Final Project: Image Classification With A Neural Network

Student Information

Name / ID: Theodore Matthew Patawari da Cunha / 2440061253

Class Information

Class: L1BC
Lecturer Name: Ida Bagus Kerthyayana Manuaba

I. Project Introduction

Every student in the Binus International Computer Science class of 2020 is tasked to make something with their knowledge of the Python programming language as well as with things that they have learned independently for their final projects. Upon starting this project, I knew that I wanted to do something with machine learning. I have been interested in machine learning for quite some time. I believe that it has a huge future potential as it can be applied in almost any industry. Machine learning may, in some circumstances, be more effective and efficient compared to manual human labor.

Initially I wanted to create a text summarizer using Recurrent Neural Networks, however, I found the task to be way too advanced for my level and I could not find a suitable dataset to work with. Hence, I decided to change my project idea but still do something with neural networks and machine learning. So, I decided to create an image classifier using machine learning as well as an interface to interact better with that image classifier.

The files for my project can be found on my Github page:
https://github.com/TheodoredaCunha/ImageRecognition.git

II. Project Specification

Project Purpose:

      The purpose of this project is to create a program that can open a pre trained neural network and run tests on it. The loaded neural network can be retrained, evaluated, and tested as the user wills. If there happens to be any changes with the loaded neural network's weights, the user can choose to save the neural network as a completely new model.

      When making this project, I envisioned this project to be a program that can help people review and test neural networks with a clear interface.

Project Audience:

      This project can be used by anybody who needs to classify images automatically as well as people who need to run tests with neural networks in order to learn its basic workings.

Project Aim:

      To have a program that serves as a clear interface for users to do different things with an image classifying neural network.

Requirements:
- An pre-trained neural network
- A menu system that allows users to navigate themselves through the program
- A good image pre-processing functionality to improve the quality of webcam images
- A dataset (training data and testing data) to serve as a baseline for neural network's accuracy

III.   Solution Design

Overview

This program allows users to open a pre trained neural network that was trained to classify images. The user can then give their own data, in the form of a picture taken by the webcam, for the neural network to process. The neural network will then attempt to distinguish what kind of object is in the picture.

External Libraries used in this project:
- Tensorflow (for machine learning)
- Keras (for machine learning; to make the syntax simpler)
- NumPy (for array/list processing)
- Matplotlib (to display images)
- OpenCV (to process images)
- IO (to process images)
- IPython (to display JavaScript, since the webcam uses JavaScript)
- Google.colab.output (to deal with JavaScript)

Different Elements in the Program

1. The Dataset

For this project, I have decided to use a dataset called the fashion_mnist dataset. This is one of the datasets provided by the Keras machine learning library that I used to create the neural network for this project. (The images below are actually in black and white with a filtered color. Purple is black and yellowish lime is white).

This dataset contains 60000 training images of fashion attires. The dataset contains images of T-shirts, shirts, coats, bags, trousers, dresses, sneakers, sandals, ankle boots, and pullovers. Each image is a grayscale image made up of 784 pixels, 28 pixels in width and 28 pixels in height. Each image in the dataset will correspond to a training label. The label consists of a single number between zero to nine. Each number from zero to nine will represent a type of fashion attire according to the attire presented in its corresponding image. For instance, in the dataset, an image of an ankle boot will correspond to the number 9.

The overarching goal of the neural network is to learn the patterns that exist in the images of the dataset and attempt to classify any 28 by 28 pixels image to a label.

The dataset also consists of some more testing images and testing labels. They have the same concept as the training images and labels, but the testing images and labels will not be included in the set of data that the neural network will learn from. The goal of this is to make sure that the neural network can accurately predict a label for any image it has not come across before.

2. *The Neural Network:*

```
#making the model
moodel = Sequential()
moodel.add(Conv2D(32, kernel_size = (3,3), activation = "relu", input_shape = (28, 28, 1)))
moodel.add(Conv2D(64, (3, 3), activation='relu'))
moodel.add(MaxPooling2D(pool_size = (2,2)))
moodel.add(Dropout(0.50))
moodel.add(Flatten())
moodel.add(Dense(128, activation='relu'))
moodel.add(Dense(70, activation='relu'))
moodel.add(Dropout(0.5))
moodel.add(Dense(10, activation='softmax'))
moodel.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = ["accuracy"])

moodel.summary()
```

A neural network learns by accepting a series of training data as well as the examples of the expected output of those specific data. The neural network will then go over the provided data to look for patterns or correlations (Sanderson, 2017). In a way, the neural network is trying to figure out its own algorithm to solve a problem. A neural network is made up of layers and each layer consists of a number of neurons (Sanderson, 2017). A neuron holds a numerical value called an activation, which represents parts of our data (Sanderson, 2017). In image recognition for instance, each neuron in the first layer might represent one pixel (Sanderson, 2017). Each neuron in a layer is connected to all other neurons in the next layer (Sanderson, 2017). The connection between neurons are called weights, which are numerical values that determine how crucial a particular neuron is in determining the correct output (Sanderson, 2017). Throughout training, the neural network will continuously adjust the weights to accurately predict the labels (Sanderson, 2017).

Before the neural network for this project can be saved into a file, it must be built, compiled, and trained on the fashion_mnist dataset. The neural network used for image classification is a special type of neural network called a convolutional neural network. Convolutional neural networks are the most effective neural networks when dealing with spatial data (Saha, 2018)**.** Images are spatial data. It would be more difficult for the neural network to learn if it sees each pixel in an image as a piece of data independent of the surrounding pixels. In reality, the significance of a single pixel in an image is determined by its surrounding pixels.

The Type of Layers in the Neural Network For This Project:
1. Convolutional 2D Layer

The Convolutional Network has a kernel size of three by three pixels. This means that when the image passes through this layer, there will be a three by three window that goes through the image and filters out any unnecessary pixels (Saha, 2018). The output of this layer will be a smaller array of pixels (Saha, 2018).
2. MaxPooling2D

Just like the kernels in a Convolutional 2D Layer, a MaxPooling2D layer will act like a window that goes through chunks of the input data and only keeps the maximum (most computationally significant) value (Saha, 2018). The output will also be a smaller array of the pixels (Saha, 2018).
3. Dropout

The Dropout layer gets rid of random data points in the input in order to make sure that the neural network does not overfit (Dropout layer, n.d.). Overfitting refers to a phenomenon where the neural networks learns the training data "too much", to the point

where it can only accurately predict values for the training data and not any new data (Brownlee, 2016). My analogy for overfitting is like when someone studies for a mathematics exam but only memorizes the answers from previous maths exercises instead of trying to actually comprehend the topics. When a new maths question is presented, that person might not know how to answer the question.

4. Flatten

As the name suggests, the Flatten layer simply turns any multidimensional array to a one long single-dimensional array. The reason this is necessary is because the next layer is a Dense layer, which accepts a single-dimensional array as input

5. Dense

The Dense layer is a straightforward layer. It simply accepts an input, processes the input according to the activation and the weights of the neurons, and provides an output. In this project, the output of this layer (which is also the final output of the neural network) is a single array with a length of ten.

As stated earlier in this report, the labels provided by the fashion_mnist data takes the form of an integer between zero and nine. However, since this is a classification machine learning problem, it serves the learning neural network well if the labels take the form such as the following (array index starts from zero):
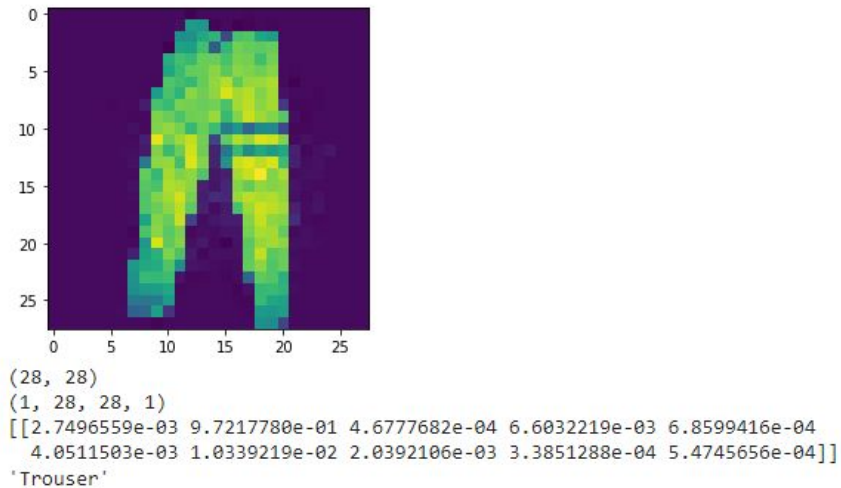
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.] for label $= 9$
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.] for label $= 0$
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.] for label $= 7$
Etc.

This type of format makes classification much easier. There are ten floats inside the array. Each float in the array signifies the probability for the correct label to be the index of the number in the array.

```
(28, 28)
(1, 28, 28, 1)
[[2.7496559e-03 9.7217780e-01 4.6777682e-04 6.6032219e-03 6.8599416e-04
  4.0511503e-03 1.0339219e-02 2.0392106e-03 3.3851288e-04 5.4745656e-04]]
'Trouser'
```

This is an image of a trouser that I took with my webcam (as I was testing the trained neural network with a new, user taken photo). These are the names of the possible attire names that the neural network could guess stored in a list called "class_names":

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

The neural network, when given the image of the trouser, outputted the following:

```
[[2.7496559e-03 9.7217780e-01 4.6777682e-04 6.6032219e-03 6.8599416e-04
  4.0511503e-03 1.0339219e-02 2.0392106e-03 3.3851288e-04 5.4745656e-04]]
```

Another algorithm was used to find the index number of the highest value in the output array and get the string value of the attire name, from the variable "class_names", according to that index. In this case, the number with the highest value in the output array of 10 floats is 9.7217780e-01 (the most probable answer to be correct), which is in index one of the array (since the index of the arrays/lists in Python starts with zero). So, all that is left to do is to get the value of index one from the array "class_names", which is "Trouser". This is a good sign as it indicates that the neural network is working properly.

Finally, the model can be saved in a Google Drive folder. This can be done easily as Keras has a built in function for this. This can be done with the ".save()" method that the model has.

```
moodel.save("/content/gdrive/MyDrive/ImageRecognition/model/")
```

3. The Main Program

The main program is made in a separate Google Colab file. The main program has a pretty linear flow. Once the user loads the program, it will prompt the user to enter in a directory name to which the pre trained neural network model is saved in. The model will then be loaded from the directory (the directory refers to a Google Drive path).

```
Model filename/directory... ImageRecognition/model/
```

After the model is loaded, the model summary will be printed out and the model will be evaluated against the provided fashion_mnist testing data. After the model is evaluated, the program will print out the neural network's loss and accuracy.

```
Model: "sequential_3"
_____
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_____
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496
_____
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
_____
dropout_2 (Dropout)          (None, 12, 12, 64)        0
_____
flatten_2 (Flatten)          (None, 9216)              0
_____
dense_4 (Dense)              (None, 128)               1179776
_____
dropout_3 (Dropout)          (None, 128)               0
_____
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_____
313/313 [==============================] - 7s 20ms/step - loss: 0.3768 - accuracy: 0.8668
The model has a loss of 0.3767894506454468
The model is 87% accurate
```

Now that the model is loaded, the user is prompted to either load some test data, give the neural network some user provided photos to predict, or to exit the program.

```
        Menu
 ==============================
 1: Get Test Data Examples
 2: Test with Your Own Data
 Anything Else: Exit
 1,2, or anything else? [_____]
```

1. Load Some Test Data

Once the user chooses to load some test data, the program will print out the list of all the attire names that the model can recognize. The user must then enter in the name of the attire to be searched for as well as the amount of images the program should show. Once all of the required parameters are given, the program will loop through the test data and show the image of the searched attire in the Google Colab console as many times as the user indicated earlier.
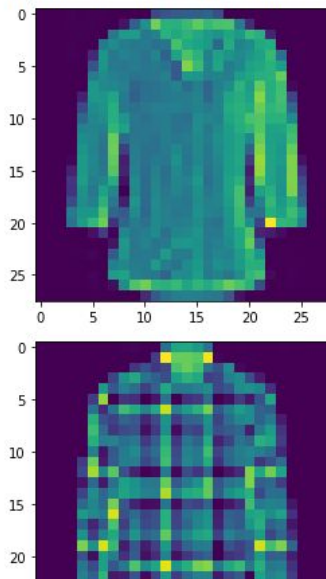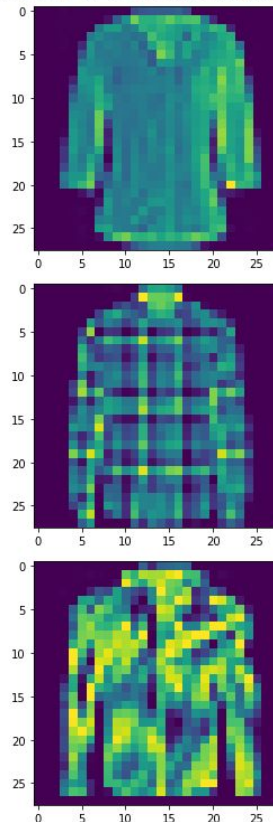
2. Give the Neural Network Some User Provided Photos to Predict

If the user decides that they wanted to provide their own test examples, they can do that. The program will run a piece of JavaScript code that creates a capture button and a webcam input.


Once the user presses the capture button, the image will be captured from the webcam and processed to fit into the neural network. The preprocessing for the neural network is as follows:

1. Get the string representation of the image
2. Convert the string into a numerical representation
3. Resize the image to fit the 28x28 pixel requirement
4. Turn the image into grayscale
5. Clean up any noise, by turning darker pixels to black
6. Reshape the data to fit the neural network's needs

The neural network accepts an image with the shape (1, 28, 28, 1). To break that down, the numerical representation of that image must be stored in one large array. Inside that array, there must be 28 other arrays. Each of this array will represent the pixels in each of the 28 rows of pixels. Inside each of the first 28 arrays, there are 28 other elements, each representing one pixel. These elements take the form of another array with an integer inside it.

```
TAKING PHOTO
==========================================
```
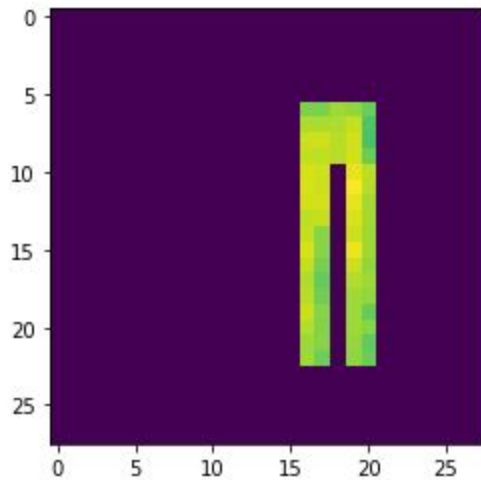


```
(1, 28, 28, 1)
WARNING:tensorflow:5 out of the last 111 calls to <function Model.make_prec
[[7.1672176e-04 4.2561099e-01 1.2925276e-04 2.6937472e-03 3.5863832e-04
  3.7212804e-01 6.9503405e-04 1.9431973e-01 1.1381671e-03 2.2096364e-03]]
Trouser
```

       The neural network will then attempt to give the right prediction. The user will then be asked whether or not the prediction was right. If it is right it will not do any further process and will ask the user whether or not the user wants to take another picture.

       If the prediction is wrong (which might happen a lot because the model itself is only about 87% accurate and the user provided image will not be as clean as the fashion_mnist provided images), the user is given the option to train the model further. The model will be trained from the fashion_mnist data again and the user gets to pick how much epoch will be necessary and how much of the data is used to retrain the model. After training concludes, the user will be asked whether or not they want to take another picture
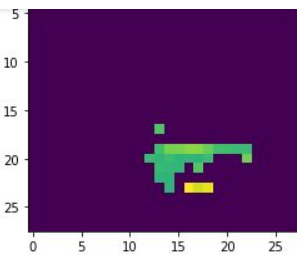

```
(1, 28, 28, 1)
WARNING:tensorflow:6 out of the last 112 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7fa537c6
[[0.00334546 0.0183134  0.00057686 0.01252269 0.11075057 0.29854372
  0.00779487 0.00219056 0.26163834 0.28432354]]
Sandal
Is the result accurate? 1 = yes, anything else = no: no
Should the model be retrained? 1 = yes, anything else = no: 1
epochs: 1
number of data: 200
7/7 [==============================] - 7s 1s/step - loss: 0.4766 - accuracy: 0.8300 - val_loss: 0.3767 - val_accuracy: 0.8666
```

If the user chooses that they do not want to take anymore pictures and the model has been retrained, the user will be asked whether or not the model should be saved. If so, the user must provide a path for the directory to which the model can be saved.

```
Should the model be saved? 1 = yes, anything else = no: 1
New model name or directory: ImageRecognition/model3/
INFO:tensorflow:Assets written to: /content/gdrive/MyDrive/ImageRecognition/model3/assets
Goodbye...
```

My Drive > ImageRecognition > model3 ▾

Name ↓

📁 variables

📁 assets

📄 saved_model.pb

Bibliography

Sanderson, G. [3Blue1Brown]. (2017, October 5). *But what is a Neural Network? | Deep Learning, Chapter 1* [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk&t=0s&ab_channel=3Blue1Brown

Saha, S. (2018, December 16). *A Comprehensive Guide to Convolutional Neural Networks -- the ELI5 way.* Towards Data Science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks -the-eli5-way-3bd2b1164a53#:~:text=A%20Convolutional%20Neural%20Network%20(ConvNet,differentiate%20one%20from%20the%20other.

Dropout layer. (n.d). Keras. Retrieved January 10, 2021, from https://keras.io/api/layers/regularization_layers/dropout/

Brownlee, J. (2016, March 21). *Overfitting and Underfitting With Machine Learning Algorithms*. Machine Learning Mastery. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/