CSCE 636 Final Project Report

Chen Liang

05/01/2020

Research Topic

This project is a deep learning solution to detect punching events. As inspired by smart home applications, the model should be able to detect person punching walls or something else from the video, to ensure the safety of family members. The model should be able to output a probability indicating whether or not someone is punching something at any given time in the video.

Dataset

Over the past 3 submissions, various datasets are used, and the last submission used a combination of previous dataset with some additional videos found online. Videos that labeled as 'punching' related are selected from dataset HMDB51, UCF101, and Kinetics. Also, some videos on YouTube that related to human actions and punching are also downloaded to expand the existing dataset. At the end, there are 66 video clips on punching and 86 video clips on not punching. Notice that punching clips are usually longer than not punching videos, and thus the total number of frames are about balanced.

Improvements and New Ideas since Last Submission

In the last submission, the model is changed from CNN+LSTM on images to LSTM on OpenPose, and gained a 30% accuracy improvement on validation dataset, where the final validation accuracy is 96.7% on a balanced dataset. For this submission, the model and data processing methods are further improved to enhance the overall performance of the program, which can be summarized into the following key points:

1. Training Dataset Improvement

Enlarge Dataset and Data Quality Improvement

The previous dataset I used contains some very low-resolution videos, especially from HMDB51 dataset, which makes the OpenPose result noisy. Also, some of the videos are upside down or rotated, which also decreases the correctness of OpenPose estimations. Thus, all these low-quality videos are replaced by some newly added YouTube videos.

Data Diversity Improvement

To cover a more diverse pose, for some videos, an additional point of view was added. This was added on top of the original dataset so that the model can be trained on punching event from different viewpoints.

2. Preprocessing Improvement

For some of the videos, OpenPose can occasionally falsely detect some 'people' in the video. This could be noise, and also could be mirror reflection, or something very similar to person (e.g. person-like punching bags). Thus, for this submission, two additional preprocessing steps are added. First, the skeletons in different

frames are matched based on the relative Euclidean Distance. For example, if there are one skeleton at frame t1 but two skeletons at frame t2. The skeleton in t1 will be matched to the closest skeleton in t2. This helps program keep track of the correct skeleton and avoid being influenced by noise data. Second, the program will set to focus on larger and more complete skeletons. An example could be that the program will choose a skeleton which is larger and has more joint points in the frame to track instead of a very smaller skeleton. However, this will still be corrected by the first preprocessing mechanism mentioned above to avoid falsely tracking on noise data.

Figure 1: Example of false positive skeleton (On the left on the punching bag)

3. Model Update

To limit the model from being overfitted, in this submission the LSTM units is changed from 256 to 64 to simplify the model. Also, a dropout rate of 0.2 was added to mitigate overfitting issue. More details will be covered in Model section below. In addition, temporal dimension for the model is increased from 10 to 20, where the unit is frame. In other words, the LSTM model will now consider 20 consecutive frames instead of 10, so that it can get a better understanding of the person's action instead of focus too much on details.

4. Add Postprocessing

It was found in last submission that the model will have some false positive 'peaks' in the plot. To mitigate this issue, if the positive result is too short to be true (e.g. it is impossible for someone to punch something in 1ms and suddenly stop punching), the postprocessing will suppress that peak. This helps the model to generate a more clear and accurate result.

Notice that this is not added in test code since the model needs to be graded by Professor and TA and I want to show the original unmodified output for correct evaluation. Thus, noting is applied to the model result in test code.

The following sections will talk about the full details of the project (including changes I made for this submission).

Preprocessing

Preprocessing program's task is to process, convert, and clean raw data generated by OpenPose, and organize them into correct matrix format. Firstly, the program will extract X and Y coordinates of joint points from the data, and convert all of them to relative X and Y positions, where converted new joint points' coordinates are all relative to the head of the person in order to track relative actions more accurately. The program will then combine data generated from 20 consecutive frames into one matrix. The shape of matrix is (20, 50), where 20 is the temporal dimension (number of consecutive datapoints to be used in training) and 50 is relative X and Y values for joint points. To keep track of the correct skeleton, the preprocessing program will find closest skeletons by calculating the Euclidean Distance between two skeleton key point arrays in consecutive frames to make sure it is not matched with false positives. Also, the program will prioritize to larger and more complete skeleton. This is determined by the square of Euclidean Distance between the joint points and the head point.

After preprocessing the raw data, the processed key points are organized to a 3D array with shape (x, 20, 50), where x is the number of frames in all samples, 20 is the temporal dimension, and 50 is x and y values for all key points. This can then be used for training/validation/testing.

Model

Structure

Since punching event is time-dependent, the model should have recurrent structure or time distributed layers. The final model used for the last submission is based on LSTM. (Model structures of past submission can be found in the 'Extra' section in the end of this report) Since after preprocessing the input is relatively clear and only contains 20x50 matrix as input, the model is designed to be concise but functioning, so that it can both predict with an acceptable accuracy and not too complicated which might cause overfitting or inefficient training. Thus, the model is set as a sequential model and the complete structure is shown below:


```
model = models.Sequential()
model.add(layers.LSTM(64, dropout=0.2,))
model.add(layers.Dense(2, activation='softmax'))
```

The model is designed to be simple and a dropout is also added to avoid overfit.

Input/Output shape

The input of the model is a 3D matrix. The size of each training sample is a matrix of shape (20, 50), where 20 is number of consecutive lists of key points in this block (i.e. temporal dimension), and 50 is the number of X and Y values for key points for each frame. The output is the probability of either class 0 or class 1 where 1 stands for 'punching' and 0 stands for 'not punching'.

```
Model: "sequential_2"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                multiple                  29440
_________________________________________________________________
dense_2 (Dense)              multiple                  130
=================================================================
Total params: 29,570
Trainable params: 29,570
Non-trainable params: 0
_________________________________________________________________
```

Hyperparameters

Unit of LSTM is set to 64, which then directly mapped to a dense layer with two units. The dropout rate for LSTM is set to 0.2. Loss function uses sparse categorical cross entropy, and evaluation method is accuracy. Model is trained for 10 epochs.

Performance Evaluation

Training and Validation

The model is trained on 91144 samples (each sample is a 20x50 matrix) and validate on `10128` samples (10% percent). The final validation accuracy is 98.27%, which shows an improved performance than last submission. Detailed result is shown below:

```
Train on 91144 samples, validate on 10128 samples
Epoch 1/15
91144/91144 [==============================] - 40s 435us/sample - loss: 0.1364
- accuracy: 0.9469 - val_loss: 0.0937 - val_accuracy: 0.9654
Epoch 2/15
91144/91144 [==============================] - 37s 406us/sample - loss: 0.0780
- accuracy: 0.9725 - val_loss: 0.0718 - val_accuracy: 0.9758
Epoch 3/15
91144/91144 [==============================] - 36s 400us/sample - loss: 0.0714
- accuracy: 0.9756 - val_loss: 0.1028 - val_accuracy: 0.9702
Epoch 4/15
91144/91144 [==============================] - 37s 402us/sample - loss: 0.0602
- accuracy: 0.9796 - val_loss: 0.0668 - val_accuracy: 0.9732
Epoch 5/15
91144/91144 [==============================] - 39s 429us/sample - loss: 0.0569
- accuracy: 0.9810 - val_loss: 0.1168 - val_accuracy: 0.9557
Epoch 6/15
91144/91144 [==============================] - 39s 432us/sample - loss: 0.0530
- accuracy: 0.9816 - val_loss: 0.0592 - val_accuracy: 0.9804
Epoch 7/15
91144/91144 [==============================] - 40s 436us/sample - loss: 0.0523
- accuracy: 0.9819 - val_loss: 0.0672 - val_accuracy: 0.9741
Epoch 8/15
91144/91144 [==============================] - 39s 424us/sample - loss: 0.0516
- accuracy: 0.9821 - val_loss: 0.0921 - val_accuracy: 0.9651
Epoch 9/15
91144/91144 [==============================] - 38s 416us/sample - loss: 0.0481
- accuracy: 0.9839 - val_loss: 0.0698 - val_accuracy: 0.9776
```

```
Epoch 10/15
91144/91144 [==============================] - 38s 421us/sample - loss: 0.0461
- accuracy: 0.9841 - val_loss: 0.0887 - val_accuracy: 0.9720
Epoch 11/15
91144/91144 [==============================] - 40s 442us/sample - loss: 0.0437
- accuracy: 0.9853 - val_loss: 0.0712 - val_accuracy: 0.9709
Epoch 12/15
91144/91144 [==============================] - 38s 422us/sample - loss: 0.0439
- accuracy: 0.9856 - val_loss: 0.0614 - val_accuracy: 0.9773
Epoch 13/15
91144/91144 [==============================] - 38s 421us/sample - loss: 0.0438
- accuracy: 0.9854 - val_loss: 0.0678 - val_accuracy: 0.9745
Epoch 14/15
91144/91144 [==============================] - 38s 414us/sample - loss: 0.0424
- accuracy: 0.9851 - val_loss: 0.0753 - val_accuracy: 0.9742
Epoch 15/15
91144/91144 [==============================] - 38s 418us/sample - loss: 0.0394
- accuracy: 0.9865 - val_loss: 0.0465 - val_accuracy: 0.9827
```

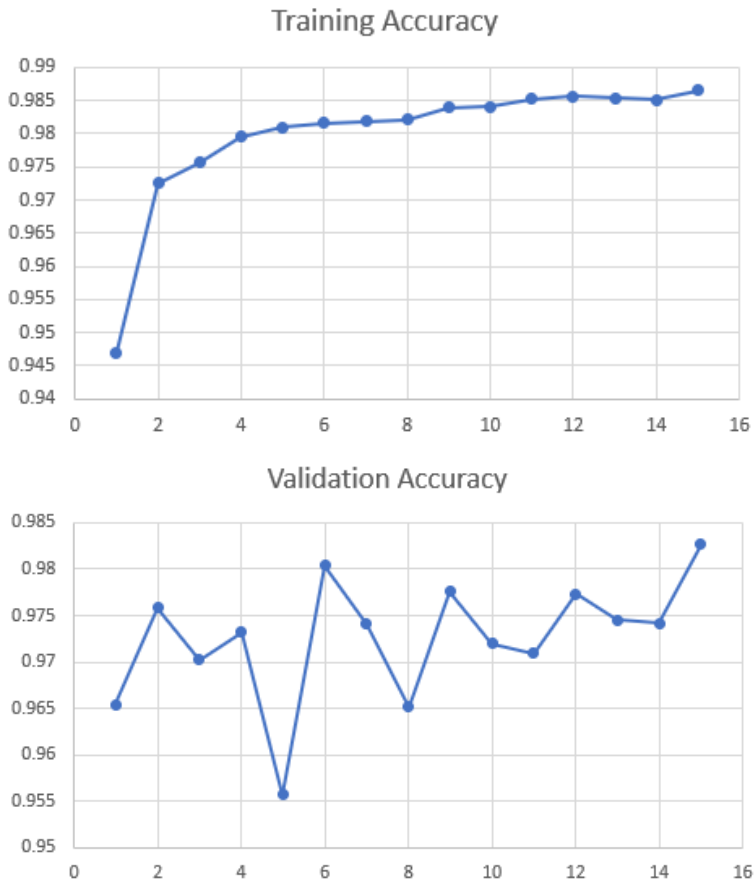

Figure 2: Left: Training Accuracy. Right: Validation Accuracy.

Model Testing Performance

To better understand how is the model's performance on videos instead of focusing on the accuracy of classifying each 20 consecutive OpenPose data, the model is tested on 5 videos. Among these 5 videos, 3 of them are punching or has punching action, and 2 of them are not punching. The YouTube links to these videos can be found in file 'FiveSampleVideoLink.txt' in the submission zip (also on Github).

Below shows one sample from each class (punch and not punch) and the plot of probability of punching.

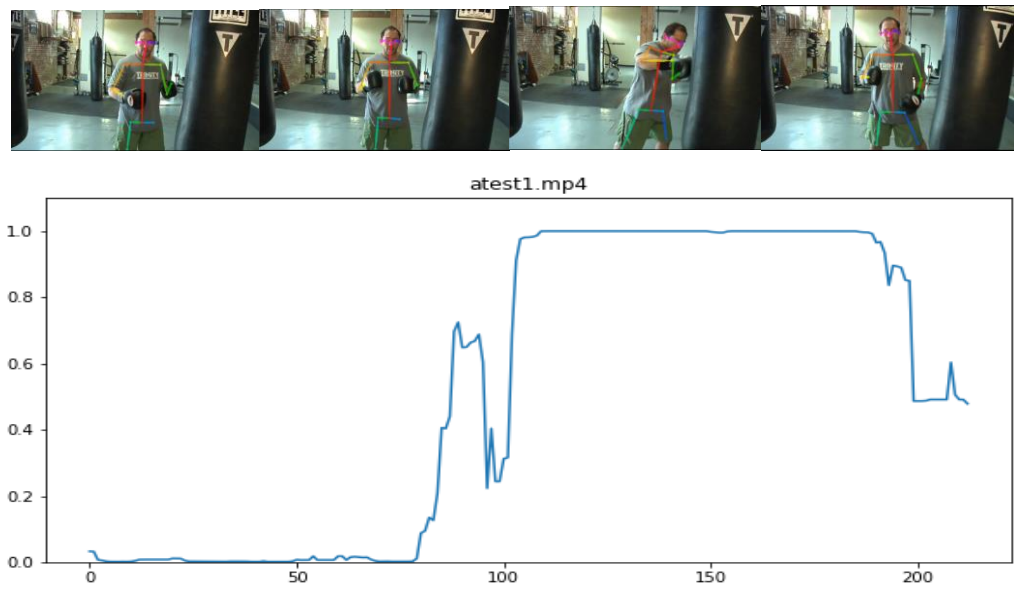

Figure 3: Plot of probability for Punching video and its corresponding video snapshots

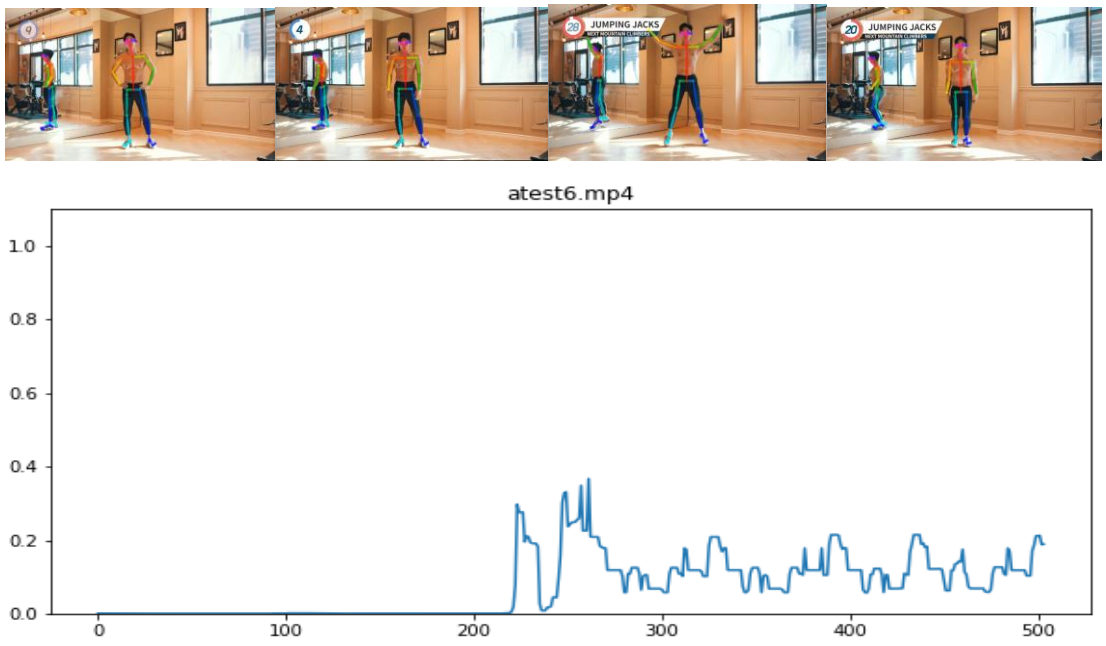

Figure 4: Plot of probability for Not Punching video and its corresponding video snapshots

Code Files and Customized Testing

All code has been uploaded to Github. A link can be found along with this submission zip file on eCampus. Model is also uploaded to Github so you don't have to download model separately. To train the model, see proj8_train.ipynb. To test on your own video, use uploaded .bat file. Instructions are in the README file.

Extra: Different ideas and improvements tried so far

In the past submission, three different models are tried for this task. By trying new structures and improve model and preprocess functions, the performances has been increased in the past few submissions. Below listed basic ideas of models and findings in the past submissions. Details can be found in the report for each submission.

Idea 1: CNN on merged image

The initial idea on classifying punching event is to merge 5 consecutive images to a single image, and thus each merged image will show some additional information about person's action in a short amount of time. The model is based on CNN layers and input is 2D images. However, the model does not perform well. It was shown that if the number of images to be merged is small, then the merged image cannot show trend well, but if the number is large, then the image is very blurry and cannot be distinguished. This idea and model do not work well on handling temporal information in videos with a validation accuracy of 63%. However, this is still an attempt about applying CNN on this task.

Idea 2: Time Distributed CNN and LSTM on a Sequence of Images

The next idea, which is based on the initial idea, is to convert CNN layers to be time distributed. By doing so the model will not use merged images but instead directly train on a sequence of images. The model is composed of several CNN layers at first, and then ended up with a LSTM and a dense layer. However, the performance is still low. In addition, since the model is changed to time distributed, there are many more parameters than the previous one. At that time the dataset is still small and the quality is low, and no pre-trained model was used, this model does not perform very well on both validation and testing set.

Idea 3: LSTM on OpenPose Key Points

Instead of taking images as input, OpenPose is used to get the joint points of person in the video. The output is then used as input in LSTM layer. By setting the temporal dimension to 20, the model can capture punching event with an acceptable accuracy, and has a validation accuracy on key point block for each frame for 98%. This model, by using OpenPose as a pre-process of videos, extracted the key information from video so that the model can focus on person's motion directly instead of finding features from images.

Although some ideas and models do not work, trying them still give me an opportunity to get a deeper understanding of the structure, model, and task itself.