LAB 8 CIFAR-10

EE 104 – Christopher Pham

Eric Reyes

San Jose State University 2021

This project is split into multiple parts with the main cnn.ipynb acting as the main entrypoint.

The libs folder containers helper code using the cnn.ipynb as follows:

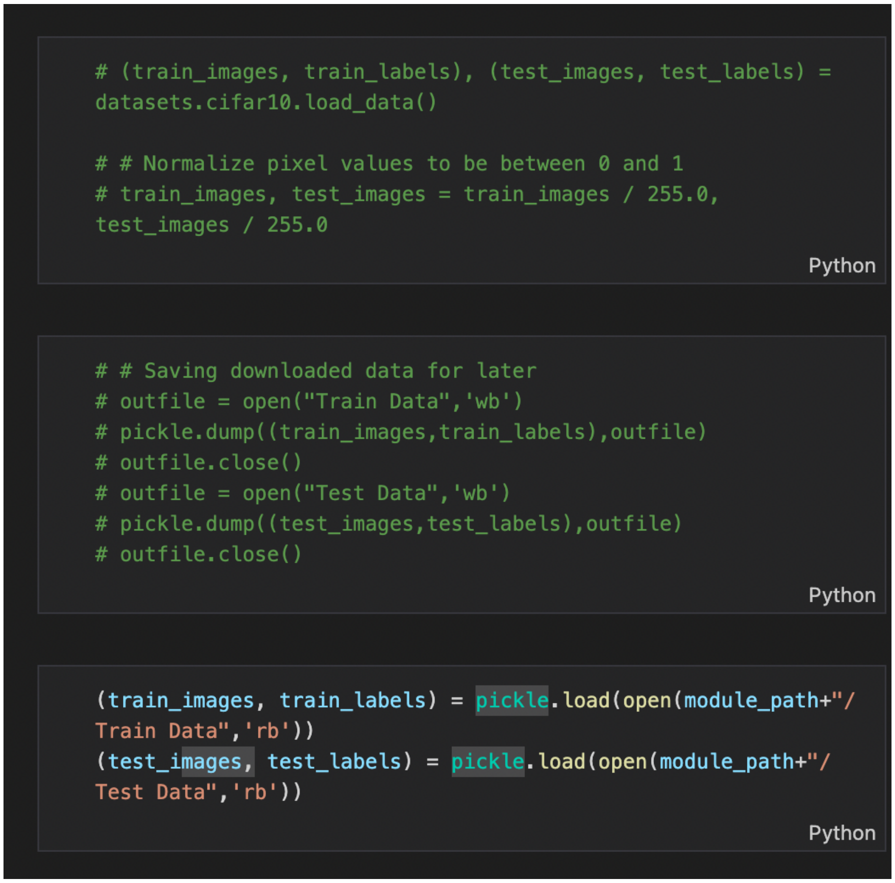
* const.py - contains constant values used in the code. Origin links are provided by the instructor and also individually supplied where needed.

* dataset.py - contains helper code that loads image dataset into a compatable format for training and testing

* models.py - contains model construction functions used to both create, tune and load the 4 different models.

Much of the code is commented out appropriately and will be repeated here for a relevant flow of operations.

Cnn.ipynb is the main entrypoint and loads modules from the libs folder. The CIRFAR-10 dataset is loaded directly from tensorflow. Figure 1 shows the code that both loads and saves the dataset. Saving the dataset for later use and loading it back in with pickle is much faster than redownloading the dataset on each run.



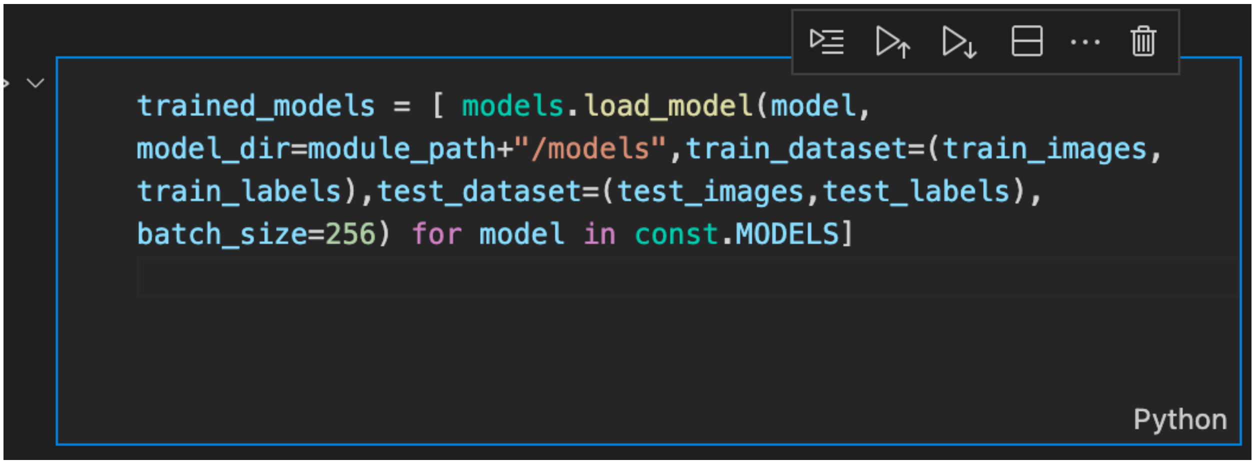
```python
# (train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

# # Normalize pixel values to be between 0 and 1
# train_images, test_images = train_images / 255.0,
test_images / 255.0
```

```python
# # Saving downloaded data for later
# outfile = open("Train Data",'wb')
# pickle.dump((train_images,train_labels),outfile)
# outfile.close()
# outfile = open("Test Data",'wb')
# pickle.dump((test_images,test_labels),outfile)
# outfile.close()
```

```python
(train_images, train_labels) = pickle.load(open(module_path+"/
Train Data",'rb'))
(test_images, test_labels) = pickle.load(open(module_path+"/
Test Data",'rb'))
```

Figure 1 Loading Data

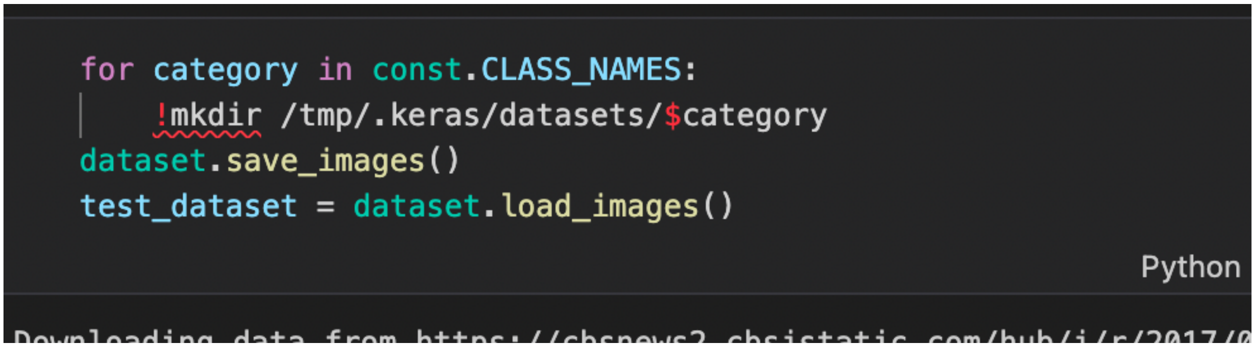
All models are then loaded into an array and trained if they cannot be found by using the load_model function from the custom models module.



```
trained_models = [ models.load_model(model,
model_dir=module_path+"/models",train_dataset=(train_images,
train_labels),test_dataset=(test_images,test_labels),
batch_size=256) for model in const.MODELS]
```

FIGURE 2 LOADING AND TRAINING ALL MODELS

A folder for each category in the CIFAR-10 dataset needs to be created before being loaded. The save_images function downloads each picture from the custom origins listed in the const module while the load_images function loads the downloaded images into a single dataset.

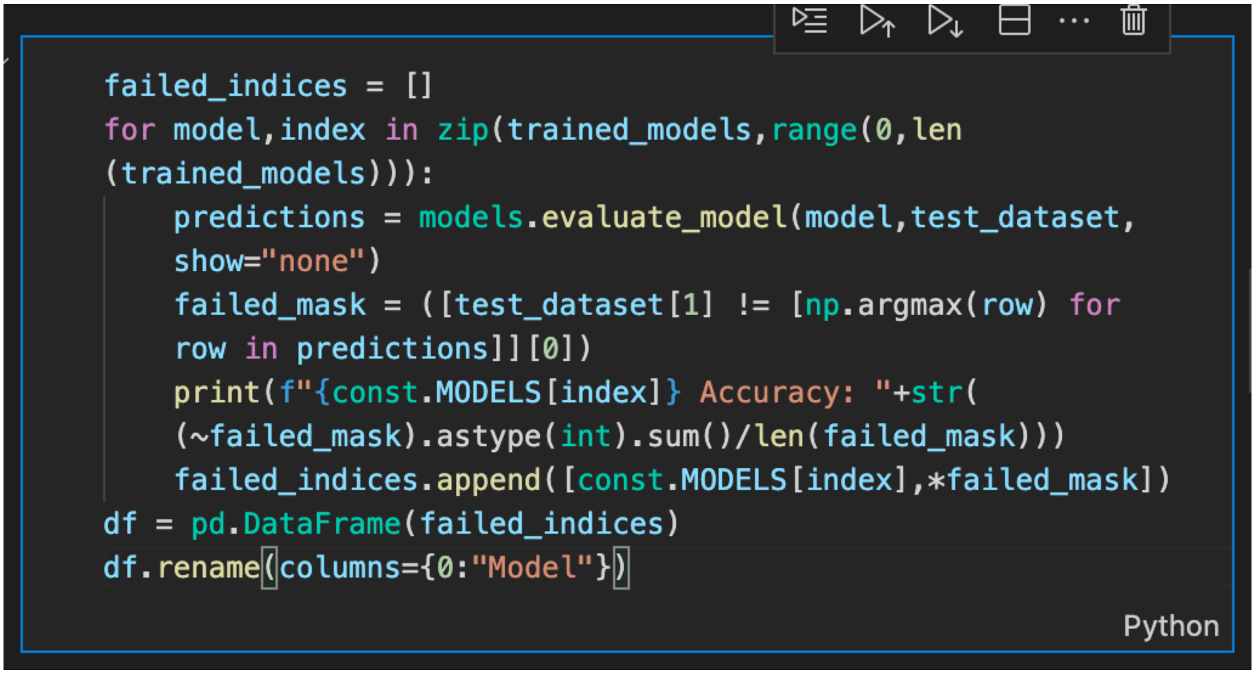


```
for category in const.CLASS_NAMES:
    !mkdir /tmp/.keras/datasets/$category
dataset.save_images()
test_dataset = dataset.load_images()
```

```
Downloading data from https://cbsnews2.cbsistatic.com/hub/i/r/2017/0
```

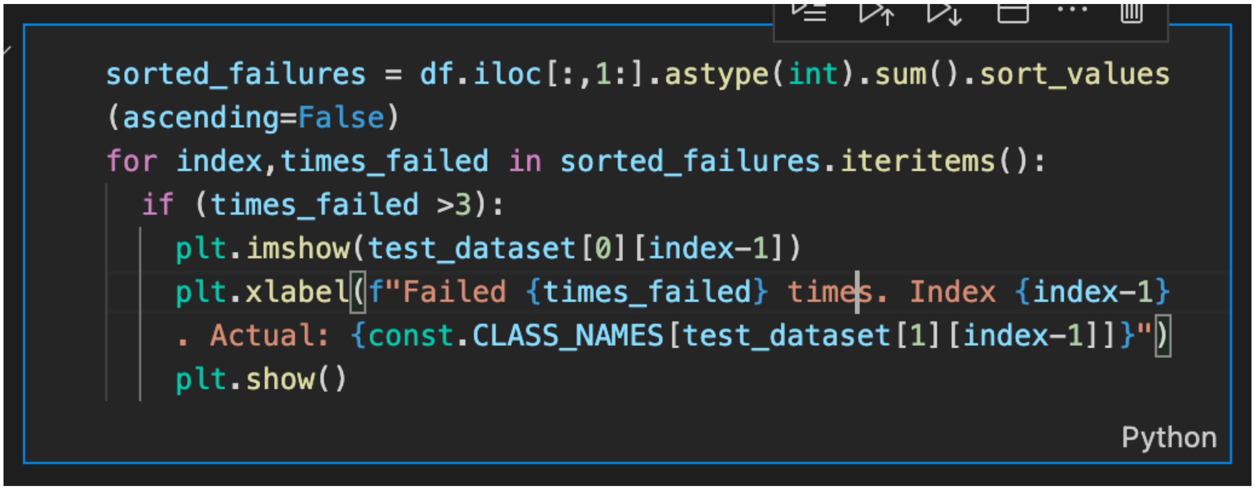
FIGURE 3 SAVES AND LOADS CUSTOM IMAGES INTO A DATASET

The next portion of the code evaluates the predicted values of the custom dataset and returns the accuracy of each model. A dataframe is also created that outlines the index of each custom image and which models failed to evaluate the image at that index.



```python
failed_indices = []
for model,index in zip(trained_models,range(0,len
(trained_models))):
    predictions = models.evaluate_model(model,test_dataset,
    show="none")
    failed_mask = ([test_dataset[1] != [np.argmax(row) for
    row in predictions]][0])
    print(f"{const.MODELS[index]} Accuracy: "+str(
    (~failed_mask).astype(int).sum()/len(failed_mask)))
    failed_indices.append([const.MODELS[index],*failed_mask])
df = pd.DataFrame(failed_indices)
df.rename(columns={0:"Model"})
```

Figure 4 Evaluates accuracy against custom dataset



```python
sorted_failures = df.iloc[:,1:].astype(int).sum().sort_values
(ascending=False)
for index,times_failed in sorted_failures.iteritems():
  if (times_failed >3):
    plt.imshow(test_dataset[0][index-1])
    plt.xlabel(f"Failed {times_failed} times. Index {index-1}
    . Actual: {const.CLASS_NAMES[test_dataset[1][index-1]]}")
    plt.show()
```

Figure 5 Identifies which images failed the most times

Figure 5 takes the generated dataframe and plots the images that failed greater than 3 times.