[Team 9] Proj-C2: [An Enhanced LSTM Model for Terrain Identification from Time Series Data]

Taufiq Islam Protick (tprotic) Ruoyun Ma (rma9) Ramin Niroumand (rniroum)

I. Methodology

A. Data Preprocessing

We synchronized the sampling frequency of the IMUs (40 Hz) and the terrain labels (10 Hz) wtih window warping: a technique which selects a random time range, then down samples or up samples it [1]. We used this method to subsample the IMUs to make the number of IMU samples equal to the number of samples of the terrain labels.

B. Making time series from Sequential data

To convert this sequential data into number of data points, we used the idea of window cropping [1]. We represent one data point as IMUs (after down sampling) of w consecutive time-steps, where w represents a window. Initially, we chose $w = 21$. For each data file, we slid this window from left to right one step at a time until we reach the end. In this way, one data point becomes a time series from $t = 1$ to $t = w$ second, and the corresponding label for that data point will be the label of the terrain the subject was walking on at time $t = w/2$.

C. Handling class imbalance in the data

There was a huge class imbalance in the data (class 0: 251733, class 1: 13804, class 2: 18267, class 3: 51609 samples). How we handled this imbalance is described in detail in Section II-A1.

D. Choosing the best optimizer and hyperparameters

To enhance the baseline model, we chose among a set of optimizers (SGD, RMSPorp, and Adam) and two hyperparameters: 1) the learning rate of the optimizer and 2) the dropout rate in the recurrent layer of our model.

E. Varying w

We varied the size of w from 21 to 31 (inclusive) at every odd number to find out which window size worked best for the task. As both the IMUs and the labels are now sampled at 10 Hz, a window of size 21 means that we are considering a time series of length 2.1 seconds to predict the terrain label the subject was walking on at the middle of this time-span. Our intuition is that, we may need to vary w size to get an understanding of how long (or short) a time series should be in length which is ideal to predict the terrain the subject was walking on.

F. Varying the size of the data set

At first, we started to work with a very small data set. Our choice of datasets of varying size and their findings can be found at Section II-A1.

G. The classification model

We tried to model our time series classification with an LSTM because it handles the vanishing gradient problem and can provide more stable predicting ability in time series classification [3]. We used an LSTM network of one recurrent layer of 100 neurons followed by a fully dense output layer of 4 neurons with softmax activation representing four classes. In the recurrent layer, ReLUs were used as activations. To enhance this baseline model, we did hyperparameter tunings with our choices of w size, dataset size, optimizer, dropout rate at the recurrent layers and learning rate of the optimizer which is described in Section II.

As our toolboxes in this phase, we used python's keras and tensorflow for as our deep learning framework. Specifically, we used Sequential library from keras.models; LSTM, and Dense from keras,layers; SGD, Adam, RMSProp from keras.optimizers. We also used scikit learn to split the dataset into training and validation. We used classification_report from sklearn.metrics to generate the accuracy, precision, recall and f1 scores.

II. Model training and Selection

In this section, we will discuss in detail about how and with which data the models were trained and what hyperparameters were tuned.

A. Model Training

Before training, the data had to be further processed to handle imbalance. Next, the data were divided into test-train-validation split. In this Section we describe them in detail.

1) Handling imbalances in the data by data augmentation: After our hyperparameters, optimizer, and w size were chosen, we finally chose an optimum model and trained it on three different types of datasets. Our choices for datasets vary in size and handle class imbalance differently or not at all.

i **Choice 1 (handles class imbalance):** Initially we down sampled the top 3 over-represented class (class 0, 2, 3) so that their sample sizes match with the sample size of the most under-represented class. (N = 55,216 samples with 13,804 samples from each class).

ii **Choice 2 (does not handle class imbalance):** Next, we chose the original data set with no augmentation. (N =

335,413 samples; class 0:251733, class 1: 13804, class 2: 18267, class 3: 51609).

iii **Choice 3 (handles class imbalance):** Finally, we up sampled the top 3 under-represented class (class 1, 2, 3) to match their size with the most over-represented class (class 0). To do this, we needed to do sampling with replacement for the under represented classes (N = 1,006,932 samples with 55,216 samples from each class).

2) Training/validation split: For all three choices at Section II-A1, we randomly shuffled our training set and extracted 20% data to create a "separate" test dataset. This test dataset was never used in the training process. During the training phase, 64% data were used for training and the rest 16% data were used for validation.

B. Model Selection

1) Choosing the best optimizer, learning rate, and dropout rate: We compared three optimizers (SGD, RMSProp, and Adam) on the baseline model and chose the optimizer based on their performances on validation accuracy. Figure 1 (left) shows the validation accuracy run over 50 epochs with each optimizer. Although the learning curve fluctuates at places for the validation curves, it is clear that the **Adam (blue curve Figure 1 (left)) optimizer** performs the best overall.

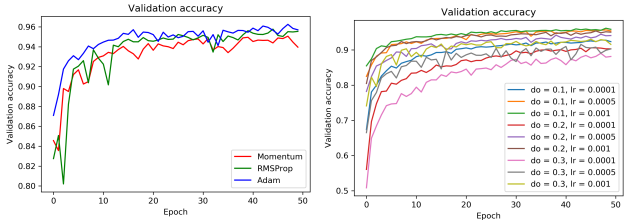

Fig. 1. (Left) The comparisons among baseline models trained with optimizers. (Right) The comparisons among tuned model with 9 combinations of dropout rate (choice = 0.1, 0.2, 0.3) and learning rate (choice = 0.0001, 0.0005, 0.001)

The model with no regularization showed significant overfitting as in all three optimizers as the training accuracy levels off at 1, and the validation accuracy is always lower than the corresponding training accuracy. To reduce overfitting, we employed dropout. Dropout randomly drops some neurons along with their connections during training. This prevents neurons from coadapting too much, hence a model using dropout is less prone to overfitting [4]. As another hyperparameter, we tuned the learning rate of the optimizer. We found that a **Dropout rate of 0.1** and a **learning rate of 0.001** was provided the best validation accuracy (**green curve in Figure 1 (right)**).

2) Varying w: The tuned model (with an Adam optimizer, dropout rate = 0.1 and learning rate = 0.001) was trained with time series of varying w size in the range [21,31].

Figure 2 shows the validation accuracy of six models trained with time series data with six different size of w. Although, it is not clear which window size performs best, **we observed that $w = 31$ (dark brown curve in Figure 2) has a better and consistent performance** over all epochs. So for the rest of the steps, we chose a time series of window length $w = 31$.

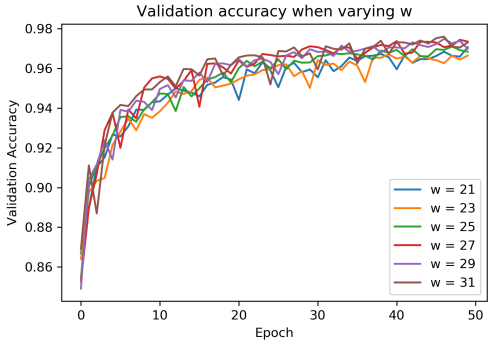

Fig. 2. The validation accuracy of different models trained with time series of different length of w

3) Choosing data set of different size from Section II-A1: In Section II-A1, we mentioned three choices for the size of the dataset. We started with smaller sized dataset (Choice 1) and gradually increased its size in Choice 2 and 3. Of these choices, both choice 1 and 3 handled the class imbalance by down sampling and up sampling respectively as we described in Section II-A1. In choice 2, we chose to train the tuned model with the original dataset with no handling of class imbalance.

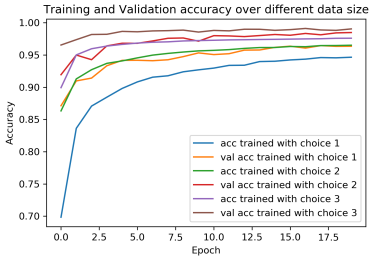

Fig. 3. The training and validation accuracy when tuned model is trained with three choices of dataset mentioned in Section II-A1

Figure 3 clearly shows that the validation accuracy of the model trained with **choice 3 (dark brown curve)** outperforms the other two models. Over all epochs, the choice-wise training accuracy remains always less than its corresponding validation accuracy. Therefore, no model trained with either choice 1, 2, or 3 overfit the data. Moreover, the curves suggests that if a model is trained with more data, it performs better no matter class imbalance is handled or not.

III. Evaluation

In this Section, we will discuss the performance of the final model (**dropout rate = 0.1, learning rate = 0. 001, trained with an Adam optimizer, trained with time series of window size $w = 31$, and where the dataset we chose is choice iii of II-A1**) both on the validation and the separate test data. Finally, we will show the final model's predictions of two random time snippets of 25 seconds from the competition test data.

A. Performance on the validation data

The final model was run for 20 epochs to summarize the impact of how less it overfits the data.

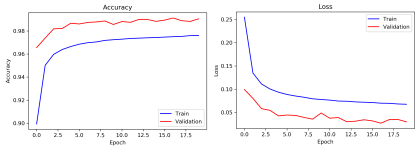

Fig. 4. The categorical accuracy and loss of the final model.

As we can see in Figure 4, due to adding a dropout layer and a suitable learning rate, the the model does not overfit at all during training. What's more surprising is that the validation accuracy is always far better than the training accuracy over all epochs.

B. Performance on a "separate" test dataset

As we mentioned earlier, we kept a separate test dataset of 201,387 samples and never used that during training. This was done to give us an idea of how much error we can expect from the unlabeled test dataset. The confusion matrix at Figure 5 gives us an idea of the actual labels and the predicted labels of these 201,387 samples.

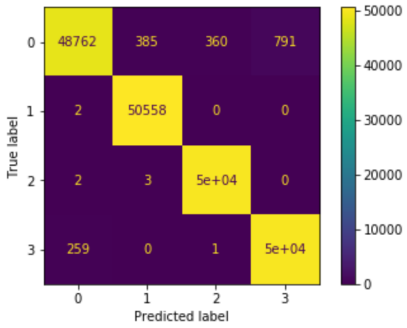

Fig. 5. The confusion matrix that shows the prediction on the "separate" test data. The labels at the left are the true labels and the labels at the bottom are the predicted labels.

A numerical analysis of the confusion matrix is found at Figure 6 which gives the numerical interpretation of the precision (what proportion of the predicted labels was actually correct), recall (what proportion of the samples was identified correctly) and the F1-score (harmonic mean of precision and recall) of the model. From Figure 6, we learn that the best precision score was for class 0 (walking on solid ground). Best recall was for class 1 (going down the stairs), and best F1- score was for class 2 (going up the stairs). The precision score means that 99.46% of the predicted labels with Class 0 were actually correct. The recall means that, of 100% of the actual samples with label 1 were identified correctly. On the other hand, the worst precision was for class 3 (walking on grass), which means only 98.45% of the predicted labels were actually class 3. The least recall comes from class 0 (walking on solid ground). This means, 96.95% of the actual class 0 samples were identified correctly.

	precision	recall	f1-score	support
Class 0	0.9946	0.9695	0.9819	50298
Class 1	0.9924	1.0000	0.9962	50560
Class 2	0.9929	0.9999	0.9964	50163
Class3	0.9845	0.9948	0.9896	50366
accuracy			0.9910	201387
macro avg	0.9911	0.9910	0.9910	201387
weighted avg	0.9911	0.9910	0.9910	201387

Fig. 6. The precision, recall, f1-score and the overall accuracy of the final model on a "separate" test dataset. The best and worst scores are marked in green and red respectively.

C. Illustration of the prediction of the competition test set

Using our final model, we predicted the labels for the competition test datasets. Figure 7 shows (assumed) improvement of the model at ProjC2 compared to ProjC1 at the same time snippet prediction. From Figure 7 (top left) it looks like

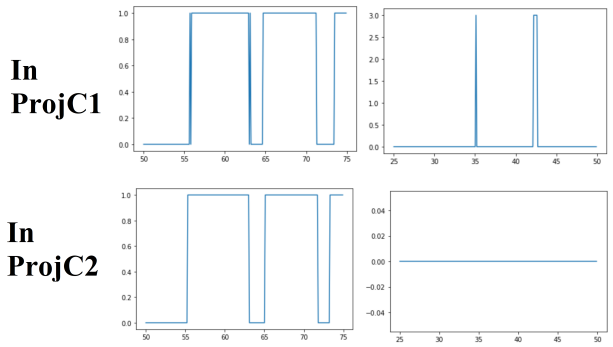

Fig. 7. Prediction of a time snippet of 25 seconds from random subject(s).

that the subject was alternating his walking terrain from solid ground (0) to down the stairs (1). However, the transition from 0 to 1 is not smooth at $t = 56$ and $t = 63$ (Notice the hazy transition from 0 to 1). In ProjC2 (bottom left), the transition was smooth.

What we infer from Figure 7 (top right) is that, large spikes from 0 to 3 at 35^{th} and 43^{rd} time step could be a prediction error, or the subject could be walking on solid ground(0) and then be slightly shifting his feet to the grass(3) for a very short interval and then going back to solid ground (0). But, from an enhanced model's prediction at ProjC2 (bottom right), it looks like that those spikes at ProjC1 predictions could be an error.

REFERENCES

[1] Wen, Qingsong, et al. "Time series data augmentation for deep learning: A survey." arXiv preprint arXiv:2002.12478 (2020).

[2] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[3] Kim, Kangil, et al. "Stable forecasting of environmental time series via long short term memory recurrent neural network." IEEE Access 6 (2018): 75216-75228.

[4] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.