Final Project Report - Embedded-NIST

Mei Qi Tang
Electrical Eng.
McGill University
Montreal, Canada
mei.q.tang@mail.mcgill.ca

Tyrone Wong
Electrical Eng.
McGill University
Montreal, Canada
tyrone.wong@mail.mcgill.ca

***Abstract*—This document is the project report for the final project in ECSE 421: Embedded Systems class. Inspired by the MNIST dataset, this project consists of designing a neural network for recognizing handwritten digits, using the accelerometer sensor on the myRIO board. This report will describe the methodology used to build the dataset as well as the neural network, present the design parameters used, and analyze the results and performance metrics such as the training loss, the computation time, and the effectiveness of the approach.**

***Index Terms*—MNIST Dataset, MyRIO, Digit Classification, Neural Network, LabVIEW**

I. Description

A neural network consists of an input layer, an output layer, and one or more hidden layers. Each of these layers are composed of linear classifier neurons that each calculates the dot product of an input vector $\mathbf{x}$ of dimension N with a weight vector $\mathbf{w}$:

$$net = \sum_{n=0}^{N-1} w_i x_i \tag{1}$$

We did not implement an optional bias in our neuron equation for this project. The calculated net is then passed into an activation function, in our case the sigmoid function:

$$\sigma(net) = \frac{1}{1 + e^{-net}} \tag{2}$$

The neural network used for this project has the same amount of input nodes as the number of data sampled for all axes, per digit drawing. This number will depend on our specific sampling rate during data collection stage. The number of hidden layers could be 1 or more but we have been able to achieve satisfactory results with only one. The number of nodes per hidden layer has great impact on the time it takes to complete each training iterations. We had to lower down the number of nodes from our initial pick to decrease the total training time needed. Finally, the network will have 10 output nodes representing the 10 digits we could be drawing.

The input vector $[i_0, i_1, ..., i_{2n}]$ is equal to the signal vector $[a_{x1}, a_{x2}, ..., a_{xn}, a_{y1}, a_{y2}, ..., a_{yn}]$, where n is our sample rate. The 10 output nodes are represented by the output vector $[o_0, o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9]$ for the 10 digits.

The loss function used to measure the error between the computed output prediction of the $\mathbf{y_n}$ training iteration and the expected target output $\mathbf{t_n}$ is the Mean Squared Error (MSE) function:

$$J_n(\mathbf{W}) = \frac{1}{2}(\mathbf{y_n} - \mathbf{t_n})^2 \tag{3}$$

This loss function defines an error that is a function of the network weights $\mathbf{W}$, which are first initialized with a random number of uniform distribution ranging from $\mathbf{W} \in [-1, 1]$. To train the weights, we will be optimizing using the gradient descent algorithm, which updates the weights iteratively by stepping along the error gradient towards a local minimum following this update rule:

$$\mathbf{W_{new}} = \mathbf{W_{old}} - \alpha \nabla J(\mathbf{W}) \tag{4}$$

A small learning rate $\alpha \ll 1$, such as 0.001, should be used to avoid overshooting a local minimum when iterating through the optimization algorithm. The gradient can be defined as a derivative of the loss function with respect to the network weights:

$$\nabla J(\mathbf{W}) = \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = \frac{\partial J(\mathbf{W})}{\partial y} \frac{\partial y}{\partial \mathbf{net}} \frac{\partial \mathbf{net}}{\partial \mathbf{W}} \tag{5}$$

The gradient descent algorithm would be used for the backward propagation step, after the forward pass.

The 10 digits are one-hot encoded as below:

TABLE I
One-hot encoding of digits '0' - '9'

Digit	Encoding
0	1000000000
1	0100000000
2	0010000000
3	0001000000
4	0000100000
5	0000010000
6	0000001000
7	0000000100
8	0000000010
9	0000000001

II. Design

Throughout our neural network design process, we have tried several design parameters and done test runs for each. We will only present our best design but will explain our design

decisions. The following subsections will discuss the other parameters that changed throughout our design process.

A. Data Collection

We used a filter cut-off frequency of 0.5 to smooth out our signals since accelerometer data tend to be noisy and hard to reproduce. Training on unfiltered data also leads to the Mean Squared Error (MSE) varying without much stability.

We collected 20 samples for each digit, giving us a dataset of 200 samples in total. We chose to collect 20 samples as an improved accuracy from the initial 10 samples per digit we took.

Each sample was collected over 700 time steps, each with a 5ms delay in between, which is around 3.5s in total. 700 time steps was chosen because it allowed us to draw big enough figures, with big enough acceleration per stoke, helping differentiate between similar digits like 0 and 6 for example. Since 700 was the number of data points sampled per axis, collecting over 2 axes (digit is drawn over a 2D plane) gives 1400, double that number of points.

We initially drew digits in the air, with smaller radius, and in a smooth fashion. The prediction results were effective only for certain digits, but it would be confusing over similar digits. Two main modifications were made to improve on prediction accuracy:

1) In order to produce better prediction accuracy, we have decided to draw digits in a more "square" fashion, making each drawing stroke very straight and defined. This helped significantly with differentiating between the digits.

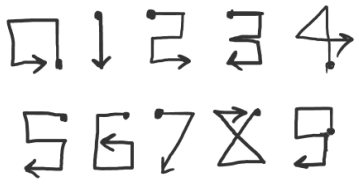

Fig. 1. Square-shaped drawing motion

2) Another modification to our methodology used for data collection was that we decided to draw figures horizontally, hover over the table. This allows us to put visual tape markings on the table in order to keep the tests reproducible in a controlled environment. This way, we could quantify and accurately measure our range of motion when testing drawing with different digit sizes. This ultimately increased our drawing motion consistency and the prediction accuracy.

B. The Neural Network Model

The following table summarizes the chosen design parameters for our Neural Network:

- From the sampling time used in our Data Collection, our Neural Network will take in 1400 inputs consisting of 700 acceleration data points in the x-axis and 700 acceleration data points in the y-axis.

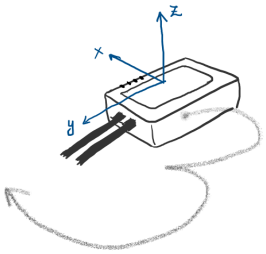

Fig. 2. Board configuration with respect to drawing motion

TABLE II
NEURAL NETWORK DESIGN PARAMETERS

Design Parameter	Value
# hidden layers	1
# nodes per hidden layer	100
# input	1400
# output	10
learning rate	0.01
validation set split	20%

- Our Neural Network is only using 1 hidden layer, since adding an extra layer introduces additional layer nodes, increasing greatly the training time. Each training iteration would need to go through two forward propagation and then two backward propagation. Even though, we would not be able to achieve great prediction accuracy with just one layer, our final design was predicting with satisfactory results meeting the performance requirements of this project.
- We chose to use only 100 hidden layer nodes to decrease the training time. When we initially used 500 nodes, each training iteration would take 40 seconds to complete, which we decided was too long for the 50-percent accuracy required for this project.
- We chose a 0.01 learning rate for the gradient descent since it needs to be $\ll 1$, but a too small learning rate would take too long for each training iteration. With 0.01, we did not observe any problematic overshooting. We were able to stop training at a relatively stable local minimum once we see that the Validation MSE starts increasing.
- We chose a validation split of 20%, giving a 80:20 ratio between the amount of data used for training and for validation. Hence, for the 20 samples per digits we collected, 16 samples will be used for training and 4 will be used for validating. This is a common ratio to choose as the validation set normally ranges between 15% and 25%.

Using these parameters with our collected data, we trained our model until we observed the Validation MSE to be consistently increasing while the training MSE is consistently decreasing. The best performing model's final training values were as follows:

The Training and Validation MSE are both within expected

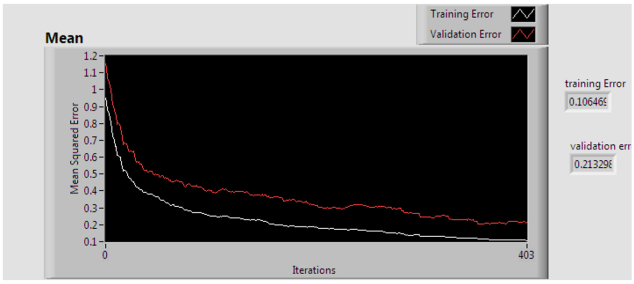

Fig. 3. Plot of training and validation mean squared error

TABLE III
BEST TRAINED MODEL TRAINING PARAMETERS

Training Parameters	Value
Training MSE	0.0106
Validation MSE	0.213
Training Time	21m20s
# Iterations	403
Time per iteration	3.2s

range of < 0.5 as defined per the project requirements (50% for at least 5 digits). The total training time is also within manageable range of around 20 minutes, letting us experiment with various tests to try different design parameters.

III. RESULTS

Using our best trained weights, we ran the inference engine to predict digits. The output array is one-hot encoded, where the index with the highest probability is the predicted digit. Our prediction success rate are recorded to be:

TABLE IV
BEST TRAINED MODEL PREDICTION PERFORMANCE

Digit	Success Rate	Satisfies Requirements
0	5/10	✓
1	0/10	✗
2	10/10	✓
3	7/10	✓
4	10/10	✓
5	5/10	✓
6	7/10	✓
7	9/10	✓
8	10/10	✓
9	9/10	✓

We can see that except the digit 1, all other digits satisfy the minimum performance requirement. We can also observe that the next least performing digits are 0 and 5.

We believe that the digit 1 does not predict well because there is too much time of inaction, where it is just sampling idle time. this makes it easy to be inconsistent in when we start drawing, when we stop drawing, and how much force to put in the stoke.

We also analyze that digits with diagonals, such as 4, 7, and 8 have the highest success rate. Diagonals means that both the x-axis and the y-axis have been recording data, instead of sampling inaction. Hence, we believe that the drawing motion should ideally contain diagonal strokes and last for the full duration of the sampling period.

A. Modifications to Improve Performance

We have done other tests with some variations and have observed the following to optimize performance:

- We should draw digits with a big range of motion
- We should draw digits with as much acceleration as possible, ideally with defined strokes of motion
- We should draw digits for the full duration of the sampling time if possible
- We should start and finish consistently at relatively the same time for each digit

These modifications should be applied for both Data Collection and Prediction stages.

IV. METHODOLOGY

A. Testing Plan

The general approach taken in our design process was to experiment with different ways to draw the digits and with trying different design parameters. We have ran the following tests before arriving at our best trained model:

1) Using 500 layer nodes: training takes 40 seconds per iteration.
   Result: We did not let it run overnight and stopped the training at around 0.4 training MSE.
2) Using 100 layer nodes: training takes 3.5 seconds per iteration.
   Result: The total training was 20 minutes, a good length for testing out different designs.
3) Using a slow and smooth drawing motion: this required a long sampling time if we wanted to draw with bigger range of motion.
   Result: Poor prediction ($< 10\%$).
4) Using a fast but smooth drawing motion: this helped us increase the magnitude of the acceleration values while reducing the sampling time.
   Result: Better prediction ($\sim 40\%$) but some digits were still confused with others.
5) Using a fast but sharp (square) drawing motion: this helps increase the magnitude of the acceleration, reduce the sampling time, while having a high motion definition for each digit.
   Result: Good prediction ($> 50\%$), similar digits were distinguished.
6) Tried to start drawing as early as possible, while sampling, and end drawing as late as possible to take full sampling duration.
   Result: Satisfactory prediction ($> 70\%$). These configurations allowed us to have our best training model.

B. Testing Procedure

For each of the tests, it was important for us to follow a consistent testing procedure to ensure reliable and reproducible test results. The following steps were taken for each test:

1) All files and folders should follow consistent naming convention.
2) Create new test folder to hold the inputs, targets, collected data samples for each digit, the trained weight matrices, and any other relevant files specific to the test.
3) Write down the design parameters used for test (type of drawing motions, #nodes in layer, learning rate, validation split, sampling time, etc.) as well the results (training time, # training iterations, Training and Validation MSE, prediction success rate, etc.).
4) The samples collected for each digit can be plotted in Excel to validate our samples, debug and analyze what could be improved, and help us replicated the motion when predicting. Here is an example of an analysis done with plotting the samples for each digit:

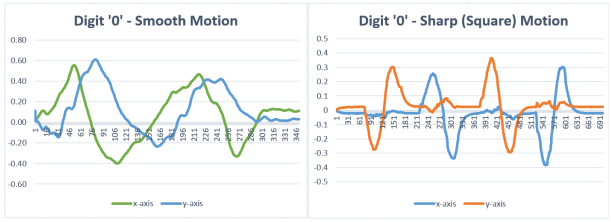

Fig. 4. Plot of digit '0' in smooth versus sharp motion

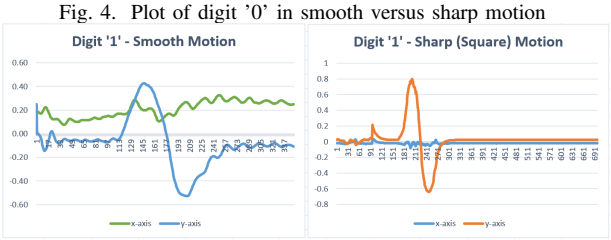

Fig. 5. Plot of digit '1' in smooth versus sharp motion

The left figures are samples drawn with a smooth motion and the right figures are samples drawn with a sharper, square motion. We can see that drawing with a sharper motion not only produces more definition in the data but also introduces more uniqueness to it. For example, the digit 0 has its x-axis peaks almost overlapping with its y-axis peaks. However, the digit 1 has its peaks on the two axes offset with each other, creating more clarity in the curves.

Plotting the sample data can also help us tweak the sample time if we see that most digits complete away before the end of the sampling. We can also see the efficiency of our filter design. Moreover, these plots allowed us to be more consistent with our drawing motion in general.

5) If trying a new drawing method, collect samples for only 5 digits to reduce the number of outputs by half, to save on time. Training on a reduced sample set while maintaining the number of samples taken per digit (20) is an efficient method to do fast prototyping while not impacting our model's prediction accuracy.
6) Since we are drawing in the air horizontally, we can delimit with visual tape markings on the table to restrict the range of motion to keep our tests reproducible, in a controlled environment.

V. Conclusion

In conclusion, our Neural Network design was able to efficiently meet the prediction accuracy requirements of minimum 50% accurate over at least 5 digits. We had all digits, except the digit 1, predicting with over 50% accuracy. Additional to its satisfactory performance, our Neural Network is efficient since the training time is only around 20 minutes, with each iteration taking about 3.2 seconds. This manageable training time facilitates experimenting with different design decisions as well as quick test runs.

The Neural Network final design consisted of only one hidden layer of 100 layer nodes, taking in 1400 inputs (700 data points per axis of motion), and outputted 10 one-hot encoded outputs. Our final model's Training Mean Squared Error was 0.106 and Validation Mean Square Error was 0.213, which are in an acceptable range. This training run used a learning step of 0.01 for the gradient descent. The model used 80% of the data for training and 20% for validation.

A. Performance and Scalability Limitations

Two major performance constraints in our Neural Network design:

1) *Limited in size of dataset.* Since we are collecting samples of digits manually, it takes approximately 3.5 seconds to collect on samples, taking 20 samples per digit takes approximately 15-20 minutes. If we were to increase the dataset size to 100 samples, it would take more than an hour of drawing in the air. Since this is a very laborious task, we are limited in our dataset size. A smaller dataset means that our model is more vulnerable to variability. For examples, our current model can only yield satisfactory ($> 70\%$) accuracy if we are consistent in the drawing style, the range of drawing motion, the drawing duration, the start and end times of the drawing, etc.
2) *Limited in the number of hidden layers and layer nodes.* Our current model has a lot of data points sampled (1400 total inputs), which limits us in having a larger amount of hidden layer nodes. We currently have 100 nodes, but could improve on accuracy with more nodes such as 500. However, that would increase exponentially our training iteration time. If we could reduce our number of inputs, we could increase the number of layer nodes or hidden layers. This can be achieved by decreasing the sampling frequency to once every 10ms instead of the current once every 5ms, which would reduce our number of inputs by half. Another solution to increasing the number of nodes would be running the training on better hardware.

Due to the above performance constraints, our model would need design changes if it were to scale to larger or more complex problems. Those problems might introduce more input data points or a much larger dataset.

REFERENCES

[1] A. Cavatassi and J. Cooperstock, "Final Project: Embedded-NIST"
Phil. McGill University, Winter 2020. [Online].
Available:
http://www.cim.mcgill.ca/~jer/courses/es/labs/project.pdf.
[Accessed April 29, 2020].