Project6: Session38\_ImplementationImageClassification

Image Classification

In this project, you'll classify images from the CIFAR-10 dataset

(https://www.cs.toronto.edu/~kriz/cifar.html). The dataset consists of airplanes, dogs, cats, and other

objects. You'll preprocess the images, then train a convolutional neural network on all the samples.

The images need to be normalized and the labels need to be one-hot encoded. You'll get to apply

what you learned and build a convolutional, max pooling, dropout, and fully connected layers. At the

end, you'll get to see your neural network's predictions on the sample images.

Get the Data

Run the following cell to download the CIFAR-10 dataset for python

(https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz).

Data

CIFAR-10 is an established computer-vision dataset used for object recognition. It is a subset of the

80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10

object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair, and

Geoffrey Hinton.

Let's get the data by running the following function

In [2]:

Explore the Data

The dataset is broken into batches to prevent your machine from running out of memory. The

CIFAR-10 dataset consists of 5 batches, named data\_batch\_1, data\_batch\_2, etc.. Each batch

contains the labels and images that are one of the following:

airplane

automobile

bird

cat

deer

dog

frog

horse

ship

truck

Understanding a dataset is part of making predictions on the data. Play around with the code cell

below by changing the batch\_id and sample\_id. The batch\_id is the id for a batch (1-5). The

sample\_id is the id for a image and label pair in the batch.

Ask yourself "What are all possible labels?", "What is the range of values for the image data?", "Are

the labels in order or random?". Answers to questions like these will help you preprocess the data

and end up with better predictions.

from urllib.request import urlretrieve

from os.path import isfile, isdir

from tqdm import tqdm

import tarfile

cifar10\_dataset\_folder\_path = 'cifar-10-batches-py'

class DLProgress(tqdm):

last\_block = 0

def hook(self, block\_num=1, block\_size=1, total\_size=None):

self.total = total\_size

self.update((block\_num - self.last\_block) \* block\_size)

self.last\_block = block\_num

if not isfile('cifar-10-python.tar.gz'):

with DLProgress(unit='B', unit\_scale=True, miniters=1, desc='CIFAR-10 Dataset

urlretrieve(

'https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz',

'cifar-10-python.tar.gz',

pbar.hook)

if not isdir(cifar10\_dataset\_folder\_path):

with tarfile.open('cifar-10-python.tar.gz') as tar:

tar.extractall()

tar.close()

The following are some helper functions students can use

in their code

In [3]: import pickle

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelBinarizer

def \_load\_label\_names():

"""

Load the label names from file

"""

return ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse

def load\_cfar10\_batch(cifar10\_dataset\_folder\_path, batch\_id):

"""

Load a batch of the dataset

"""

with open(cifar10\_dataset\_folder\_path + '/data\_batch\_' + str(batch\_id), mode=

batch = pickle.load(file, encoding='latin1')

features = batch['data'].reshape((len(batch['data']), 3, 32, 32)).transpose(0

labels = batch['labels']

return features, labels

def display\_stats(cifar10\_dataset\_folder\_path, batch\_id, sample\_id):

"""

Display Stats of the the dataset

"""

batch\_ids = list(range(1, 6))

if batch\_id not in batch\_ids:

print('Batch Id out of Range. Possible Batch Ids: {}'.format(batch\_ids))

return None

features, labels = load\_cfar10\_batch(cifar10\_dataset\_folder\_path, batch\_id)

if not (0 <= sample\_id < len(features)):

print('{} samples in batch {}. {} is out of range.'.format(len(features)

return None

print('\nStats of batch {}:'.format(batch\_id))

print('Samples: {}'.format(len(features)))

print('Label Counts: {}'.format(dict(zip(\*np.unique(labels, return\_counts=True

print('First 20 Labels: {}'.format(labels[:20]))

sample\_image = features[sample\_id]

sample\_label = labels[sample\_id]

label\_names = \_load\_label\_names()

print('\nExample of Image {}:'.format(sample\_id))

print('Image - Min Value: {} Max Value: {}'.format(sample\_image.min(), sample\_

print('Image - Shape: {}'.format(sample\_image.shape))

print('Label - Label Id: {} Name: {}'.format(sample\_label, label\_names[sample\_

plt.axis('off')

plt.imshow(sample\_image)

def

"""

Preprocess data and save it to file

"""

features = normalize(features)

labels = one\_hot\_encode(labels)

pickle.dump((features, labels), open(filename, 'wb'))

def preprocess\_and\_save\_data(cifar10\_dataset\_folder\_path, normalize, one\_hot\_encod

"""

Preprocess Training and Validation Data

"""

n\_batches = 5

valid\_features = []

valid\_labels = []

for batch\_i in range(1, n\_batches + 1):

features, labels = load\_cfar10\_batch(cifar10\_dataset\_folder\_path, batch\_i

validation\_count = int(len(features) \* 0.1)

*# Prprocess and save a batch of training data*

\_preprocess\_and\_save(

normalize,

one\_hot\_encode,

features[:-validation\_count],

labels[:-validation\_count],

'preprocess\_batch\_' + str(batch\_i) + '.p')

*# Use a portion of training batch for validation*

valid\_features.extend(features[-validation\_count:])

valid\_labels.extend(labels[-validation\_count:])

*# Preprocess and Save all validation data*

\_preprocess\_and\_save(

normalize,

one\_hot\_encode,

np.array(valid\_features),

np.array(valid\_labels),

'preprocess\_validation.p')

with open(cifar10\_dataset\_folder\_path + '/test\_batch', mode='rb') as file:

batch = pickle.load(file, encoding='latin1')

*# load the training data*

test\_features = batch['data'].reshape((len(batch['data']), 3, 32, 32)).transpo

test\_labels = batch['labels']

*# Preprocess and Save all training data*

\_preprocess\_and\_save(

normalize,

one\_hot\_encode,

np.array(test\_features),

Image\_classification

\_preprocess\_and\_save(normalize, one\_hot\_encode, features, labels, filename):

np.array(test\_labels),

'preprocess\_training.p')

def batch\_features\_labels(features, labels, batch\_size):

"""

Split features and labels into batches

"""

for start in range(0, len(features), batch\_size):

end = min(start + batch\_size, len(features))

yield features[start:end], labels[start:end]

def load\_preprocess\_training\_batch(batch\_id, batch\_size):

"""

Load the Preprocessed Training data and return them in batches of <batch\_size

"""

filename = 'preprocess\_batch\_' + str(batch\_id) + '.p'

features, labels = pickle.load(open(filename, mode='rb'))

*# Return the training data in batches of size <batch\_size> or less*

return batch\_features\_labels(features, labels, batch\_size)

def display\_image\_predictions(features, labels, predictions):

n\_classes = 10

label\_names = \_load\_label\_names()

label\_binarizer = LabelBinarizer()

label\_binarizer.fit(range(n\_classes))

label\_ids = label\_binarizer.inverse\_transform(np.array(labels))

fig, axies = plt.subplots(nrows=4, ncols=2)

fig.tight\_layout()

fig.suptitle('Softmax Predictions', fontsize=20, y=1.1)

n\_predictions = 3

margin = 0.05

ind = np.arange(n\_predictions)

width = (1. - 2. \* margin) / n\_predictions

for image\_i, (feature, label\_id, pred\_indicies, pred\_values) in enumerate(zip

pred\_names = [label\_names[pred\_i] for pred\_i in pred\_indicies]

correct\_name = label\_names[label\_id]

axies[image\_i][0].imshow(feature\*255)

axies[image\_i][0].set\_title(correct\_name)

axies[image\_i][0].set\_axis\_off()

axies[image\_i][1].barh(ind + margin, pred\_values[::-1], width)

axies[image\_i][1].set\_yticks(ind + margin)

axies[image\_i][1].set\_yticklabels(pred\_names[::-1])

axies[image\_i][1].set\_xticks([0, 0.5, 1.0])

In [21]:

Stats of batch 3:

Samples: 10000

Label Counts: {0: 994, 1: 1042, 2: 965, 3: 997, 4: 990, 5: 1029, 6: 978, 7: 101

5, 8: 961, 9: 1029}

First 20 Labels: [8, 5, 0, 6, 9, 2, 8, 3, 6, 2, 7, 4, 6, 9, 0, 0, 7, 3, 7, 2]

Example of Image 5:

Image - Min Value: 9 Max Value: 255

Image - Shape: (32, 32, 3)

Label - Label Id: 2 Name: bird

%matplotlib inline

%config InlineBackend.figure\_format = 'retina'

import numpy as np

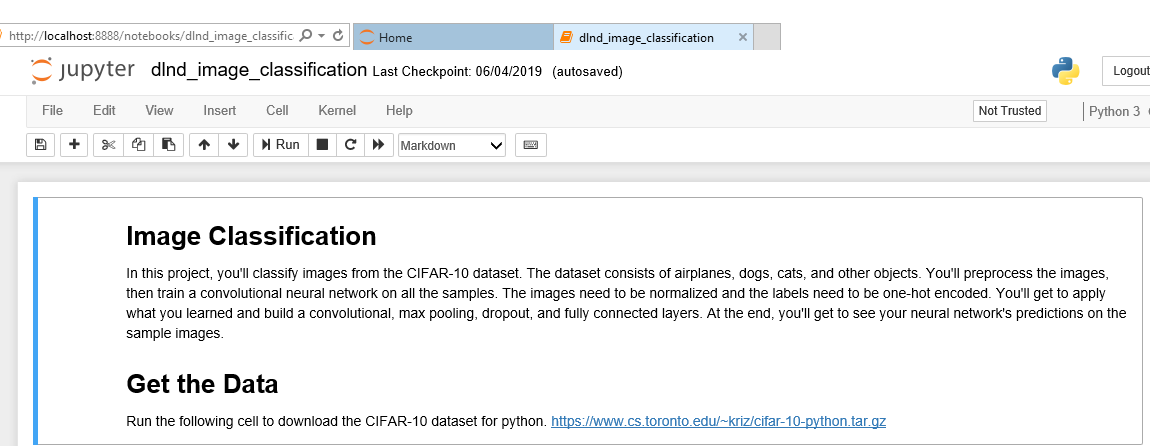
*# Explore the dataset*

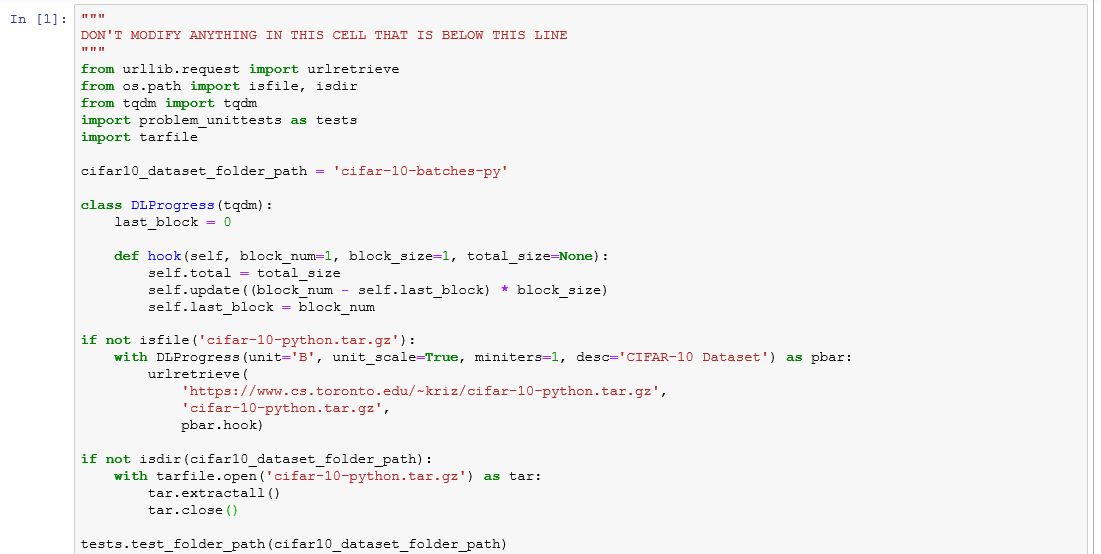
batch\_id = 3

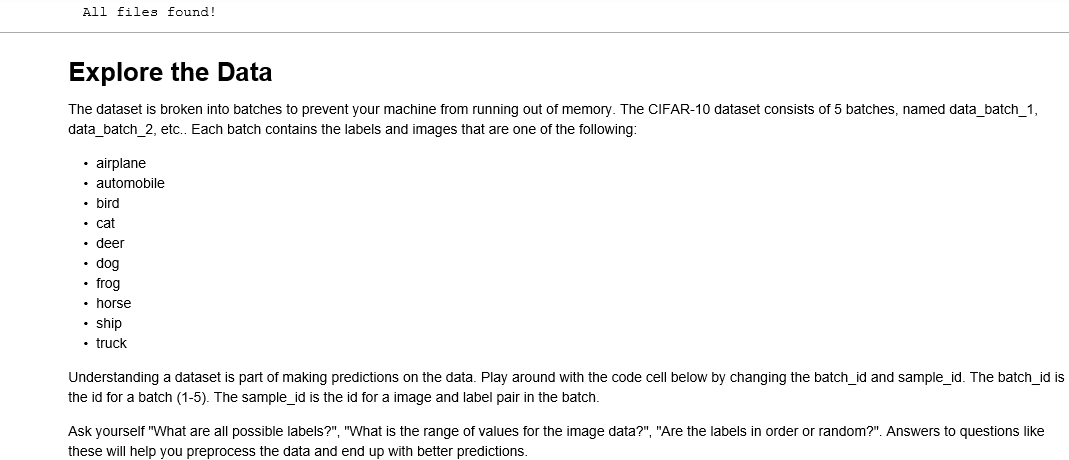
sample\_id = 5

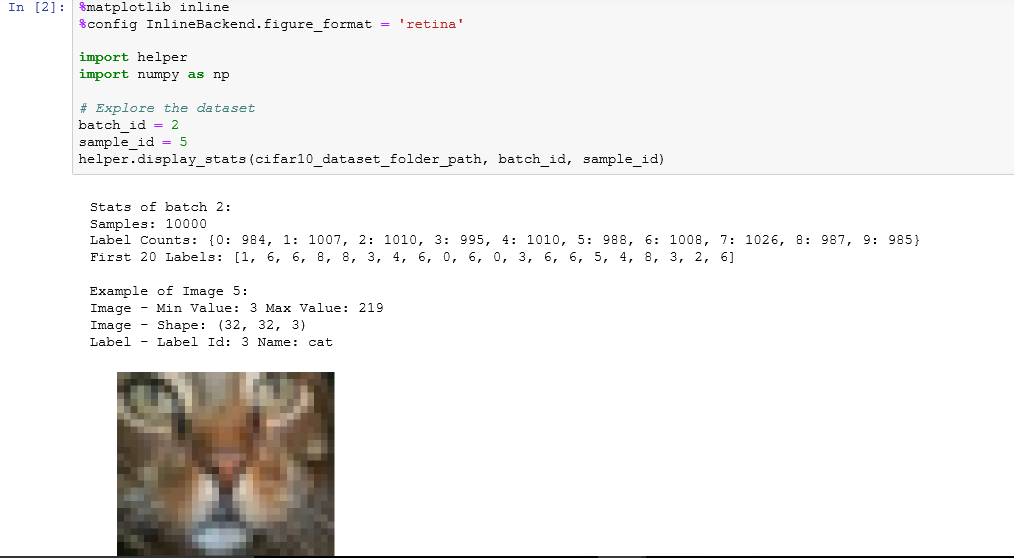
display\_stats(cifar10\_dataset\_folder\_path, batch\_id, sample\_id)

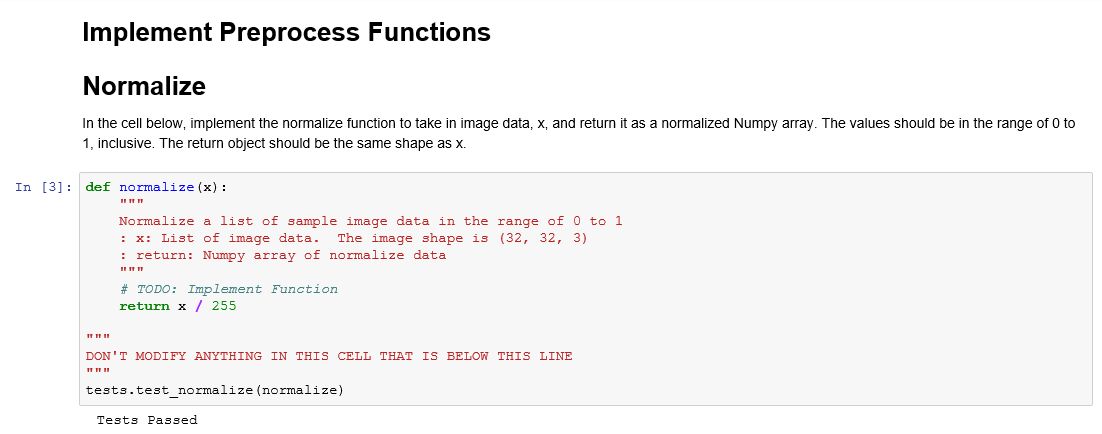
**OUTPUT**

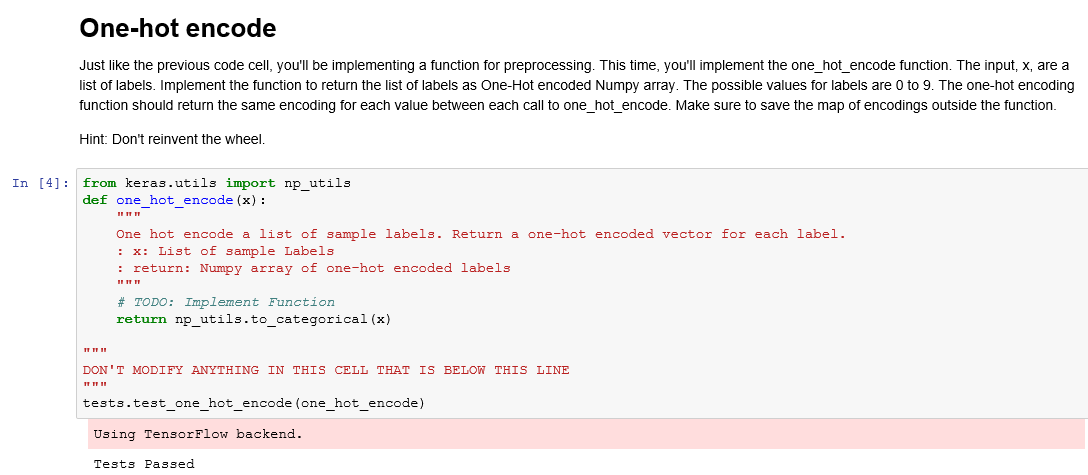
****

****

****

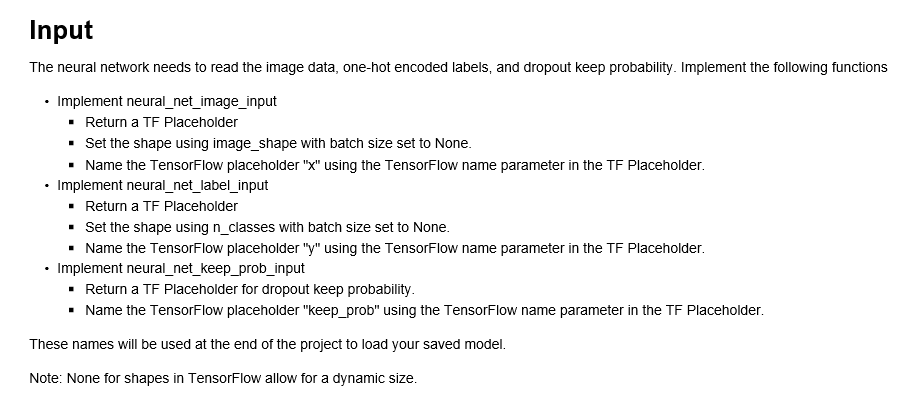
****

****

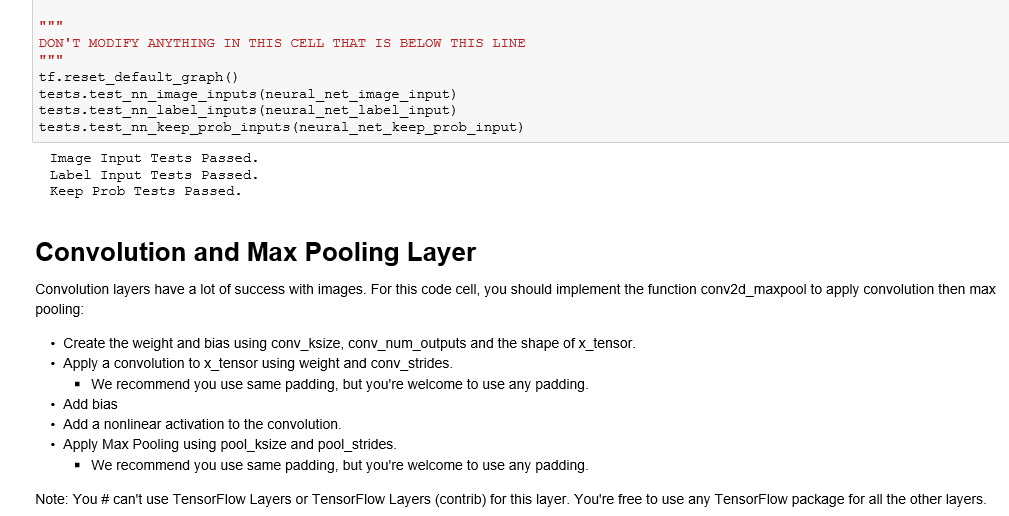
****

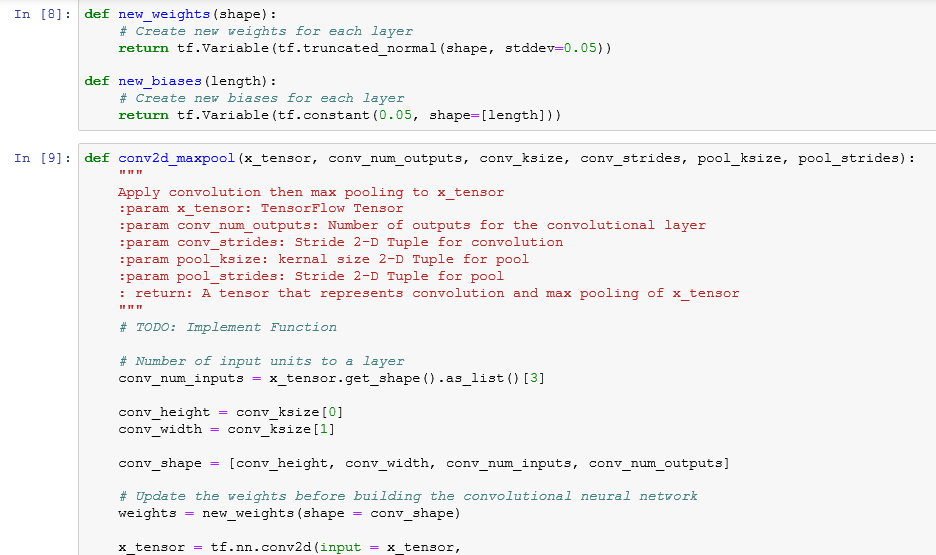
****

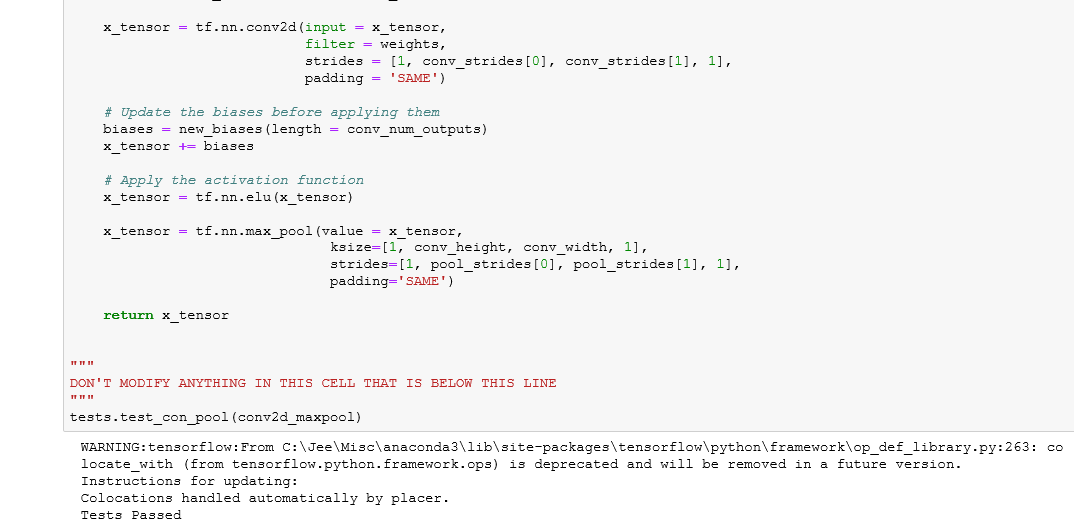
****

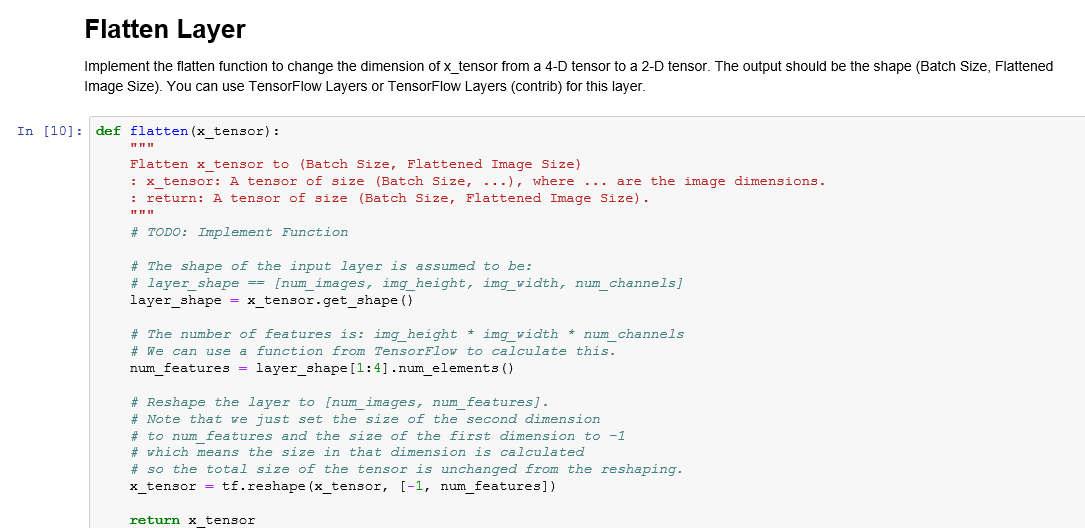
****

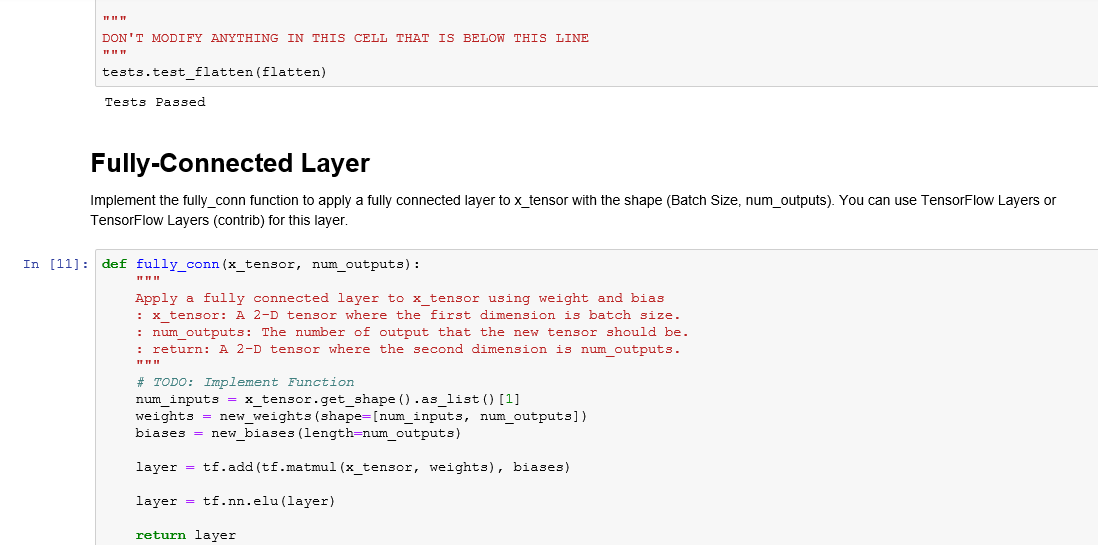
****

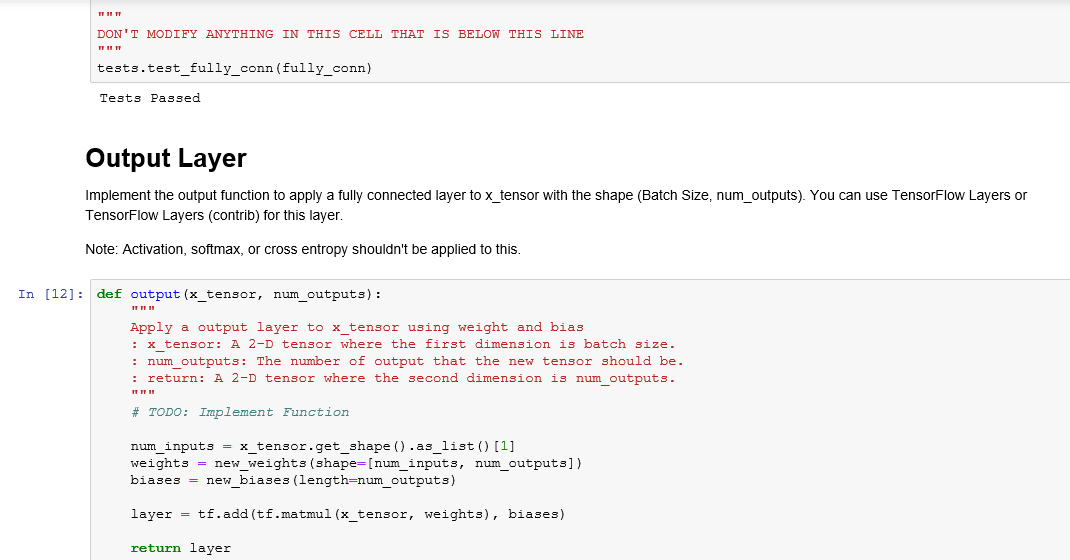
****

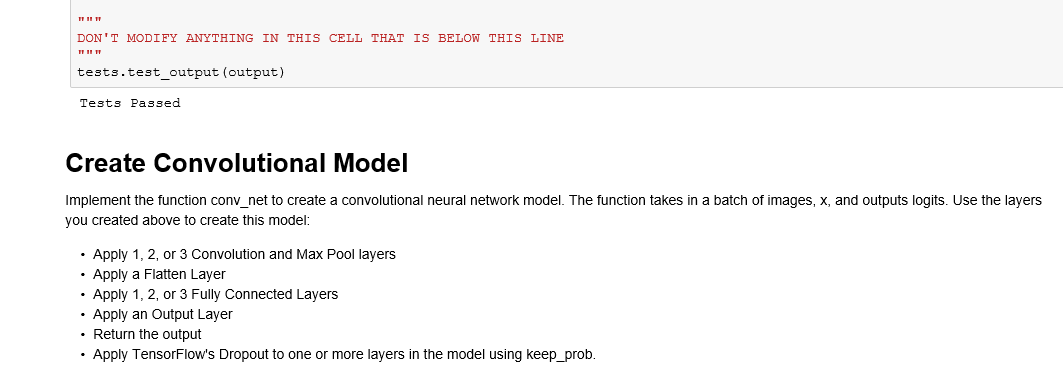
****

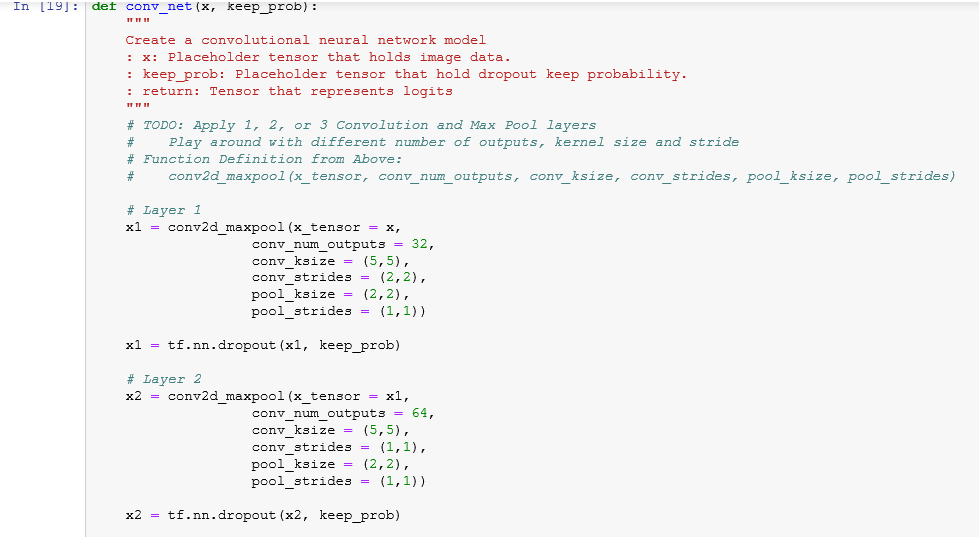
****

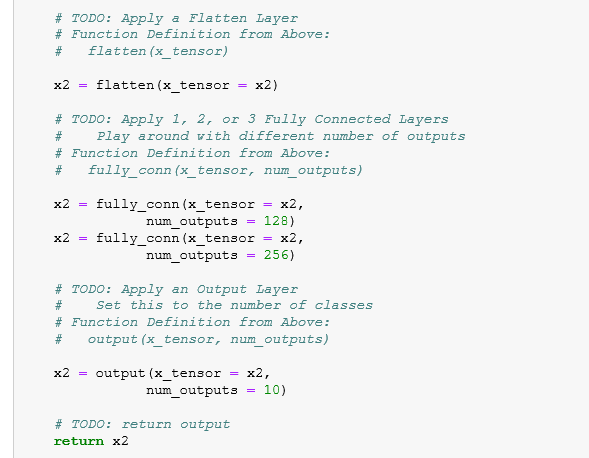
****

****

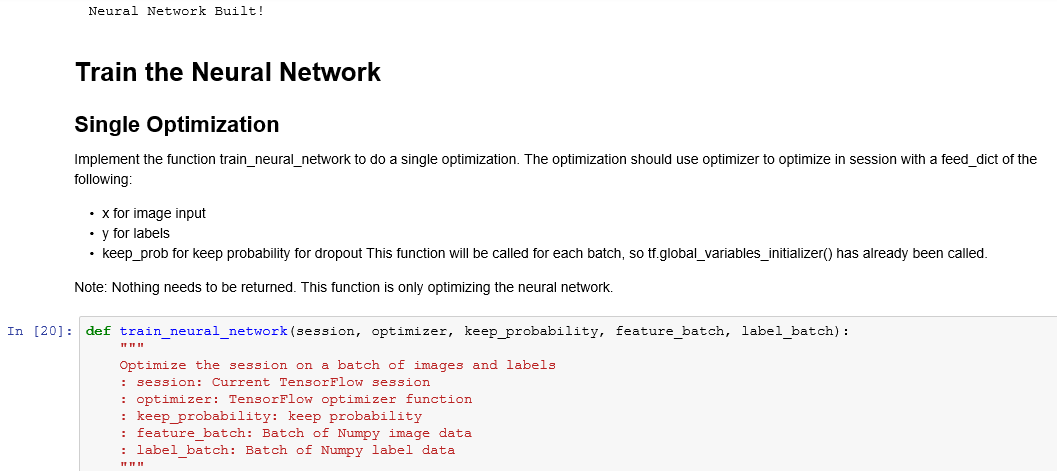
****

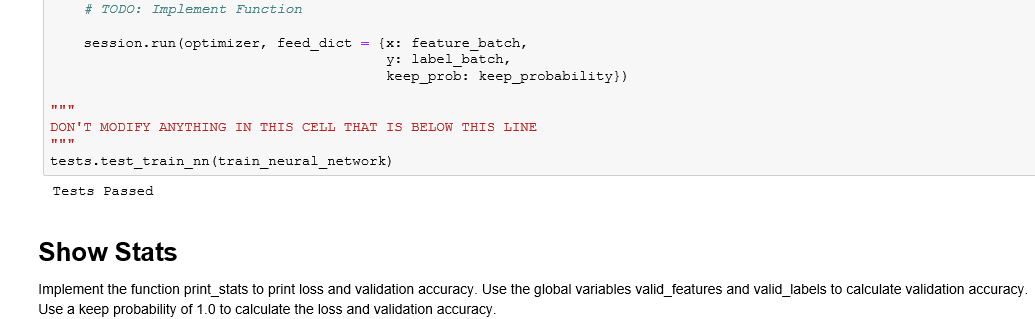
****

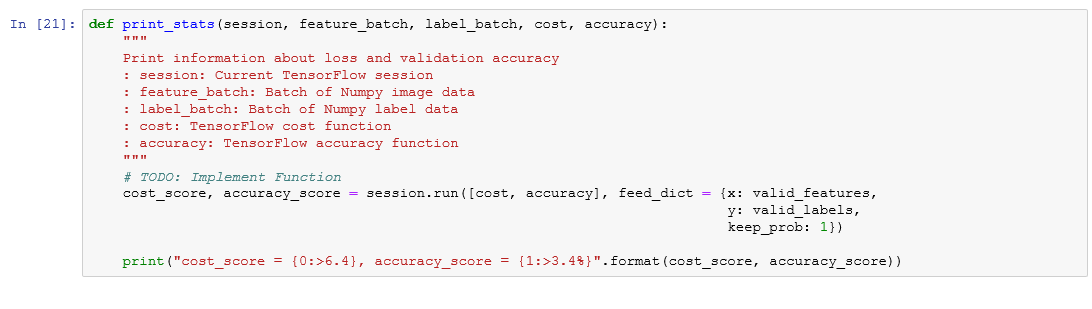
****

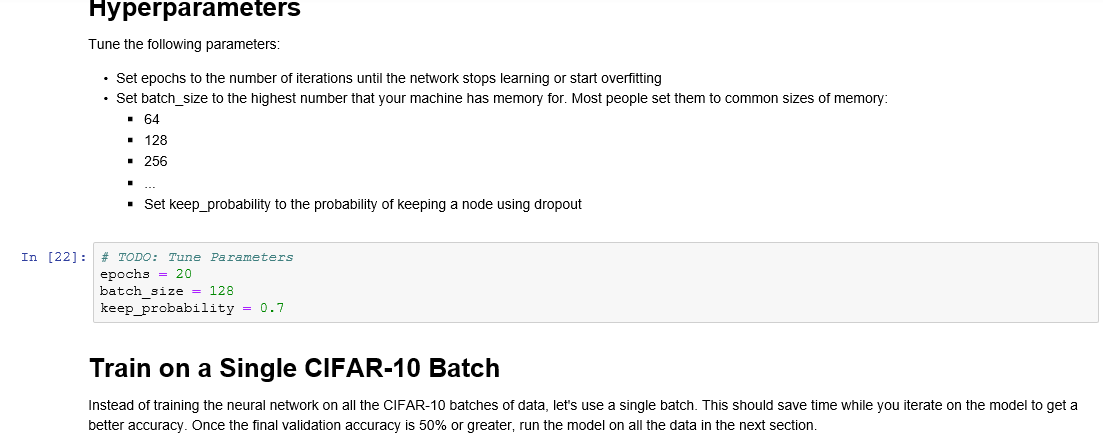
****

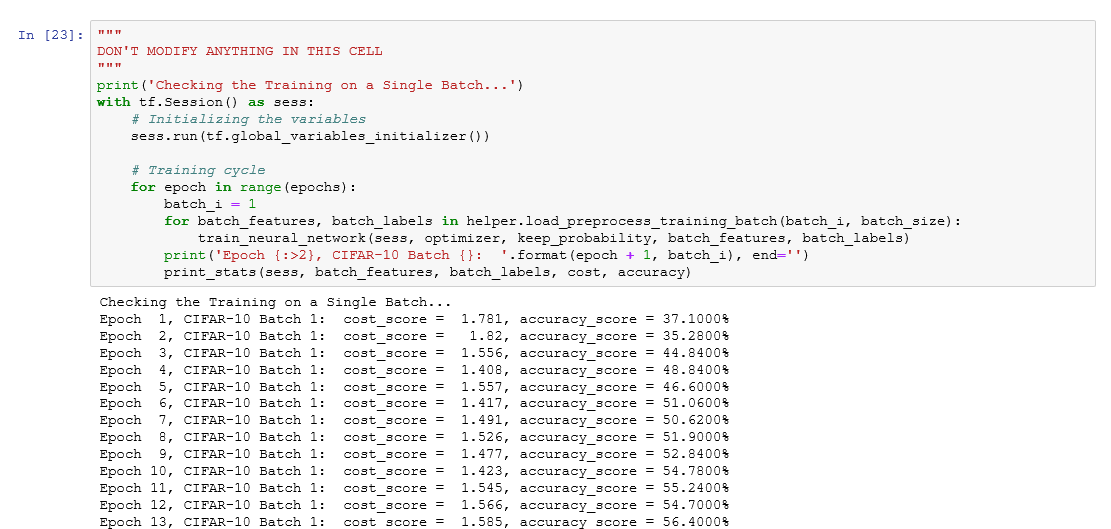
****

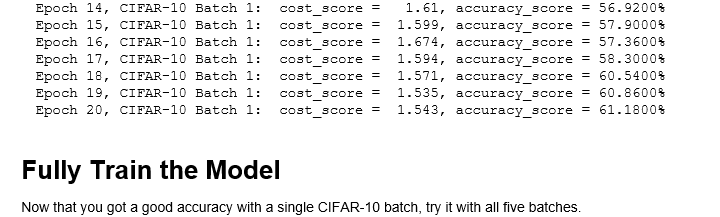
****

****

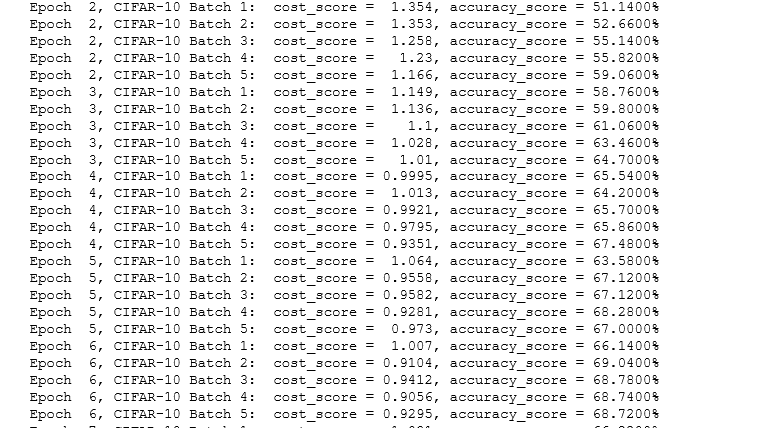
****

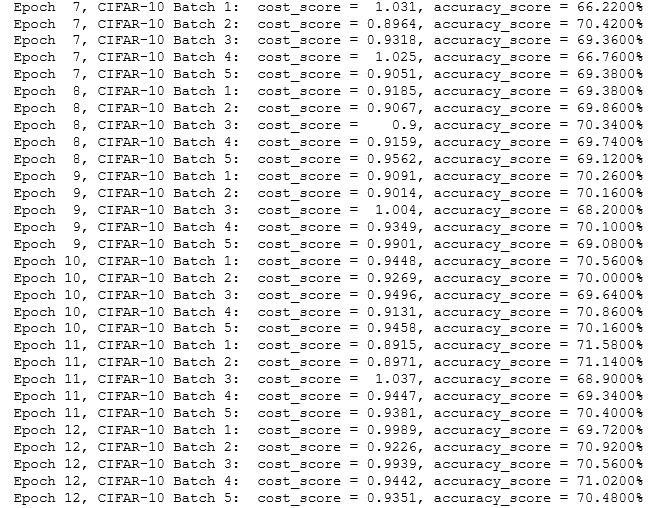
****

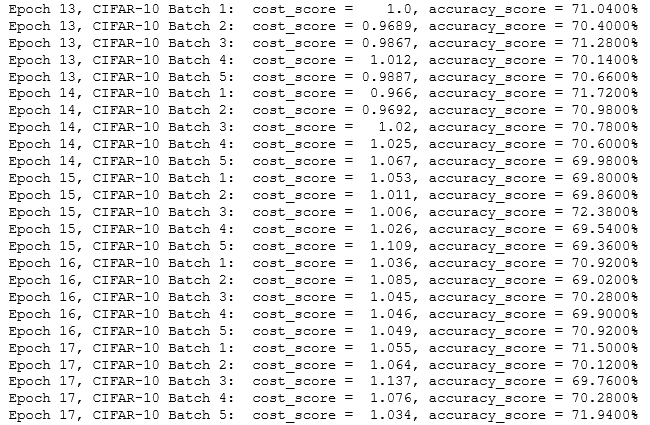
****

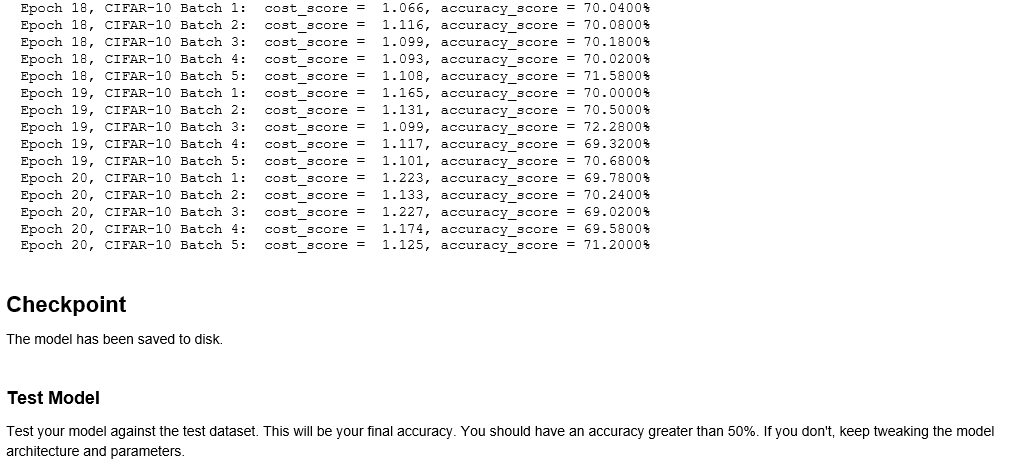
****

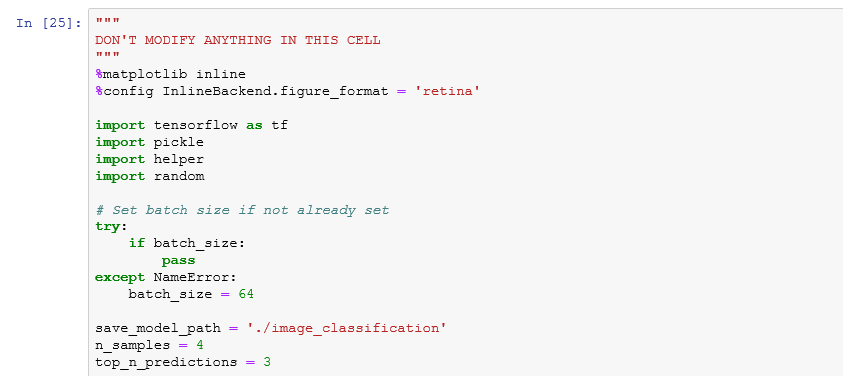
****

****

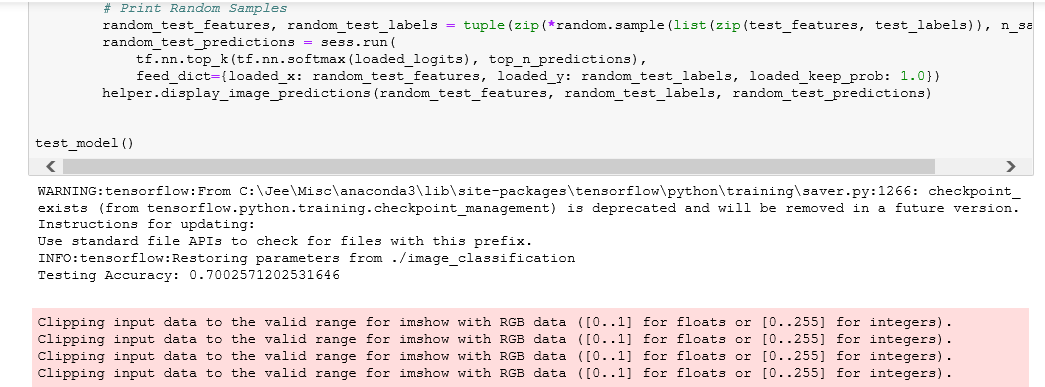
****

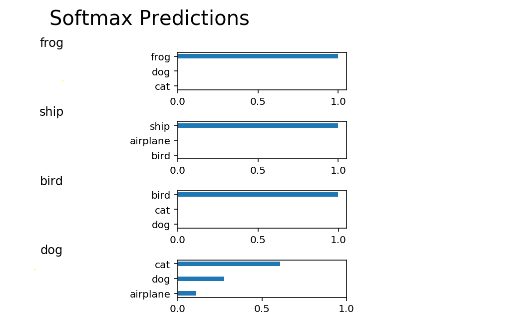
****

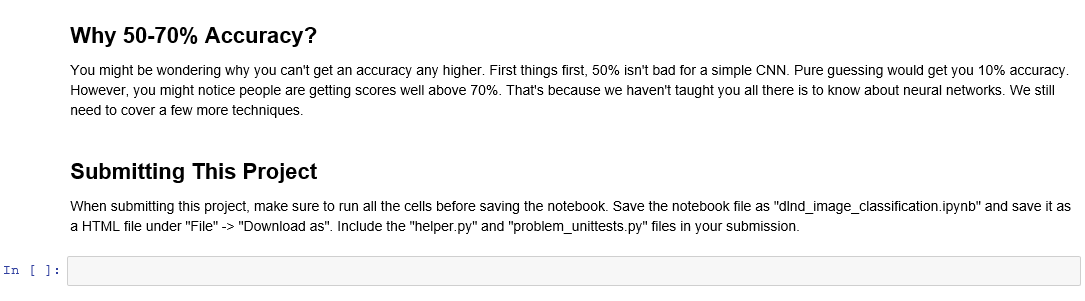
****

****

****

****

****

****