Group Project: Final Report

Group 2 Team name: APLUS

Zhicong Fan
zhicong2

Yuqing Xu
yuqingx3

ABSTRACT

The project is to recognize human activities in short videos by using the LSTM RNN model. This report includes some work that relates to our project, some details of the data, neural network architectures, and results of trained models.

INTRODUCTION

In this project, we are going to train models to recognize human activities in short videos. It is so important as it shows interpersonal relations and provides information to understand a person. Also, it has applications in many fields like health, security, entertainment, etc. During these years, some improper behaviors are reported in restaurant such as pizza hut, making these restaurants the worst places to go to. As specified by Adrian Rosebrock, human activity recognition can be used to monitor a food service worker has washed their hands after visiting the restroom or handling food that could cause cross-contamination. Also, our project is motivated by a seminar that discussed LSTM and used it to predict future behaviors, early activity recognition, and Volumetric velocity forecasting in the ocean. Traditional neural networks are not able to classify what is happening during every point in a short video. But a LSTM (a recurrent neural networks) could help solve this problem. For the LSTM, it does not have to deal with the long-term dependencies problem and there exists a sigmoid function in each node of the LSTM acting like a gate, letting how much of the parameters should flow into the networks. This time, we only focus on human activities. The input to the network will be time-series data denoted the track of the movement of volunteers by wearing a smartphone on their waist and using its embedded accelerometer and gyroscope to capture the value of 3-axial linear acceleration and 3-axial angular velocity at a constant rate, which is more than seven thousands of observations and nine features. Each observation of a feature has a vector of length 128, denoting the movement in a range of time. The output will be from 1-6, and each denotes an activity.

RELATED WORK

Our project is based on Chevalier's human activity recognition project[1], but we normalized and augmented the data, giving us more data in a manageable range. We used different model structures: an ensemble model is used to compare. As we used Recurrent Neural Networks (RNN) with Long Short-Term Memory cells (LSTMs), which we have never touched on, some papers helped us. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition [3] talks about a proposed generic deep framework for activity recognition based on convolutional and LSTM recurrent units with multimodal wearable sensors, which is a similar topic of ours as our dataset also used some sensors to detect the movements and reflect those movements by data in different dimensions. Sequence to Sequence Learning with Neural Networks [2] presents a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure using LSTM architecture to solve the dimensionality requirement for DNNs. Deep learning in neural networks: An overview [4] is an overview of several deep learning topics, which gives a wide range and deep understanding of many aspects. Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors [5] proposed a deep network architecture using residual bidirectional long short-term memory, including bidirectional connection can concatenate the positive time direction and the negative time direction. These give us a clearer view of long short-term memory and neural networks we will use.

DATA

The data is numeric data and comes from https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI HAR Dataset.zip
Shape of x_train:(7352, 128, 9) Shape of x_test:(2947, 128, 9)
The first dimension is the number of observations, the second dimension is the 128 timesteps per observation, and the third dimension is the input parameters.
Shape of y_train:(7352, 1) Shape of y_test:(2947, 1)
The contributor of the source pre-processed the sensor signal data we accessed. They applied noise filtering median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise. The acceleration signal was separated into body and gravity acceleration using another low pass Butterworth filter. We applied normalization and augmentation on the dataset to obtain more data within a specific range.
The original x data was separated into nine text files, and each denotes a feature. We will need to combine them and convert the text file into the data of NumPy arrays and the normal structure of x. We transformed y into one-hot output. 70% of observations are selected as training data, and the rest 30% are test data. Nothing else is required to process the data.

This is the first observation of one of the features body acceleration in X direction (first 10 of 128 element vectors in the first observation and first feature): 1.8085150e-004 1.0138560e-002 9.2755740e-003 5.0658970e-003 1.0810250e-002 4.0451010e-003 4.7573750e-003 6.2136470e-003 3.3067440e-003 7.5719410e-003

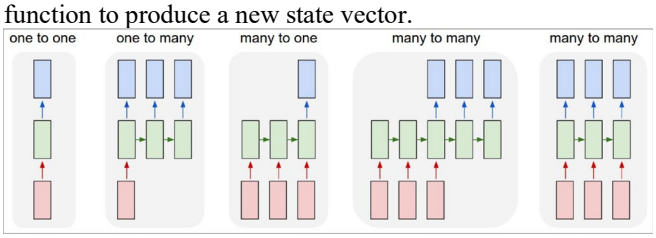
METHODOLOGY

First Neural Network Architecture:

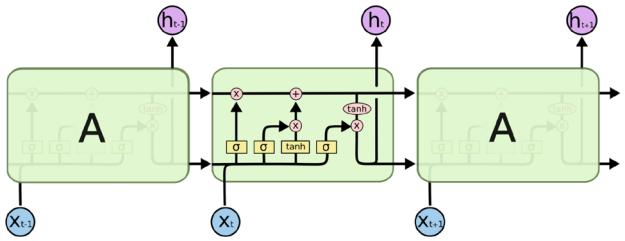
1. Two hidden layers are used.
2. In the hidden layer, there are 32 neurons.
3. RELU activation function is used in the network

In this code, we used the implemented code of RNN in TensorFlow and used the function tf.keras.layers.LSTMCell. After generating 2 LSTM cells, a tf.contrib.rnn.StackedRNNCell function is used to combine all the cells.

Recurrent Neural Networks take several input vectors as input and then output several vectors as specified. In this project, we used the many to one structure as we are going to predict labels of behaviors in the videos. RNN combines the input with their state vector with a learned and fixed function to produce a new state vector.

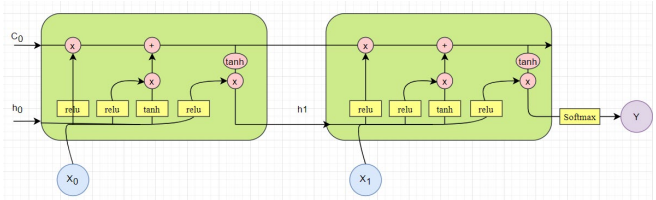


LSTM is a special kind of Recurrent Neural Networks and we used the many to one structure to predict the labels of the behaviors input. LSTM has a similar structure as the RNN but in each node of the LSTM, there are four neural network layers interacting in sigmoid or tanh function. Then the information in this node directly flows into the next node.



For the second model, we use the mean of the ensemble model consisting of the first neural network architecture.

For our model, the LSTM structure for the first model would look like:



In these LSTM cells, each parameter at moment t can be defined as follows:

$$f_t = relu(W_f[h_{t-1},x_t]+b_f) \quad (1)$$

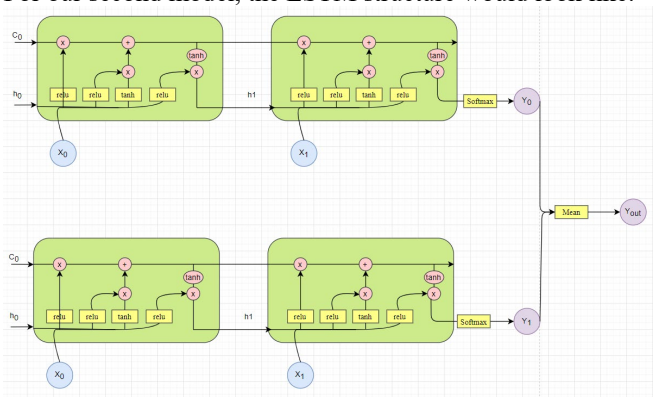
$$i_t = relu(W_i[h_{t-1},x_t]+b_i)$$

$$Ct = ftC_{t-1} + i_t tanh(W_f[h_{t-1},x_t]+b_f)$$

$$O_t = relu(W_f[h_{t-1},x_t]+b_f)$$

$$Y_t = o_t tanh(C_t)$$

For our second model, the LSTM structure would look like:



In these LSTM cells, each parameter at moment t can be defined as follows:

$$f_t = relu(W_f[h_{t-1},x_t]+b_f) \quad (2)$$

$$i_t = relu(W_i[h_{t-1},x_t]+b_i)$$

$$Ct = ftC_{t-1} + i_t tanh(W_f[h_{t-1},x_t]+b_f)$$

$$O_t = relu(W_f[h_{t-1},x_t]+b_f)$$

$$Y_t = o_t tanh(C_t)$$

$$Y_{ensemble} = (Yt_1+Yt_2)/2$$

General Description:

After downloaded and preprocessed the dataset as described in the DATA part, we need to define the labels we want to classify: *"WALKING", "WALKING_UPSTAIRS", "WALKING_DOWNSTAIRS", "SITTING", "STANDING", "LAYING"*. After that, we define the number of neurons in the hidden layer, the learning rate, the training epochs, and the batch size Then, as mentioned previously, TensorFlow embedded functions are used to generate the RNN cells. Since we used cross-entropy in our model, we have to make the labels into one-hot labels. In which np_eye is used to implement. Then, we start to train the model. During the training procedure, losses and accuracies are documented inside our local lists. Finally, we use the data in these lists to plot graphs. Then we repeat the training procedure for the ensemble model except that the model structure is changed.

Overview of Hyperparameters:

1. Learning Rate: 0.0028:
   This is used for the adam optimizer and we got this value by choosing the highest accuracy from testing for 10 times.
2. Lambda loss = 0.0017:
   This is used for l2 calculation, deciding how much we want from the result.
3. Training iterations = 112,5300:
   This is the epoch for training the dataset. This number is chosen because it is the fastest iteration to reach about 95% accuracy.

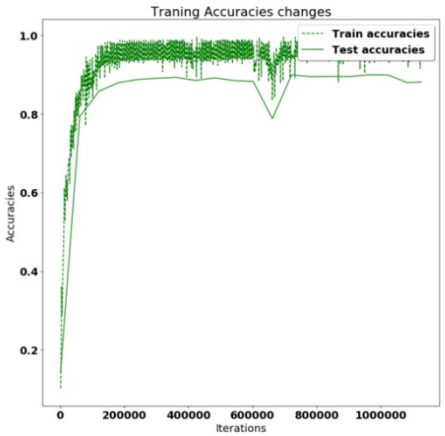
4. Batch size = 1500:
   This is used to divide the dataset into several groups so that the training is efficient.
5. Display index = every 60000 iterations:
   This is used to display the current batch loss and accuracy on the training or testing dataset.

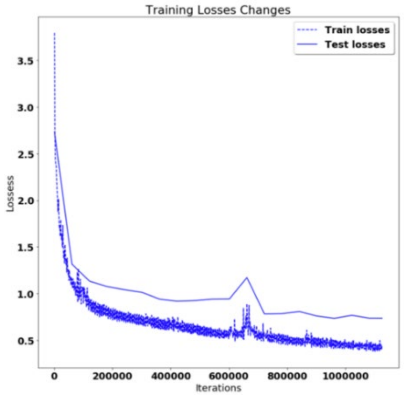
We are using SoftMax cross-entropy with logits as the cost function with l2 loss and Adam optimizer to prevent the neural network from overfitting the data.
We chose Adam because it really does not matter which optimizer we choose. They all work the same.

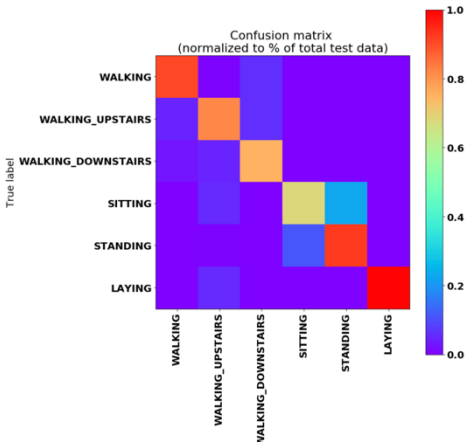
We chose to use CrossEntropy because:
1. We learned this in the class
2. We are trying to classify it into six different labels.

RESULTS

Tables & Graphs:
Model1:

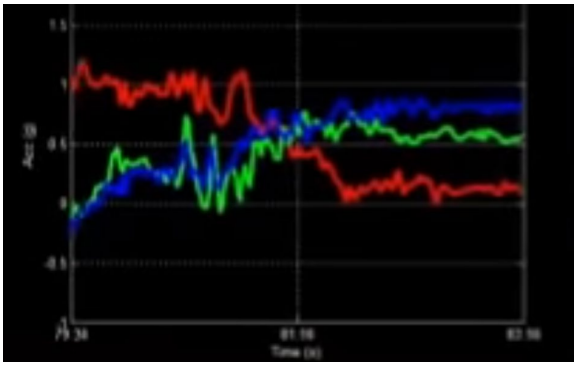




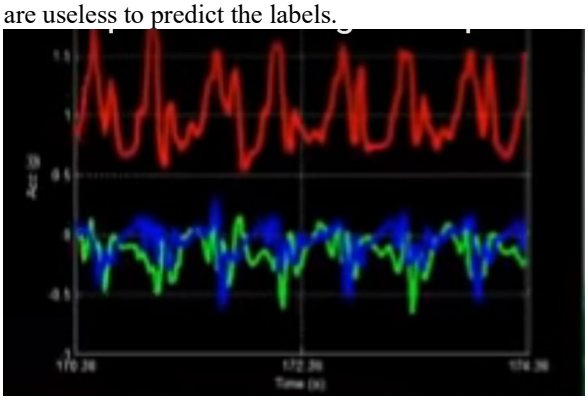


TESTING SCORES	PERCENTAGE
Precision	88.35%
Recall	88.09%
F1 score	88.02%
Testing accuracy	88.09%

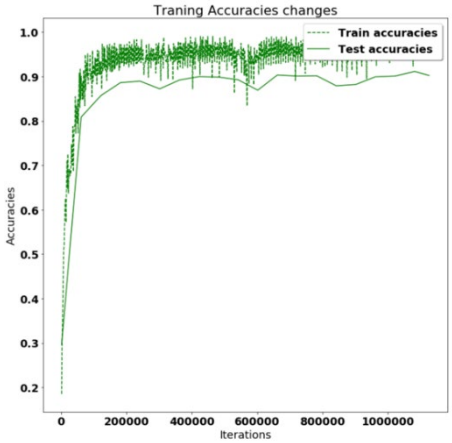
For the first LSTM model, it achieves the accuracy over 90% after training for 120000 and during the training it starts to bouncing between 91% and 94% on the training data, and bouncing between 85% to 89% on the testing data. After the training is finished, we can see that the losses drop to 0.7 which was 3.8 at the beginning. All the four scores of the first model are about 88% which mean we did a great job on classifying true positive labels. We can see that, the most easiest label to predict is walking and laying, sure you can imagine that. When a person lays down, the axis of the signal crossed with each other so for sure it is much easier to detect.

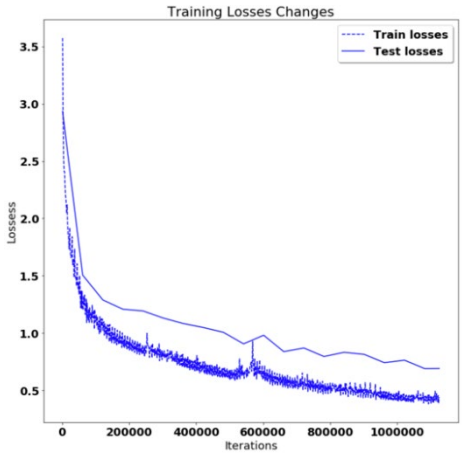


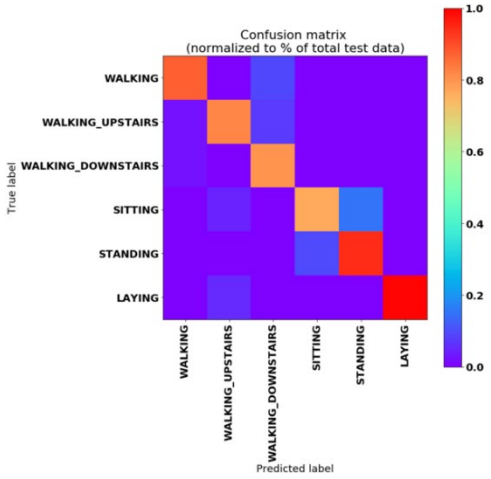
And for walking, the signals are constantly changing, which makes it also easier to classify. But sit and stand is just one movement and stay still, so for most of the signals afterwards are useless to predict the labels.



Model2:







TESTING SCORES	PERCENTAGE
Precision	90.56%
Recall	90.23%
F1 score	90.24%
Testing accuracy	90.23%

For the ensemble model, we can see that all the testing scores went up about 2% which means our ensemble model could work better than a single model. And the ensemble model did a much better job in classifying sitting and standing which is the hardest part in this project.

This is the right way to approach our goal because LSTM is the model designed to recognize videos while other classic models are not able to tell us which label it is during every point in the video.

For the ensemble model, I have tried to implement a ensemble model that contains more than three models or implementing a simple tree model as to predict the residual of the predictions. But my computer is not capable of doing these approaches. Even just compiling the model and not even training it would take me more than half hour, making it difficult to debug.

GitHub: https://github.com/MichaelFan36/STAT430

CONCLUSION

In this project, we used the UCI machine learning dataset of human movements to predict their activities using LSTM RNN models and finally got an accuracy of about 90 percent by comparing two models and picking the one that can predict better. We have learned how to manage time to work efficiently in a group by working separately in the usual and working together when needed. For future idea or improvement of the activity recognition project, maybe we can collect the dataset by ourselves from the internet and use those data without preprocessed by others to truly learn the algorithms of detecting movements of them in the video. And also, we can try other different ensemble model structures such as XGBoost, models more than 3 or other Adaboost or Gradient Boosting Strategies. Besides this, we could add more iterations on training, because I found when training epochs are doubled, the accuracy on predicting the training dataset went up to 98% for the single model. For the ensemble model, it took too long to run so I made only half of my original training epochs.

Both of us did 50 percent of the work on the project. We chose the project from a list of topics we are interested in and found the one we both agreed with and found the resources and dataset on it separately. Zhicong tried different hyperparameters, loss functions, and optimizers to adjust the model to improve the model's robustness since different datasets would be used. Yuqing processed the data to a format easy to use to model the neural networks, summarized the papers, made the report, and wrote the report in detail. We constructed the neural network and trained it together, and Zhicong did the work that showed the network results.

REFERENCES

[1] Guillaume Chevalier, LSTMs for Human Activity Recognition, 2016, https://github.com/guillaume-chevalier/LSTM-Human-Activity-Recognition

[2] Sutskever, I., Vinyals, O., and Le, Q. V., "Sequence to Sequence Learning with Neural Networks", arXiv e prints, 2014.

[3] Ordóñez, F.J.; Roggen, D. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. Sensors 2016, 16, 115.

[4] Jürgen Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, Volume 61, 2015, Pages 85-117, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2014.09.003.

[5] Yu Zhao, Rennong Yang, Guillaume Chevalier, Ximeng Xu, Zhenxing Zhang, "Deep Residual Bidir-LSTM for Human Activity Recognition Using Wearable Sensors", Mathematical Problems in Engineering, vol. 2018, Article ID 7316954, 13 pages, 2018. https://doi.org/10.1155/2018/7316954