Team 36: Proj-C: Terrain Identification from Time Series Data

Sai Naga Vamshi Chidara
schidar@ncsu.edu

Venkata Pranav Singaraju
pvsingar@ncsu.edu

Krishna Saurabh Vankadaru
kvankad@ncsu.edu

I. Methodology

For the Terrain Identification task at hand, we started with pre-processig the data following the procedure described in the Data Pre-processing section below. In phase 1 we have used Random Forest Classifier[1] with SMOTE [2] technique. But the model was overfitting the data. In phase 2, we implemented Neural Networks and specifically LSTM and to compensate the class imbalance we have implemented Weighted Cross Entropy[3] instead of using SMOTE. For splitting the data set into training and validation, we have used the window size[4] approach as the data is sequential and LSTMs are good at capturing the information from a sequential data.

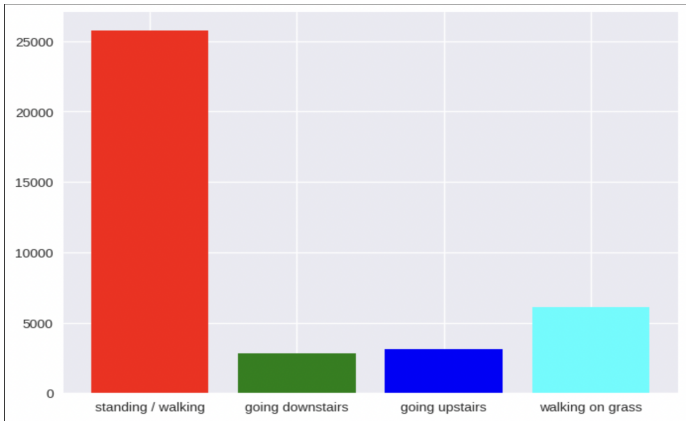
A. *Data Preprocessing*

The training dataset for terrain identification task contains data for 8 subjects (Subject001 - Subject008) and the test dataset contains data for 4 subjects (Subject009 - Subject012). Each subject has files related to measurements, labels and time accros the sessions.

The file _x_time contains the timestamps for the accelerometer and gyroscope measurements where as the file _y_time contains the timestamps for the labels. The units are in seconds and the sampling rate is 40 Hz for _x_time and 10 Hz for _y_time. Files _y contains labels 0 to 3 where label 0 indicates standing or walking in solid ground, 1 indicates going down the stairs, 2 indicates going up the stairs, and 3 indicates walking on grass.

First, for each subject, we merged all the _x files into one single file along with time column. Similary, we merged _y files. Then, for each subject, we merge the file containing accelerometer and gyroscope readings with the file containing labels using the ”time” column (outer join). Here, while doing an outer join, because of the difference in the sampling rate (40 Hz for _x and 10 Hz for _y), we have labels for only a quarter of the x samples because of this difference in sampling rate.

To fill the missing labels, we use interpolation technique with padding, where the missing label for a sample is the label for the sample before it. Now, we merged all the files into one single file which will be used for training and validation. By plotting this data, we identified that there is severe imbalance in the dataset with majority of data samples having 0 as their class label. So, we used Weighted Cross Entropy in the loss function to compensate it.



B. *Training and validation split*

*1) **Session Leave Out**:* We took a Session leave out appraoch for train validation split, where we considered 2 common sessions of all the subjects as the validation set and remaining data as training set. We trained models with this dataset and evaluated their performance. We identified that we could do better with a window approach where we will define a window size and consider consecutive data record of window size as one window.

*2) **Window Size Approach**:* Sequential property will be lost if we randomly divide the dataset. We divided the data into windows and then we had split the data into training and validation. With this, we ensured that we mantain the sequential property of the data. We had trained using different window sizes and identified 120 as the ideal window size.

II. Model Training and Selection

A. *Model Architectures*

In the first phase of the Terrain Identification project, we considered Random-Forest Classifier. In the next phase, we considered LSTM and bi-directional LSTM[5] as the data is sequential i.e. Time Series data. The main idea behind using a Bi-directional LSTM is that the current prediction of the terrain not only depends on few previous measurements (xyz accelerometers & gyroscope values), but also on few of the future measurements in the sequence. This intuition is taken into consideration as we implemented window size approach.

A batch size of 512 is taken in all the models that we have implemented. Following are the different architectures around LSTM and Bi-LSTM that we have considered for the 3 phases of round 2.

*1) **Bi-LSTM+LSTM+40WS**:* In this model architecture, the input is fed to a single layer of Bi-LSTM(hidden size = 128) and then a single layer of LSTM(hidden size = 128) is added sequentially. The output from the LSTM layer is fed to a dense layer (64x64). The activation function used here is ReLU between the dense layer and also on the output from the dense layer. Below picture depicts the model architecture.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─LSTM: 1-1                              [-1, 40, 256]             139,264
├─LSTM: 1-2                              [-1, 40, 128]             197,632
├─Linear: 1-3                            [-1, 40, 64]              8,256
├─ReLU: 1-4                              [-1, 40, 64]              --
├─Linear: 1-5                            [-1, 40, 4]               260
├─ReLU: 1-6                              [-1, 40, 4]               --
==========================================================================================
Total params: 345,412
Trainable params: 345,412
Non-trainable params: 0
Total mult-adds (M): 0.34
==========================================================================================
Input size (MB): 0.47
Forward/backward pass size (MB): 0.14
Params size (MB): 1.32
Estimated Total Size (MB): 1.92
==========================================================================================
```

*2) **Bi-LSTM+LSTM(No of layers=2,dropout=0.5)+40WS**:* In this model architecture, we have considered single layer of Bi-LSTM and 2 layers of LSTM. Both have the same hidden size of 128. A 0.5 dropout is added to the first layer of LSTM.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─LSTM: 1-1                              [-1, 120, 256]            139,264
├─LSTM: 1-2                              [-1, 120, 128]            329,728
├─Linear: 1-3                            [-1, 120, 64]             8,256
├─ReLU: 1-4                              [-1, 120, 64]             --
├─Linear: 1-5                            [-1, 120, 4]              260
├─ReLU: 1-6                              [-1, 120, 4]              --
==========================================================================================
Total params: 477,508
Trainable params: 477,508
Non-trainable params: 0
Total mult-adds (M): 0.47
==========================================================================================
Input size (MB): 1.41
Forward/backward pass size (MB): 0.41
Params size (MB): 1.82
Estimated Total Size (MB): 3.64
==========================================================================================
```

*3) **Bi-LSTM+LSTM(layers=2,dropout=0.5)+120WS**:* In the previous two models, we have taken the window size of 40 which is nothing but data corresponding to 1 second. Later, we realized that measurements corresponding to 1 second are too less to identify the terrain as the person can move only one step in one second. So, in this model, we have choosen 120 as the window size which is 3 seconds of data and considered the same architecture as (2).

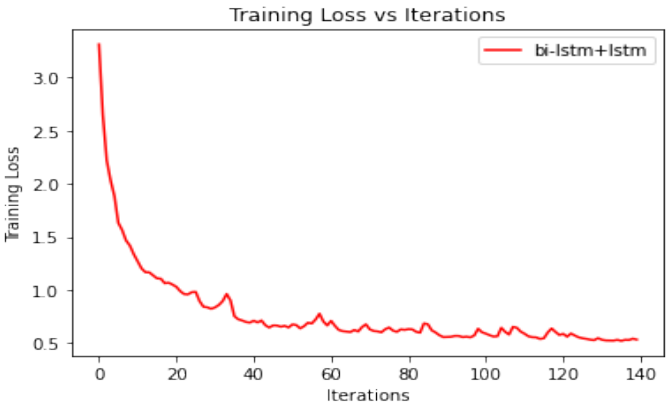
B. Model Training

In the training data files, we have 8 subjects and the xyz accelerometers and xyz gyroscope measurements recorded for each subject for different sessions. In the first phase, we explored subject-leave-out and used it to split the dataset. But, as we have taken window size, we split the entire data into several windows and each window is a sample. Now, these samples are shuffled randomly and split into 70% and 30% for training and validating.

*1) **Weighted Cross Entropy**:* In the previous phase, we implemented SMOTE technique to increase the samples of the minority class by generating synthetic data. This didn't help us in improving the performance as model underwent over-fitting due to synthetic data. Hence, in this phase, we have considered weighted cross entropy to counter attack the data imbalance. We know that, class 0 is the majority class and all other classes comes under minority classes. More weight is assigned to the minority class as a miss-classification in minority classes should yield a bigger loss compared to a miss-classification of majority class. The weights for the classes in the weighted cross entropy is given by

$$Class\_x\_Weight = 1 - \frac{\text{Class X Training Samples}}{\text{Total Training Samples}}$$

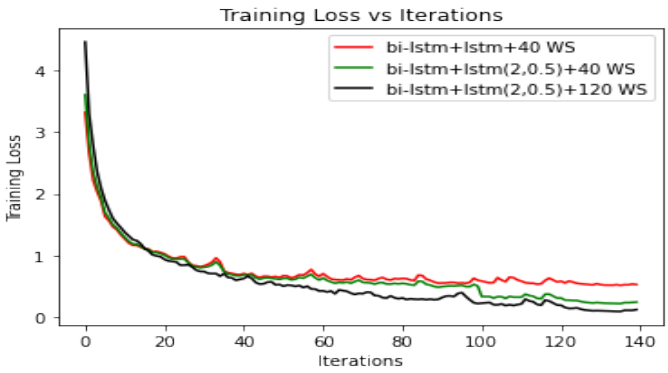
*2) **Adam Optimizer with StepLR Scheduler**:* Due to limited computing resources, we have trained models only for 140 epochs. We have considered different learning rates starting from 0.001 to 0.1 along with Adam optimizer[6], but the loss wasn't consistent. The model with learning rate 0.001 gave the least training loss, but after 50 epochs, the gradient steps over-shot and hence we implemented scheduler with stepLR with step size of 50 epochs and 0.5 value for gamma. Below is the graph of training loss with epochs.



C. Model Selection

For the phase 1 of the project we used the Random Forest classifier[1] which is a classical machine learning technique. For the next phase, we used Bi-LSTM and LSTM

architectures along with a dense layer. The hyper-parameters that we have tuned are batch-size, window size, hidden size (for LSTM layers), learning rate, number of LSTM layers and dropout value. The graph of training loss with epochs for all the three architectures that we have used is given below.



In the above plot, the model with lowest training loss is model-3 and we used it for the final submission.

III. Evaluation

As mentioned earlier, we have 30% of the total data as validation set which helped us in evaluating the model's performance. The main metric that we considered is the macro-average F1 score. Apart from it, th other metrics that we observed are class-wise F1 score, Precision and Recall. Based on the validation set scores of the three different architectures , the last model (Bi-LSTM+LSTM(layers=2,dropout=0.5)+120WS) gave the best macro-average F1 score, which in turn gave the best F1 score on the test set data set. Below table displays different metrics that were captured on the validation set.

Table I
Average Metric Values

Model Name	Accuracy	Precision	Recall	F-1 Score
Model-1	0.94	0.91	0.93	0.92
Model-2	0.95	0.91	0.93	0.92
Model-3	**0.96**	**0.93**	**0.94**	**0.93**

From the above table, we can see that the model-3 has the best macro average F1 score compared to other models. We can observe that the model-3 improved only by 1%, but, the class 3 scores of other models is very less compared to class 3 score of model 3. Hence, even though there is 1% increase in the F1-score, our results on the test set significantly improved by 3%. Table II depicts the model-3's per class metrics.

Table II
Per Class Metric Values

Class	Precision	Recall	F1-Score
0	0.97	0.97	0.97
1	0.94	0.94	0.94
2	0.94	0.92	0.93
3	0.88	0.92	0.90

IV. Results & Conclusion

We have submitted three predictions on test scores for the three models that we have selected and trained. Table III depicts the performance of all the three models on test set.

Table III
F1 Scores on Test Set

Model Name	Macro Avg F1-Score
Random Forest Classifier (20 estimators)	0.5688443884
Bi-LSTM+LSTM+40 WS	0.8581920173
Bi-LSTM+LSTM(layers=2,dropout=0.5)+40 WS	0.8702780213
Bi-LSTM+LSTM(layers=2,dropout=0.5)+120 WS	**0.90321956**

From the table, we can see that the F1-Score increased from phase to phase of the project and the highest value is 0.903. While training Neural networks, we haven't considered SMOTE technique to up sample the minority class data or down sample the majority class data points. Then we implemented the weighted cross entropy loss to counter attack the class imbalance. An interesting design would be combining data augmentation (without over-fitting the minority class) with class weights. But, because of the less computing resources, we have considered only weighted cross entropy. Below are few key take aways from the project.

1. Using Bi-LSTM for sequential data and especially when the current label is governed by previous time and future time measurements (window size).
2. Using Weighted Cross Entropy loss function to compensate class imbalance.
3. Using Adam Optimizer with StepLR scheduler to avoid over shooting of the step-size.
4. Increasing hidden size and implementing layer normalization may not work for smaller networks.

References

[1] "https://machinelearningmastery.com/random-forest-for-time-series-forecasting". In.

[2] "https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques". In.

[3] Sheng Lu et al. "Dynamic Weighted Cross Entropy for Semantic Segmentation with Extremely Imbalanced Data". In: *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*. DOI: 10.1109/AIAM48774.2019.00053.

[4] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. "Applying LSTM to Time Series Predictable through Time-Window Approaches". In: *Artificial Neural Networks — ICANN 2001*. 2001.

[5] Mehak Khan et al. "Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification". In: *The Journal of Supercomputing* (July 2021). DOI: 10.1007/s11227-020-03560-z.

[6] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015*. URL: http://arxiv.org/abs/1412.6980.