MSBA –Optimization 2
Project 1 – Image Classification

The goal of this assignment is to create a neural network that has greater than 99% accuracy on the MNIST dataset. To do this you will try various network structures. Crucially, you will fit a convolutional neural network and a transformer neural network. Your job is to find the best architecture you can for each of these and compare the 2 structures. To present this information, you will build an interactive webpage. It is important to be able to write code that does machine learning that is accessible to more than just you!

Cross-validation is the typical preferred method to fit hyper-parameters (network structure), but this is difficult for neural networks because they are typically slow to train. Therefore, we will just use a validation set approach for this assignment. Fortunately, TensorFlow will do this for you automatically; set validation split = 0.2 when you train the model.

For the CNN, you can use multiple convolutional layers, different filter sizes, different number of filters, multiple max pool layers, several dense layers, regularizers, dropout layers, different batch sizes or number of epochs; whatever gets you greater than 99% accuracy on the validation set.

For the transformer, you can use different hidden, key/query, and value dimensions, different number of heads for the MHSA layers, different number of layers, …

If you can get 99%, try to get 99.5% also. Once you find two network structures that give you greater than 99% accuracy on the validation set, go back and train those networks on the entire training set. How does this do on the MNIST test set? Please give a detailed answer to this question.

Plot a few of the numbers you misclassify from the test set. What are the common mix-ups of numbers in your network? Why do you think that is? Do you think it would ever be possible to get 100% accuracy?

You will most likely be better off doing this assignment on Colab rather than on your own machine. The network I trained that gets 99% accuracy took several hours to train on my laptop but took just a few minutes on Colab.

A general tip about convolution: it usually works a bit better to have many small filters than having few large filters. Also, multiple layers of convolution/maxpooling is usually helpful. But be careful that you don't shrink the data too much. Look at the model summary, at the size of the output of the flatten layer, before you use your first dense layer. If there aren't many nodes from your output layer that means you did too much

convolution/maxpooling and you removed a lot of information in the image, and you probably won't classify very accurately.

Web Page – Anvil

The traditional tools to build webpages using python are django and flask. Both require knowledge of html. A new alternative has emerged recently called anvil: https://anvil.works . Anvil is an online service that allows you to easily build webpages by dragging and dropping components onto a page, like a wordpress webpage. But then the components can rely on python code. I have no interest in anvil as a company, I just think it's a very useful tool.

Your webpage should describe your process, your network, what worked well and what didn't work well. And it should also allow a visitor to upload a csv file with the pixel intensities of a 28x28 image and then your webpage should display the image and call your tensorflow model to classify the image. Think of this as an interactive blog post.

You should NOT use anvil to fit your network, just to evaluate an uploaded image. Anvil is free to use but the free version is limited: you can't use numpy, pandas or other similar packages. There is a workaround for this that I will explain below.

The user community for Anvil is large and the forums are actively monitored by Anvil employees to answer questions. If you have a question of how to do something, you'll likely be able to find the answer on their page.

Anvil has many blog posts that teach how to use their tool. Here are a few that may be helpful.

1. https://anvil.works/learn/examples/meter-feeder
2. https://anvil.works/blog/plotting-in-matplotlib
3. https://anvil.works/docs/media
4. https://anvil.works/learn/tutorials/jupyter-notebook-to-web-app
5. https://anvil.works/forum/t/how-to-add-multiple-pages-within-an-app/74/3
6. https://anvil.works/docs/client/python/alerts-and-notifications
7. https://anvil.works/docs/deployment/quickstart
8. https://anvil.works/docs/users

Accessing Python Packages with Anvil

Many python packages are unavailable to Anvil, such as numpy, pandas, and tensorflow. To build a webpage that can use those packages there is a workaround.

Anvil can call python functions that live on other computers. For example, you can write a jupyter notebook on your computer that has some functions in it and tell anvil to access those functions. Then when someone uses your webpage, if that function needs to be called, it will be executed on your machine. This is the anvil uplink feature. Tutorial 4 above uses this feature.

The uplink feature requires your computer to be turned on and running your jupyter notebook when the page is called. This is burdensome. Another work around (which you will need to do for the project) is to host your python code on an AWS (or similar cloud service) virtual machine. The simplest way to do this, that I know of, is through AWS Lightsail. Go to aws.amazon.com and sign up for an account. You must sign up with a credit card, but you'll be able to launch a machine for free that won't charge your card.

There are MANY different types of AWS virtual machines. The simplest one is called lightsail. Lightsail simply aggregates a few AWS services (EC2, S3, …) into a user-friendly virtual machine.

Once you create and login to your AWS account, in the top search bar search for lightsail and click the link. Under the instances tab, click the button to create an instance. Select Linux/Unix as the platform and select the Django app for the blueprint (this will just make sure python is installed on your instance). Then go down and pick whichever rate plan is currently free. As of me writing this, there are 2 plans that have 3 free months. Just remember to delete this instance when the semester is over to avoid being charged after the 3 months.

Once the instance is created (this may take 5-10 minutes of waiting for AWS to create the machine), go back to the lightsail page and on the instance tab there will be an icon of >_. Click that and it will open a new window that is the terminal for your machine. You can pip install any packages you want there, just like any other computer.

You need to be able to transfer files (like your code and your tensorflow model) into your machine too; you do that using an ftp app called filezilla. Instructions can be found here: https://lightsail.aws.amazon.com/ls/docs/en_us/articles/amazon-lightsail-connecting-to-linux-unix-instance-using-sftp

You'll need to write your python code in a .py file, instead of a jupyter notebook. At the bottom of your .py file be sure to include anvil.server.wait_forever(). This will make sure your functions stay available to be called by your webpage. Once you're ready to run your python, go to the virtual terminal, cd to where the code is located and run:

nohup python filename.py &

nohup is short for no hang up. This means when you log out of your AWS account your script will keep running. When you run the above line, the terminal will output a

number; this is the process ID for the python script that is running. Be sure to take note of that number somewhere and don't lose it! It is important that you include the & at the end, or the process ID won't be printed to the screen. To make your python script stop running later, in another session of the virtual terminal you can run:

kill id_number

where id_number is the number you wrote down above.

Your anvil webpage should:

1. Require a user to sign in with email and password
    a. Create an account for me with email: dan and password: Optimization1234
        i. I know dan is not an email address, but that's actually ok if you create the account for me!
    b. Do not allow users to create new accounts automatically.
    c. The rest of the page should only be accessible if a user is logged in.
2. Have a detailed description of how you built your NNs, some pictures of well classified vs ill classified images, and analysis of why some images are classified poorly. Think of this as a medium article you would write about the mnist dataset and your network.
3. Allow the user to upload a csv file
    a. The csv file should have 28 rows and 28 columns, with no header row. Each entry may be the 0-255 pixel intensity, or the scaled 0-1 pixel intensity; your code should figure out which one!
    b. If the file uploaded is not a csv or if it is not in the correct format the webpage should alert the user that the file is not correct.
4. If the file works, the upload button, and any description of what to upload, should disappear.
5. Classify the image using both NNs and display it.
6. Have a button that allows the user to start over: upload a new file and classify another image.
7. Publish the app at the following url: MSBAoptim2-{YOUR group number}.anvil.app
8. All your backend code that needs to use python packages should be hosted on a cloud service, like AWS or GCP or … Please submit a screengrab of your cloud service running your code. We will not announce when we will grade your assignment, so it's important that you cloud server is up and running for a few weeks.
9. Make your site look nice, 50% of your grade will come from the appearance of your site.
10. Include a link to clone your anvil code in your submission.
11. The code you are running on your cloud backend should be submitted along with your links to the public site and your anvil clone.