: 0.000001
CuttinginKitchen: 0.001270

For Jumping_Jacks.mp4 10.jpg:
JumpingJack: 0.987884
Lunges: 0.000267
WallPushups: 0.009904
BrushingTeeth: 0.000003
CuttinginKitchen: 0.001942

For Jumping_Jacks.mp4 100.jpg:
JumpingJack: 0.990512
Lunges: 0.000234
WallPushups: 0.007830
BrushingTeeth: 0.000002
CuttinginKitchen: 0.001423

For Jumping_Jacks.mp4 101.jpg:
JumpingJack: 0.993600
Lunges: 0.000158
WallPushups: 0.005203
BrushingTeeth: 0.000001
CuttinginKitchen: 0.001038

For Jumping_Jacks.mp4 102.jpg:
JumpingJack: 0.994338
Lunges: 0.000128
WallPushups: 0.004700
BrushingTeeth: 0.000001
CuttinginKitchen: 0.000833

For Jumping_Jacks.mp4 103.jpg:
JumpingJack: 0.995120
Lunges: 0.000099
```

Jumping jack and Cutting in kitchen video predictions (Fig 3.)

Different Training Video Sizes



```
For Brushing Teeth.mp4.jpg:
JumpingJack: 0.156864
Lunges: 0.464094
WallPushups: 0.001713
BrushingTeeth: 0.349442
CuttinginKitchen: 0.027888

For Brushing_Teeth.mp4.jpg:
JumpingJack: 0.003067
Lunges: 0.041196
WallPushups: 0.000133
BrushingTeeth: 0.845562
CuttinginKitchen: 0.110042

For Cutting_Cucumber.mp4.jpg:
JumpingJack: 0.001276
Lunges: 0.000082
WallPushups: 0.000079
BrushingTeeth: 0.899930
CuttinginKitchen: 0.098633

For Jumping Jack.mp4.jpg:
JumpingJack: 0.021791
Lunges: 0.754181
WallPushups: 0.002611
BrushingTeeth: 0.001612
CuttinginKitchen: 0.219804

For Jumping Jack.mp4_Flip.jpg:
JumpingJack: 0.073468
Lunges: 0.168864
WallPushups: 0.005381
BrushingTeeth: 0.003167
CuttinginKitchen: 0.749120

For Jumping_Jacks.mp4.jpg:
JumpingJack: 0.990655
Lunges: 0.003188
```



```
Training Epoch 0 ---Training Accuracy: 0.281250  Validation Accuracy: 0.156250  Validation Loss: 1.618514
Training Epoch 1 ---Training Accuracy: 0.218750  Validation Accuracy: 0.250000  Validation Loss: 1.598388
Training Epoch 2 ---Training Accuracy: 0.500000  Validation Accuracy: 0.312500  Validation Loss: 1.579245
Training Epoch 3 ---Training Accuracy: 0.375000  Validation Accuracy: 0.406250  Validation Loss: 1.538058
Training Epoch 4 ---Training Accuracy: 0.437500  Validation Accuracy: 0.375000  Validation Loss: 1.462888
Training Epoch 5 ---Training Accuracy: 0.531250  Validation Accuracy: 0.406250  Validation Loss: 1.412405
Training Epoch 6 ---Training Accuracy: 0.625000  Validation Accuracy: 0.406250  Validation Loss: 1.359970
Training Epoch 7 ---Training Accuracy: 0.687500  Validation Accuracy: 0.406250  Validation Loss: 1.297009
Training Epoch 8 ---Training Accuracy: 0.750000  Validation Accuracy: 0.468750  Validation Loss: 1.212561
Training Epoch 9 ---Training Accuracy: 0.781250  Validation Accuracy: 0.531250  Validation Loss: 1.127725
Training Epoch 10 ---Training Accuracy: 0.843750  Validation Accuracy: 0.593750  Validation Loss: 1.050288
Training Epoch 11 ---Training Accuracy: 0.875000  Validation Accuracy: 0.593750  Validation Loss: 0.998636
Training Epoch 12 ---Training Accuracy: 0.875000  Validation Accuracy: 0.656250  Validation Loss: 0.932211
Training Epoch 13 ---Training Accuracy: 0.875000  Validation Accuracy: 0.656250  Validation Loss: 0.884629
Training Epoch 14 ---Training Accuracy: 0.906250  Validation Accuracy: 0.687500  Validation Loss: 0.833039
Training Epoch 15 ---Training Accuracy: 0.906250  Validation Accuracy: 0.687500  Validation Loss: 0.787224
Training Epoch 16 ---Training Accuracy: 0.906250  Validation Accuracy: 0.687500  Validation Loss: 0.745018
Training Epoch 17 ---Training Accuracy: 0.937500  Validation Accuracy: 0.687500  Validation Loss: 0.704051
Training Epoch 18 ---Training Accuracy: 0.937500  Validation Accuracy: 0.718750  Validation Loss: 0.655629
Training Epoch 19 ---Training Accuracy: 0.937500  Validation Accuracy: 0.750000  Validation Loss: 0.606563
Training Epoch 20 ---Training Accuracy: 0.968750  Validation Accuracy: 0.812500  Validation Loss: 0.556919
Training Epoch 21 ---Training Accuracy: 0.968750  Validation Accuracy: 0.812500  Validation Loss: 0.519957
Training Epoch 22 ---Training Accuracy: 0.968750  Validation Accuracy: 0.812500  Validation Loss: 0.467887
Training Epoch 23 ---Training Accuracy: 0.968750  Validation Accuracy: 0.843750  Validation Loss: 0.436701
Training Epoch 24 ---Training Accuracy: 0.968750  Validation Accuracy: 0.843750  Validation Loss: 0.395141
Training Epoch 25 ---Training Accuracy: 0.968750  Validation Accuracy: 0.875000  Validation Loss: 0.371986
Training Epoch 26 ---Training Accuracy: 0.968750  Validation Accuracy: 0.875000  Validation Loss: 0.351447
Training Epoch 27 ---Training Accuracy: 0.968750  Validation Accuracy: 0.906250  Validation Loss: 0.337660
Training Epoch 28 ---Training Accuracy: 1.000000  Validation Accuracy: 0.875000  Validation Loss: 0.325486
Training Epoch 29 ---Training Accuracy: 1.000000  Validation Accuracy: 0.843750  Validation Loss: 0.312674
Training Epoch 30 ---Training Accuracy: 1.000000  Validation Accuracy: 0.843750  Validation Loss: 0.294862
Training Epoch 31 ---Training Accuracy: 1.000000  Validation Accuracy: 0.843750  Validation Loss: 0.264677
Training Epoch 32 ---Training Accuracy: 1.000000  Validation Accuracy: 0.906250  Validation Loss: 0.235831
Training Epoch 33 ---Training Accuracy: 1.000000  Validation Accuracy: 0.937500  Validation Loss: 0.214515
```

Training and prediction values for 80 videos (Fig 4.)

Performance Explanation

Different Network Layers

(Check the network depth folder for the files)

We tested our framework for 3 different network models (using the middle,middle+4,middle flipped frames of each video for training and the middle and middle flip frame for testing) with 3 ,4 and 5 convolutional layers (Fig 1.) . In the three network as you can see the the training accuracy increasing constantly and the overall time taken to run all the epochs was about 10 minutes for the 3-layered network, 15 for the 4-layered and about 25 for the 5-layered. We found that the more layers we added the less was the increase in the training accuracy and more abrupt were the changes in the validation accuracy (although it increased in the end but it was abrupt). We believe that it must be due to overfitting of the data, due to the increase in complexity of the model. Another reason why we feel it overfitted the data is because after it reaches the end of the epoch the training accuracy stops increasing. This was coherent in all the 3 network types, the reason being that the number of iterations (1500) for training should have been less to have avoiding that. The predictions were where we were most surprised because all the self recorded videos were being misclassified to a random class. We concluded that the self recorded videos (for example) were not similar enough to the UCF videos. But, for the videos collected from the internet (for example) were misclassified less. As you can see above the network with the 5 layer predicted much better than the 3 and the 4 layers. So, even with overfitting errors we got about 70-80% correct classification for the 5 layers. It reduced as we reduce the layers. So, in conclusion having more network depth is good (for instance 4 layers), but having more training data would have resulted in a better model at the cost of performance.

Different Class Predictions

(Check the predcition_classes file for the predictions)
For the different class tests we chose 1 video from 20 different classes namely; baby crawling, bench press, fencing, front crawl, hammering, hammer throw, juggling balls, jump rope, knitting, mopping floors, nunchucks, parallel bars, pizza tossing, pull ups, punch, push ups, soccer juggling, taichi, discuss throw and typing. From the prediction file we see that taking the middle frame for the all these videos and compared to the 3 layered model (default model) we saw that bench press and parallel bars being classified as lunges due to similar upward and downward motion or pull ups and being classified as jumping jacks due to the upward motion. But some of them like pizza tossing being classified as brushing teeth was also funny to see as you can see in the figure (Fig. 2) that the viewpoint is what could have caused that

Predicting All Frames for a video

(Please refer predicting all frames folder for the files)

For explaining this we are taking the example of a jumping jack video (Fig 3.) for which we used all the frames and tested on what it classified it to and saw how the classification changed as the video progressed in frames. It classified all the frames as jumping jacks with a probability of about 98% using the 3 layered model. This means that the video was very similar to the training data that we used to train to get such accurate results. Also, the model overfit it due to the 1500 iterations. Looking at the other files we found that except for the brushing teeth video all the other files for cutting, wall push ups were classified to either brushing teeth or lunges, suggesting that the model was trained more to classify lunges and brushing teeth which resulted in such discrepancies.

Different Training Video Sizes

(Please refer Different Training Video Sizes folder for the files)

For the different training video sizes we can used four different quantities namely 20,30,60,80 (Fig 4.). We ran the 3 layered model on all these types of training sets and recording the training and prediction values. Looking at the training values we saw that for the sizes it was pretty much reaching 100% training accuracy with high validation accuracy and low validation loss at the end of the iterations, which could be due to the high number of iterations, so it seems fine. But, when you see the prediction files for 30 and 20 video sizes you notice a huge amount of miss classification when compared with 60 and 80 video sizes. This was clearly due to the result of the low size of training data, because of less varieties of videos reduced the overall performance thus resulting in overfitting to only those 20 or 30 videos and not being able to classify different videos.

Task performed in the assignment

1. Performed cross validation
2. Test videos for prediction
3. Test on own videos
4. Use different network depths
5. Trained using multiple frames including mirrored frames
6. Used different quantities of training data
7. Tested performance on all frames of a video
8. Tested on other classes

References

1. http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network-for-image-classification/ Used this example to design our model