Computer Vision CAP 5415 – Fall 2017 – Programming Assignment III

Girish Kumar Kannan – UCF ID 4196719 – University of Central Florida

Action Recognition

Action Recognition was attempted with both CNN and Non-CNN methods and as reported.

Preprocessing :

I am using videos as inputs and converting them to frames during training. The given dataset had images and videos in some classes, just videos in some classes and videos with incorrect frames in some classes. Therefore, the videos were made from the pictures for all classes and the classes which had only the videos – the video alone was taken. These videos are copied to another directory and are used during program.

Method 1 – CNN

Feature Extraction : I used the architecture of AlexNet version 1.0 to design my CNN model which I can run in my computer (with low configuration GPU). The image is attached in the GitHub Link provided along. My architecture can be represented in simple words as follows : Image(360x480) → 4 Convolutional Layer Branch → 4 Maximum Pooling Layer Branch → 8 Convolutional Layer Branch → 8 Maximum Pooling Layer Branch → 16 Convolutional Layer Branch → Another 16 Convolutional Layer Branch → Another 16 Convolutional Layer Branch → 16 Maximum Pooling Layer Branch → 16 Dense Layers Branch → 1 Dense Layer → Softmax.

Classifier Design : CNN by itself does classification along with feature extraction due to the layers. However, classification method done by the CNN cannot be specifically explained. It is an automatic process.

Evaluation : For CNN, I did not implement Leave One Out Cross Validation. Instead, I randomly sampled one video from each class, as Test Dataset and Rest of the Videos are my Training Videos. Since there is no LOO done, my accuracy, sensitivity and specificity values are not averages from epochs, but it is just one run values. I obtained Accuracy scores implementing all methods to reduce overfitting – using L2 regularization, dropouts and early stopping. When executed, I realized that even though my accuracy was above 95% at some instances, the sensitivity measure did not match the accuracy value. In my case, Sensitivity is same as Accuracy as I define Sensitivity as Matched predictions divided by Total Predictions, while Specificity is Non-Matched predictions divided by Total Predictions (1 - Sensitivity).

Results : The Maximum Sensitivity my model yielded was not over 20% as opposed to the necessity of being over 90%. Therefore, I had to discard the use of CNN and switch over to another method – using Histogram of Oriented Gradients and Support Vector Machines.

Method 2 – SVM with HOG

Feature Extraction : I used Histogram of Oriented Gradients to get the Features of the frames in the videos. Initially I reduce the frames to 90x120 pixels. This is done as more the size of image, more the features obtained from HOG method – so more time taken to fit and evaluate in the SVM. I use 9 bins for angular orientation with 8x8 block size and 1x1 cell size. The trade-off here is, more the cell size – more features, more the orientation of nbins – more the features and smaller the block size – more features. Thus, a combination was chosen that can be executed in my computer without compromising performance and time. I am using HOG over other techniques like SIFT(Angular Histograms), HAAR(Cascading), and SURF(Patches) as HOG is simpler and faster to execute and easy to feed-in into SVM, in my opinion.

Classifier Design : Support Vector Machines. I am using all the four kernels – Linear, Gaussian, Sigmoid and Polynomial. For my Linear kernel, I have chosen C as 1.0, Gaussian with C as 1.0 and sigma is 1/n_features, Sigmoid with C as 1.0, Gamma as 1.0 and Coefficient as 1.0, and Polynomial with C as 1.0, Degree as 1.0, Gamma as 1.0 and Coefficient as 1.0. After executing a few epochs on each, Polynomial with degree 1 yielded close accuracy as Linear. Sigmoid took the same time as Linear and yields the close accuracy of Gaussian. Linear took less time while Gaussian took maximum time to execute (870sec Linear vs. 2100sec Gaussian per epoch). With respect to Time constraints, Linear performed well while given a lot of time, Gaussian performed well. The features as a row vector for all frames were passed on to SVM to get output predictions.

Evaluation : I used two evaluation techniques : LeaveOneOut and Random One Sample Selection. LeaveOneOut implemented for total number of videos taking one video out as test data. Random One Sample Selection – One video from each class is randomly chosen as Test data and rest of the videos are used as Train data. Here the Accuracy is the accuracy in prediction while Sensitivity is the percentage of matching predictions and Specificity is percentage of non-matching predictions. I have only run less than 10 epochs as my computer configuration cannot handle all epochs. Thus, manually averaging over the epochs LOO produced around 96% and Random One Sample Set obtained around 88%.

Result : Appreciable results were produced as described above. I had to use Linear Kernel to compare results due to low computer configuration. Classification using SVM and HOG performed better than my CNN model.

Another inference was that I could not understand what happened within CNN in TensorFlow but I could understand better in SVM implementation. Therefore, I found non-CNN method worthwhile.

The programs written are not excessively yet sufficiently commented. All files provided and procured from internet that was used as image data are attached in the final submission file.