Action Detection for Smart Homes

Wangyang He

UIN: 625004872

Nickname: heswaggy

Submission 10

05/03/2021

Abstract

The topic I choose for the “Smart Home” project is the “fall” action detection. I built my own model with VGG16 and some other layers for this detection project. I increased my model’s accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model’s overall accuracy is at around 75% with 1600 videos trained.

The topic I choose for the “Smart Home” project for submission three is the “coughing” action detection. I built my own model with VGG16 in the beginning, but then switched to Xception for better accuracy and some other layers for this detection project. I increased my model’s accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model’s overall accuracy is at around 82.3% with about 1000 videos trained.

The topic I choose for the “Smart Home” project for submission five is the “Hand washing” action detection. I built my own model with VGG16 in the beginning, but then switched to Xception for better accuracy and some other layers for this detection project. I increased my model’s accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model’s overall accuracy is at around 83.8% with about 1430 videos trained.

The topic I choose for the “Smart Home” project for submission six is the “Cleaning bathroom” action detection. I built my own model with VGG16 in the beginning, but then switched to Xception for better accuracy and some other layers for this detection project. I

increased my model's accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model's overall accuracy is at around 96.6% with about 1100 videos trained.

The topic I choose for the "Smart Home" project for submission eight is the "Cleaning windows glass doors etc" action detection. I built my own model with VGG16 in the beginning, but then switched to Xception for better accuracy and some other layers for this detection project. I increased my model's accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model's overall accuracy is at around 86.2% with about 1800 videos trained.

The topic I choose for the "Smart Home" project for submission ten is the "Washing feet" action detection. I built my own model with VGG16 in the beginning, but then switched to Xception for better accuracy and some other layers for this detection project. I increased my model's accuracy by changing parameters such as epochs, steps per epoch, optimizer learning rate, train and test data split rate, batch size and many more. I improved the training process of my model by purchasing the Google CoLab Pro account, which adds more GPU memory and speed. I used the Kinetics 700 (2020 Version) as my training and testing dataset. My model's overall accuracy is at around 82.3% with about 1500 videos trained.

1. Topic

The topic for my project is action detection on “fall”.

The second topic for my project is action detection on “coughing”.

The third topic for my project is action detection on “Hand washing”.

The fourth topic for my project is action detection on “Cleaning bathroom”.

The fifth topic for my project is action detection on “Cleaning windows glass doors etc”.

The sixth topic for my project is action detection on “Washing feet”.

2. Motivation

The reason for choosing this project is because I want to make something that will help people's safety at home. Falling at home can sometimes result in bad consequences. For example, my grandpa fell at home and had to stay on wheel chair for rest of his time. So, I really wanted to do something about this issue, which is why I picked this topic.

On top of fall detection, to make home safety even better, I choose the topic of "coughing" for my second detection target. During COVID, coughing is one of the symptoms so it is very important to detect the action of coughing.

Making sure your hands are always clean especially when you just get home from outside is important, everyone should make sure they don't bring germs into the house for their family's safety, so for the third topic after coughing, I choose "Hand washing".

The motivation for choosing "Cleaning bathroom" as my fourth topic is that bathroom is a place that could easily contain millions of bacteria, especially in the toilet. I choose cleaning bathroom with the main target of cleaning toilet to be one of the actions that could affect people's health at home.

The reason for choosing cleaning windows and glasses as my fifth topic is because this is a very common activity at home, it also involves both indoor and outdoor videos because glasses are double sided. This topic will be important to detect so that the smart home system can track the number of times the house owner cleans their home, and could set alarms and reminders to keep the cleanness inside their home.

I choose "Washing feet" as my sixth and final topic because it matches with my third topic "Hand washing", with the same motivation, I think most people nowadays under-estimate how dirty could our feet get, while keeping other parts of our body clean, don't forget the feet too!

3. Related Works

- The paper [1] focus on fall action detection. It uses fall detection as a challenge in the public health care domain, it gives a comprehensive survey of different systems for fall detection and their underlying algorithms.
- The paper [2] focus on fall action detection. It is a survey of systems, algorithms and sensors, for the automatic early detection of the fall of elderly persons.
- The paper [3] focus on fall action detection. It introduces a new trend towards the integration of fall detection into smartphones as well as the use of machine learning methods in the detection algorithm.
- The paper [4] focus on fall action detection. This paper gives a survey of fall detection for elderly and patient, focusing on identifying approaches and principles of the existing fall detection methods.
- The paper [5] focus on fall action detection. This paper presents a smartphone-based fall detection system that monitors the movements of patients, recognizes a fall, and automatically sends a request for help to the caregivers.
- The paper [6] focus on cough action detection. In this paper, deep neural networks (DNN) are applied to model acoustic features in cough detection. A two step cough detection system is proposed based on deep neural networks(DNN) and hidden markov model(HMM).
- The paper [7] focus on cough action detection. This paper reimplemented two approaches from prior work and investigated their performance in two different scenarios across devices. It propose an efficient convolutional neural network architecture and an ensemble based classifier to reduce the cross-device discrepancy.
- The paper [8] focus on cough action detection. This paper presents a system that employs a wearable acoustic sensor and a deep convolutional neural network for detecting coughs.
- The paper [9] focus on washing hands action detection. In this work a deep learning (DL) based approach has been proposed to evaluate the autonomy of children with ASD while performing the hand-washing task.
- The paper [10] focus on table cleaning action detection. In this paper, A lightweight Deep Convolutional Neural Network (DCNN) has been proposed to recognize the food litter on top of the table.
- The paper [11] focus on human activity recognition. In this study, the authors analyze sensor readings from accelerometers and gyroscopes using long short-term memory (LSTM) recurrent neural networks to identify human activities and present a location-aware approach to improve the recognition accuracy.

4. Proposed Model

I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the VGG16 model, which is what I learned from the class textbook. On top of the VGG16, I used suggestions from the medium article, added layers such as GRU, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.001. A summary of my model looks like:

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_2 (TimeDist (None, 15, 512)           14714688
_________________________________________________________________
gru_2 (GRU)                  (None, 64)                110976
_________________________________________________________________
dense_10 (Dense)             (None, 1024)              66560
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               65664
_________________________________________________________________
dropout_8 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_14 (Dense)             (None, 2)                 130
=================================================================
Total params: 15,491,074
Trainable params: 776,386
Non-trainable params: 14,714,688
_________________________________________________________________
```

For submission 3, I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the Xception model, which I found on the Keras application website, this is the highest accuracy model listed on the website. On top of the Xception, I used suggestions from the medium article, added layers such as LSTM, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.0001. A summary of my model looks like:

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_8 (TimeDist (None, 5, 2048)           20861480
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                540928
_________________________________________________________________
dense_33 (Dense)             (None, 512)               33280
_________________________________________________________________
dropout_17 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_34 (Dense)             (None, 128)               65664
_________________________________________________________________
dropout_18 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_35 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_36 (Dense)             (None, 2)                 130
=================================================================
Total params: 21,509,738
Trainable params: 648,258
Non-trainable params: 20,861,480
_________________________________________________________________
```

For submission 5, I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the Xception model, which I found on the Keras application website, this is the highest accuracy model listed on the website. On top of the Xception, I used suggestions from the medium article, added layers such as LSTM, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.0001. A summary of my model looks like:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, 5, 2048)           20861480
_________________________________________________________________
lstm (LSTM)                  (None, 64)                540928
_________________________________________________________________
dense (Dense)                (None, 512)               33280
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 21,509,738
Trainable params: 648,258
Non-trainable params: 20,861,480
```

For submission 6, I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the Xception model, which I found on the Keras application website, this is the highest accuracy model listed on the website. On top of the Xception, I used suggestions from the medium article, added layers such as LSTM, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.0001. A summary of my model looks like:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, 5, 2048)           20861480
_________________________________________________________________
lstm (LSTM)                  (None, 64)                540928
_________________________________________________________________
dense (Dense)                (None, 512)               33280
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 21,509,738
Trainable params: 648,258
Non-trainable params: 20,861,480
_________________________________________________________________
```

For submission 8, I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the Xception model, which I found on the Keras application website, this is the highest accuracy model listed on the website. On top of the Xception, I used suggestions from the medium article, added layers such as LSTM, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.0001. A summary of my model looks like:

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed_4 (TimeDist (None, 5, 2048)           20861480
_________________________________________________________________
lstm_4 (LSTM)                (None, 64)                540928
_________________________________________________________________
dense_16 (Dense)             (None, 512)               33280
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 128)               65664
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_18 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_19 (Dense)             (None, 2)                 130
=================================================================
Total params: 21,509,738
Trainable params: 648,258
Non-trainable params: 20,861,480
_________________________________________________________________
```

For submission 10, I build my model with learning and code based on the textbook “Deep Learning with Python” and a medium article “Training a neural network with an image sequence — example with a video as input”. For the base model, I used the Xception model, which I found on the Keras application website, this is the highest accuracy model listed on the website. On top of the Xception, I used suggestions from the medium article, added layers such as LSTM, dense, dropout and time distributed layers. The optimizer I used for this model is “Adam”, at learning rate 0.0001. A summary of my model looks like:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, 5, 2048)           20861480
_________________________________________________________________
lstm (LSTM)                  (None, 64)                540928
_________________________________________________________________
dense (Dense)                (None, 512)               33280
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 21,509,738
Trainable params: 648,258
Non-trainable params: 20,861,480
_________________________________________________________________
```

5. Dataset

The dataset I used to train the model is Kinetics 700 (2020 Version).

A link to this dataset: https://deepmind.com/research/open-source/kinetics

This dataset contains 700 classes with at least 700 video clips from different YouTube videos. A sample data looks like:

	label	youtube_id	time_start	time_end	split
0	clay pottery making	---0dWlqevI	19	29	train
1	news anchoring	---aQ-tA5_A	9	19	train
2	using bagging machine	---j12rm3WI	14	24	train
3	javelin throw	--07WQ2iBIw	1	11	train
4	climbing a rope	--0NTAs-fA0	29	39	train

This dataset contains many labeled actions with its YouTube video ID, start time and end time of the clip. There are in total of 542902 labels in the dataset's training data as shown below:

```
[ ]  print(data.label)

0           clay pottery making
1                news anchoring
2          using bagging machine
3                  javelin throw
4               climbing a rope
                     ...
542897            washing dishes
542898             juggling fire
542899              taking photo
542900            brush painting
542901              changing oil
Name: label, Length: 542902, dtype: object
```

By using Pandas data frame, I was able to take out the label that I need, which is the label named "falling off chair" in this case. There are total of 805 training videos for this label as shown below:

```
[ ]  print(data[data.label == ('falling off chair')])

                    label   youtube_id  time_start  time_end  split
840     falling off chair  -5hw88bD4mE           3        13  train
944     falling off chair  -6ezg_-7Bck          10        20  train
2029    falling off chair  -Fmo9kHEV7M           0        10  train
2537    falling off chair  -KN26OH63WU           5        15  train
2831    falling off chair  -MqpifyByYs           0        10  train
...                   ...          ...         ...       ...    ...
539608  falling off chair  zaGM-9_DGe4           0        10  train
539967  falling off chair  zc_uLy45ZJU          10        20  train
541545  falling off chair  znz5AOG_AYY           0        10  train
541902  falling off chair  zqpoWFfiyvM           0        10  train
542363  falling off chair  zumeyo_1LwY          35        45  train

[805 rows x 5 columns]
```

For submission 3, the target label is “coughing”, there are total of 560 training videos for this label as shown below:

```
[ ]  print(data[data.label == ('coughing')])

             label   youtube_id  time_start  time_end  split
90616   coughing  A_FZjBew_ss           0        10  train
94647   coughing  B5LvpaWdwcs           5        15  train
95474   coughing  BBCuDtrV9gU           7        17  train
96174   coughing  BGYWV5zFR2U          32        42  train
96312   coughing  BHjCQuZ3l94         112       122  train
...          ...          ...         ...       ...    ...
532349  coughing  yiZ7mm_tRxg           7        17  train
534686  coughing  z1AqT-8Yu3A           0        10  train
534689  coughing  z1BSEeTgk-E           0        10  train
537402  coughing  zJNHNSYARcM          84        94  train
539394  coughing  za2xznAzN_0          14        24  train

[560 rows x 5 columns]
```

For submission 5, the target label is “washing hands”, there are total of 973 training videos for this label as shown below:

```
[ ]  print(data[data.label == ('washing hands')])

                label   youtube_id  time_start  time_end  split
296     washing hands  -1Hub6Ps_cc          47        57  train
1516    washing hands  -BL2GD3GBfE         592       602  train
1668    washing hands  -ChLS3YLStk          94       104  train
2666    washing hands  -LUN6528w3I          28        38  train
3582    washing hands  -TAINJnhrvU           0        10  train
...               ...          ...         ...       ...    ...
522293  washing hands  xZd0YH8C2F8          77        87  train
522625  washing hands  xbDodTSD6zE         114       124  train
526417  washing hands  y3NbJsrCecI          68        78  train
528324  washing hands  yFjrrLoPZ4s           2        12  train
531919  washing hands  yh9aL3dGuBQ           5        15  train

[973 rows x 5 columns]
```

For submission 6, the target label is “cleaning toilet”, there are total of 730 training videos for this label as shown below:

```
[ ]  print(data[data.label == ('cleaning toilet')])

                  label   youtube_id  time_start  time_end  split
220     cleaning toilet  -0o2B1AEdoQ           7        17  train
1724    cleaning toilet  -D9Azl-MLjI           0        10  train
2556    cleaning toilet  -Ke-6s7IBkU          23        33  train
3083    cleaning toilet  -OjYE3Z0vPY         144       154  train
3124    cleaning toilet  -P8Hq7Nc_lQ          93       103  train
...                ...          ...         ...       ...    ...
539893  cleaning toilet  zfeLD9P4-q0           2        12  train
541032  cleaning toilet  zoNd629rwOY         121       131  train
541751  cleaning toilet  zuTPOKPQog0           4        14  train
541770  cleaning toilet  zufm7uGFkuY          66        76  train
542074  cleaning toilet  zxL44Ksu1XQ         168       178  train

[730 rows x 5 columns]
```

For submission 8, the target label is “cleaning windows”, there are total of 907 training videos for this label as shown below:

```
print(data[data.label == ('cleaning windows')])
```

```
                 label   youtube_id  time_start  time_end  split
563     cleaning windows  -3as098jwNs           0        10  train
1040    cleaning windows  -7MqfSPKj6c           2        12  train
1303    cleaning windows  -9QfMtbGuq0           6        16  train
1563    cleaning windows  -Bk9MdOqzTY          23        33  train
1983    cleaning windows  -FOfyuyabWQ          16        26  train
...                  ...          ...         ...       ...    ...
542454  cleaning windows  zgNV3xKWStA          86        96  train
542733  cleaning windows  ziMV2sYVg14           5        15  train
543586  cleaning windows  zoxvVbxOFws           1        11  train
544340  cleaning windows  zvKPwIVjhOA         229       239  train
544373  cleaning windows  zvWVe2cIAQ4           1        11  train

[907 rows x 5 columns]
```

For submission 10, the target label is “Washing feet”, there are total of 745 training videos for this label as shown below:

```
print(data[data.label == ('washing feet')])
```

```
                label   youtube_id  time_start  time_end  split
902     washing feet  -6F0zYW3gUQ         110       120  train
1427    washing feet  -AR-hLSb2Bk          28        38  train
2393    washing feet  -J-DV0OBshM         118       128  train
3635    washing feet  -TPM9qZU7ZQ          15        25  train
4232    washing feet  -ZFww1qGkJs         118       128  train
...              ...          ...         ...       ...    ...
541973  washing feet  zdHU8kuiX98         349       359  train
542346  washing feet  zfW-iqH5fc0          25        35  train
542398  washing feet  zfpK0ulWcT8          42        52  train
544485  washing feet  zwUuMdt295Y          99       109  train
544525  washing feet  zwojAMytFyo         358       368  train

[745 rows x 5 columns]
```

I downloaded all the videos with label of “falling off chair” to use as my training data, also I downloaded the same number of videos with label that isn’t “falling off chair” also to used during my training process. So, there are about 1600 videos used to train my model.

For submission 3, I downloaded all the videos with label of “coughing” to use as my training data, also I downloaded the same number of videos with label that isn’t “coughing” also to use during my training process. So, there are about 1100 videos used to train my model.

For submission 5, I downloaded all the videos with label of “washing hands” to use as my training data, also I downloaded the same number of videos with label that isn’t “washing hands” also to use during my training process. So, there are about 1430 videos used to train my model.

For submission 6, I downloaded all the videos with label of “cleaning toilet” to use as my training data, also I downloaded the same number of videos with label that isn’t “cleaning toilet” also to use during my training process. So, there are about 1100 videos used to train my model.

For submission 8, I downloaded all the videos with label of “cleaning windows” to use as my training data, also I downloaded the same number of videos with label that isn’t “cleaning windows” also to use during my training process. So, there are about 1800 videos used to train my model.

For submission 10, I downloaded all the videos with label of “washing feet” to use as my training data, also I downloaded the same number of videos with label that isn’t “washing feet” also to use during my training process. So, there are about 1500 videos used to train my model.

6. Model Training and Performance

For my base model, I used VGG16, with input shape (150, 150, 3). I added layers such as below:

```
conv_base = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(150, 150, 3))
conv_base.trainable = False

def action_model(shape=(NBFRAME, 150, 150, 3), nbout=2):

    # Flatten output of conv_base
    mod = Sequential()
    mod.add(conv_base)
    mod.add(GlobalMaxPool2D())
    # Build our model for training
    model = Sequential()
    model.add(TimeDistributed(mod, input_shape=shape))
    # LSTM for time series
    model.add(GRU(64))
    # Build the classifier
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(nbout, activation='sigmoid'))
    return model
```

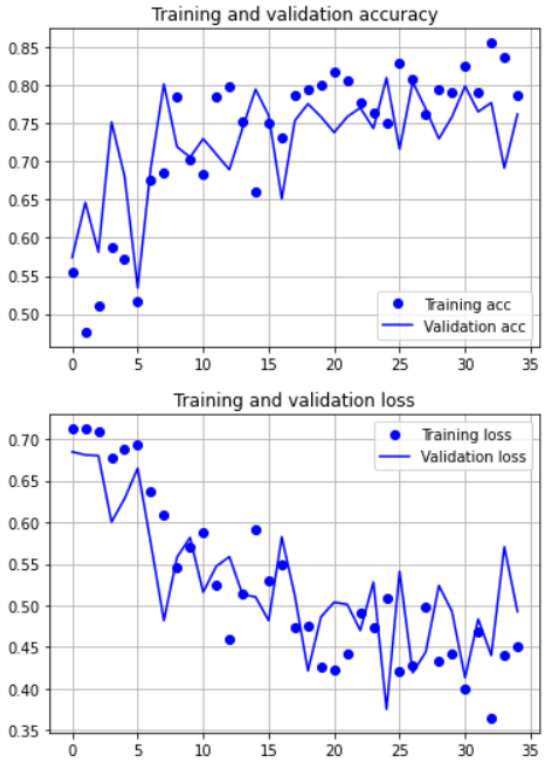
The optimizer I used was Adam, with learning rate 0.001 as shown below:

```
from keras.optimizers import Adam

optimizer= Adam(0.001)
model.compile(optimizer=optimizer ,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

I choose 35 as the number of my epochs, and 20 steps per epoch. Initially I started with 20 epochs and resulted in underfitting, so I changed to 50 epochs, which resulted in overfitting. Based on the accuracy and loss graphs, I could tell around 30 or 40 epochs was my best result, so I ended up with 35 epochs.

As the results, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:



As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 75%. The best I've gotten was 78% and worst was 56%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

In submission 3, for my base model, I used Xception, with input shape (150, 150, 3). I added layers such as below:

```
conv_base = Xception(weights='imagenet',
                     include_top=False,
                     input_shape=(150, 150, 3))
conv_base.trainable = False

def action_model(shape=(NBFRAME, 150, 150, 3), nbout=2):

    # Flatten output of conv_base
    mod = Sequential()
    mod.add(conv_base)
    mod.add(GlobalMaxPool2D())
    # Build our model for training
    model = Sequential()
    model.add(TimeDistributed(mod, input_shape=shape))
    # LSTM for time series
    model.add(LSTM(64))
    # Build the classifier
    # model.add(Dense(1024, activation='relu'))
    # model.add(Dropout(.5))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(nbout, activation='sigmoid'))
    return model
```

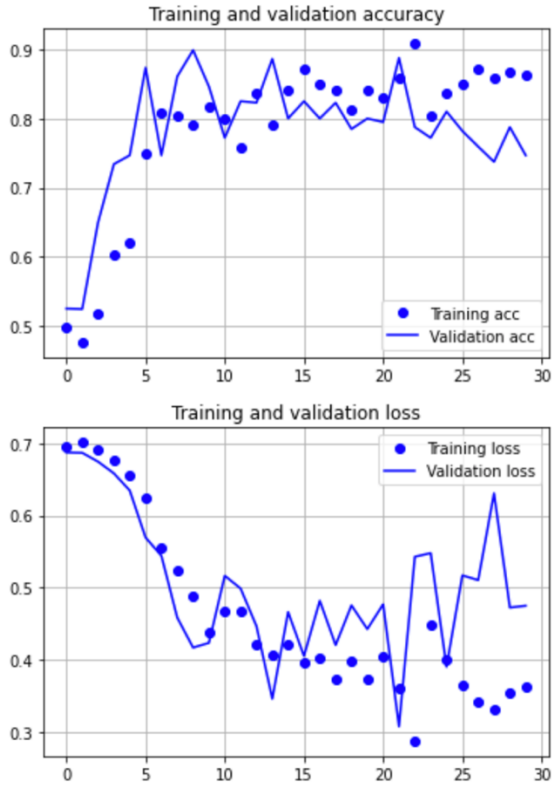
The optimizer I used was Adam, with learning rate 0.0001 as shown below:

```
from keras.optimizers import Adam

optimizer= Adam(0.0001)
model.compile(optimizer=optimizer ,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

I choose 30 as the number of my epochs, and 30 steps per epoch. Initially I started with 20 epochs and resulted in underfitting, so I changed to 40 epochs, which resulted in overfitting. Based on the accuracy and loss graphs, I could tell around 30 or 40 epochs was my best result, so I ended up with 30 epochs.

As the results, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:



As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 83%. The best I've gotten was 83% and worst was 56%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

In submission 5, submission 6, submission 8 and submission 10, for my base model, I used Xception, with input shape (150, 150, 3). I added layers such as below:

```
conv_base = Xception(weights='imagenet',
                     include_top=False,
                     input_shape=(150, 150, 3))
conv_base.trainable = False

def action_model(shape=(NBFRAME, 150, 150, 3), nbout=2):

    # Flatten output of conv_base
    mod = Sequential()
    mod.add(conv_base)
    mod.add(GlobalMaxPool2D())
    # Build our model for training
    model = Sequential()
    model.add(TimeDistributed(mod, input_shape=shape))
    # LSTM for time series
    model.add(LSTM(64))
    # Build the classifier
    # model.add(Dense(1024, activation='relu'))
    # model.add(Dropout(.5))
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(.5))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(nbout, activation='sigmoid'))
    return model
```

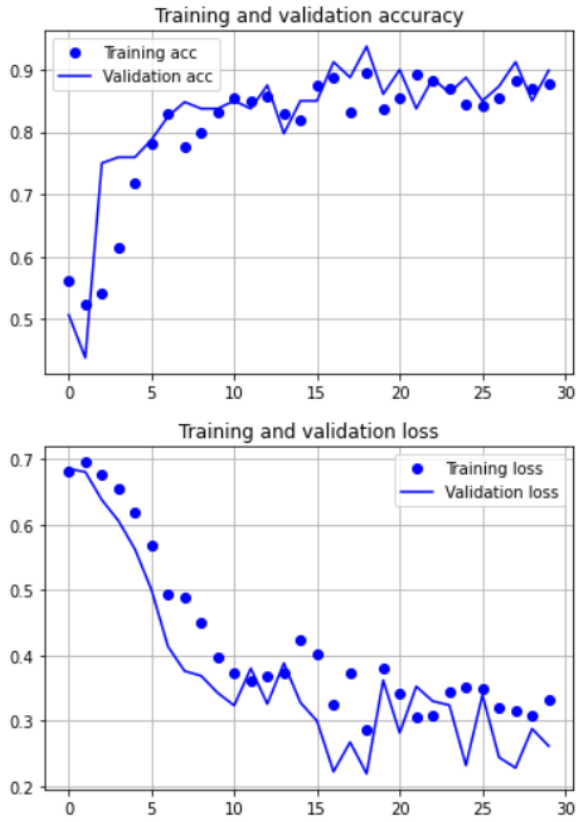
The optimizer I used was Adam, with learning rate 0.0001 as shown below:

```
from keras.optimizers import Adam

optimizer= Adam(0.0001)
model.compile(optimizer=optimizer ,
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

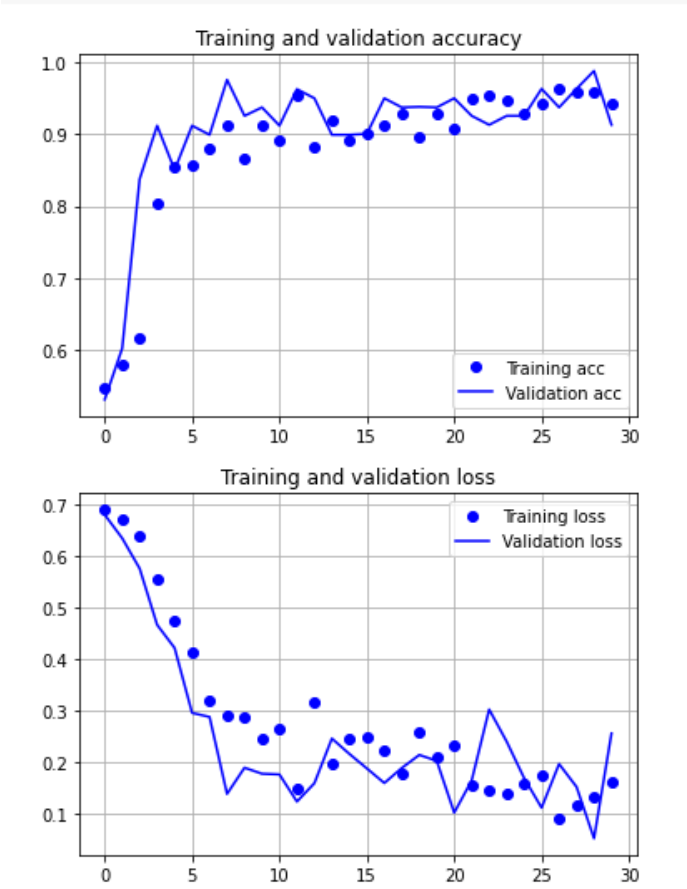
I choose 30 as the number of my epochs, and 30 steps per epoch. Initially I started with 25 epochs and resulted in underfitting, so I changed to 45 epochs, which resulted in overfitting. Based on the accuracy and loss graphs, I could tell around 30 or 40 epochs was my best result, so I ended up with 30 epochs.

As the results, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:



As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 84%. The best I've gotten was 84% and worst was 65%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

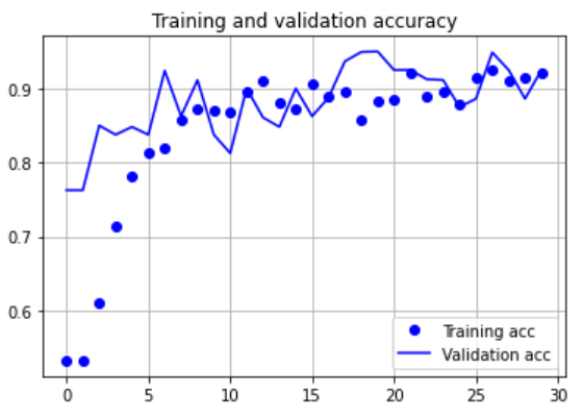
For submission 6, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:

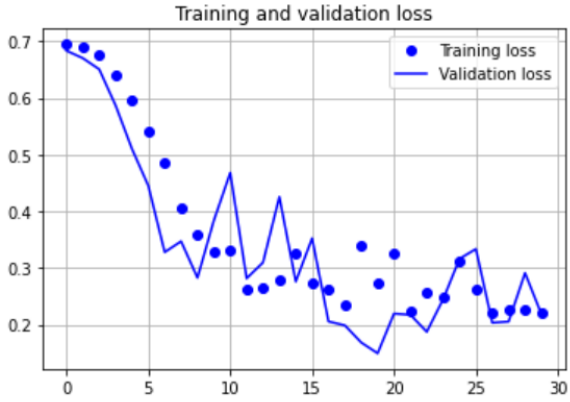


As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 96%. The best I've gotten was 96% and worst was 74%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

For submission 8, I choose 30 as the number of my epochs, and 35 steps per epoch. Initially I started with 25 epochs and resulted in underfitting, so I changed to 45 epochs, which resulted in overfitting. Based on the accuracy and loss graphs, I could tell around 30 or 40 epochs was my best result, so I ended up with 35 epochs.

As the results, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:

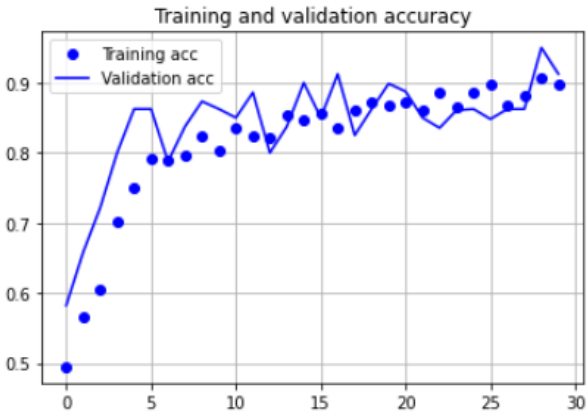


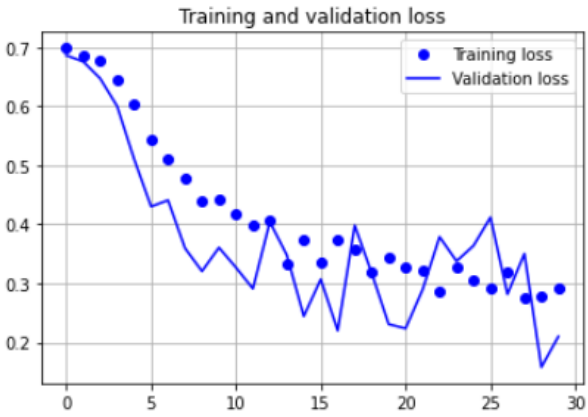


As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 86%. The best I've gotten was 86% and worst was 56%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

For submission 10, I choose 30 as the number of my epochs, and 35 steps per epoch. Initially I started with 25 epochs and resulted in underfitting, so I changed to 45 epochs, which resulted in overfitting. Based on the accuracy and loss graphs, I could tell around 30 or 40 epochs was my best result, so I ended up with 35 epochs.

As the results, my model's training and validation accuracy increased and loss decreased as shown in the graphs below:





As you can see, there was no overfit or underfit in both of the categories. And my model accuracy ended up around 83%. The best I've gotten was 83% and worst was 62%. I think a key factor about this model is the quality of videos. I manually looked through some of the training videos, many of them either had bad quality, or bad lighting, or only part of the human body was shown. This could strongly affect the accuracy of my model.

7. Performance on YouTube Videos

7.1 How to detect “moments” of target action

I first choose some videos to test the performance of my model, by doing that, I downloaded those videos from YouTube. Then, I loaded my saved model .h5 file, and made a function where it will check the video with my model. For every two seconds, the model will check if the video contains the target action, if it does, then it will save the 2 second video clip’s information such as start and end time, video ID, to a .json file.

7.2 Video Found “Moments” in iLab Website

The label of my target action is “fall”, in total I found 69 moments in 12 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-2” in the observer filed to see these videos and moments.

For submission 3, the label of my target action is “coughing”, in total I found 524 moments in 7 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-3” in the observer filed to see these videos and moments.

For submission 5, the label of my target action is “Hand washing”, in total I found 2909 moments in 48 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-5” in the observer filed to see these videos and moments. This data will be uploaded by the professor or TA in this submission due to issues of the iLab website.

For submission 6, the label of my target action is “Cleaning bathroom”, in total I found 5713 moments in 19 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-6” in the observer filed to see these videos and moments. This data will be uploaded by the professor or TA in this submission due to issues of the iLab website.

For submission 8, the label of my target action is “Cleaning windows glass doors etc”, in total I found 4089 moments in 26 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-8” in the observer filed to see these videos and moments. This data will be uploaded by the professor or TA in this submission due to issues of the iLab website.

For submission 10, the label of my target action is “Washing feet”, in total I found 2052 moments in 17 videos. You can simply search on the website with “CSCE636Spring2021-heswaggy-10” in the observer filed to see these videos and moments. This data will be uploaded by the professor or TA in this submission due to issues of the iLab website.

7.3 Performance Accuracy

For all submissions, I defined False-Positive Rate as the number of testing clips that got an accuracy score above 80% but does not contain my target label action. In submission 2, my FPR was about 45%; for submission 3, my FPR was about 5%; for submission 5, my FPR was about 2%; for submission 6, my FPR was about 15%; for submission 8, my FPR was about 16%, for submission 10, my FPR was about 8%. I choose about 50 clips that contains my target actions to test my FPR for each submission, the formula to get the estimation on my FPR is: # of False-Positive / # of clips tested.

For all submissions, I defined False-Negative Rate as the number of testing clips that got an accuracy score below 80% but does contain my target label action. In submission 2, my FNR was about 24%; for submission 3, my FNR was about 3%; for submission 5, my FNR was about 2%; for submission 6, my FNR was about 11%; for submission 8, my FNR was about 8%, for submission 10, my FNR was about 4%. I choose about 50 clips that contains my target actions to test my FNR for each submission, the formula to get the estimation on my FNR is: # of False-Negative / # of clips tested.

In my opinion, the FPR of the model started too high at 45% for the initial submission, but after fixing my model and implementation, it quickly decreased under 10% for most of future submissions, which satisfies my expectation. The FNR also started too high at 24%, which means my model was not detecting my action well, but after making adjustments such as downloading higher video quality, adding more training data, fitting the best epochs, it decreased to below 10% for all future submissions. This also satisfies my expectation.

7.4 Performance: Efficacy of Action Detection

After training my model, I used it to detect YouTube videos that contains my target action. I first find videos that contains my target action on YouTube, then get the video ID for downloading purpose. After downloading all the videos, I feed the video into my trained model, with the algorithm that the model will check the video in 2 second clips, if my target action appears with an accuracy above 80% in those 2 second, it will generate a JSON block into the JSON file with all the information needed.

By estimation, my program can detect 6000-10000 moments per hour, because it takes about 20 minutes to detect an hour-long video, which can product about 2000-3500 moments.

The efficiency of my algorithm is satisfactory, I save a lot of time downloading videos because I can put all video IDs in one code and download them together.

However, each time I test on a new video I need to change the JSON information and also manually change the target testing video. Therefore, while it is not the worse algorithm, it can get a lot better with smarter algorithm.

8. Improve Accuracy and Efficiency

8.1 Improve Accuracy of Model on Youtube Videos

To better improve my model accuracy on practical videos from Youtube, I improved video quality on both training and testing videos. The software I used to download videos from Youtube is called “youtube_dl”. I made this plug-in to choose the highest video quality possible when it downloads a video. For my testing videos, I manually looked for videos with high resolution such as 1080p.

This method worked pretty well, I found out the better quality of videos to train/test, the better accuracy it will get. For example, the accuracy average for my submission 2 was around 75% with high FPR and FNR, but after using this method, the accuracy went up to about 85% with low FPR and FNR.

To view this improvement on the iLab website, you can simply check the accuracy score of my moments, which it sometimes gotten to above 95% in later submissions.

8.2 Improve Efficiency of Action/Emotion Detection

To improve my efficiency of detection in Youtube videos, I added more false training videos and looked for less true-positive dominated videos on Youtube. During the training process of my model, I added the same number of false video samples as the true video samples, which makes my training data half true and half false. This will help my model detect better since it has seen the same number of false samples and true samples. To find better videos on Youtube for testing, I looked for videos that contains some length of my target actions while it also contains many other actions. This will test the true performance of my model.

This method worked out very well, after adding more false samples during my training process and testing process, my FPR went down from about 45% to as low as 2%. This is clearly the right way to solve this issue, and a great method to improve performance.

To view this improvement on the iLab website, you can simply check the accuracy score of my moments, which it sometimes gotten to above 95% in later submissions.

9. **Code in TAMU Github**
   Github Link: https://github.tamu.edu/hwy893747147/CSCE636-Spring2021-ProjectSubmission2-WangyangHe

10. References

[1] Mubashir, M., Shao, L., & Seed, L. (2013). A survey on fall detection: Principles and approaches. Neurocomput., 100, 144–152. https://doi.org/10.1016/j.neucom.2011.09.037

[2] N. Noury et al., "Fall detection - Principles and Methods," 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007, pp. 1663-1666, doi: 10.1109/IEMBS.2007.4352627.

[3] Igual, R., Medrano, C. & Plaza, I. Challenges, issues and trends in fall detection systems. BioMed Eng OnLine 12, 66 (2013). https://doi.org/10.1186/1475-925X-12-66

[4] Xinguo Yu, "Approaches and principles of fall detection for elderly and patient," HealthCom 2008 - 10th International Conference on e-health Networking, Applications and Services, 2008, pp. 42-47, doi: 10.1109/HEALTH.2008.4600107.

[5] Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., & Vecchio, A. (2012, 2012/12/01/). A smartphone-based fall detection system. Pervasive and Mobile Computing, 8(6), 883-899. https://doi.org/https://doi.org/10.1016/j.pmcj.2012.08.003

[6] J. Liu, M. You, Z. Wang, G. Li, X. Xu and Z. Qiu, "Cough detection using deep neural networks," 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2014, pp. 560-563, doi: 10.1109/BIBM.2014.6999220.

[7] F. Barata, K. Kipfer, M. Weber, P. Tinschert, E. Fleisch and T. Kowatsch, "Towards Device-Agnostic Mobile Cough Detection with Convolutional Neural Networks," 2019 IEEE International Conference on Healthcare Informatics (ICHI), 2019, pp. 1-11, doi: 10.1109/ICHI.2019.8904554.

[8] J. Amoh and K. Odame, "DeepCough: A deep convolutional neural network in a wearable cough detection system," 2015 IEEE Biomedical Circuits and Systems Conference (BioCAS), 2015, pp. 1-4, doi: 10.1109/BioCAS.2015.7348395.

[9] D. Berardini, L. Migliorelli, S. Moccia, M. Naldini, G. D. Angelis and E. Frontoni, "Evaluating the autonomy of children with autism spectrum disorder in washing hands: a deep-learning approach," 2020 IEEE Symposium on Computers and Communications (ISCC), 2020, pp. 1-7, doi: 10.1109/ISCC50000.2020.9219648.

[10] Yin J, Apuroop KGS, Tamilselvam YK, Mohan RE, Ramalingam B, Le AV. Table Cleaning Task by Human Support Robot Using Deep Learning Technique. Sensors. 2020; 20(6):1698. https://doi.org/10.3390/s20061698

[11] W. Chen, C. A. Betancourt Baca and C. Tou, "LSTM-RNNs combined with scene information for human activity recognition," 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom), 2017, pp. 1-6, doi: 10.1109/HealthCom.2017.8210846.