The Classified Project
AA_04_12
By: James Perdue

Table of Contents

Motivation

First off, this project is called the classified project because it will be focusing on using classification algorithms in Python. This task will be accomplished by using techniques that are within the scope of data science. The field of data science has a straightforward appeal. For instance, the potential ability for computers to analyze large sets of data is fascinating. For quite some time now, computers have conquered the term calculator, which was once used to describe number-crunching people. Now, people can use computers for almost every computation. Peoples' interest in data and what can be done with data is a big contributing factor to this project. Another reason for the motivation behind this project is that there are some people that are close to the author who have messed around with machine learning in the past, so their interest in the subject has also played a part in the motivation for this project. Also, these people that have been working on projects similar to this one will be able to assist with most questions that come up as the project progresses, which will hopefully make up for the fact that only one person will be working on this project. Another reason for the motivation of this project is that data science seems to be a great field to get into. Many companies hire data analysts to come calculate trends that affect bottom lines. Often times, these data scientists are able to find problems or things to improve on from the data that they feed to the computer. Being able to use data to find a bigger picture is an amazing thing and will be a valuable skill to learn for when looking for a job in the future. This project will serve as test run to see if data-based programming is enjoyable. Hopefully, this project will be able to give an idea of what type of programming would be interesting to pursue in the future. If the programming in this project is not enjoyable, then another field of programming will open its doors. Data science seems like it will be worth getting into for now, so its doors appear to be wide open.

Objectives

Like what was said previously in the motivation section, this project will be using classification algorithms to sort a data set. The goal of this project will be to input a data set into a Python program, have the program recognize the data, sort the data into categories, and output the result. Algorithms will be used in order to recognize the data and sort the data. The program will be able to take in pictures of M&Ms, recognize the M&M pictures, and sort them into color categories. The algorithms used for this would be an algorithm to recognize the M&M, an algorithm to recognize the M&M's color, and the algorithm to sort the pictures based on the colors. In order to accomplish this problem, the program will need to be able to distinguish the colors in the picture, the shapes in the picture, and the way in which picture can be classified using the similar colors and shapes in the picture. Either the picture will not be recognized as an M&M, or the picture will be recognized as an M&M that has a distinctive color. The result will be outputted once the machine is able to analyze these factors. Since there is two months to implement this idea into code, the ways in which the data will be sorted will not be very complicated (color and shape). Now that the idea for this project has been solidified, there will be time needed in order to research methods to accomplish this project idea. After that, there will be prototypes built to solve parts of the main problem, which will eventually combine into accomplishing the entire classification problem. Finally, the finished product will be optimized so that the algorithms will be efficient, and the final product can be viewed in all of its glory. If the project goes well, it will be able to follow these steps and become a project worthy of praise. This project is meant to be fun, while also challenging the programming knowledge that the maker has. Hopefully, this project will allow the creator and viewers to pursue a greater understanding of data manipulation using algorithms in programming.

Problem Formulation

This project will be done in Python, so everything that will be used has to be usable in that language. Currently, this project is looking at the Tensorflow API with the Keras interface. These libraries seem to be the best way to accomplish the goals of this project, so the current objective is to understand them. By understanding these libraries, the project will be able to implement algorithms that can recognize the shapes of the M&M pictures and their distinct colors. The layout of the code should start with the file i/o reader, which will take in the pixels of the picture file. The next part should establish a base case for what the program is looking for in each picture in order to be able to classify each picture by color and shape. The output should tell the user whether the input was an M&M or not, and the color of that M&M if it is one. That is the rough idea of what the project will look like, but it will most likely change throughout working on the project.

Now, the project has been completed. The problems with trying to accomplish the original goal of this project became obvious once code came into the picture. When trying to feed data into a neural network, it is crucial to have all of that data in the same format. This means, in the case of this project, that the pictures had to all be the same amount of pixels, the same dimensions, and colored similarly. Because of this, the M&M idea became close to impossible to accomplish given the scope of this project. So, the data that was used in this project was public data that data scientists made to be put into neural networks like the one this project creates. The data set that was used for this project is called the cifar10 data set. It can be found through this url(https://www.cs.toronto.edu/~kriz/cifar.html) which will be included in the reference section of this project's document. The cifar10 has ten different classes of pictures. The classes are airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The data set is split into a training group of pictures and labels, and a testing group of pictures and labels. It is significant to split the data set into training and testing data sets because the model needs to be taught a reference to classify the pictures that tested against it. This process is similar to how classes work for students at school. First the student needs to be given a reference, then they will be tested on similar material. This is oversimplifying what is actually happening, but it gets the general idea across.

The main algorithms used in this project relate to the model that is created. Machine learning starts with a model which can be though as the brains of the operation. This model is then given rules which are called layers which it uses to analyze and understand the dataset. The initial creation of a model in this project establishes three rules. This is shown below:

SOURCE CODE

```
model = tf.keras.Sequential([  tf.keras.layers.Flatten(input_shape=(32, 32,3)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

The first layer is used to simplify the data so that the computer can consume the information easier. The dense functions are then used to give the neural network the way in which it is supposed to perceive the data. The accuracy that was accomplished for this project was close to fifty percent, which is obviously not great, but the project creator ran out of time to mess with it. The ways to improve the accuracy of the neural network is to add more layers, convert the pictures to grayscale, or other methods that were not found. Fifty percent is definitely not great, but if more time was poured into this project then that would definitely be changed for the better.

Once the model is established, the training can begin:

SOURCE CODE

```
model.fit(train_images,train_labels, epochs=20) #Fits the model to the training data. Start of learning
```

```
Train on 50000 samples
Epoch 1/20
50000/50000 [==============================] - 33s 661us/sample - loss: 1.8684 - accuracy: 0.3233
Epoch 2/20
50000/50000 [==============================] - 16s 322us/sample - loss: 1.6785 - accuracy: 0.3982
Epoch 3/20
50000/50000 [==============================] - 16s 316us/sample - loss: 1.5977 - accuracy: 0.4278
Epoch 4/20
50000/50000 [==============================] - 17s 337us/sample - loss: 1.5488 - accuracy: 0.4455
Epoch 5/20
50000/50000 [==============================] - 17s 348us/sample - loss: 1.5136 - accuracy: 0.4570
Epoch 6/20
50000/50000 [==============================] - 18s 365us/sample - loss: 1.4879 - accuracy: 0.4666
Epoch 7/20
50000/50000 [==============================] - 17s 338us/sample - loss: 1.4711 - accuracy: 0.4752
Epoch 8/20
50000/50000 [==============================] - 16s 328us/sample - loss: 1.4502 - accuracy: 0.4817
Epoch 9/20
50000/50000 [==============================] - 17s 332us/sample - loss: 1.4331 - accuracy: 0.4878
```

```
50000/50000 [==============================] - 17s 333us/sample - loss: 1.3912 - accuracy: 0.5033
Epoch 13/20
50000/50000 [==============================] - 17s 332us/sample - loss: 1.3853 - accuracy: 0.5028
Epoch 14/20
50000/50000 [==============================] - 17s 337us/sample - loss: 1.3781 - accuracy: 0.5077
Epoch 15/20
50000/50000 [==============================] - 18s 356us/sample - loss: 1.3661 - accuracy: 0.5108
Epoch 16/20
50000/50000 [==============================] - 17s 340us/sample - loss: 1.3560 - accuracy: 0.5152
Epoch 17/20
50000/50000 [==============================] - 15s 307us/sample - loss: 1.3473 - accuracy: 0.5162
Epoch 18/20
50000/50000 [==============================] - 15s 303us/sample - loss: 1.3425 - accuracy: 0.5180
Epoch 19/20
50000/50000 [==============================] - 15s 302us/sample - loss: 1.3376 - accuracy: 0.5213
Epoch 20/20
50000/50000 [==============================] - 15s 303us/sample - loss: 1.3317 - accuracy: 0.5246
```

As you can see, the accuracy of the computer gets better with each new iteration of training. The accuracy ended at .5246 which translates to 52.46%. This means that the computer was able to predict the label of the training pictures about half of the time after training twenty times.

Then, the model is tested against the test data set once. This test data set still adheres to the same rules as the training set, but it is different.

SOURCE CODE

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2) #Seeing the models performance on the test dataset


print('\nTest accuracy:', test_acc) # Will be worse accuracy than the training data set, which means the model is overfitted so it performs worse on new information than its old information
```

OUTPUT

```
10000/10000 - 1s - loss: 1.4447 - accuracy: 0.4873

Test accuracy: 0.4873
```

As you can see, the neural network was less accurate when testing the test data set. That means that the model could be overfitted, meaning that the computer has gotten used to the training info enough that it is drawing conclusions that would normally not be reached without repetition. A little bit of overfitting is okay though, so it is good that the test accuracy is hovering around 50% as well.

There are other parts to explain, but the creator is running out of time.

Implementation

The project was accomplished using an IDE called Jupyter Notebook where Python was used to write source code. The libraries used were tensorflow, keras (which is now included in the tensorflow import with tensorflow v2), matplotlib, and numpy. Numpy was used to structure the data, matplotlib was used to show the pictures, tensorflow and keras were both used to create the models and conduct machine learning.

Final Reflection

Overall, this project was quite enjoyable. The creator wishes that he did not procrastinate it as much so he could have done a better job, but the overall product is fine. Neural networks are important for the future of computers. In a world that is controlled by profits and losses, it is crucial to do anything that can give the upper hand to succeed. That is why the power of computers is being used to predict what moves companies should make. The data is out there and ready to be used. All that is needed in order to translate that data into usable inferences is a mind great enough to draw relevant conclusions. The power to predict the future is truly incredible. Machine learning does not inherently predict the future, but it can result in something that can look like it can.

References

- https://www.cs.toronto.edu/~kriz/cifar.html(Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.)