**CS506 Programming for Computing**

**HOS07A– Deep Learning – Image Classification**

4/10/2025 Reviewed by Clark Ngo

05/14/2025 Reviewed by Ashwin Paturi

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* **Screenshots may be different from your environment.**
* The directory path shown in screenshots may be different from yours.
* There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
* Some steps may not be explained in detail. If you are not sure what to do:

1. Consult the resources from the course.
2. If you cannot solve the problem after a few tries (usually 15 -30 minutes), ask a TA for help.

**Learning Outcomes**

Students will be able to:

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

**Resources**

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

**Section 1: Preparation:** Get started with your virtual environment here: <https://cityuseattle.github.io/docs/git/github_codepsace/#codespaces>

1. Create a codespace on your central repository, similar to the screenshot below, except for the course name.

**A screenshot of a computer

Description automatically generated**

1. In your terminal, type the following to install TensorFlow: pip install tensorflow

A screenshot of a computer program

Description automatically generated

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



1. Type the following to install matplotlib: pip install matplotlib

A screen shot of a computer

Description automatically generated

1. Once the installation is successful, you should see a similar message



1. Open the Jupyter Notebook. Under the Module07 folder, create a new file named Image\_Manipulation.ipynb, and click on the file to open the notebook.
2. Type the following into the Image\_Manipulation.ipynb file to import the TensorFlow and Matplotlib we installed.

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text

Description automatically generated

**Section 2: Import the dataset**

* 1. In this HOS, we will use the Fashion\_MNIST dataset. The source code can be found here: <https://github.com/zalandoresearch/fashion-mnist>. Feel free to experiment with the repo before continuing.
  2. Under the same file, in a new block, type the following:

Text

Description automatically generated

* 1. You should see a similar result when running the above block:

Text

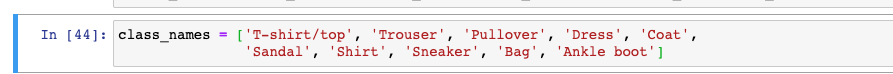
Description automatically generated

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

Table

Description automatically generated

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



* 1. Now that we have the dataset 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

* 1. When we call train\_images.shape, we get the result of (60000, 28, 28), meaning there are 60000 images in the test data set, each 28x28 pixels.
  2. len(test\_labels) gave us 10000, meaning there are 10000 labels in the test dataset. The same logic is applied to the rest of the code in the above screenshot.
  3. Now, for visualization, let’s print out some images so we have a better idea of what we are working with:

Graphical user interface, text, application, email

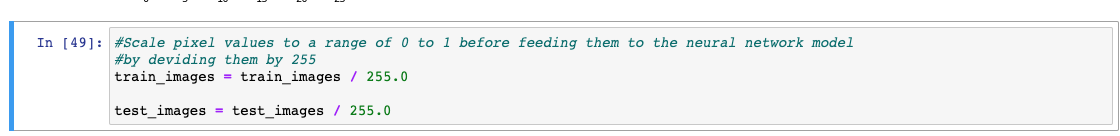
Description automatically generated

* 1. Scroll through the result to see different images.
  2. Now, let’s explore the details of a single image:

Graphical user interface

Description automatically generated

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



* 1. 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

**Section 3: Build the model**

* 1. Building the neural network requires configuring the model's layers and compiling the model.
  2. Layers in neural networks: When you look at any pictures on the computer, they are made up of 3 intensities of color: Red, Green, and Blue:

Chart, shape, bubble chart

Description automatically generated

* 1. Each pixel in the image ranges between 0 and 255; the numbers represent how intense the color would be. For example:

A close up of a window

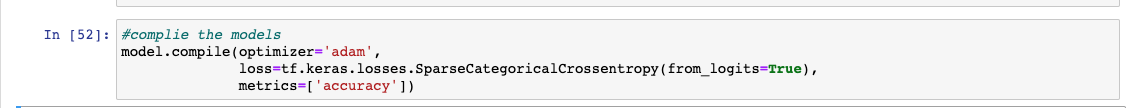
Description automatically generated

* 1. Thus, in simple words, since digital images are made up of a combination of different colors layered on top of each other with different intensities, we need to set the layers of our data before training them.
  2. Type the following to set up the layers:

A picture containing text

Description automatically generated

* 1. Type the following into a new block to compile the model:



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

* 1. Type the following into a new block to train the model:

A picture containing text

Description automatically generated

* 1. Pay attention to the result, showing us the details of data loss and accuracy.
  2. 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

**Section 3: Make Prediction**

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

A screenshot of a computer code

Description automatically generated

A picture containing diagram

Description automatically generated

* 1. 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

* 1. 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 ten different articles of clothing.
  2. 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 at the beginning of this HOS, 9 represents ankle boots:

Table

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

* 1. 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

**Section 4: Verify Prediction**

* 1. Type the following code into a new block:

Diagram

Description automatically generated

Save your Jupyter Notebooks with all Outputs.

**Push your work to GitHub Codespaces**

Follow instructions here: <https://cityuseattle.github.io/docs/git/codespaces_submission/>