Natural Network Project Demo

March, 22, 2020

Yijun Zhang 629009140

1. Topic

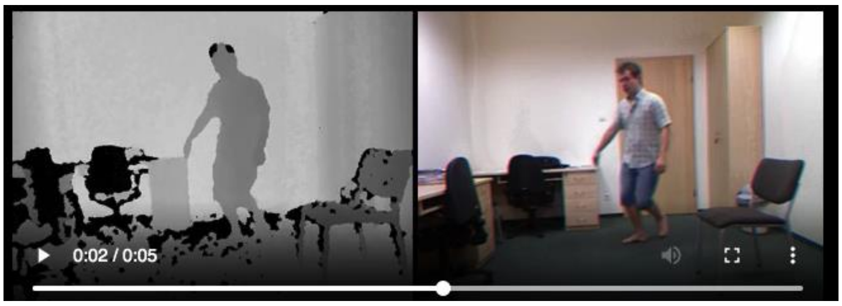
Falling Detection.

The figure shows how the probability of falling changes according to time of the video.

2. Dataset

1) UR Fall Detection Dataset: http://fenix.univ.rzeszow.pl/~mkepski/ds/uf.html

This dataset includes 30 videos of falling and 40 videos of daily activity(don't have falling).
Falling videos name like: "fall-01-cam0"
Daily activity videos like: "adl-01-cam0"



It also includes label file urfall-cam0-falls.csv and urfall-cam0-adls.csv .

Extracted features from depth maps are stored in CSV format. Each row contains one sample of data corresponding to one depth image. The columns from left to right are organized as follows:

- sequence name - camera name is omitted, because all of the samples are from the front camera ('fall-01-cam0-d' is 'fall-01', 'adl-01-cam0-d' is 'adl-01' and so on).
- frame number - corresponding to number in sequence.
- label - describes human posture in the depth frame; '-1' means person is not lying, '1' means person is lying on the ground; '0' is temporary pose, when person "is falling", we don't use '0' frames in classification.

I transform the label using the rule:

- For the falling videos, original label '-1' means 0 (not falling), and orginal label '0' and '1' means 1 (falling).
- For the adl videos, all frames are labeled 0 (not falling).

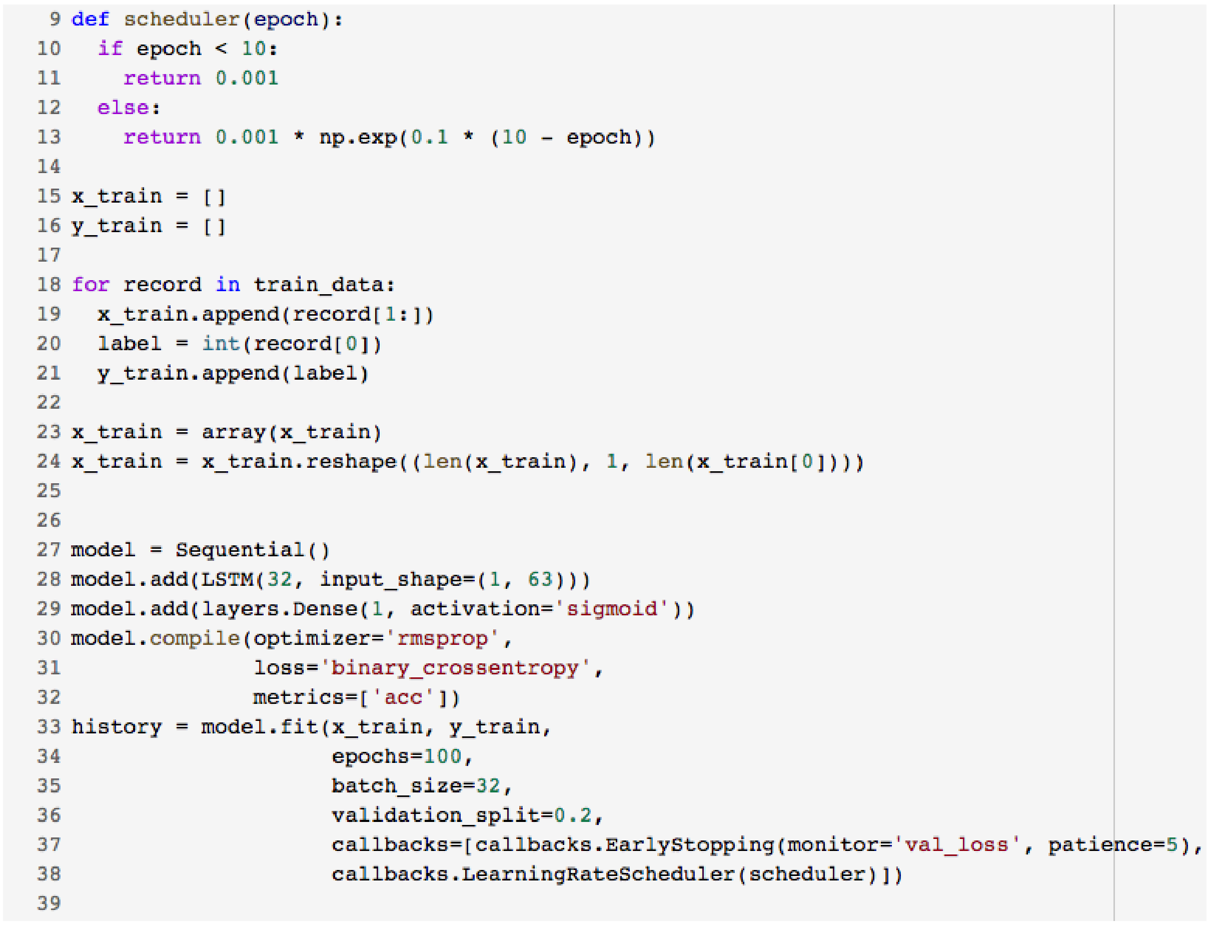
Currently, I only used one dataset. In the following days, I will train the model using more dataset.

3. LSTM model

3.1. Architecture

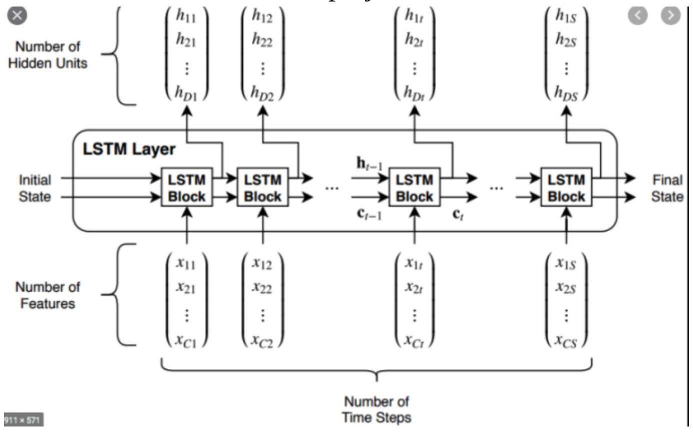
This model builds one hidden layer of LSTM and one dense layer as output layer.

A snippet of code of constructing the model is shown as follows:



```
def scheduler(epoch):
  if epoch < 10:
    return 0.001
  else:
    return 0.001 * np.exp(0.1 * (10 - epoch))

x_train = []
y_train = []

for record in train_data:
  x_train.append(record[1:])
  label = int(record[0])
  y_train.append(label)

x_train = array(x_train)
x_train = x_train.reshape((len(x_train), 1, len(x_train[0])))


model = Sequential()
model.add(LSTM(32, input_shape=(1, 63)))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=100,
                    batch_size=32,
                    validation_split=0.2,
                    callbacks=[callbacks.EarlyStopping(monitor='val_loss', patience=5),
                    callbacks.LearningRateScheduler(scheduler)])

```

The architecture used in this project:



3.2. Input: Shape of Tensor

x_train shape is (17283, 1, 63)

x_test shape is (2160, 1, 63)

I transform it into a float32 array of shape (17283, 63)

3.3. Output: Shape of Tensor

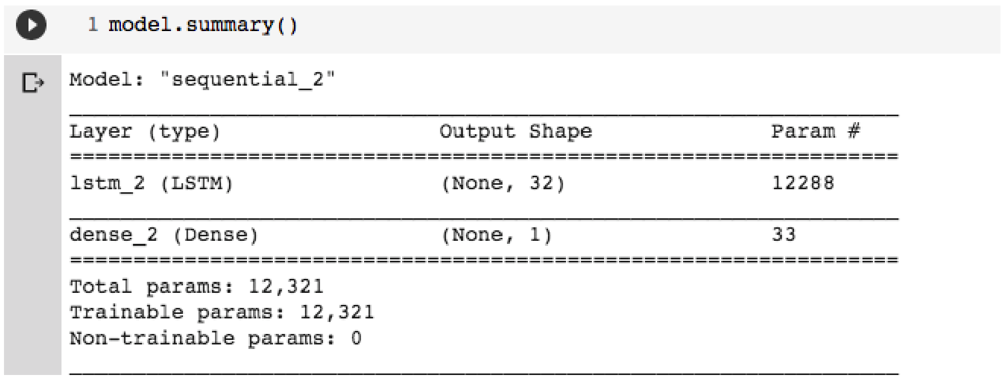
y_train shape is (17283,)

y_test shape is (2160,)

3.4. Shape of Output Tensor of Each layer

Output of first hidden layer: (17283, 32)

Output of output layer: (17283, 1)



```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 32)                12288
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 12,321
Trainable params: 12,321
Non-trainable params: 0
_________________________________________________________________
```

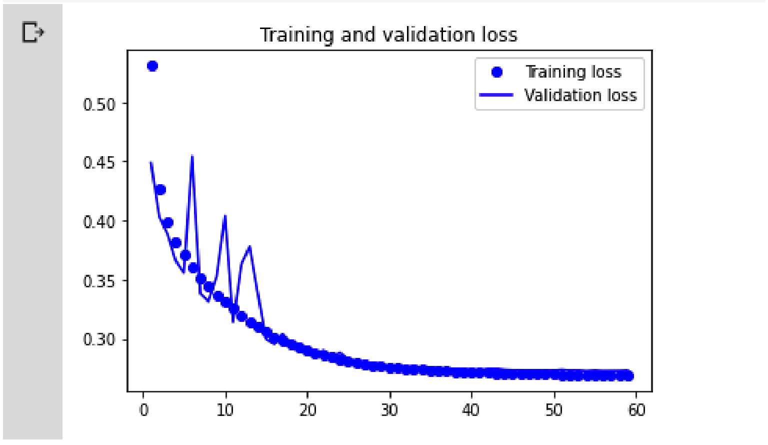
4. Hyperparameters

4.1. List of Hyperparameters

In this project, there are three hyperparameters I choose to tune: epochs.

4.2. Range of Value of Hyperparameters Tried

Epochs: I use learning rate decay(function scheduler), which means after the 10th epoch, learning rate will decrease gradually. I also use early stopping, which means when validation loss is not decrease for 5 epochs in a row.



5. Annotated Code

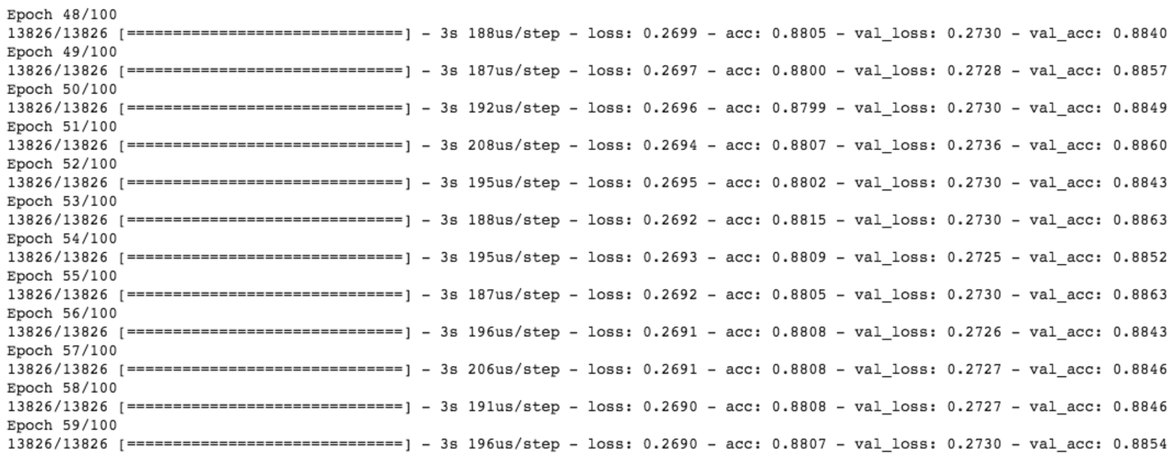
Github repository:
https://github.com/YJZFlora/Fall_Detection_Deep_Learning_Model

Training code: model.py

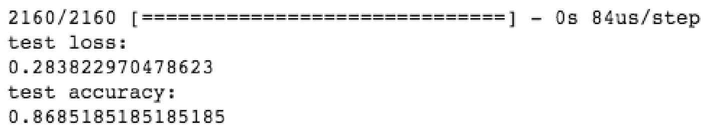
Execute code: execute_model.py

6. Training and Testing performance

```
13826/13826 [==============================] - 11s 828us/step - loss: 0.5312 - acc: 0.7591 - val_loss: 0.4487 - val_acc: 0.8045
Epoch 2/100
13826/13826 [==============================] - 3s 204us/step - loss: 0.4272 - acc: 0.8069 - val_loss: 0.4035 - val_acc: 0.8198
Epoch 3/100
13826/13826 [==============================] - 3s 196us/step - loss: 0.3987 - acc: 0.8174 - val_loss: 0.3883 - val_acc: 0.8221
Epoch 4/100
13826/13826 [==============================] - 3s 203us/step - loss: 0.3818 - acc: 0.8206 - val_loss: 0.3660 - val_acc: 0.8325
Epoch 5/100
13826/13826 [==============================] - 3s 210us/step - loss: 0.3707 - acc: 0.8276 - val_loss: 0.3556 - val_acc: 0.8386
Epoch 6/100
13826/13826 [==============================] - 3s 206us/step - loss: 0.3602 - acc: 0.8299 - val_loss: 0.4543 - val_acc: 0.7880
Epoch 7/100
13826/13826 [==============================] - 3s 191us/step - loss: 0.3515 - acc: 0.8351 - val_loss: 0.3380 - val_acc: 0.8397
Epoch 8/100
13826/13826 [==============================] - 3s 205us/step - loss: 0.3449 - acc: 0.8368 - val_loss: 0.3314 - val_acc: 0.8473
Epoch 9/100
13826/13826 [==============================] - 3s 196us/step - loss: 0.3369 - acc: 0.8447 - val_loss: 0.3531 - val_acc: 0.8421
Epoch 10/100
13826/13826 [==============================] - 3s 201us/step - loss: 0.3314 - acc: 0.8436 - val_loss: 0.4037 - val_acc: 0.8088
Epoch 11/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.3259 - acc: 0.8462 - val_loss: 0.3138 - val_acc: 0.8588
Epoch 12/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.3192 - acc: 0.8509 - val_loss: 0.3629 - val_acc: 0.8244
Epoch 13/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.3138 - acc: 0.8540 - val_loss: 0.3780 - val_acc: 0.8296
Epoch 14/100
13826/13826 [==============================] - 3s 195us/step - loss: 0.3099 - acc: 0.8544 - val_loss: 0.3381 - val_acc: 0.8461
Epoch 15/100
13826/13826 [==============================] - 3s 198us/step - loss: 0.3057 - acc: 0.8587 - val_loss: 0.2998 - val_acc: 0.8675
Epoch 16/100
13826/13826 [==============================] - 3s 206us/step - loss: 0.3012 - acc: 0.8605 - val_loss: 0.2953 - val_acc: 0.8707
Epoch 17/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.2975 - acc: 0.8642 - val_loss: 0.3038 - val_acc: 0.8704
Epoch 18/100
13826/13826 [==============================] - 3s 209us/step - loss: 0.2954 - acc: 0.8634 - val_loss: 0.2958 - val_acc: 0.8750
Epoch 19/100
13826/13826 [==============================] - 3s 193us/step - loss: 0.2922 - acc: 0.8667 - val_loss: 0.2916 - val_acc: 0.8759
Epoch 20/100
13826/13826 [==============================] - 3s 194us/step - loss: 0.2905 - acc: 0.8672 - val_loss: 0.2868 - val_acc: 0.8782
Epoch 21/100
13826/13826 [==============================] - 3s 191us/step - loss: 0.2879 - acc: 0.8682 - val_loss: 0.2871 - val_acc: 0.8739
Epoch 22/100
13826/13826 [==============================] - 3s 193us/step - loss: 0.2862 - acc: 0.8683 - val_loss: 0.2900 - val_acc: 0.8765
Epoch 23/100
13826/13826 [==============================] - 3s 193us/step - loss: 0.2844 - acc: 0.8701 - val_loss: 0.2841 - val_acc: 0.8782
Epoch 24/100
Epoch 25/100
13826/13826 [==============================] - 3s 189us/step - loss: 0.2805 - acc: 0.8744 - val_loss: 0.2823 - val_acc: 0.8802
Epoch 26/100
13826/13826 [==============================] - 3s 197us/step - loss: 0.2795 - acc: 0.8739 - val_loss: 0.2812 - val_acc: 0.8817
Epoch 27/100
13826/13826 [==============================] - 3s 189us/step - loss: 0.2786 - acc: 0.8738 - val_loss: 0.2797 - val_acc: 0.8802
Epoch 28/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.2775 - acc: 0.8753 - val_loss: 0.2798 - val_acc: 0.8817
Epoch 29/100
13826/13826 [==============================] - 3s 199us/step - loss: 0.2772 - acc: 0.8763 - val_loss: 0.2787 - val_acc: 0.8849
Epoch 30/100
13826/13826 [==============================] - 3s 206us/step - loss: 0.2762 - acc: 0.8760 - val_loss: 0.2768 - val_acc: 0.8846
Epoch 31/100
13826/13826 [==============================] - 3s 200us/step - loss: 0.2757 - acc: 0.8772 - val_loss: 0.2767 - val_acc: 0.8857
Epoch 32/100
13826/13826 [==============================] - 3s 195us/step - loss: 0.2749 - acc: 0.8781 - val_loss: 0.2760 - val_acc: 0.8860
Epoch 33/100
13826/13826 [==============================] - 3s 213us/step - loss: 0.2741 - acc: 0.8795 - val_loss: 0.2767 - val_acc: 0.8814
Epoch 34/100
13826/13826 [==============================] - 3s 203us/step - loss: 0.2737 - acc: 0.8785 - val_loss: 0.2757 - val_acc: 0.8814
Epoch 35/100
13826/13826 [==============================] - 3s 224us/step - loss: 0.2730 - acc: 0.8789 - val_loss: 0.2757 - val_acc: 0.8854
Epoch 36/100
13826/13826 [==============================] - 3s 203us/step - loss: 0.2726 - acc: 0.8786 - val_loss: 0.2752 - val_acc: 0.8846
Epoch 37/100
13826/13826 [==============================] - 3s 196us/step - loss: 0.2723 - acc: 0.8780 - val_loss: 0.2744 - val_acc: 0.8852
Epoch 38/100
13826/13826 [==============================] - 3s 207us/step - loss: 0.2720 - acc: 0.8796 - val_loss: 0.2746 - val_acc: 0.8857
Epoch 39/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.2717 - acc: 0.8799 - val_loss: 0.2742 - val_acc: 0.8826
Epoch 40/100
13826/13826 [==============================] - 3s 191us/step - loss: 0.2713 - acc: 0.8793 - val_loss: 0.2737 - val_acc: 0.8849
Epoch 41/100
13826/13826 [==============================] - 3s 189us/step - loss: 0.2711 - acc: 0.8791 - val_loss: 0.2734 - val_acc: 0.8852
Epoch 42/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.2709 - acc: 0.8794 - val_loss: 0.2741 - val_acc: 0.8857
Epoch 43/100
13826/13826 [==============================] - 3s 197us/step - loss: 0.2705 - acc: 0.8807 - val_loss: 0.2745 - val_acc: 0.8869
Epoch 44/100
13826/13826 [==============================] - 3s 190us/step - loss: 0.2705 - acc: 0.8801 - val_loss: 0.2736 - val_acc: 0.8852
Epoch 45/100
13826/13826 [==============================] - 3s 195us/step - loss: 0.2702 - acc: 0.8796 - val_loss: 0.2732 - val_acc: 0.8840
Epoch 46/100
13826/13826 [==============================] - 3s 203us/step - loss: 0.2701 - acc: 0.8804 - val_loss: 0.2733 - val_acc: 0.8840
Epoch 47/100
13826/13826 [==============================] - 3s 199us/step - loss: 0.2699 - acc: 0.8798 - val_loss: 0.2729 - val_acc: 0.8863
```



```
Epoch 48/100
13826/13826 [==============================] - 3s 188us/step - loss: 0.2699 - acc: 0.8805 - val_loss: 0.2730 - val_acc: 0.8840
Epoch 49/100
13826/13826 [==============================] - 3s 187us/step - loss: 0.2697 - acc: 0.8800 - val_loss: 0.2728 - val_acc: 0.8857
Epoch 50/100
13826/13826 [==============================] - 3s 192us/step - loss: 0.2696 - acc: 0.8799 - val_loss: 0.2730 - val_acc: 0.8849
Epoch 51/100
13826/13826 [==============================] - 3s 208us/step - loss: 0.2694 - acc: 0.8807 - val_loss: 0.2736 - val_acc: 0.8860
Epoch 52/100
13826/13826 [==============================] - 3s 195us/step - loss: 0.2695 - acc: 0.8802 - val_loss: 0.2730 - val_acc: 0.8843
Epoch 53/100
13826/13826 [==============================] - 3s 188us/step - loss: 0.2692 - acc: 0.8815 - val_loss: 0.2730 - val_acc: 0.8863
Epoch 54/100
13826/13826 [==============================] - 3s 195us/step - loss: 0.2693 - acc: 0.8809 - val_loss: 0.2725 - val_acc: 0.8852
Epoch 55/100
13826/13826 [==============================] - 3s 187us/step - loss: 0.2692 - acc: 0.8805 - val_loss: 0.2730 - val_acc: 0.8863
Epoch 56/100
13826/13826 [==============================] - 3s 196us/step - loss: 0.2691 - acc: 0.8808 - val_loss: 0.2726 - val_acc: 0.8843
Epoch 57/100
13826/13826 [==============================] - 3s 206us/step - loss: 0.2691 - acc: 0.8808 - val_loss: 0.2727 - val_acc: 0.8846
Epoch 58/100
13826/13826 [==============================] - 3s 191us/step - loss: 0.2690 - acc: 0.8808 - val_loss: 0.2727 - val_acc: 0.8846
Epoch 59/100
13826/13826 [==============================] - 3s 196us/step - loss: 0.2690 - acc: 0.8807 - val_loss: 0.2730 - val_acc: 0.8854
```



```
2160/2160 [==============================] - 0s 84us/step
test loss:
0.283822970478623
test accuracy:
0.8685185185185185
```

7. Instruction on how to test trained model

Instruction:
https://github.com/YJZFlora/Fall_Detection_Deep_Learning_Model/blob/master/README.md

7.1. Install Dependencies

- python 3
- Keras
- Tensorflow
- pandas
- Numpy

7.2. Execution

```
python3 execute_model.py <directory of body landmarks for a video>
```

video demo: https://www.youtube.com/watch?v=r2CNC9QNPMg

7.3. Video Link

https://www.youtube.com/playlist?list=PLYkX3-wtcreE-4tlAK8zQ4hFrgpoR_OUC