Team 26: Proj-C: Terrain Identification from Time Series Data

Himanshu Gupta (hgupta6)
hgupta6@ncsu.edu

Tirth Patel (tdpatel2)
tdpatel2@ncsu.edu

Harsh Kachhadia (hmkachha)
hmkachha@ncsu.edu

I. METHODOLOGY

For the classification task at hand, initially, we preprocessed data as per the procedure described in the Data Preprocessing subsection in Methodology. Based on our final time-series data post preprocessing, and performing literature survey about various classifiers (classical machine learning techniques), we curated some Deep Learning models and trained them on our processed data.

We trained and validated these classifiers on two variants of data to record the effects of Class Imbalance on training and the performance. These data variants will be:

- Dataset obtained after Data Preprocessing.
- Dataset obtained after Data Preprocessing and using SMOTE.

After performing base comparison of the curated deep learning models with default parameters, we planned not to use the data which was processed with SMOTE. The main reason for this was that SMOTE augments the dataset with duplicate rows of the minority class labels to balance the dataset. Therefore, the data processed with SMOTE will have similar rows which may lead our model to over-fit over the minority class and effects the overall performance of the model.

For the three different models we trained and tested on the dataset, we found that Bidirectional LSTM + LSTM model out-performed the other models. Furthermore, we planned to tune the Learning rate and dropout values (hyper-parameters) on the selected model. Hyper-parameter tuning made sure that we got the best performing model on our validation set and improve our F1 score over the unseen dataset.

A. Data Preprocessing

Our training dataset consists of continuous session of data for multiple subjects (Subject001 - Subject008). Each of the subject in dataset has 4 files named as ”subject_x”, ”subject_x_time”, ”subject_y_time”, ”subject_y”. Our test dataset consists of continuous session of data for multiple subjects (Subject008 - Subject012). Each of the subject in dataset has 4 files named as ”subject_x”, ”subject_x_time”, ”subject_y_time”.

Initially we started with joining the ”subject_x files” for all subjects with their respective ”subject_x_time” files and named each file as ”subject_x_meas_time.csv”. Similarly we joined the ”subject_y files” for all subjects with their respective subject_y_time files and named each file as ”subject_y_label_time.csv”. This gives us the accelerometer and gyro-sensor reading with their time in a single file and y_label and y_time in a single file.

Now, for each subject, we will merge their ”subject_x_meas_time.csv” file and ”subject_y_label_time.csv” files based on the ”time” column in both the files. The result of merging gave us a single file for a subject ”subject_final.csv” that will contain all the columns relating to that subject in the dataset. Since, we know there is a difference in sampling rate between the x_data samples and y_data samples due to which we observe the final labels for only 1/4 of the dataset.

We applied interpolation technique to fill out the missing data in each ”subject_final.csv” files. This interpolation technique used the method ”pad”, where it will fill the gap between two labels in the dataset. After interpolation, we get the ”subject_interpolated.csv” files, in which, finally, all our tuples will have a label. In the next stage of processing, we concatenated all the ”subject_interpolated.csv” into a single file to ”Interpolated_combined.csv” that has the data for all the subjects and will be used for training and validating our model.

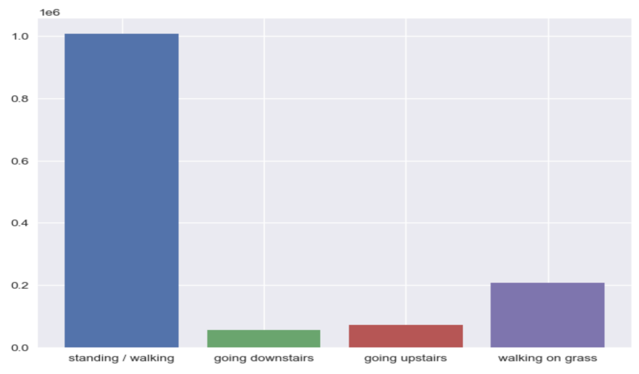


Fig. 1. Imbalanced Classes for Complete Dataset

After going through the data in the combined file and plotting it, we observed a severe class imbalance problem for both single subject and for whole combined data. Figure 1 shows the imbalanced class between the dataset. Now, by applying SMOTE, we were able to solve the class imbalance problem, and thus, obtain 2 datasets (one with balanced classes/labels and the other with imbalanced classes/labels)

Even though, we did SMOTE to balance our dataset previously during 1st predictions, we are not using any such class balancing techniques this time for Neural Networks model. SMOTE simply duplicates the instances of minority classes for oversampling but, that can cause our deep neural network model to over-fit over those minority classes.

II. MODEL TRAINING AND SELECTION

A. *MODEL TRAINING*

After interpolating and combining the data from all subjects(described in prev section), we are performing the below 3 steps to make the data is ready for feeding into our deep neural network models -

1) One-Hot Encoding: - We are performing One-hot encoding on our y-labels which will convert our y-label from integers (0,1,2,3) to categorical encodings. The new labels for each class looks like - class0 = [1,0,0,0], class1=[0,1,0,0], class2=[0,0,1,0] and class3=[0,0,0,1]

2) Standardization of Dataset: - We are standardizing the data using StandardScalar library from sklearn which will subtract the mean to center them on 0.0 and divide by the standard deviation to give the standard deviation of 1.0. This will allow our data to be on a similar scale and reduce the bias.

3) Creating Window Frames: - We are dividing the dataset into multiple window batches of size 40. These will allow the Neural network model to identify the pattern within each window time frame. This also increases the dataset dimension from 2 to 3 which is required to be the input shape for our deep neural network models.

B. *MODEL SELECTION*

Once the dataset was ready, we split into training and validation sets. The training set was 90% of the entire dataset and validation set was 10% of the entire dataset.

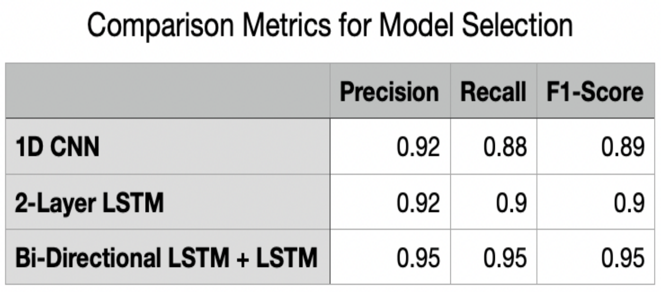
To perform classification, we have considered the following 3 Neural Net Classifiers for our task:

- 1D CNN Model
- 2 Layer LSTM Model
- Bidirectional LSTM + LSTM

Bidirectional LSTM + LSTM model refers to 1 layer of Bidirectional LSTM + 1 layer of LSTM model. Refer to Figure 3 to check the model structure.

After training the model, we used the validation dataset to evaluate the performance of our model on the basis of F1 score, Accuracy, Recall, precision.

On comparing the result metrics of these 3 models, we observed that the third model(Bidirectional LSTM + LSTM) performed far better than the other two neural network



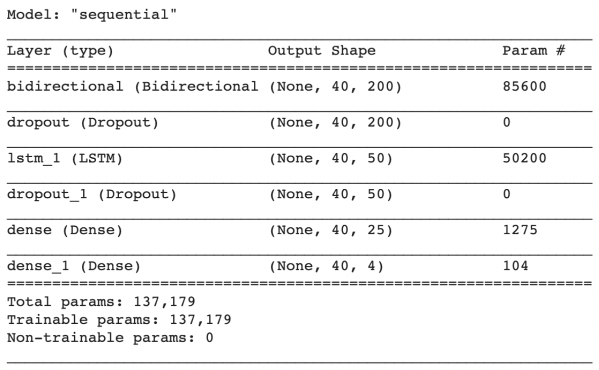
Comparison Metrics for Model Selection

	Precision	Recall	F1-Score
1D CNN	0.92	0.88	0.89
2-Layer LSTM	0.92	0.9	0.9
Bi-Directional LSTM + LSTM	0.95	0.95	0.95

Fig. 2. Metrics Comparison between different Neural Network models

models, and thus, we selected the third model as our final model. Refer the table in Figure 2 for comparison of these models over various .

From the metrics comparison, we can see that the Bidirectonal LSTM + LSTM model outperforms the other models in terms of various metrics and also, the F1 score over the unseen test dataset. The structure of our BIdirectional LSTM + LSTM model can be seen in the figure 3.



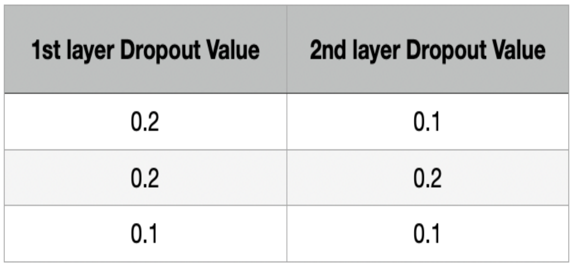
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectional (None, 40, 200)           85600
_________________________________________________________________
dropout (Dropout)            (None, 40, 200)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 40, 50)            50200
_________________________________________________________________
dropout_1 (Dropout)          (None, 40, 50)            0
_________________________________________________________________
dense (Dense)                (None, 40, 25)            1275
_________________________________________________________________
dense_1 (Dense)              (None, 40, 4)             104
=================================================================
Total params: 137,179
Trainable params: 137,179
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. Model Structure for BidirectionalLSTM + LSTM Model

After selecting the Bidirectional LSTM + LSTM Model, we performed hyper-parameter tuning to achieve the best possible performance from our selected model.

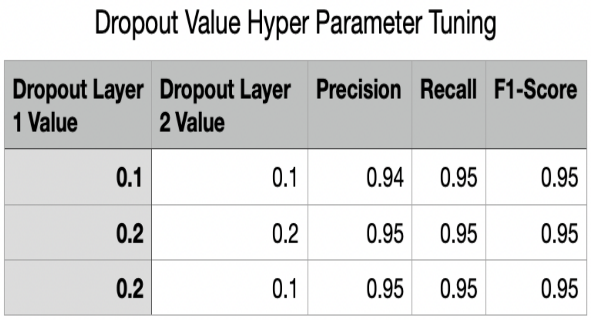
We decided to tune the dropout values and learning rates to see which combination gave us the best model. First we planned to tune the dropout values keeping the learning rate constant at 0.001. As observed from the model structure in Fig 3, we have 2 Dropout layers in our model, hence, we tested different combinations of Dropout values for these 2 layers. The multiple combination we tried are shown in Fig 4.

We found that the dropout values combination of 0.2 and 0.1 to work best for our model. Refer Figure 5 for metric comparison for various Dropout values:



1st layer Dropout Value	2nd layer Dropout Value
0.2	0.1
0.2	0.2
0.1	0.1

Fig. 4. Dropout Value combinations



Dropout Value Hyper Parameter Tuning

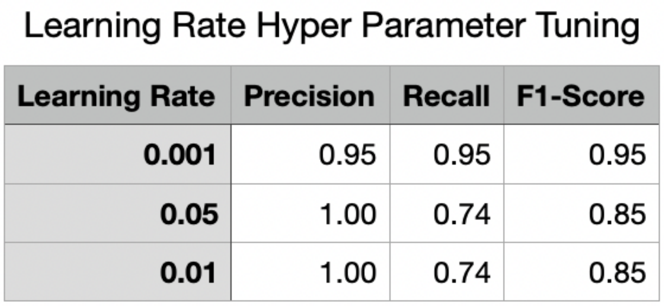
Dropout Layer 1 Value	Dropout Layer 2 Value	Precision	Recall	F1-Score
0.1	0.1	0.94	0.95	0.95
0.2	0.2	0.95	0.95	0.95
0.2	0.1	0.95	0.95	0.95

Fig. 5. Dropout Value Hyper Parameter Tuning

Next, we tried to tune the learning rate while keeping the dropout values constant at our selected dropout value combination. We tested the models with following learning rates:

- 0.001
- 0.05
- 0.01

We found that the learning rate of 0.001 to work best for our model. Refer figure 6 for metric comparison:



Learning Rate Hyper Parameter Tuning

Learning Rate	Precision	Recall	F1-Score
0.001	0.95	0.95	0.95
0.05	1.00	0.74	0.85
0.01	1.00	0.74	0.85

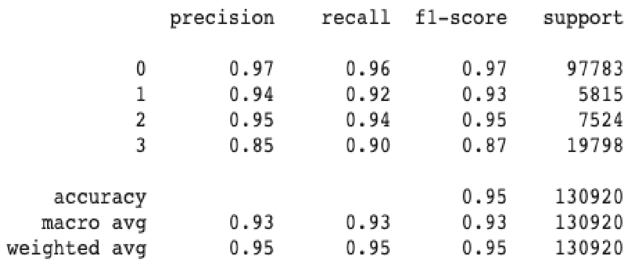
Fig. 6. Learning Rate Hyper Parameter Tuning

After a thorough tuning of hyper-parameters (dropout values, learning rate), we came to conclusion and finalized the following hyper-parameter values for our selected model, which gave us the best F1-Score of 86.1 on the unseen test dataset:

Hyper-parameter Values:
Dropout Value Combination: 0.2 (Layer 1) & 0.1 (Layer 2)
Learning Rate: 0.001

III. Evaluation

Amongst the various models we tried, the Bi-directional LSTM layer stacked with LSTM layer performed best on the unseen test data. It gave the best F1 score of 86.1 on the unseen test dataset. Figure 7 shows the performance of our model on the validation dataset.

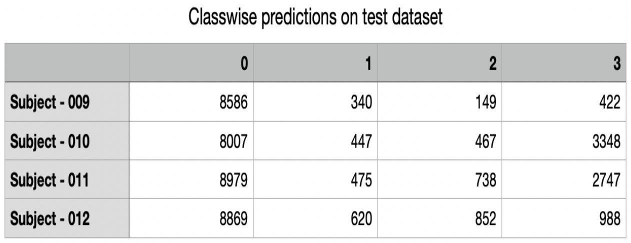


```
              precision    recall  f1-score   support

           0       0.97      0.96      0.97     97783
           1       0.94      0.92      0.93      5815
           2       0.95      0.94      0.95      7524
           3       0.85      0.90      0.87     19798

    accuracy                           0.95    130920
   macro avg       0.93      0.93      0.93    130920
weighted avg       0.95      0.95      0.95    130920
```

Fig. 7. Metric Score on the validation dataset

IV. Test-case illustration

Figure 8 shows the illustration of our predictions over the test dataset. It shows the number of classes that are predicted for each of the subjects.



Classwise predictions on test dataset

	0	1	2	3
Subject - 009	8586	340	149	422
Subject - 010	8007	447	467	3348
Subject - 011	8979	475	738	2747
Subject - 012	8869	620	852	988

Fig. 8. Class wise predictions on the test dataset

We took a small subsample from the prediction label of subject 9. Figure 9 shows the trend plot during that timeframe. From the plot, we can observe that initially the person was walking on ground, then the person starts climbing the stairs, then again walks on a flat terrain and at the end we can see going downstairs and then walking on flat terrain in small intervals.

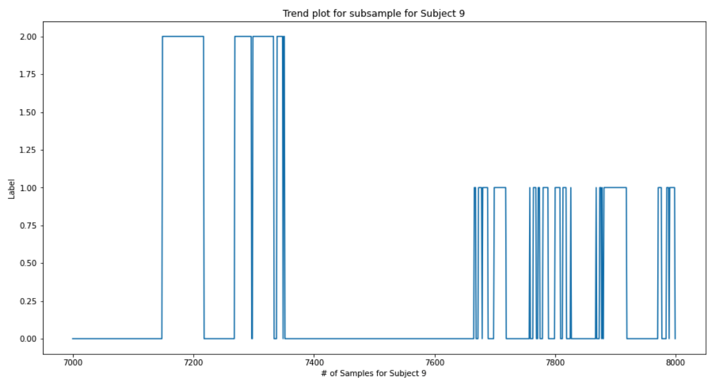


Fig. 9. ”Trend plot for subsample for Subject 9”

REFERENCES

[1] Chawla, Nitesh Bowyer, Kevin Hall, Lawrence Kegelmeyer, W.. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair.953.

[2] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. NIPS 2017: 3146-3154

[3] Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[4] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324

[5] J. Brownlee, “Lstms for human activity recognition time series classification,” Machine Learning Mastery, Sep 2018. [Online]. Available: https://machinelearningmastery.com/how-to-develop- rnn-models-for-human-activity-recognition-time-series-classification/

[6] Gers, Felix Eck, Douglas Schmidhuber, Jürgen. (2001). Applying LSTM to Time Series Predictable through Time-Window Approaches. 669-676. 10.1007/3-540-44668-0_93.