[Team 29] ProjC2 [Terrain Identification From Time Series Data]

Radek Pudelko (rmpudelk@ncsu.edu), Melvin Orichi Socana (msorichi@ncsu.edu), Stenson Sampson (sdsampso@ncsu.edu)

METHODOLOGY

There have been significant changes in our data processing and model architecture since our initial model. We still follow a CNN approach and use a window of data to make our predictions which was originally inspired by audio processing neural networks [1]. Our team believes that CNNs are the most intuitive approach as a window of a data as it considers the spatial relationship within the data [2]. The data surrounding the action we want to predict should contain enough features for classification. This window of data has grown from 1 second to 2 seconds (80 samples per channel) centered around the time to predict. 1 second was originally chosen because that roughly corresponds to the gait cycle time [3]. Now with 2 seconds, the model can see 2 gait cycles, which we found beneficial for performance.

In the collaborative phase, we split the raw train sessions of data into train and validation sets before processing it, now we process all the data according to our 2 second window before shuffling it and splitting the data off into train, validation, and test sets. Data normalization has not changed, we subtract by the mean and divide by the standard deviation. When splitting the data, we first split a test set of 25% from the whole data set. Then we take a validation split of 25% of the remaining data. This corresponds to 56.25%/18.75%/25% train/validation/test splits. From here, data is resampled to account for imbalance. Resampling the train, validation, and test sets results in 30,000:10,000:10,000 samples for train, validation, and test sets. This resampling results in a down sampling of solid ground and grass classes and up sampling of upstairs and downstairs classes.

There were major changes to the model architecture. We now use the predictions from 7 models using the same architecture but trained from different train/validation sets. After training, these models are then evaluated using a global test set. From this evaluation, the f1 scores are taken to be used as weights for predictions. This way, models with higher f1 scores carry more weight in their vote and avoid the issue of split vote between 2 or more classes as each weight is unique.

The number of convolutional, pooling and fully connected (fc) layers have been doubled and follow the architecture in figure 1. Dropout and batch normalization were added to improve model generalization. A dropout of .2 was used after each convolutional layer and .5 after each fc layer. Batch normalization follows the pooling layers and the fc layers. This increase in model capacity is useful for the model to extract the features needed for classification, especially now that the amount of data entered per prediction has doubled with the window time.

We also added a custom filter which takes the consensus prediction from the 7 models and filters it according to the length of the predictions. This filter monitors the current state of the predictions and looks at the length of predictions after a change in state occurs. If changes in state are less than a minimum time period of 1.2 seconds, then they are replaced with the previous state. For instance, the model may be predicting walking on solid ground followed by .3 seconds of walking upstairs. The walking upstairs prediction is replaced with the original state of solid ground as .3 seconds does not meet the minimum filter length to be considered a true event. The output of this filter is the final output for our model.

The consensus approach establishes greater confidence in the predictions, particularly where there is a transition in state, as most of the models must agree to the prediction. This also helps to prevent spikes from entering to predictions as it is unlikely that all models would predict a spike at the same location given that they were trained of different sets of data.

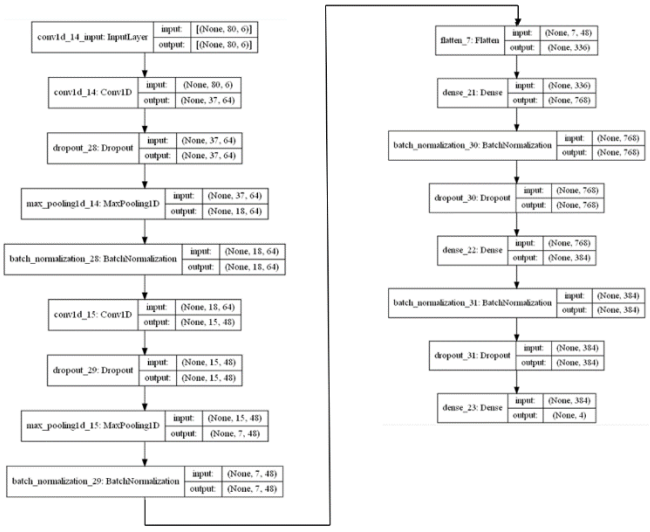

Figure 1. Single Model Architecture

We continued using the TensorFlow library in python. The layers are defined under tensorflow.keras.Layers. The input layer is shaped (80, 6) corresponding to 80 samples of sensor data (2 second @ 40 Hz) and 6 channels for accelerometer and gyroscope x,y and z signals. The input goes through a 1D convolution layer (Conv1D) with 64 filters (kernels), kernel size 8 and stride of 2. Next is dropout (Dropout) of .2, then max pooling (MaxPool1D) using a pool size of 2. This is followed by batch normalization (BatchNormalization). This repeats, except the convolutional layer uses 48 filters with kernel size of 4 and stride of 1. Then, the output is flattened (Flatten) so that it can be run through

fully connected (fc) layers (Dense). From here, a fc layer with 768 neurons was used. This is followed by batch normalization and dropout of .5. Then this repeats, but with a fc using 384 neurons. These convolutional and fully connected layers are activated by ReLu. Finally, the last layer has 4 output neurons activated by softmax.

MODEL TRAINING AND SELECTION

Model Training

As mentioned above, train, test and validation sets are split and resampled following data preprocessing. Data preprocessing involves normalizing the data followed by splitting it into 2 second windows centered around the prediction time. The distribution of data from all subjects and sessions is as follows: standing/walking on solid ground (0) (75%), walking upstairs (1) (4.1%), walking downstairs (2) (5.5) and walking on grass (2) (15.4%). It is noted that taking 2 second windows centered around the prediction time removes a small number of labels near the beginning and end of each file (1s) from the set of possible labels as there is not enough data to have a centered label. Truncating this small amount of the data at the start and end allowed for the output labels and original sensory information to synchronize for processing.

Model Selection

One of the challenges in creating neural networks is selecting the optimal hyperparameters. For every layer used, there may be one or more additional hyperparameters adding additional dimensions to consider in the search space, not to mention other hyperparameters such as learning rate, batch size, etc. [4].

The hyperparameters for these layers were chosen using a random search with the keras_tuner library. Random search was used to determine the number of filters, kernel size and number of strides for the convolutional layers, the pool size for the pooling layers and the number of neurons in the fc layers. The tuner was run targeting the best validation accuracy as its metric employing early stopping with a patience of 5 epochs. Additionally, Adam was used with a learning rate of .0001. The search space is shown table 1.

Table 1 - Hyper Parameter Settings

Hyperparameter Search Space		
Layer	Feature	Search Space
Conv1	Filters	48,64
	Kernels	4,6,8
	Strides	2,4
Pooling1	Pool Size	2,4
Conv2	Filters	32,48
	Kernels	1,2
	Strides	2,3
Pooling2	Pool Size	2,3
Dense1	Neurons	512:128:1536
Dense2	Neurons	256:64:512

EVALUATION

The loss and accuracy plots were very similar across our 7 models. Presented in figures 2 and 3 are the loss and accuracy scores for one of our models. Interestingly, the loss and accuracy for the validation sets were better than the actual train sets.

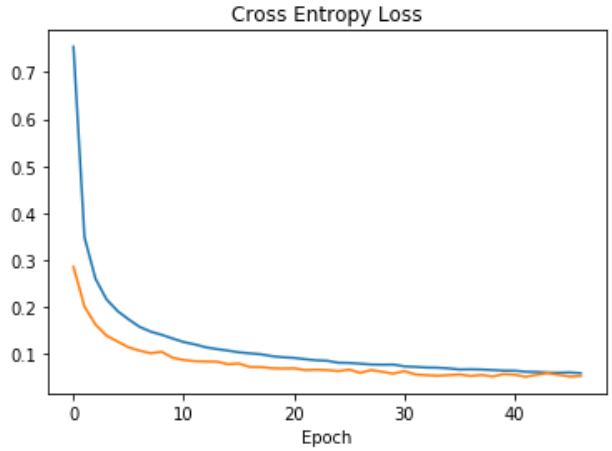

Figure 2. Single Model Loss

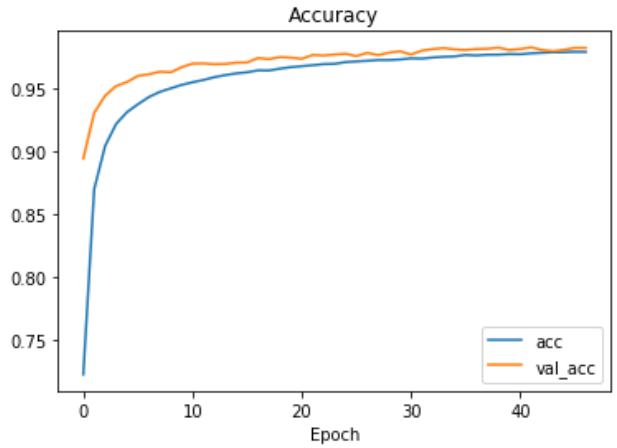

Figure 3. Single Model Accuracy

Presented in table 2 is the evaluation of our consensus model (7 models) on the global test set. This is unfiltered as the test does not contain sequences, but random 2s windows of data from different sessions.

Table 2. Consensus Model Evaluation using Test Set

	Score				
Metrics	Overall	0	1	2	3
Accuracy	98.3	96.7	99.1	98.9	98.3
Precision	98.3	96.7	99.1	98.9	98.3
Recall	98.3	96.5	99.6	99.6	97.3
F1	98.3	96.6	99.4	99.3	97.8

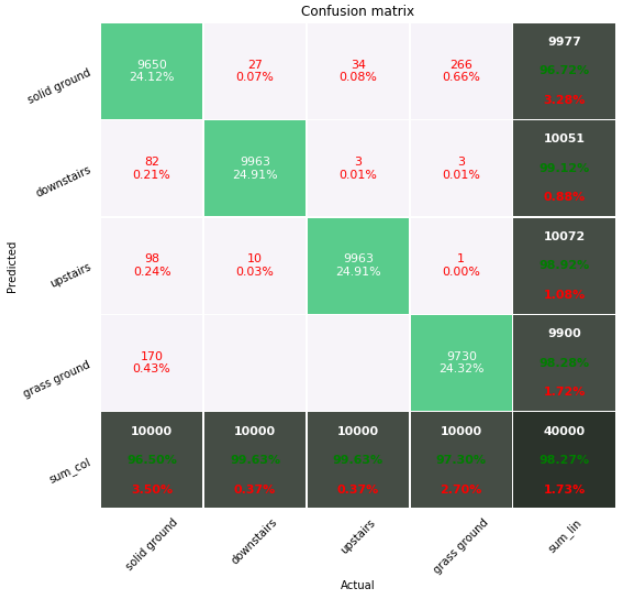

Figure 4. Consensus Model Confusion Matrix on Test Set

Unfortunately, because we mixed all the data from the various subjects and sessions, it is difficult to evaluate the model in combination with the custom filter as the individual models have been trained on some parts of all the sequences. Therefore, we tabulate the average scores for the model predicting for the raw session data in table 3.

Table 3. Consensus Model with Filter Evaluation Scores

	Score				
Metrics	Overall	0	1	2	3
Accuracy	92.8	99.8	87.8	89.2	94.2
Precision	92.8	99.8	87.8	89.2	94.2
Recall	99	97.1	100	100	98.9
F1	95.6	98.4	93.4	94.2	96.4

Finally, we can look at the model predictions for one of the test sets, which have not been seen before. Compared to our previous model's predictions, these can be judged much more easily by a human. Previously, the predictions, while having ~87% accuracy, were littered with spikes, making it very difficult to tell what the model was predicting. Now with the consensus model and filter, we have achieved a score of 92.4% from prediction 3 and spikes are mostly gone. From our examination, errors are most likely due to predicting too soon or too late when there is a change in class. For instance, the model may predict that the person begins walking upstairs .5 seconds before they do.

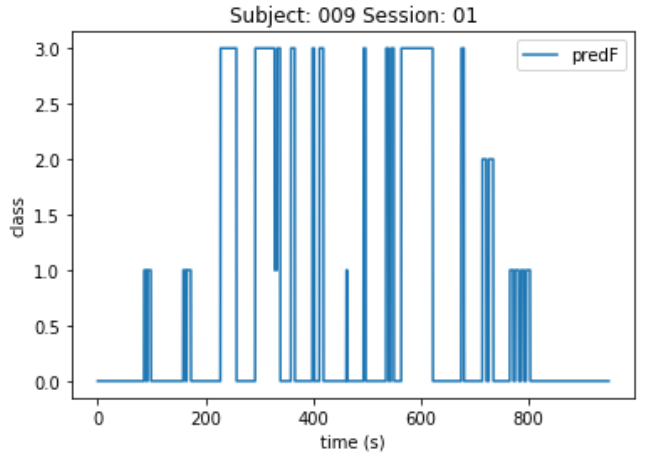

Figure 5. Model Predictions

Shown in figure 5 is the predictions from our model for subject 9 session 1. From this figure, it is easy to tell when the subject is walking on solid ground and when they begin to walk to change classes such as walking on grass after 200 seconds.

REFERENCES

[1] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, "End-to-end environmental sound classification using a 1D convolutional neural network," Expert Systems with Applications, vol. 136, pp. 252–263, Dec. 2019.
[2] M. Z. Hashmi, Q. Riaz, M. Hussain, and M. Shahzad, "What lies beneath one's feet? terrain classification using inertial data of human walk," Applied Sciences, vol. 9, no. 15, p. 3099, Jan. 2019.
[3] "Gait cycle," Gait Cycle - an overview | ScienceDirect Topics. [Online]. Available: https://www.sciencedirect.com/topics/engineering/gait-cycle. [Accessed: 16-Oct-2021].
[4] M. Zeng et al., "Convolutional Neural Networks for human activity recognition using mobile sensors," 6th International Conference on Mobile Computing, Applications and Services, 2014, pp. 197-205, doi: 10.4108/icst.mobicase.2014.257786.