Action Detection for Smart Homes

Wangyang He

UIN: 625004872

Nickname: heswaggy

Submission 3

03/11/2021

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.

1. Topic

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

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

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.

3. **Related Works**

https://www.sciencedirect.com/science/article/pii/S0925231212003153
https://ieeexplore.ieee.org/abstract/document/4352627
https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-12-66
https://ieeexplore.ieee.org/abstract/document/4600107
https://www.sciencedirect.com/science/article/pii/S1574119212000983
(Not enough time to read and summarize these papers, will do in future submissions.)

Submission 3:
https://ieeexplore.ieee.org/abstract/document/6999220?casa_token=qJrAOvJp19UAAAAA:A74-zij2jCB-KZqrZVs_jMJr4FCRjy-h26CYKQ3wJ5jXar2S1QovzL4pF-w1MBCTb3HxND2S
https://ieeexplore.ieee.org/abstract/document/8904554?casa_token=FvbZznpB-3IAAAAA:yWg84GP675vWCZEe8aeFs50Ck6evjm4YeEUAOc41l2i_UMGz6nZMGteqfZywlYxCd-d9KPZu
https://ieeexplore.ieee.org/abstract/document/7348395
https://arxiv.org/abs/2006.07743
(Not enough time to read and summarize these papers, will do in future submissions.)

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
_________________________________________________________________
```

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]
```

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.

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
```

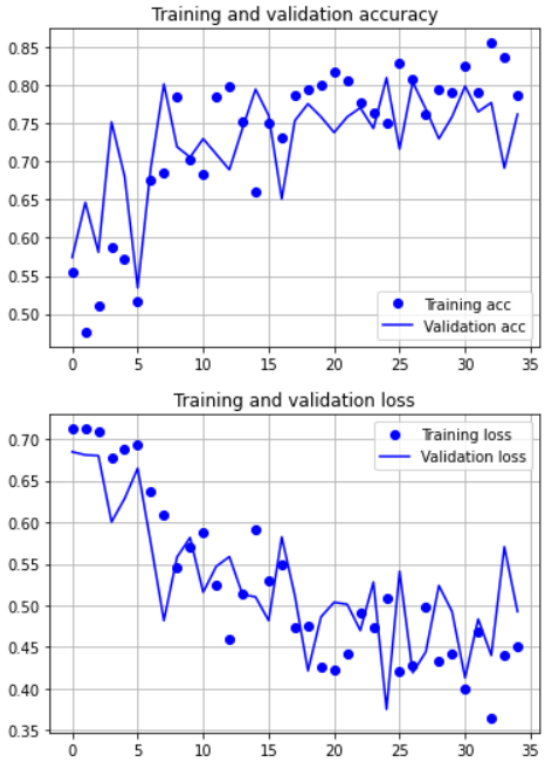
The optimizer I used was Adam, with learning rate 0.001 as shown below:

```
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
```

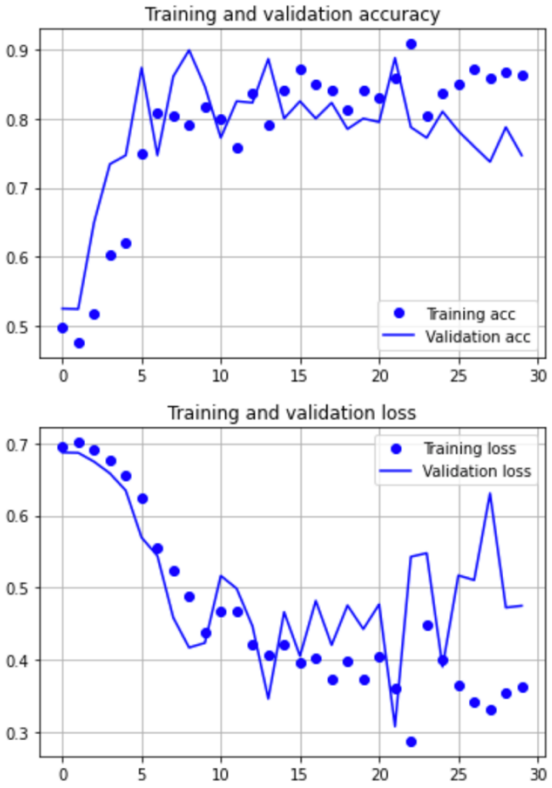
The optimizer I used was Adam, with learning rate 0.0001 as shown below:

```
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.

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.

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.

7.3 Performance Accuracy

(Will do later)

7.4 Performance: Efficacy of Action Detection

(Will do later)

8. **Improve Accuracy and Efficiency**
   (will do some research on it in the future)

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