Final Project: Embedded-NIST

ECSE 421: Embedded Systems
Department of Electrical and Computer Engineering
McGill University
Version 2.1

Adam Cavatassi and Jeremy Cooperstock

Winter 2020

1 Introduction

You have previously been provided with the data set that was used to train Position-NET. For your project, you will design your own neural network for an entirely new task, recognizing handwritten digits using the myRIO board. Specifically, you will "draw" numbers in the air, and use the on-board myRIO accelerometer as a sensor (or use data collected from a group who did so). How you use the data harvested by the myRIO will be up to you. Using your designed network, you must build an inference engine that will accept a new input in the form of a hand-drawn number in the air, and output its guess ("prediction") as to which number was drawn. As per recent postings on Moodle, if you are unable to run a local instance of LabVIEW that can communicate with your myRIO board, please make use of one or more datasets that will be provided by your classmates.

2 Embedded-NIST

Embedded-NIST is inspired by the MNIST dataset [1], often used as the basis for simple neural network exercises. MNIST is a dataset of thousands of small 28×28 grayscale images of hand-written digits ranging from 0 to 9. When the images are flattened to an input vector of length 784, a vanilla neural network can be easily built to recognize which of the 10 classes to which the input image belongs.

2.1 Data Collection

You are required to collect data, consisting of sequences of accelerometer values, and the associated digits represented by each sequence, for example, as shown in Figure 1. You will then use these labelled data to train and test your neural network. Designing the training set involves determining any data processing you might want to perform, deciding how many training examples should be used for each class, and the mapping from raw accelerometer data to the input vector for your network.

2.2 Network Design

You are free to choose the parameters that govern the design of the neural network for this project, i.e., the size of the input/output vectors, the number and size of the hidden layers, the loss function, learning rate, and activation function. Optimizing a neural network can be time-consuming, and a modular implementation, which you have hopefully achieved through the previous labs in this course, will likely help speed up your design process, allowing you to investigate the effectiveness of various network configurations more quickly.

2.3 External Code

For file manipulation and data shaping needed to build your training set, e.g., to clean or trim your raw data, you are permitted to use an external software environment such as MATLAB, Python, or Excel. However, any operations required for the real-time inference engine must be performed entierly within LabVIEW. For example, assuming you have a trained network, you are *not* allowed to input a new data sample e.g., drawing a 3, then stop

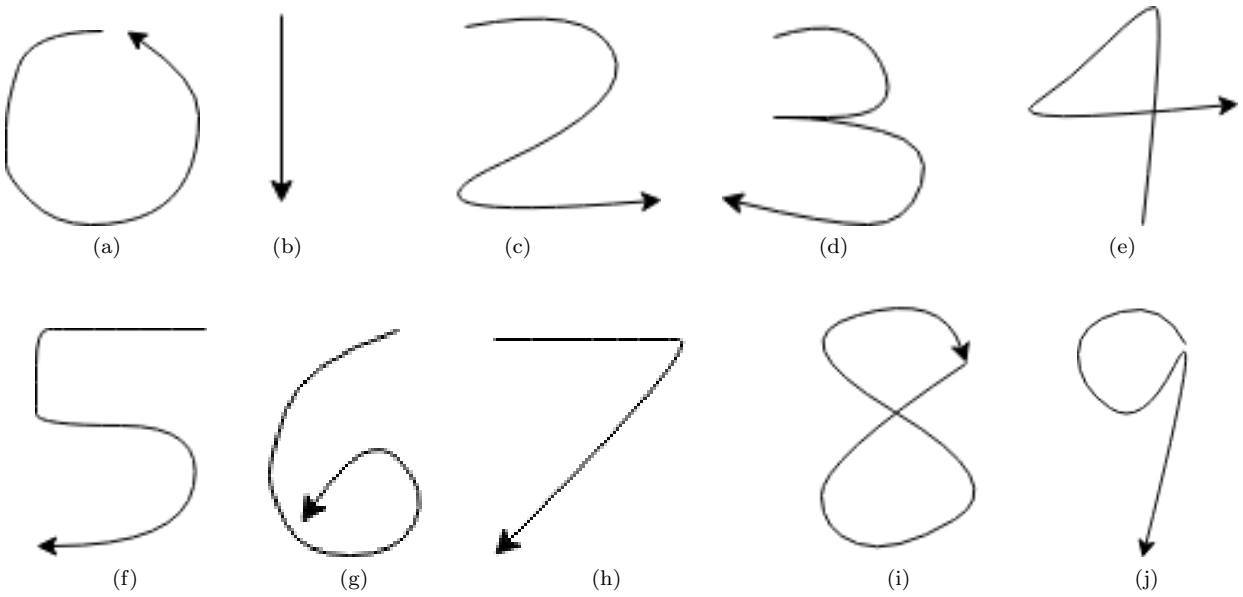

Figure 1: Examples of hand-drawn numbers on a 2-dimensional plane.

your LabVIEW code, carry out some computation with Python, and use those results to resume your LabVIEW code. For those running LabVIEW using pre-recorded data, e.g., from your peers, you should read the raw sensor values from the CSV file in lieu of the accelerometer. All other specifications here apply *mutatis mutandis*.

3 Submission requirements

Much like in previous labs, you must submit a .zip file containing your project file, all VI files that are part of your project, and screenshots of both the final block diagram and final front panel for all VIs. You must also submit the code used for any scripts used to manipulate training data. The front panel of your training VI should include a plot of training and validation set errors over time, as well as a data collection interface. You will also be required to submit a URL to a private YouTube video with a maximum length of 300 seconds that demonstrates the full functionality of the network training, including an error plot followed by the inference engine that you trained in real-time. New: This video will serve in lieu of the originally planned in-class demonstration of the performance of your system. If you saved a set of weights that perform well, you may use those to demonstrate the inference engine. However, you must *also* demonstrate that your system is able to learn the weights, as was stipulated for Lab 4. You should demonstrate the best possible inference engine you are able to achieve, convincing the viewer that your your weights were synthesized from your code and your own data. Your network outputs can be determined by you, as long as they are very clear.

Show that each hand-drawn digit is clearly recognized by the neural network after training. You are also required to submit a formal report in the IEEE conference format. The report should outline the methodology used to build your data set and neural network, and should disclose all design parameters explored as well as results from the project such as training loss and computation time. The report should evaluate the effectiveness of your approach.

4 Assessment Criteria

Criterion	Unsatisfactory	Bare minimum	Satisfactory	Good	Excellent
A screen capture of all block diagrams and front panels are present.	no screen capture of any block diagrams or front panels	only one screen capture is present		multiple screen captures are present, but submission is missing screen captures of some sub-VIs used in the project	all screen captures are present depicting all sub-VIs used in the project
A video with maximum length of 300 seconds demonstrating the functionality and performance of your Embedded-NIST digit recognizer	A URL to a video of the lab submission in operation was not provided, or the video does not demonstrate the Embedded-NIST digit recognizer being trained or tested.	The video offers a minimal demonstration of the training and testing procedures and operation of the recognizer.	The video offers a reasonable demonstration of the training and testing procedures and operation of the recognizer.	The video also includes some discussion of a challenge encountered and how this challenge was tackled or resolved by the team.	The video also includes some discussion of system performance and the implications to scalability to larger problems.
A .zip file is provided containing all project and VI files. The code compiles and runs as described in the README.	no .zip file is provided	.zip file is provided, but is missing files required to peer-assess the submission, or used an incompatible version of LabVIEW	LabVIEW files are provided, but code does not compile	LabVIEW code compiles, but instructions for training or testing the network are not provided or are otherwise unintelligible	LabVIEW code compiles, instructions are clear and allow for effective assessment of project.
The submission exhibits an effective Embedded-NIST digit recognizer.	Recognition performance not demonstrated.	Testing procedure results are provided, but unclear how testing was carried out.		Testing procedure makes it clear that the test data was kept separate from the training data, and test results indicate plausible recognition performance (better than 50% correct recognition on samples of at least 3 of the 10 digits).	Testing procedure makes it clear that the test data was kept separate from the training data, and test results indicate reasonable recognition performance (better than 50% correct recognition on samples of at least 5 of the 10 digits).

References

[1] URL: `http://yann.lecun.com/exdb/mnist/`.