Team 33 Project 1.2: Terrain Identification

Krishna Murali (kmurali2), Mit Pandya (mpandya), James Rotbert (jgrotber)

I. Methodology

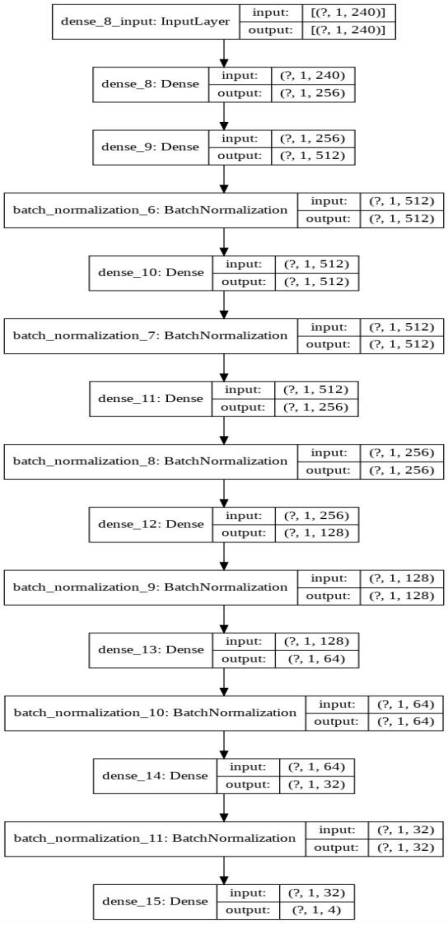
The team's task was to take given training and test data from six accelerometer and gyroscope sensors, positioned on the leg of various subjects. Using this data, four different types of terrain identifications were predicted: solid ground, downward stairs, upward stairs, and grass.

The overall methodology of the approach was derived from the paper, "Three-Dimensional Vibration-Based Terrain Classification for Mobile Robots" (Bai, Guo, Zheng, 2019). The team used the Pandas, Scikit, Tensorflow and Keras Python toolboxes, as well as the pyplot library from Matplotlib.

The final model selected was a Convolutional Neural Network. The input layer to this model is a 256-neuron dense layer. The input shape to this layer is a 1x240 vector. These 240 data points correspond to 40 timesteps of data from the 6 sensors (6 * 40 = 240). It is important to note that each subject and subject ID was treated as a separate grouping of data during training. In other words, data from each subject was fed into the neural network sequentially. It was essential that the model's predictions of terrain classification were derived not for instantaneous points in time, but rather a series of data rows spanning several cycles. This way, the model could more accurately be able to predict the terrain the subject is experiencing using information about how the sensor data changes over time. For the final rows, zero-padding was appended to the dataset. This also allowed for smoothing transitions between sequential subjects and subject IDs.

As seen below, the model contains 7 dense layers, each implementing the Swish activation function. The Swish activation function is given by $f(x) = x \cdot \text{sigmoid}(x)$. Originally, the Relu activation function was used, but further research concluded that better results can be achieved using the Swish activation function as discovered by Google. Between each layer, batch normalization with no dropout is performed to enable maximising the use of available training data. In the final layer, softmax activation is used to derive the one-hot encoded predictions for the 4 classifications of terrain types.

While an RNN with an LSTM was tested as well, the CNN with the 40-time-step input had better performance. The model structure is shown below.



II. Model Training and Selection

A. *Model Training*

In this experiment, there were several misalignments between the X and Y data. The first challenge was that the accelerometer and gyroscope sensors measured data at 40 Hz, while the sampled "Y" dataset containing the classifications was only provided at 10 Hz. The second challenge was that the X time dataset contained 3 decimal places, while the Y dataset only had two.

In order to match the frequencies of both the _x and _y time sampling, we added 0.005s to each Y time and used left join with nearest neighbour values for the Null values in order to create a full merged dataset.

The Sklearn function "train_test_split" was used to divide the data such that 30% of the rows were randomly selected for validation, while 70% remained for training.

The next step was to standardize and scale the x data, following best practices for neural networks. The *StandardScaler* function was used to perform this task. This function simply subtracts each value from the mean of the sensor dataset, and then divides this by the standard deviation of the dataset.

Exploratory data analysis showed us that the datasets contained a significantly higher number of '0' classifications (corresponding to solid ground) than the other classifications (1, 2, and 3). In order to balance the training dataset, the Kmeans Smote data augmentation method was used on the training dataset. The validation data was not augmented. The figure below shows the distribution of the classifications before and after augmentation.

Classification	Training Samples Before KSmote	Training Samples After KSmote
0	712382	712382
1	57263	145003
2	42221	145005
3	145092	145092

The data was augmented to give better representation to classes 1, 2, and 3. To further assist these underrepresented classes, class weights were added as follows after iterating through multiple different weight values. These class weights allowed the underrepresented classes to receive more careful tuning during training:

Classification	Class Weight
0	1
1	50
2	45
3	10

The model was trained using a Nesterov SGD momentum of 0.9. This momentum value was selected after iterating various values between 0.9 and 0.99 and choosing the most optimal.

Categorical Cross Entropy was used as the loss function because this is a multi-class classification problem, as opposed to a binary classification which would use binary cross entropy.

B. Model Selection

In tuning the model, the following parameters were considered:

1. Number of timesteps for the input layer (input shape)
2. Number of neurons in the hidden layers
3. Inclusion of batch normalization layers and the batch size
4. Training with momentum and momentum type.

We used one hot encoding to convert the Y predictions into categorical labels. For the number of neurons in the hidden layers, we followed guidance from the papers cited.

The decision to include the batch normalization layer was driven by our desire to avoid overfitting the data. The Batch Normalization layer stabilizes the training process and results in a higher performing model. After testing various batch sizes, a final batch size of 400 was selected, which corresponds to 10 seconds of 40Hz data.

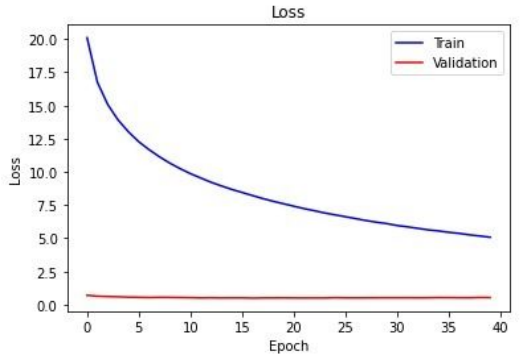
While initially the team decided to include dropout to improve the robustness of the model, we discovered it lowered the accuracy substantially and decided not to include a dropout layer. The prevention of overfitting was accomplished by the batch normalization.

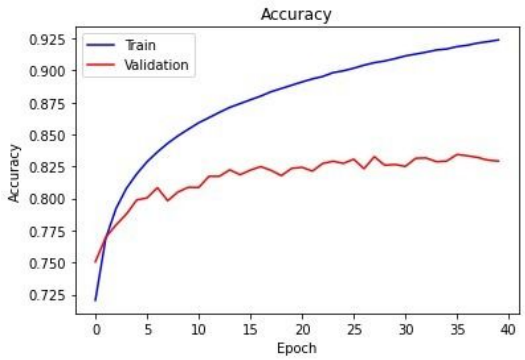
III. EVALUATION

The final results of the training and validation are outlined in the table below:

Result	Accuracy	Loss
Training Set	0.9238	5.078
Validation Set	0.8292	0.5333

The following plot illustrates the loss and accuracy of the model during 40 epochs of training.





The precision, recall, accuracy and F1 scores for each class, and the average results of these metrics is shown in the table below:

Class	Precision	Recall	F1 Score
0	0.88	0.91	0.89
1	0.64	0.56	0.60
2	0.54	0.51	0.53
3	0.70	0.65	0.68
Macro Avg	0.69	0.66	0.68
Weighted Avg	0.82	0.83	0.83

The class weights and the feeding of sequences of data over time (rather than instantaneous data) were the driving factors of improved performance over the baseline model, which implemented an LSTM with no timesteps. It is important to note that including class weights can change the range of the loss during training. In our case, the weights for class 1 and 2 were substantially higher than classes 0 and 3. This meant during training, the model more frequently incorrectly predicts a zero as a one or two, for example. In other words, there is an unbalanced preference to higher weight classes.

SGD was selected over the Adam optimizer so that momentum could be used. Momentum was important for this application because the model was fed sequential timesteps that were mutually dependent. Momentum allowed these timeseries to have a smooth transition.

References

A. Bai, J. Guo, H. Zheng, “Three-Dimensional Vibration-Based Terrain Classification or Mobile Robots,” School of Astronautics, Harbin Institute of Technology, Harbin 150001, China, 2019

B. SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research, by N. V. Chawla K. W. Bowyer L. O. Hall W. P. Kegelmeyer

C. Nesterov SGD Momentum: Nesterov's Accelerated Gradient and Momentum as approximations to Regularised Update Descent, by Aleksandar Botev, Guy Lever and David Barber, Department of Computer Science University College London, 2016