[Team 42] ProjC2 Technical Report: IMU-to-Terrain Classification

Alexander Kyu
Email: awkyu@ncsu.edu

Anthony Kyu
Email: amkyu@ncsu.edu

Michael McKnight
Email: mpmcknig@ncsu.edu

I. Methodology

The Alexnet paper [1] is a well-known example of applying a deep convolutional neural network to a classification task. In the paper an architecture was proposed with five convolutional layers followed by three fully connected dense layers. After the first two dense layers, dropout layers of 0.5 were used to prevent overfitting of the data. Many other of the techniques used in the network were to help prevent overfitting as the nature of their problem had an incredible amount of data. Some of these techniques included flipping the down sampled images, overlapping the max pooling kernels, and using ReLU for the activation layers. We used this architecture for inspiration for our own. We did not use some of the proposed techniques such as the parallel GPU usage and the synthetic data from flipping, as our use case has significantly less data points. We also had to construct images from our data and hand crafted features, which will be described in more detail later. We also found that after our optimization search Swish was a more effective activation function than ReLU for our model.

When optimizing our model, we decided to use Swish as the main activation function. While the Rectified Linear Unit (ReLU) seems to be one of the most widely used activation functions, another new function has been shown to work better among multiple datasets and models. One journal showed that simply replacing ReLU on deeper models improved top-1 classification accuracy for ImageNet by 0.9% for Mobil NASNet-A and 0.6% for Inception-ResNet-v2. This improvement seems to be a result of Swish--like ReLU--being unbounded above, bounded below, but also non-monotonic and smooth [2]. These properties were desirable to use in the chosen model due to its ability to avoid saturation from the unboundedness, strong regularization from is bounded bottom, ability to express small negative outputs from small negative inputs from its non-monotonicity, and its ability to optimize and generalize well due to its smoothness.

In another publication, a deep neural network was developed to classify terrain for an autonomous vehicle using IMU data. The autonomous vehicle here was driven on five different terrains and the IMU recorded acceleration in all three directions, and yaw, roll, and pitch--the gyroscopic data in our case. Several features were extracted from this data, including the averages and standard deviation with a 5 point window. All together a model was created with inputs of the Y acceleration, Pitch, Roll, and Yaw average and the Roll standard deviation. A single hidden layer with ten neurons was used and led to an output layer with five neurons – one for each terrain class. Training was done by feeding in five datasets with 100 samples each, one for each terrain, creating a balanced set [3]. The features used here inspired several features engineered in our model, specifically averages and standard deviations within a window.

Therefore, based on the literature review above and our validation results (discussed later), a Deep Convolutional Neural Network was developed as our final model with the architecture shown in Figure 1. The dropout layers were added between the 2D convolutional and max pooling layers to prevent overfitting.

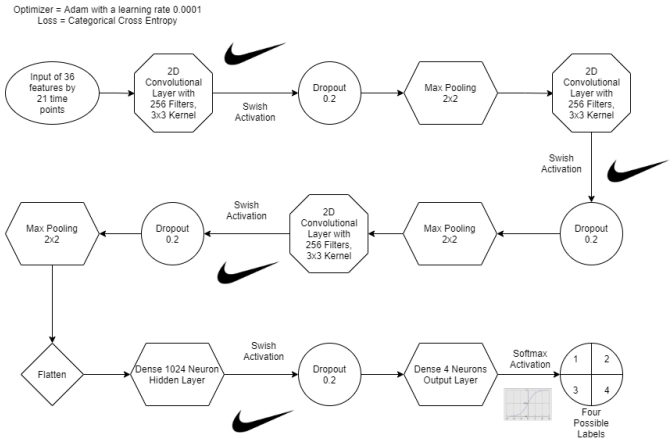

Fig 1. Model architecture consists of 36 hand-crafted feature inputs, three convolutional layers with swish activations, dropout layers, and a dense layer outputting four probabilities for each label.

II. Model Training and Selection

A. *Model Training*

There were numerous methods of preprocessing the data before using specific models. First, the data was down sampled to 10 Hz to match the x data time point closest to the corresponding y data label. Afterwards, several other features

were generated. Derivatives of the given data were taken to generate two extra features, jerk and angular acceleration, for each axis, X, Y, and Z, for both acceleration and angular velocity, respectively. Additionally, a moving average and standard deviation were taken across the data using a window size of 21 (~2 seconds), with the corresponding data point as the center of the window. With these additional features, additional insight into the variation of the IMU data was revealed which is important for terrain identification.

From these 36 total features, the data was then reorganized into matrices of window size by features (21 x 36) and was stored as .npz files for each dataset. Each matrix corresponded to a single label.

To deal with imbalanced data, the smallest class size was first identified for the training set – class 1 or walking upstairs. Then each of the other classes were randomly sampled to achieve equal amounts of data totaling 55216 total samples (13804 samples for each class). 50216 were used in training while 5000 were used in validation. These were split chronologically as LSTM models were initially considered. However, in hindsight, with our chosen model, splitting this data randomly would have probably resulted in better training and validation datasets as the convolutional model does not rely on any time series aspect, unlike an LSTM.

B. Model Selection

The models proposed above were all tested against several metrics, such as categorical validation accuracy and F1 scores. However, the model with the highest F1 score on Gradescope was the Convolutional Neural Network (Figure 1). Therefore, we selected that model as our final model, and worked to optimize it.

A down sampled (1%) grid search was performed, using the "talos" library, to determine the optimal number of convolutional layer filters (128, 256, 512), number of dense layer neurons (256, 512, 1024), Adam optimizer learning rate (0.0001, 0.001, 0.01), the type of activation function in the convolutional layers (ReLU, Swish), the dropout rate (0.2, 0.3, 0.5), and the batch size (36, 144, 512, 2048). From this initial grid search, we determined which parameters had little effect (batch size, number of dense layer neurons, and number of convolutional layer neurons) and what values of parameters led to extremely low Validation F1 Scores (learning rate of 0.01 or higher). Based on these results, another, more-detailed, down sampled (50%) grid search was performed, further optimizing the convolutional layer filters (128, 256), the learning rate (0.0001, 0.001), activation function (ReLU or Swish), and dropout (0.2, 0.3, 0.5). The dense layer neurons and batch size were set at 1024 and 36, respectively, because these were determined to be the best values for these parameters from the first experiment. Below is the table showing the grid search results which shows that 256 Convolutional Layer Filters, a Dropout of 0.2, a Learning Rate of 0.0001 and a Swish activation function gives the optimal Deep CNN model. Time and GPU constraints did not allow for the investigation of other hyperparameters.

Table 1. Hyper-parameter combinations for experiment two (varying Convolutional Layer Filters, Dropout, Learning Rate for Adam Optimizer, and Activation Function). Red indicates the best performing combination in all metrics.

Trial	Convolutional Layer Filters	Dropout	Learning Rate (Adam)	Activation Function	Validation F1 Score
7	256	0.2	0.0001	Swish	0.922191
8	256	0.3	0.0001	Swish	0.907754
6	128	0.2	0.001	ReLU	0.897388
10	128	0.3	0.001	ReLU	0.884192
0	128	0.3	0.0001	Swish	0.875788
1	128	0.3	0.0001	ReLU	0.872288
5	256	0.2	0.001	Swish	0.868881
3	256	0.5	0.0001	ReLU	0.827860
9	256	0.3	0.001	Swish	0.810955
4	128	0.5	0.001	ReLU	0.776382
11	256	0.5	0.0001	Swish	0.766347
2	128	0.5	0.0001	Swish	0.709840

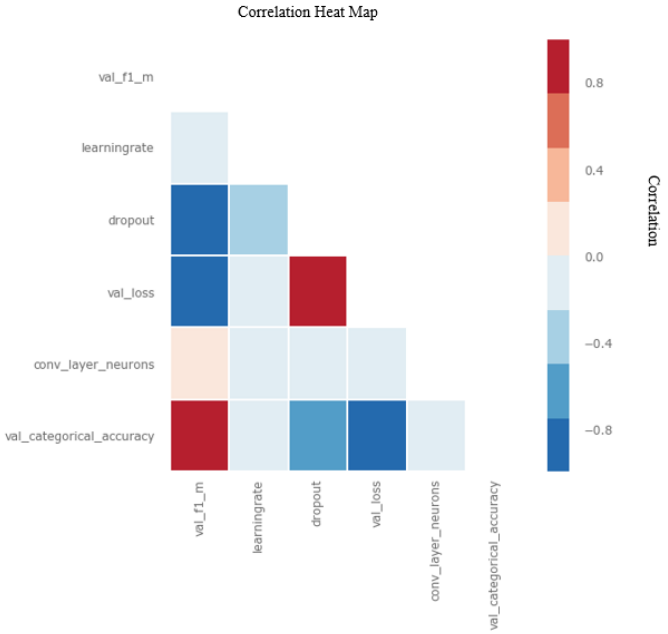

Fig 2. Heat map of the correlation between validation metrics and hyperparameter values.

Furthermore, Table 1 shows that Swish performed marginally better than ReLU activation function, which can be observed between Trial 0 and 1. From Figure 2, that higher dropout is negatively correlated with high validation F1 scores, indicating that dropout should be kept to smaller values such as 0.2 rather than 0.5. This is also observed in Figure 3. Meanwhile, other hyperparameters such as convolutional layer neurons/filters and learning rate are weakly correlated to high

validation F1 scores (Figure 2). Looking at the boxplots (Figure 3), there seems to be no significant difference between the different learning rates and number of convolutional layer neurons/filters. Since there was a time constraint, we were unable to perform more experiments to determine if there is a significant difference between these different values.

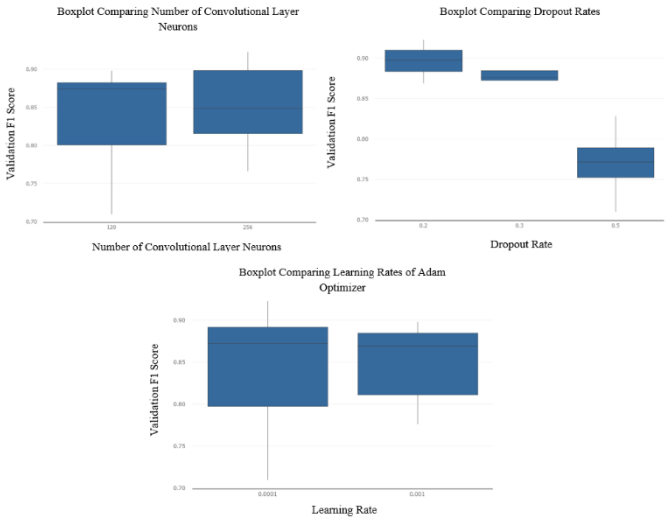

Fig 3. Boxplots comparing varying hyperparameter values against each other based on Validation F1 Scores.

III. EVALUATION

Table 2. Shows the precision, recall, accuracy, and F1-scores for each class, and shows the total accuracy of the model's predicted classifications against test data.

Class	Precision	Recall	F1-Score	Accuracy
Solid Ground (0)	0.7663	0.9008	0.8281	0.9197
Downstairs (1)	0.9677	0.9935	0.9804	0.9935
Upstairs (2)	0.9753	0.9737	0.9745	0.9724
Grass (3)	0.9463	0.7610	0.8436	0.7498
			Total Accuracy	0.9200

Table 3. Shows the 5-Fold Validation F1 Score Results Using the Testing Data. The testing data was split into five groups, and an F1 Score was generated for each group (fold).

Fold	1	2	3	4	5
Validation F1 Score	0.8648404538430648	0.8694104216593623	0.8933375553109537	0.8859451752563804	0.8679025316231771

The metrics listed in the table above were calculated using custom functions to improve the accuracy of the precision, recall and F1 scores. The accuracy for each class was calculated by looking along the diagonal of a confusion matrix. The classes for Solid Ground and Grass terrains did not achieve as high metrics as the classes for Downstairs and Upstairs. This seems to indicate that our model more accurately identifies Downstairs and Upstairs terrains better than Solid Ground and Grass terrains (Table 2).

We also did a 5-fold validation of F1-scores using the test data to further validate and evaluate the model (Table 3). The graphs below show the training and validation loss and categorical accuracy achieved by our model from the validation data that was generated splitting the augmented training data (Figure 4). Despite the very high accuracy and F1 scores, our model's F1 score in Gradescope was determined to be 0.809 which was quite a bit lower. This comparatively low F1 score is likely due to just natural variation in the data. Furthermore, our model may have resulted in better scores if the training testing data was split more evenly across classes.

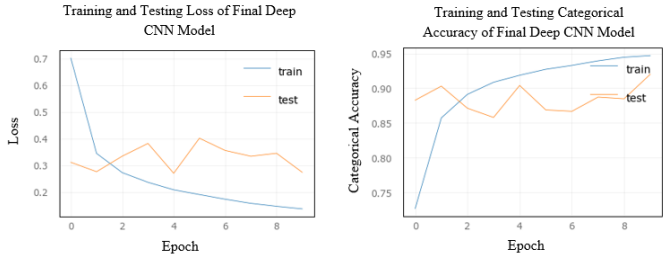

Fig 4. Line graph showing training and testing loss and categorical accuracy from our final Deep CNN model's predictions over several epochs.

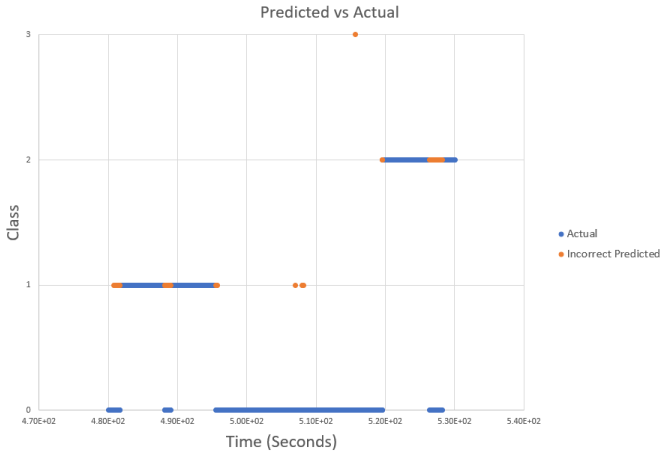

Fig 5. 50 second snapshot of predicted data for subject 1 that shows the model's misclassification when compared to the ground truth.

In Figure 5, we show a test case of our model classifying the terrain for a portion of subject 1's IMU data. From the figure, one can see that the model tends to misclassify in the transition between terrain types, but the time lag is minimal in that boundary region. The model also tended to not be able to respond to quick changes in the terrain types (shown in the stair regions). This may be caused by the 2-second window size used whereas a smaller window size could possibly respond more quickly. The model did not have any large issues with random misclassifications with only a few errant data points in the middle of a terrain type.

REFERENCES

[1] P. Ramachandran, B. Zoph, and Q. Le Google Brain, "SWISH: A SELF-GATED ACTIVATION FUNCTION,." Accessed: Mar. 22, 2021. [Online]. Available: https://arxiv.org/pdf/1710.05941v1.pdf?source=post_page.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[3] R. Jitpakdee and T. Maneewarn, "Neural networks terrain classification using Inertial Measurement Unit for an autonomous vehicle," *2008 SICE Annual Conference*, Aug. 2008, doi: 10.1109/sice.2008.4654717.