**Midterm Report**

professor: Kindratenko, Volodymyr

Team: Xiyue Zhu, Hangzheng Lin, Ruike Zhu

**Introduction:**

Develop a model to detect human behavior is an interesting machine learning and computer vision project to work on. Moreover, a model that can detect fall behavior in real-time can effectively decrease the negative consequences of falls. To sum up, this project aims to develop a neuron network that can detect falling behavior in real-time.

**1. Existing results:**

As a start of the project, we read a few articles and rebuild some python codes on our computer:

Here are the links:

These results are in python and are easy to be run on our computers, so it is a good starting point for the project. I am going to introduce some of them that leaves a deep impression on me in a few words:

One of the codes is using data from an accelerometer that the user needs to wear and use different accelerometer and time as 2 dimensions for the data and stack them in different orders for multiple times. Then, they used convolutional neuron networks on the 2D feature for convolution can create correlations both for time and accelerometer.

Another code uses CNN than an RNN(recurrent neuron network), in sequential. Because RNN can take the input that is continuously correlated and CNN is good at image processing. They also get decent results, although RNN is too heavy for a real-time system.

Another code worked on classifying 6 different human behavior like running and sitting. They classified them into “moving behavior” and “non-moving behavior” first and then divided each behavior using 3 neuron networks. And in this way, they decreased the difficulty of the project,

Although most of the codes does not help us directly. But in the process of reading, rebuilding, and understanding them. We have got a lot of experience and skills.

**2. Shirui‘s code**

We finally got this code for the real starting point of our project:

Link:

[https://gitlab.engr.illinois.edu/shirui/fall\_detection\_shirui](https://gitlab.engr.illinois.edu/shirui/fall_detection_shirui" \t "_blank)

**2.1 introduction for the code:**

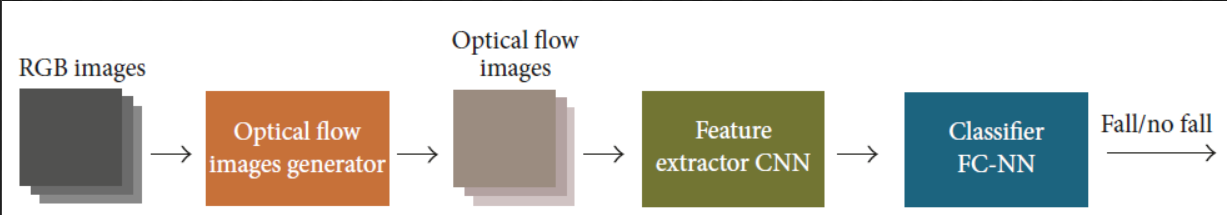
Here we have two different models: vgg and i3d model

**2.1.1 vgg model (CNN)**

Vgg is a Convolutional neuron network with specific hyperparameters. We use this for it shows competitive performance in computer vision.

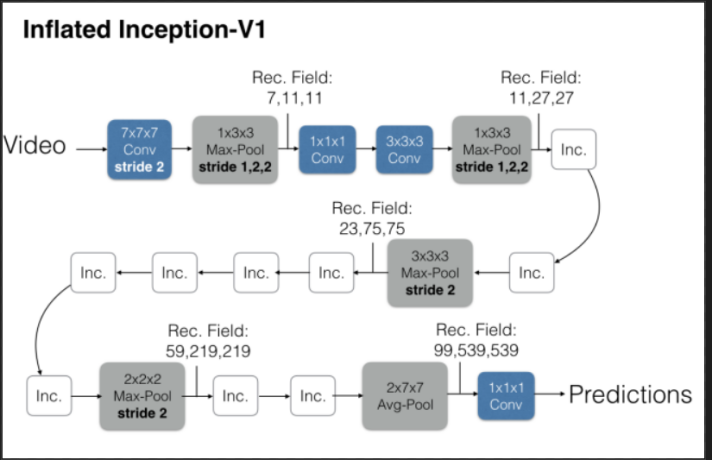
Convolutional neuron network first uses convolutional layers to extract deeper features of the images and use pooling layers to minimize the size of the extracted features. After that, we send the features into an ordinary neuron network of dense and dropout layers to get the final results.

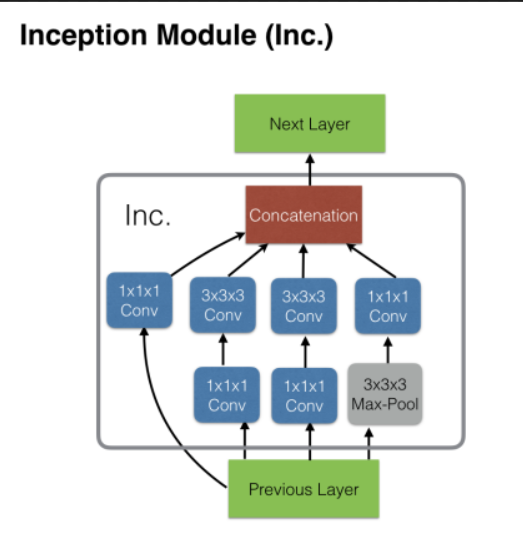
Here is the structure of the network:



**2.1.2 I3D inflated 3D model**

It is a similar model to CNN but it uses 3d convolution instead of normal convolutions,





Preprocess: stacked optical flow

Optical flow is can show the movement of each pixel and the behavior of falling have features in movement. This is why we use optical flow as a preprocess.

As we know, the video is continuous on-time, and the behavior is also continuous on time behavior. Therefore, it wasn’t enough to use one frame as input so we staked the optical flow of 10 frames as the final input of the model.

**2.2. what we do with it.**

First, since Shirui’s code only load a trained model from outside and do the testing, we build our own model of the same structure, and we train the neuron network on our own downloaded dataset. And then we do the testing on the same dataset, We get a pretty high accuracy although it is only in a small data set (about 1GB).

Although it is quite interesting to get our own network model on our PC. However, Shirui’s team has trained the neuron network on a huge dataset that is unlikely to be trained on our PC. So we used their trained model to get a better performance on other datasets.

**3. Openvino part**

**3.1 we learned how to run python in openvino**

Openvino is a tool suite developed by Intel based on its existing hardware platform, which can speed up the development of high-performance computer vision and deep learning vision applications. It supports deep learning on hardware accelerators of various Intel platforms. We will use this tool to speed up and optimize our model running.

Because there are not many complete tutorials about openvino on the Internet, we have learned the Chinese course of openvino from 51 website, and we have roughly understood the usage of openvino through the examples in the course. Then we run our own model with openvino. The specific steps are:

1. preprocess of images: we use shirui’s code to generate the optical flow, and stack the flow of 10 frames using the sliding window algorithm.

2. Then we convert the trained model into .h5 file

3. then use Keras, Tensorflow package to convert .h5 file into .pb file

4. After we get the .pb file, we run this model file in openvino by the command ‘mo.py’ in terminal to get the IR of the network. (IR is the intermediate representation of the network, including .xml (describing network topology) and .bin (including binary data of weight and deviation).

5. use IR files to run the model in openvino.

**3.2 combine shirui’s code with openvino**

Here shirui gives us two .h5 files, which refers to convolution neural network and fully connected network respectively. We load these two .h5 file to the model in part 3.1 to form our own network. And then we transform this model to .h5 file to get the .pb file, and did the same operation as step 4&5 above.

**3.3 Optical flow part**

We use stacked optical flow in the preprocessing part. Optical flow is the instantaneous velocity of the pixel motion of a space moving object in the observation image plane.

It is a method to find the corresponding relationship between the previous frame and the current frame by using the changes of pixels in the time domain and the correlation between adjacent frames in the image sequence, so as to calculate the motion information of objects between adjacent frames.

In general, the instantaneous change rate of the gray level on a specific coordinate point of a two-dimensional image plane is defined as the optical flow vector.

In short, optical flow means the instantaneous rate, or the displacement when the time interval is very small (like between two frames).

Besides, the video is continuous on time, and the behavior is also continuous on-time behavior. Therefore, it wasn’t enough to use one frame as input so we staked the optical flow of 10 frames as the final input of the model.

We realize the optical flow in several stages. Firstly we explore the optical flow in our own code, we use the Opencv package to perform the optical flow operation over the given UR data, and use it as the input of the model. Secondly, we realize the optical flow by using shirui’s optical flow result directly. Thirdly, we use shirui’s code to run optical flow on the data we have and get the result.

**Appendix**

Here are some links that we read and rebuild in part 1:

Human Activity Recognition (HAR) Tutorial with Keras and Core ML (Part 1)

https://towardsdatascience.com/human-activity-recognition-har-tutorial-with-keras-and-core-ml-part-1-8c05e365dfa0

How to Develop RNN Models for Human Activity Recognition Time Series Classification

<https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/>

Here are a few good starting points to explore this:

https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/

https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/ (https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition)

https://towardsdatascience.com/human-activity-recognition-har-tutorial-with-keras-and-core-ml-part-1-8c05e365dfa0

<https://www.mdpi.com/1424-8220/17/11/2556>

Some relevant datasets:

https://www.crcv.ucf.edu/data/UCF101.php

https://www.crcv.ucf.edu/data/UCF\_YouTube\_Action.php

<http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/>