**CS506 Programming for Computing**

**HOS08A – Computer Vision – Image Classification with CNN**

5/19/2025 Reviewed by Clark Ngo

05/22/2025 Reviewed by Naveena Moddu

05/22/2025 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.
* Create Convolutional Neural Networks

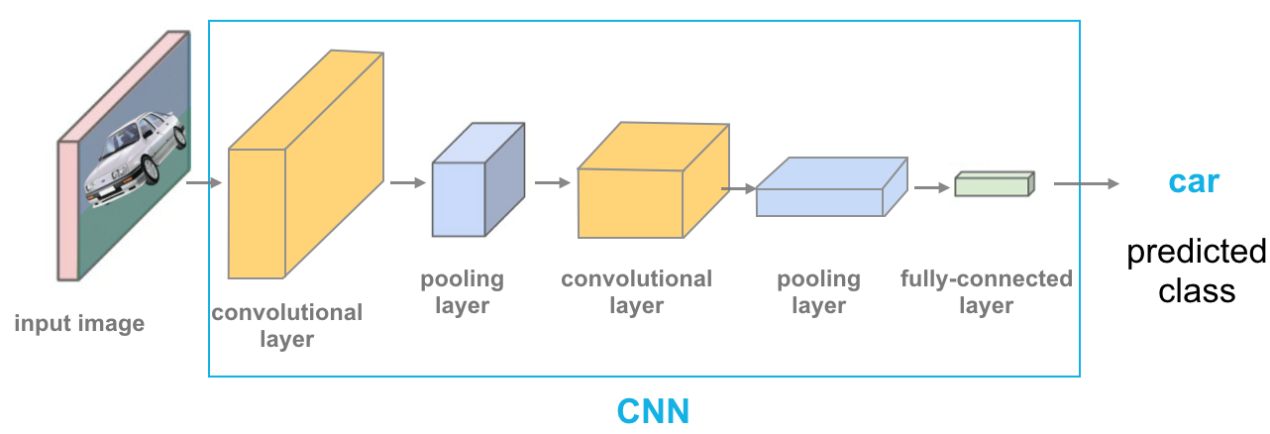
**Resources**

* Kaggle.com
* <https://randerson112358.medium.com/>

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

**Section 2: Image Classification using Keras and scikit-image**

* 1. Last week, we learned how to
     1. Use Machine Learning to classify images.
     2. Process, build, and train datasets to make and verify predictions.
  2. This week, we will continue to practice image classification, create a Convolutional Neural Network, and see how powerful Python and TensorFlow can manipulate images.



* 1. Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects, and traffic signs, apart from powering vision in robots and self-driving cars
  2. This week, we will work on the CIFAR-10 dataset. Please have a look at the dataset to know what we will be working on the [CIFAR-10 and CIFAR-100 datasets](https://www.cs.toronto.edu/~kriz/cifar.html)

A picture containing text, screenshot

Description automatically generated

* 1. This HOS will need the Keras and scikit-image libraries. Follow the instructions below to install them; skip this section if you already have them installed.
  2. Open the Terminal and type the following command to install Keras

**>> pip3 install keras**



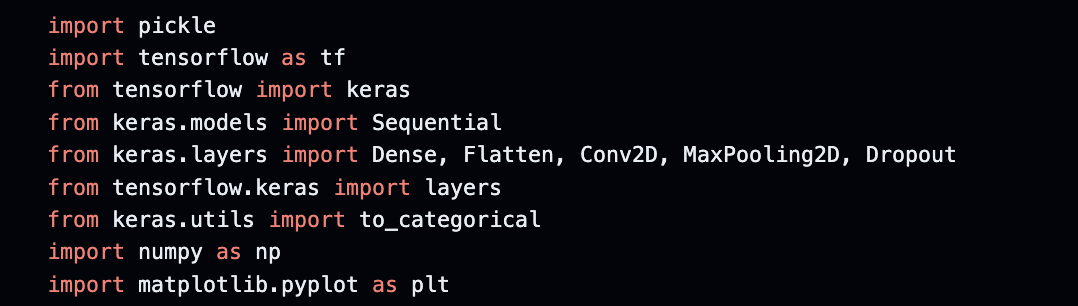
* 1. Then, type the following to install scikit-image:

**>> pip3 install scikit-image**



**Section 2: IMPORT DATASET**

* 1. Create a new file called Image\_Classification.ipynb, and click on the file to open the notebook. ￼
  2. Type the following into the Image\_Classification.ipynb file to all the libraries needed:



* 1. Type the following to a new code block to import the CIFAR-10 dataset above:



* 1. The data are stored in NumPy arrays; type the following into a new block to see the shape of the arrays



* 1. The x\_train array has 50000 rows of data, with 32x32 images in each row. The depth of image color is 3 (RGB), where R is Red, G is Green, and B is Blue.
  2. The y\_train array has 50000 rows of data, each with 1 column, and so on.
  3. Let’s print out the first image in the x\_train in the form of an array and put the following code into a new block:



* 1. As you can see, the results are arrays and hard to read; thus, let’s use Matplotlib to print the image again, this time in the picture format. Put the following code into a new block:



* 1. Although the result is an image, due to the size, we can’t tell precisely what the image is, so let’s print out the label of this image. Put the following code into a new block:

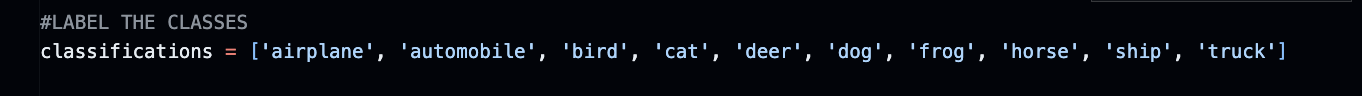


* 1. Trace back to the classes of the dataset (shown at the beginning of this HOS); the 6th classification is “Frog”:

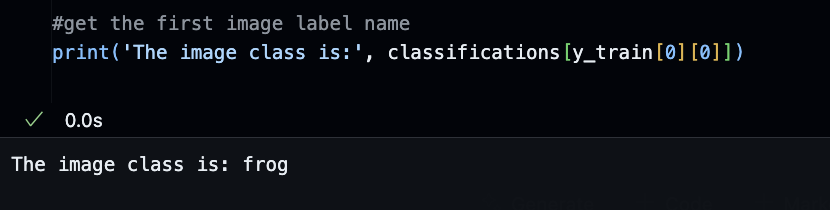
A picture containing text, screenshot

Description automatically generated

* 1. To avoid tracing back to the list, let’s create labels for our datasets. Type the following into a new block:

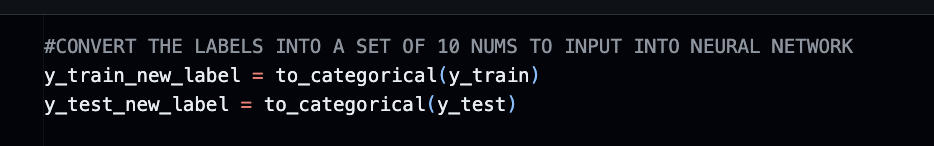


* 1. Now, let’s print the label again; we should see a label name instead of a number. Type the following into a new block:



**Section 3: Convert the labels into a set of 10 numbers to input into a neural network**

* 1. We must prepare and clean up the dataset before training it. First, convert the above labels into numbers between 0 and 1. Type the following into a new block:



* 1. Type the following into a new block to print the new labels:



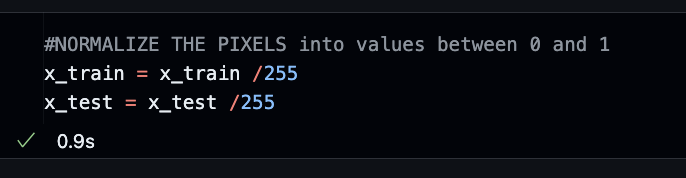
* 1. Type the following into a new block to print the new labels of the image printed above (frog):



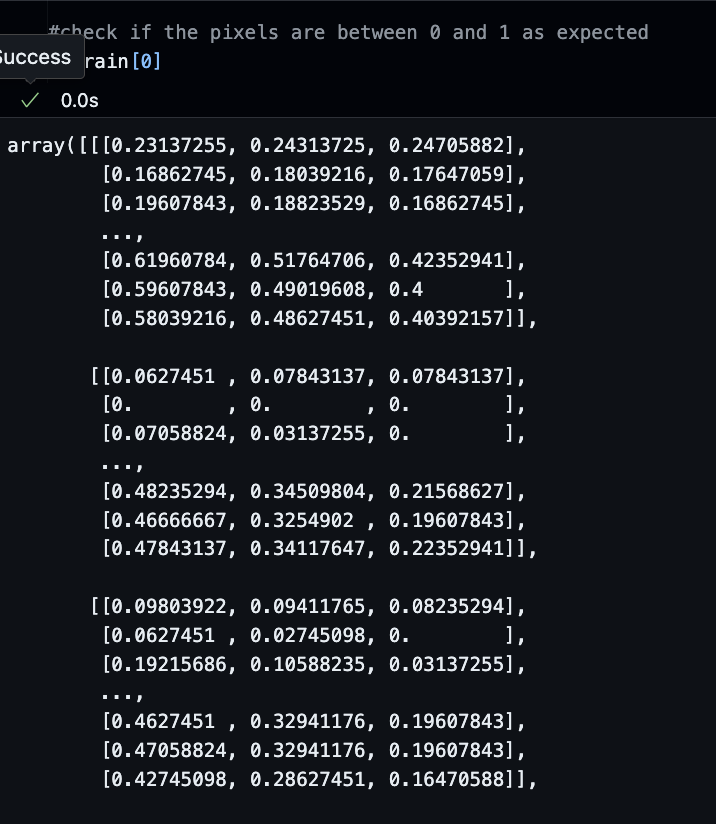
* 1. Each 0 and 1 represents a label name of all the ten labels we have. 1 illustrates the label of the image we are printing, while 0 is the opposite. Thus, based on the result above, one shows up at the 6th index. Trace back to our label list (“classifications” array). At the 6th index, the label name is “frog.” We can confirm that everything is working correctly as expected.

**Section 4: Normalize the data**

* 1. Same concept as last week: we want to normalize our data before training the dataset. To have the best performance, type the following into a new block to do so:

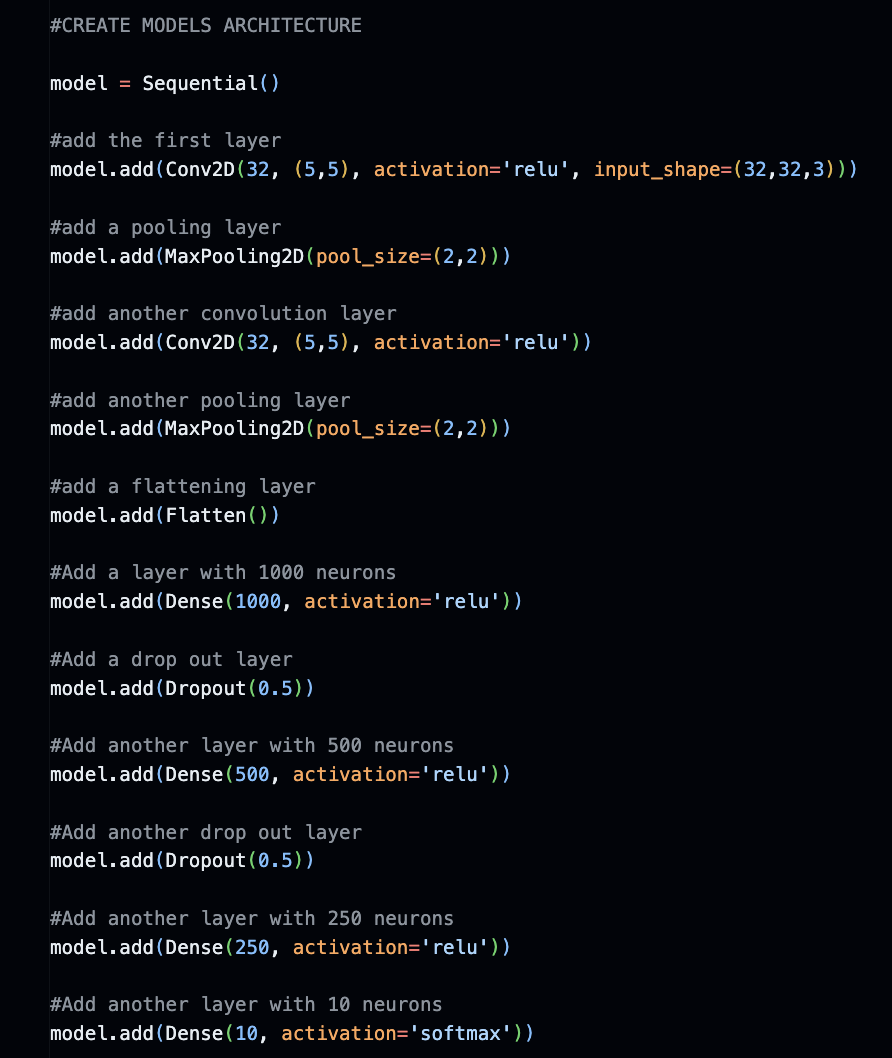


* 1. Check our result; type the following into a new block:



**Section 5: Build the model**

* 1. Type the following into a new block to create the model’s architecture:

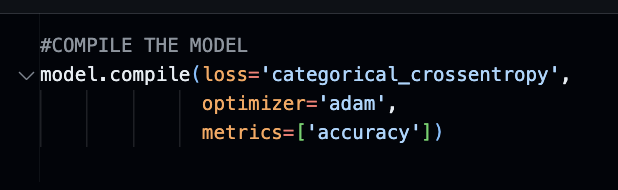


* + 1. Input Layer: It represents input image data. It will reshape the image into a single-dimensional array. For example, if your image is 64x64 = 4096, it will convert to a (4096,1) array.
    2. Conv Layer: This layer will extract features from the image.
    3. Pooling Layer: This layer reduces the spatial volume of the input image after convolution.
    4. Fully Connected Layer: It connects the network from one layer to another layer
    5. Output Layer: It is the predicted values layer.

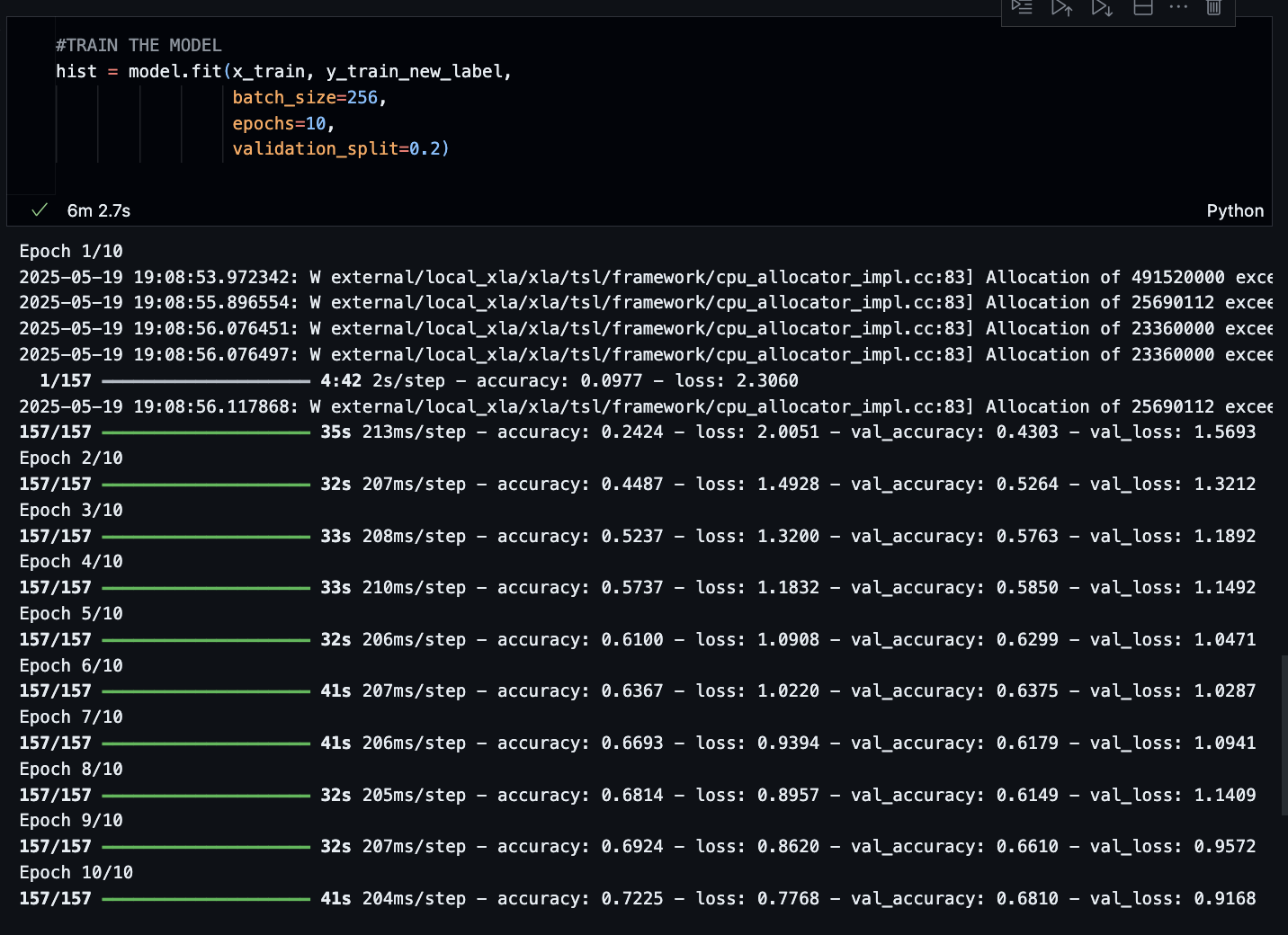
Diagram

Description automatically generated

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



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



*Note: This step might take up to 5 minutes or more.*

**Section 6: Evaluate and test the model.**

* 1. Type the following into a new block to evaluate our model accuracy and loss:



* 1. Type the following into a new block to test the model by using the dog\_test.jpeg image (given in the same Module 8 folder you are working on):



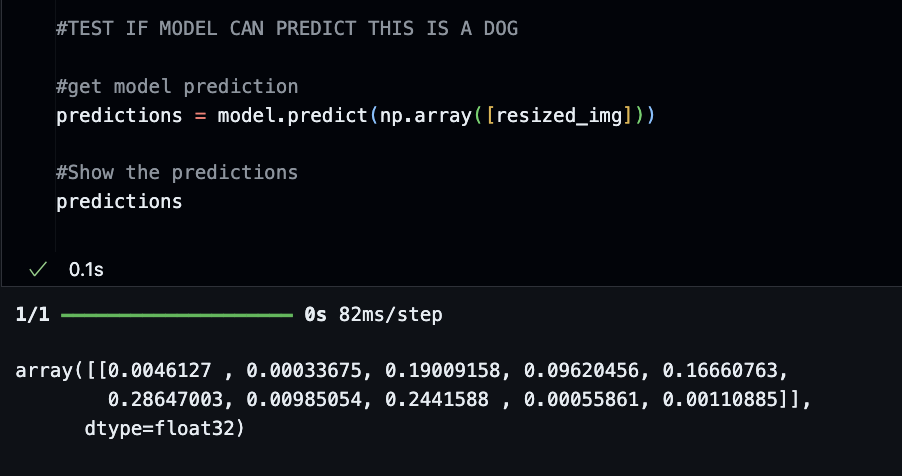
* 1. Let’s use a math plot to show this image instead; in a new block, type the following:



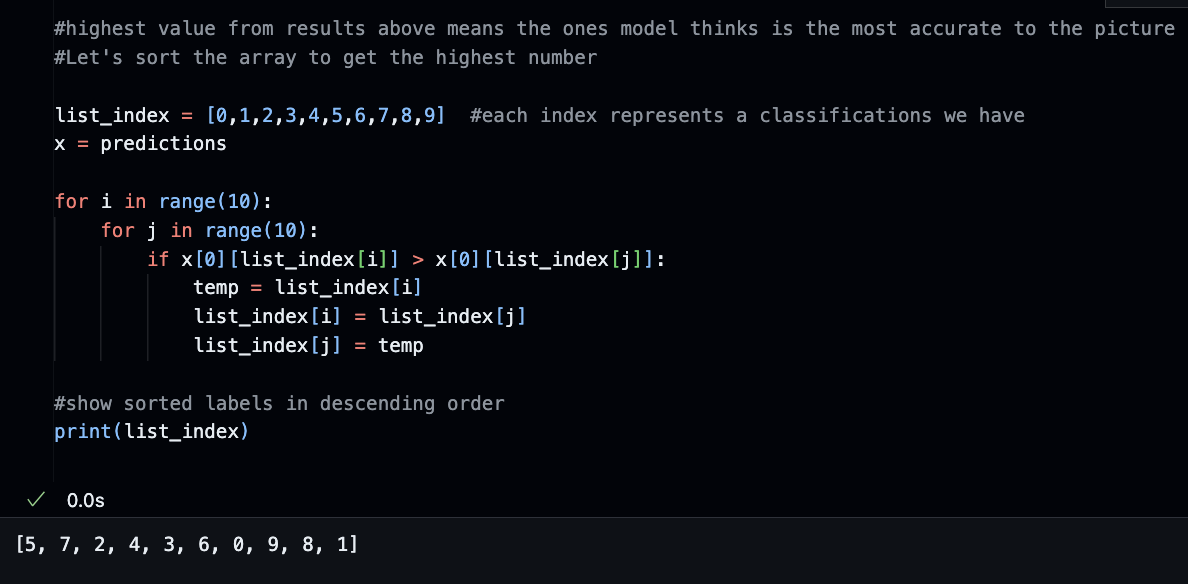
* 1. As you notice, the size of this picture is not what our model works on, so let’s resize it to 32x32, with 3RGB depth. In a new block, type the following:



* 1. Type the following into a new block:



* 1. The result was difficult to understand, so let’s convert them into numbers between 0 and 9, representing our ten labels. We will sort and print them in descending order, meaning the number shown at the beginning of the array is the result the model is most confident about. Type the following into a new block:

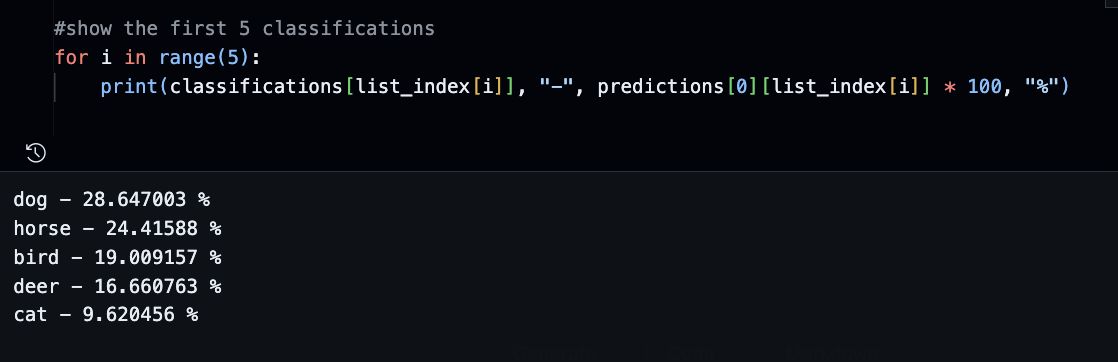


* 1. Trace back to our list of labels; the one with index 5 is “dog”:

A picture containing text, screenshot

Description automatically generated

* 1. Thus, we can be happy that our model works well. We can also print out the list of labels one more time together with the prediction. Type the following into a new block:



* 1. The model is most confident that our test image is a dog, and so on.

**Save your Jupyter Notebook with all outputs.**

**Section 3: Pushing your work to GitHub**

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

1. Go to Source Control on your GitHub Code space and observe the pending changes.
2. Type the message for your changes in the message box at the top. For example,” **Submission for Module08 – Your Name.**”
3. Click on the dropdown beside the commit button and select “**Commit & Push”** to update the changes to your repository's main branch.
   * Select **Yes** when prompted.