[Team 05] ProjC1 Report

Matthew Houk
Dept. of Electrical and Computer Engineering
North Carolina State University
Raleigh, US
mjhouk@ncsu.edu

Jeremy Park
Dept. of Computer Science
North Carolina State University
Raleigh, US
jipark@ncsu.edu

Tyrone Wu
Dept. of Computer Science
North Carolina State University
Raleigh, US
tkwu@ncsu.edu

I. Methodology

A. Data Description

For this project, we are given a collection of time series data from 8 subjects, each with varying number of trials, to try and identify the terrains from real time data. In each trial, the following 6 features have been collected in 40 Hz:

- x accelerometer
- y accelerometer
- z accelerometer
- x gyroscope
- y gyroscope
- z gyroscope

At the same time when when the 6 features above were measured, the following terrain labels have been measured in 10 Hz:

- 0 - Standing or walking on solid ground
- 1 - Going down stairs
- 2 - Going up stairs
- 3 - Walking on grass

In further analysis of the dataset, we noticed there was a heavy class imbalance. Fig. 1 depicts the distribution of the class label according to each the aggregated trails of each subject. We expect the class imbalance would skew the training and predictions of our model, however for now, we were only concerned with identifying our preferred model. In the future, we plan resolve this.

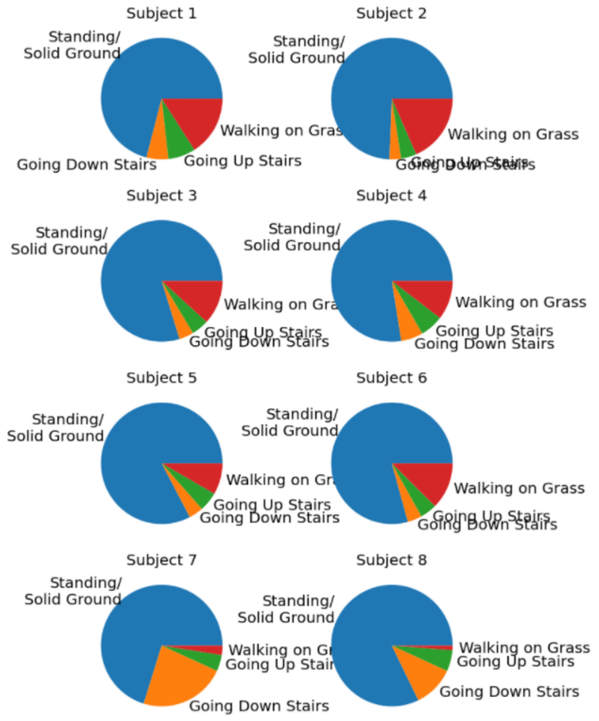
B. Data Preprocessing Method

For each trial we preprocessed the time series into windows, where each window stores 2 seconds (80 rows) of information, which is flattened into a vector, on the 6 features. The stride/timestep of the window is taken every 0.5 seconds from the beginning to the end of a trial. The label associated for each window is determined on the label it was last identified within the window. This way, the data can be processed in an LSTM model.

C. Choice of Model

For our initial model selection, we decided on choosing one of the three models: Random Forest Classifier (RFC), Support Vector Machine (SVM), and Long Short-Term Memory Network (LSTM). Although this submission was only to establish a baseline model, we wanted to get an edge for our first submission, so we tested for descent models. We expect LSTM will perform the best, but to make sure, we will implement each model using scikit-learn for RFC and SVM, and Keras for LSTM. Later in the report, we will discuss our results of our findings.

Fig. 1. Class Distribution of the 8 Subjects



II. Model Training and Selection

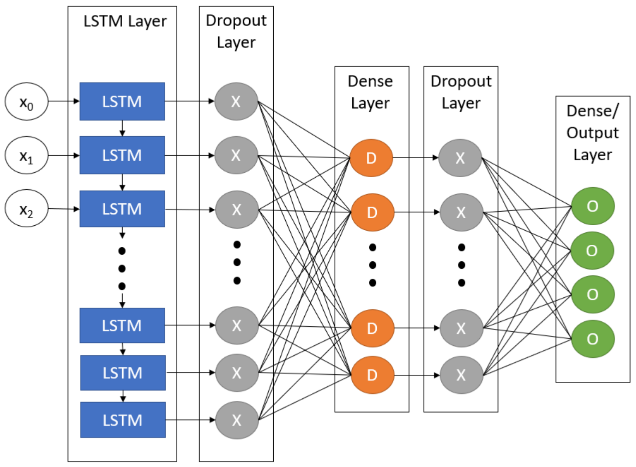
A. Model Training

For this submission, we only trained and evaluated our model on the data for subject 2 trial 1 to simplify our model selection. As mentioned in the *Data Preprocessing Method* subsection, we used this format so that our LSTM model could have the same training data as RFC and SVM. Given the temporal relationships between time-series data, traditional k-fold cross validation cannot be used, as the order of the time-series data is important [1]. Therefore, we used the TimeSeriesSplit function in sklearn, where the training set is provided by the first k folds and the testing set is the (k+1)th fold [1]. Thus, the training set grows and includes all of the folds up until the k-th fold, and we chose k=4 for this submission. For each iteration up to k folds, we train a model using a 80-20 training-validation split on the training fold. We use TimeSeriesSplit to evaluate the performance across folds, and we also compare against an 80-20 split of the entire training data. Lastly, we retrained the model using the all of the available data for subject 2 trial 1 and used that model for our final prediction submissions.

B. Model Selection

To begin with, an SVM, RFC, and LSTM were implemented in code using the default parameters of the Keras library. At this point models were compared for a baseline performance, before every model underwent basic hyper-parameter tuning, where values were modified and the delta accuracy was used to explore the parameter space and identify reasonably well performing values. This is acknowledged to not be the best method for hyper-parameter selection, with part of the intended improvements of the model being to implement more thorough hyper-parameter tuning solutions. For the final model, Fig. 3 demonstrate the model training throughout each KFold. It can be observed that the model initially overfits, before gradually becoming a better fit to the data with each fold, this can be understood as being due to insufficient data, as the amount of data for training in each fold increases with TimeSeriesSplit.

Fig. 2. Deep Network Architecture

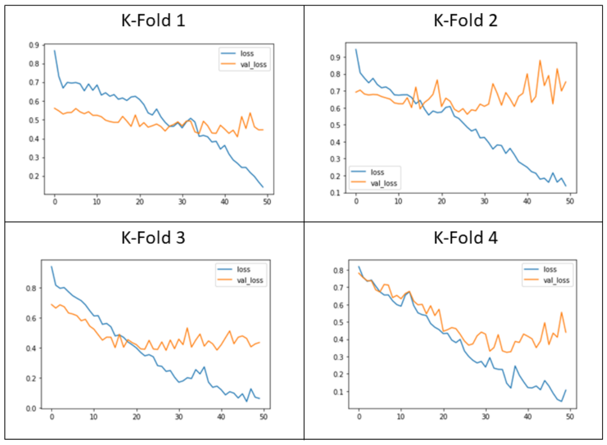


In Table I the difference in model performance between the tuned and untuned model parameters can be observed for each model tested. It can be observed that the LSTM performance exceeded the performance of both SVM and RFC classifiers. For this reason we chose to continue with the LSTM model, as seen in Fig. 2.

TABLE I
TUNING ACCURACY

Model	***Untuned***	***Tuned***
LSTM	89.23%	92.38%
SVM	67.13%	75.11%
RFC	79.53%	85.42%

Fig. 3. KFold Training



III. Evaluation

The model was evaluated upon a test set of data set aside prior to the beginning of the training period. This test set represented 20% of the original data, using a 80-20 split. The model was trained and evaluated upon data from subject 2, trial 1. The results of the final training upon which predictions were based can be found below in Table II

TABLE II
CLASSIFICATION METRICS

Test Set Performance					
Class	***Precision***	***Recall***	***Accuracy***	***F1 Score***	***Support***
Standing	0.90	0.98	0.98	0.94	457
Down Stairs	0.90	0.47	0.47	0.62	19
Up Stairs	0.92	0.40	0.40	0.56	30
Grass	0.87	0.62	0.62	0.73	53
Total	0.90	0.90	0.90	0.89	559

References

[1] "sklearn.model_selection.TimeSeriesSplit," scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html (accessed Oct. 15, 2021).