**CS506 Programming for Computing**

**HOP08 Deep Learning – Image Classification**

11/22/2020 Developed by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Use Machine Learning to classify images.
* Process, build, train datasets to make and verify predictions.

**Resources**

* <https://www.tensorflow.org/>
* Linkedin Learning

**Preparation**

***[NOTE: your environment might look different from the images below]***

1. In Visual Studio Code, open the private repository generated when you accepted the HOP08 assignment (If you cannot find that repository in your machine, you might have not cloned the repo, if so, please do before proceeding).
2. Open the terminal in VSCode by hitting the Ctrl + ~ key. Type the following to install tensorflow:

pip3 install tensorflow

Text

Description automatically generated

Once the installation is successful, you should see similar message:



Type the following to install matplotlib:

pip3 install matplotlib

Text

Description automatically generated

Once the installation is successful, you should see similar message:



1. Open Jupyter Notebook, under Module 8 folder, create a new file called Image\_Manipulation.ipynb and simply click on the file to open notebook.
2. Type the following into the Image\_Manipulation.ipynb file, to import tensorflow and matplotlib we just installed.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

**Import data set**

In this HOP, we will use the FASHION\_MNIST dataset. You can see the source code here: <https://github.com/zalandoresearch/fashion-mnist> , feel free to experiment the repo before continuing.

Under the same file, in a new block, type the following:

Text

Description automatically generated

You should see similar result when running the above block:

Text

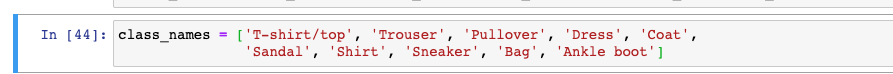
Description automatically generated

Each training and test example is assigned to one of the following labels:

Table

Description automatically generated

Since the class names are not included with the dataset, type the following to store them:



Now that we have the data set imported, we can experiment to understand our data better. Type the following and run each block to see the result:

Graphical user interface, text, application, email

Description automatically generated

When we call train\_images.shape, we got the result of (60000, 28, 28), which means there are 60000 images in the test data set, each image is 28x28 pixels.

len(test\_labels) gave us 10000, which means, there are 10000 labels in the test data set. The same logic is applied to the rest of the code in the above screenshot.

Now, for visualization, let’s print out some images so we have better idea what we are working with:

Graphical user interface, text, application, email

Description automatically generated

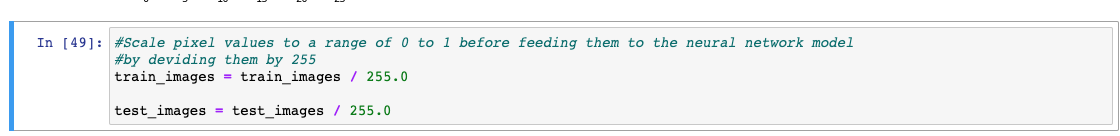
Scroll through the result to see different images.

Now, let’s explore the detail of a single image:

Graphical user interface

Description automatically generated

As you can see, each image has pretty high pixel values right now, before training our data, let’s rescale them, to produce a more accurate result and reduce the computational power required *(the bigger the image, the more pixels the computer needs to work on).* Type the following into a new block:



After rescaling the images, let’s print the first 25 images to test if everything goes as expected:

A picture containing diagram

Description automatically generated

**Build the model**

Building the neural network requires configuring the layers of the model, then compiling the model.

Layers in neural networks:

When you look at any pictures in the computer, they are actually made up from 3 intensity of color: Red, Green and Blue:

Chart, shape, bubble chart

Description automatically generated

Each pixel in the image is ranged between 0 to 255, the numbers represent how intense the color would be., for example:

A close up of a window

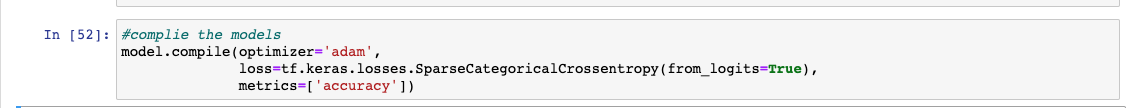
Description automatically generated

Thus, in simple words, since digital images are made up from a combination of different color laying on top of each other, with different intensity, we need to set the layers of our data before training them.

Type the following to set up the layers:

A picture containing text

Description automatically generated

Type the following into a new block to compile the model: 

You can find detailed explanation of each parameter here: <https://www.tensorflow.org/api_docs/python/tf/keras/Model/#compile>

Type the following into a new block to train the model:

A picture containing text

Description automatically generated

Pay attention to the result, where the details of data loss and data accuracy are shown to us.

Next, let’s evaluate the accuracy and loss of the model in the test model. Type the following into a new block:

Graphical user interface, text, application

Description automatically generated

**Make Prediction**

Let’s see what we have for our test data sets by typing the following code:

A picture containing diagram

Description automatically generated

Now, let’s use the trained models, to make predictions, type the following into new blocks:

Graphical user interface, text, application, email

Description automatically generated

predictions[0] is the prediction, in the format of an array of 10 numbers. They represent the model's "confidence" that the image corresponds to each of the 10 different articles of clothing. np.argmax(predictions[0]) shows us the label that has the highest “confidence” score, in this case, the result was 9. Recall the table shown in the beginning of this HOP, 9 represents ankle boots:

Table

Description automatically generated

We know that the 0th test image is an ankle boot, let’s test if the prediction can recognize the 0th test image has label number 9 (ankle boot). If we have 9 as the result, that means the prediction is correct. Type the following:

Graphical user interface, text, application

Description automatically generated

Let’s move on to graph all 10 classes. Type the following into a new block: Graphical user interface, text, application, email

Description automatically generated

**Verify Prediction**

Type the following code to a new block:

Diagram

Description automatically generated

**Push your work to GitHub**

Open the terminal from the VSCode by hitting the “control” + “~” key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 8 – Your Name”

>>> git push origin master