Team 12: Proj-C: Terrain Identification from Time Series Data

Soham Mudalgikar
Department of Operations Research
smmudalg@ncsu.edu

Parimal Rohan Mehta
Department of Computer Science
pmehta3@ncsu.edu

Omkar Rajendra Madure
Department of Industrial Engineering
ormadure@ncsu.edu

I. MOTIVATION

The primary objective of this project is to analyze and predict the terrain based on the readings of accelerometers and gyroscopes fitted at the lower limbs of subjects. We aim to develop a terrain identification system that collects the inertial data measured by normal human walk and predicts underlying terrain types. We use the Convolutional neural network model (1D); we used Keras and other Python packages to identify different types of terrain based on time series data. 1D Convolutional autoencoders are very similar to autoencoders except that all Convolutional and DownSampling operations are done in 1-D.

Auto-encoders typically have a bunch of convolutional blocks, consisting of a ReLU-activated convolutional layer followed by a Dropout layer and then the pooling layer. After that Max Pooling layer is used, which shrinks by a factor of 2. The number of convolutional filters is typically doubled in each successive convolutional block. The VGG-architectures first set this standard. Following the convolutional blocks, we see equal deconvolution blocks, with each convolutional block having the same features as the corresponding block. Hence, we use Convolutional Neural Network (1D).

II. METHODOLOGY

The data consists of several sessions of 8 different subjects collected from the sensor attached to the lower limb. The data available needs augmentation with time windows because the accelerometer and gyroscope data were sampled at 40 Hz, and labels were sampled at 10 Hz. In order to fit the data to any model, we need first to synchronize the attributes and labels and then match the sampling rate for input and class labels by downsampling the label files. Downsampling is preferred to compute evaluation metrics.

Followed by a deconvolutional block, we observe a convolutional layer with a softmax activation function and the same number of filters as the number of output classes. The output shape is [batch size, sequence length, number of classes].

We formed a group of weaker models to remove the occasional misclassifications and achieve a stable model. Hence we built an ensemble of 15 classifiers. We chose hyperparameters that produced models with less performance than our best-performing model. This is called bagging, where we use small random subsets of our data to train multiple smaller models and let them overfit to a subset of the dataset.

Layer (type)	Output Shape	Param #
input_12 (InputLayer)	[(None, 160, 6)]	0
conv1d_77 (Conv1D)	(None, 160, 16)	304
dropout_66 (Dropout)	(None, 160, 16)	0
max_pooling1d_33 (MaxPooling	(None, 80, 16)	0
conv1d_78 (Conv1D)	(None, 80, 32)	1568
dropout_67 (Dropout)	(None, 80, 32)	0
max_pooling1d_34 (MaxPooling	(None, 40, 32)	0
conv1d_79 (Conv1D)	(None, 40, 64)	6208
dropout_68 (Dropout)	(None, 40, 64)	0
max_pooling1d_35 (MaxPooling	(None, 20, 64)	0
conv1d_80 (Conv1D)	(None, 20, 64)	12352
dropout_69 (Dropout)	(None, 20, 64)	0
up_sampling1d_33 (UpSampling	(None, 40, 64)	0
conv1d_81 (Conv1D)	(None, 40, 32)	6176
dropout_70 (Dropout)	(None, 40, 32)	0
up_sampling1d_34 (UpSampling	(None, 80, 32)	0
conv1d_82 (Conv1D)	(None, 80, 16)	1552
dropout_71 (Dropout)	(None, 80, 16)	0
up_sampling1d_35 (UpSampling	(None, 160, 16)	0
conv1d_83 (Conv1D)	(None, 160, 4)	196

Total params: 28,356
Trainable params: 28,356
Non-trainable params: 0

Fig1. Ensemble Unit Summary

A. Data Preprocessing:

We divided the data into training data and testing data. The training data consisted of continuous sessions of data for multiple subjects ranging from subject_001 to subject_008. Our test data consists of data for multiple subjects ranging from subject_009 to subject_012.

B. Sampling Rate

Downsampling is used before fitting the data. The data is read session-wise and subject-wise, which is leveraged to interpolate functions on the y labels to get the x labels. The y_time and y labels are merged together, after which x_time and y_time labels are appended. Sorting of a data frame based on time attribute is done. After that, we extracted the x_time labels. To fill the gap between the two labels, we applied the interpolation technique.

C. Data Augmentation

To provide better training data for the Neural Network model, we matched the sampling frequency for the classification labels by interpolating the data using the NumPy library. As we deal with temporal data, we divided the feature sets into optimized time windows before feeding them into the model. We perform data augmentation on the entire training dataset.

D. Data Imbalance

We tried SMOTE (Synthetic Minority Oversampling Technique) for imbalanced classification. However, we did not get optimal results because of class imbalance. 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 affect the model's overall performance.

II. MODEL TRAINING AND SELECTION

A. Model Training

The data had two issues: 1) class- imbalance 2) No synchronization between x attributes and y labels. Hence attributes and labels are synchronized. We observed that x was sampled at a faster rate than y. We broadcasted the x attributes to the closest y label times. Then we down-sampled the y prediction from 40Hz to 10 Hz output sequence; for this, we used mode of all predictions-y's that fall under the same time-frame.

After that, we consider the class imbalance. We performed exploratory analysis and observed that zero is over-represented and one and two is underrepresented.

60% of the data is used for training, and 40% is reserved for testing. After this, we normalize all gyro_x (say) values present in all sequences in training set in one batch.

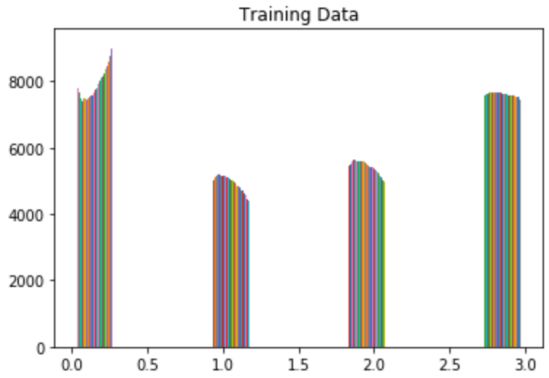

Fig 2. Histogram of Training Data

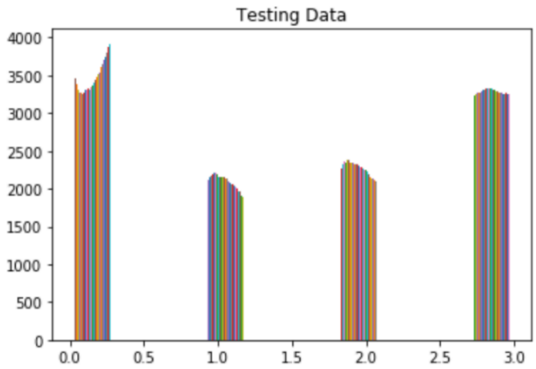

Fig 3. Histogram of Testing data

B. Model Selection

We tried three sets of parameters and chose the best parameters out of this based on validation scores. We expect that using a grid-search method, we can tune the hyperparameters better, but we were unable to perform it due to computational limitations.

We selected the parameters as that depth=4, dropout rate= 0.25, start filters= 16 gives the best performance. Batch normalization does not improve the performance and reduces the performance. We tried changing batch size, but this, too, did not significantly improve the performance. Using 16 parameters that made the model unable to fit the training data, we tried using 25.

Our custom class function is weighted categorical cross-entropy with weights for classes 0-1-2-3 are 2-1-1-2. In short, we weigh a misclassification among classes 0 and 3 three times as much as a misclassification among other classes.

III. EVALUATION

While experimenting with these parameters, we achieved 92.83% categorical accuracy on the training data and 92.26% accuracy on the test data. Training graphs are plotted for a typical ensemble unit. The validation split was set to 0.2. From the graphs, we observe that there is maximum overfitting in the training graphs.

Class	Precision	Recall	f1-Score	Support (Number samples)
0	0.86	0.90	0.88	553912
1	0.96	0.95	0.96	335346
2	0.97	0.96	0.97	362966
3	0.93	0.91	0.92	525536
Accuracy			0.92	1777760
Macro Avg	0.93	0.93	0.93	1777760
Weighted Avg	0.92	0.92	0.92	1777760

Table1. Classification report on unseen data

From the confusion matrix, we notice comparable performance in each of the classes. We observe similarities between classes 0 and 3 as the activities are similar. Also, there is less similarity between classes 1and 2. This is because the activities are opposite to each other. F1 scores of classes 1 and 2 are relatively higher than 0 and 3.

REFERENCES

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

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

[3] J. Brownlee, "Lstms for human activity recognition time series classification," Machine Learning Mastery, Sep 2018. [Online]. Available: https://machinelearningmastery.com/how-to-develop-rnnmodels-for-human-activity-recognition-time-series-classification/

[4]Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images," in IEEE Geoscience and Remote Sensing Magazine, vol. 6, no. 3, pp. 44-68, Sept. 2018, doi: 10.1109/MGRS.2018.2853555.