**CS506 Programming for Computing**

**HOP09 Computer Vision – Image Classification with CNN**

11/30/2020 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.
* Create Convolutional Neural Networks

**Resources**

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

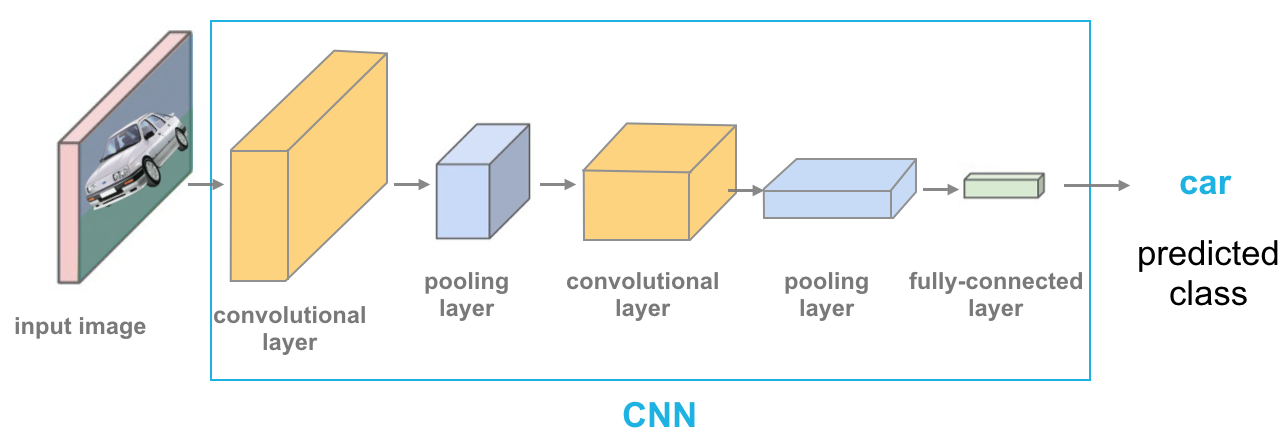
**Preparation**

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

Last week, we have learned how to

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

This week, we will continue to practice image classification, create a Convolutional Neural Netnork and see how powerful Python, TensorFlow can be used to manipulate images.



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

This week, we will work on the CIFAR-10 dataset, have a look at the dataset to know what we will be working on: <https://www.cs.toronto.edu/~kriz/cifar.html>

A picture containing text, screenshot

Description automatically generated

Let’s start!

**INSTALL LIBRARIES**

In this HOP, we will need keras and scikit-image libraries, follow the below instructions to install, if you already have them installed, skip this section.

1. Open Command Prompt (for Windows users) or Terminal (for Mac Users), type the following command to install keras:

pip3 install keras

Text

Description automatically generated

Once done, you should see the similar message as following:

Text

Description automatically generated

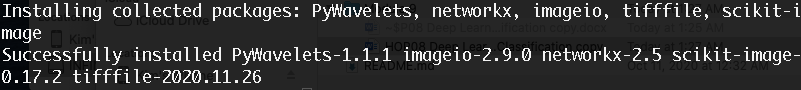
1. Then, type the following to install scikit-image

pip3 install scikit-image

Text

Description automatically generated

Once done, you should see the similar message as following:



**IMPORT DATASET**

1. In Visual Studio Code, open the private repository generated when you accepted the HOP09 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 Jupyter Notebook, under Module 9 folder, create a new file called **Image\_Classification.ipynb** and simply click on the file to open notebook.
3. Type the following into the **Image\_Classification.ipynb** file, to all the libraries needed:

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application

Description automatically generated

1. The data are stored into numpy arrays, type the following into a new block, to see the shape of the arrays:

Text

Description automatically generated

For the x\_train array: there are 50000 rows of data, with 32x32 images each row, depth of image color is 3 (RGB) where R is Red, G is Green, and B is Blue.

For the y\_train array: there are 50000 rows of data, with 1 column. So on and so forth.

1. Let’s print out the first image in the x\_train in the form of an array, put the following code into a new block:

Table

Description automatically generated

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

Graphical user interface, application

Description automatically generated

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

Table

Description automatically generated

Trace back to the classes of the dataset (shown in the beginning of this HOP), the 6th classification is “Frog”:

A picture containing text, screenshot

Description automatically generated

1. To avoid having to trace back to the list, let’s create the 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:

Graphical user interface, text, application

Description automatically generated

**CONVERT THE LABELS INTO A SET OF 10 NUMS TO INPUT INTO NEURAL NETWORK**

1. Before training the dataset, we need more preparation and clean up, first, let’s convert the above labels into a set of numbers between 0 and 1. Type the following into a new block:

Text

Description automatically generated

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

Calendar

Description automatically generated

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

Graphical user interface, text, application, chat or text message

Description automatically generated

Each 0 and 1 represents a label name of all the 10 labels we have. 1 represents the label of the image we are printing, while 0 is the opposite. Thus, based on the result above, 1 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.

**NORMALIZE THE DATA**

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

Text

Description automatically generated

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

Table

Description automatically generated

**BUILD THE MODEL**

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

A picture containing table

Description automatically generated

* Input Layer: It represent input image data. It will reshape image into single diminsion array. Example your image is 64x64 = 4096, it will convert to (4096,1) array.
* Conv Layer: This layer will extract features from image.
* Pooling Layer: This layerreduce the spatial volume of input image after convolution.
* Fully Connected Layer: It connect the network from a layer to another layer
* Output Layer: It is the predicted values layer.

Diagram

Description automatically generated

More about layers in CNNs: <https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>

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

Graphical user interface, text

Description automatically generated

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

Table

Description automatically generated

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

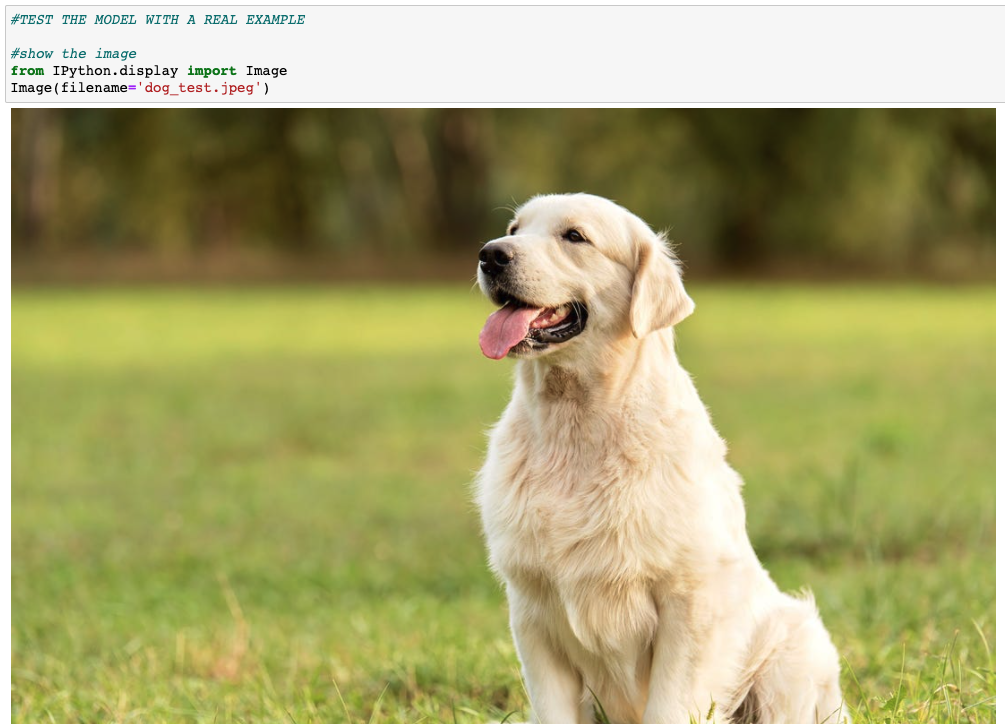
**EVALUATE AND TEST THE MODEL**

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

Graphical user interface, text

Description automatically generated

1. Type the following into a new block to test the model by using the dog\_test.jpeg image (given in the same Module 9 folder you are working on):



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

A picture containing text, grass, screenshot

Description automatically generated

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:

Graphical user interface

Description automatically generated

1. Type the following into a new block:

Graphical user interface, text, application

Description automatically generated

1. The result was not very easy to understand, so let’s convert them into numbers between 0 and 9, representing the 10 labels we have. We will sort and print them in descending order, meaning the number shows up in the beginning of the array is the result the model most confident about. Type the following into a new block:

Graphical user interface, text, application, email

Description automatically generated

Trace back to our list of labels, the one with index 5 is “dog”:

A picture containing text, screenshot

Description automatically generated

Thus, we can be happy that our model is working pretty good. We can also print out the list of label one more time together with the prediction. Type the following into a new block:

Graphical user interface, text, application

Description automatically generated

The model is most confident that our test image is a dog, showing in 100%, and so on.

**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 9 – Your Name”

>>> git push origin master