# 18-100: Intro to Electrical and Computer Engineering

# LAB9: ML Lab

## Writeup Due: Thursday, December 1st, 2022 at 10 PM

Name: _____

Andrew ID: _____

### How to submit labs:

Download from this file from *Canvas* and edit it with whatever PDF editor you're most comfortable with. Some recommendations from other students and courses that use Gradescope include:

| | |
|---|---|
| **DocHub** | An online PDF annotator that works on desktop and mobile platforms. |
| **pdfescape.com** | A web-based PDF editor that works on most, if not all, devices. |
| **iAnnotate** | A cross-platform editor for mobile devices (iOS/Android). |

*If you have difficulties inserting your image into the PDF, simply append them as an extra page to the END of your lab packet and mark the given box.* ***Do NOT insert between pages.***

If you'd prefer not to edit a PDF, you can print the document, write your answers in neatly and scan it as a PDF. *(Note: We do not recommend this as unreadable lab reports will not be graded!).* Once you've completed the lab, upload and submit it to *Gradescope*.

Note that while you may work with other students on completing the lab, this writeup is to be completed alone. Do not exchange or copy measurements, plots, code, calculations, or answer in the lab writeup.

### Your lab grade will consist of two components:

1. Answers to all lab questions in your lab handout. The questions consist of measurements taken during the lab activities, calculations on those measurements and questions on the lab material.
2. A demonstration of your working lab circuits and conceptual understanding of the material. These demos are scheduled on an individual basis with your group TA.

| Question: | 1 | 2 | 3 | Total |
|---|---|---|---|---|
| Points: | 6 | 16 | 8 | 30 |
| Score: | | | | |

# Lab Outline

This lab introduces you to Google's co-lab and TensorFlow. The goal is for you to learn about machine learning and neural networks as well as to learn how to use these powerful, free tools for your own projects.

# Sections

1. Reviewing the Source Code
2. Training and Evaluating your Model
3. Putting your Model on the Web

# Required Materials

- 1x Computer w/ Web Browser

# Setting Up the Environment

This lab uses the "Google Co-Lab" online environment to run code and train a neural network. For an introduction, please watch the following video: https://www.youtube.com/watch?v=vVe648dJOdI. [1] We recommend using Google Chrome as your browser, as Safari and Edge sometimes have issues with Co-Lab.

**Option A:**

1. Login to Google Drive (https://drive.google.com) using your Andrew ID/account and add Google Co-Lab as instructed in the video
2. While logged into Google using your AndrewID/account, download the file `handwritingv9.ipynb` from Canvas and upload it to your own Google Drive
3. Open the file by double clicking on it in Google Drive. It should open in the Co-Lab online editor.

**Option B:**

1. Download the file `handwritingv9.ipynb` from Canvas to your local machine
2. Go to https://colab.research.google.com/ and sign in with a Google account, e.g. your AndrewID/account.
3. Once logged in, go to "Upload" on the far right tab and select the `handwritingv9.ipynb` file and upload it.
4. It might take a minute or few to load so please be patient, but eventually you will be taken to the Co-Lab page and the file will be uploaded to your drive.

In the upper right, to the left of "Editing" and beneath "Comment", you'll see a "Ram/Disk" pull down. Pull it down and select "Connect to hosted runtime".

**Run the code.** You can either do it all at once, or step-by-step.

- To do it all at once, click on the "Runtime" menu and select "Run all". It'll take a while. You'll know it is done when a `handwriting.zip` file is downloaded. This file contains the model you just trained.
- To do it step-by-step, notice the numbered code blocks in the file. If you hover your mouse over a code block number, e.g. [71], it will turn into a play icon that, when pressed, will execute the corresponding code block. Walk down the file, executing each code block and waiting for it to complete, each one in turn.

[1] Python 2 is no longer supported, so the choice between Python 2 and Python 3 is no longer available.

# 1. Reviewing the Source Code

The source code is written in the *Python programming language*. It contains lines beginning with the # character. These lines are *comments*; they explain, in English, what the code does. This lab does not require that you write code. It only requires that, with the help of these comments, you understand the code that is given and can make very small changes to it, as described later in this lab. Please review the code, especially the comments, and answer the following questions.

1 pts **1.1** What is "one hot" encoding, and why is it used?

1 pts **1.2** The code includes this line. In English, what does it do, and why is it needed?

```
model.add(Flatten())
```

1 pts **1.3** The code includes the following lines. In English, what do they do, and why are they needed?

```
for dense_layer in range (HIDDEN_LAYERS):
model.add(Dense(HIDDEN_LAYER_SIZE, activation='relu'))
```

1 pts **1.4** The code includes the following line. What is achieved by using the "softmax" activation function in the output layer?

```
model.add(Dense(number_of_classes, activation='softmax'))
```

1 pts **1.5** What is accomplished by "compiling" the neural network model?

1 pts **1.6** What is accomplished by "fitting" a neural network to training data?

## 2. Training and Evaluating Your Model

Verifying that the neural network and training parameters are set as below:

```
[1045] # These parameters control different aspects of the neural network and its training

       # These are the parameters you most likely want to explore
       TRAIN_SIZE=20                # This is the total pool of examples presentable to the network for training
       BATCH_SIZE=5                 # This is the total number of examples the network should look at before making adjustments
       EPOCHS=30                    # This is the number of epochs, i.e. batches of the training set

       HIDDEN_LAYERS=1              # The number of densely hidden layers within the neural network
       HIDDEN_LAYER_SIZE=64         # The number of neurons in each dense layer. For simplicity, each dense layer is set up to be the same

       LEARNING_RATE_COEFF=1.30     # Control the rate at which adjustments to weights are made: <1 slows, 1 is default, and >1 exaggerates

       # Feel free to change this, if you'd like, but doing so may not be particularly illuminating for this lab
       TEST_SIZE=100                # This is the number of never-seen-before images used to evaluate the performance after training is complete. Do not lower this number to improve your results.
```

Figure 1: NN Training Parameters

**Hit "Runtime → Run all".**

2 pts **2.1** Get a screen capture of TensorBoard's output, adjusted to see the accuracy and loss lines, without smoothing, for both training and validation data. (*Remember to set smoothing to 0.0 and hit the*  *button, if needed, to set the scale to capture both lines.*) You may need to wait a minute or two for TensorBoard to process the data and show the plots.

☐ I have appended the screenshot to the back of my lab writeup

1 pts **2.2** Look at the plots. What do you notice about the relationship between the `epoch_accuracy` of the training data and the validation data? (*Hint: Recall that accuracy is the rate at which the neural network's strongest output is the correct output.*)

1 pts **2.3** What do you notice about the shapes of the `epoch_loss` curves? (*Hint: Recall that the loss is a measure of how far each output is from the 1 or 0 value it is would ideally exhibit.*)

1 pts **2.4** Why do you think the validation curve behaves as it does? In other words, why does it actually get worse with more training?

1 pts **2.5** Why is the effect you saw in the loss curves not apparent in the accuracy curves?

Change the code and reduce the learning rate to 1.0. Re-run the experiment. It is suggested that you "Runtime → Run all" again.

2 pts **2.6** Paste a copy of the TensorBoard plot below. Show both the new and old runs without smoothing and with proper scaling.

☐ I have appended the screenshot to the back of my lab writeup

1 pts **2.7** What do you notice changed most with a reduced learning rate? Why do you think this happened? Was there any benefit to a higher learning rate?

2 bonus

**2.8** Modify the learning rate schedule function to tune the learning rate to achieve the benefit(s) you listed for (10) above while avoiding the problem(s) you noted in (8) above. To do this, you might consider, for example, during which epoch(s) the benefit(s) of a higher learning rate began to diminish and during which epoch(s) the problems began to arise. To get credit, please paste your modified code and a new TensorBoard plot showing the improvement below.

*Modified Code*

*TensorBoard Plot*

☐ I have appended the screenshot to the back of my lab writeup

1 pts

**2.9** Why do you think the training set continues to outperform the validation set so greatly?

1 pts **2.10** Change *exactly one* of the NN training parameters shown in Figure 1 to reduce validation loss below 0.33 or to achieve an accuracy of $> 90\%$. What one parameter did you change, what did you change it to, and why?

**Hint 1:** Consider your answer above.

**Hint 2:** Try making big changes, e.g. doubling or halving values and testing.

**Hint 3:** Try making only one change at a time and evaluating it.

**Hint 4:** If something works, try doing it, e.g. doubling it or halving it, again

**Hint 5:** Remember, you only need to, and should only, change one variable, not multiple.

2 pts **2.11** Paste a copy of your new TensorBoard plot below. Please uncheck or delete runs as needed to plot only one pair of training and validation curves that meet the criteria above:

☐ I have appended the screenshot to the back of my lab writeup

1 pts **2.12** Set the training set size to 5000 and the number of epochs to 20. This should be plenty to get pretty good results from the presently defined architecture. You can take our word for it, or experiment to find your own personal best. Please give each of the range of validation losses and the range of the validation accuracies shown in the last 3 epochs.

1 pts **2.13** Change the number of hidden layers to 2, the size of each hidden layer to 128. Please give each of the range of validation losses and the range of the validation accuracies shown in the last 3 epochs and explain how they compare to your result above?

1 pts **2.14** What do you think accounts for the change you observed?

1 bonus **2.15** At this point, feel free to work on your personal best. If you'd like, expand the validation set size to get a better and more consistent metric. If you've improved upon the performance above, paste your metrics, as above, and a screenshot of TensorBoard plots, as were done earlier, below. If not, just check "N/A" below.

☐ I have appended the screenshot to the back of my lab writeup
☐ N/A

**Make sure to save and download your best model, even if it's not performing too well. You'll need this model for the next part of the lab. To do this, you'll need to uncomment and run the necessary code segments to download the `handwriting.zip` file.**

## 3. Putting Your Model On the Web

You previously downloaded a model file, `handwriting.zip`. If you downloaded multiple copies, please be careful to identify and use the model trained as best as you were able to train it, likely your most recent download. Unzip the model .zip file, leaving you with a directory tree. Important: The .zip archive contains a directory/folder named model. Please be sure it is unzipped into a location where it can do this, or it could overwrite files in an unrelated "model" directory, should one already exist in your file system.

In order to use your model via the Web, you need your own Web space. One easy way to achieve this is with your Andrew File System (AFS) UserWeb page. See here for more information: https://www.cmu.edu/computing/services/comm-collab/collaboration/afs/.

For example, one of your instructors created the following Web space https://www.andrew.cmu.edu/user/tzajdel/. The model uploaded was not trained particularly well, but you can play around with it by visiting `handwriting.html`. You can also easily upload your own webpage with your trained model by using a series of commands in a terminal window.

Whichever OS you are using, you need to open a command line interface to upload your files to the AFS. On a Windows machine, open a PowerShell window in the directory where your html files and model folder are stored on your computer. Hold Shift while Right Clicking, then select "Open PowerShell window here." On a MacOS machine, navigate to the folder containing your web html files and model folder, then click Finder > Services > New Terminal at Folder. If you are using a Linux machine, you need to open a Terminal window in the appropriate folder (you probably aren't GNU to this, Linux User).

Then, in your command line window on your system of choice, type the following command:
`scp -r * andrewid@unix.andrew.cmu.edu:~/www/`

Replace `andrewid` with your AndrewID. Enter your AndrewID password when prompted and press Enter. If successful, you will see a list of files and folders uploaded to your `www` folder on AFS. This is the folder that is publicly accessible on the web. (Command line aside for those interested: To verify that these files exist, you can Secure Shell into your AFS from your command line window: `ssh andrewid@unix.andrew.cmu.edu` and navigate around your file system. `ls` will list all the folders in a directory. `cd foldername` will navigate to that folder, and `cd ..` will go back one folder level.)

When your webpage is uploaded to your AFS `www` folder, you then need to publish it. Open a web browser and go to https://andrew.cmu.edu/kweb/publish/. Enter your AndrewID and then click Publish! Once you've done that, your files should be publicly available on https://www.andrew.cmu.edu/user/andrewid/ where `andrewid` is of course your AndrewID. If all goes well, you should see these files in your web space.

Go to the `handwriting.html` page within your web space and play with the applet a bit.

3 pts **3.1** Draw a single digit number in the black box, click the "Predict" button to see how your neural network identifies it, hit the "Clear" button, and repeat a few times. The first time you use the page, it might take a few seconds to a minute to load before the "Predict" button will work. Paste a screenshot of the output below:

☐ I have appended the screenshot to the back of my lab writeup

1 pts **3.2** Draw a smiley face, a train, plane, automobile, or whatever. Anything but a digit. Hit predict. Paste a screenshot of the result below:

☐ I have appended the screenshot to the back of my lab writeup

1 pts **3.3** How did the neural network react to this non-digit input? Why?

2 pts **3.4** How could you improve the performance of your network in this respect?

1 pts **3.5** Consider your answer to the two questions above and think about the use of Machine Learning (ML), such as trained neural networks, in systems which interact with humans and affect the human experience. What might be an important lesson to learn about training real-world systems to interact with people? If it helps, consider the population of people used to train neural networks and the much larger population of people in the world.