Project Report

Image Recognition with Pattern Recognition

Isingizwe Didier Frank 2019280803

Course: Advanced Machine Learning

January, 2020

Abstract

There exists many applications for image recognition. One of the largest that people are most familiar with would be facial recognition, which is the art of matching faces in pictures to their identities. Image recognition goes much further, however. It can allow computers to translate written text on paper into digital text, it can help the field of machine vision, where robots and other devices can recognize people and objects. In my project, the target is to begin to use machine learning, in the form of pattern recognition, to teach my program what text looks like. In this case, we'll use numbers, but this could translate to all letters of the alphabet, words, faces, really anything at all. The more complex the image, the more complex the code will need to become. When it comes to letters and characters, it is relatively simplistic.

Keywords: machine learning. machine vision. pattern recognition. image recognition.

Contents

Chapter 1

Introduction

Image recognition is referred to us the ability to recognize certain people, animals, objects or other targeted subjects through the use of algorithms and machine learning concepts. The term "image recognition" is linked to "computer vision," which is an overarching label for the process of training computers to "see" like humans, and "image processing," which is a catch-all term for computers doing intensive work on image data, however pattern recognition is the automated recognition of patterns and regularities found in data.

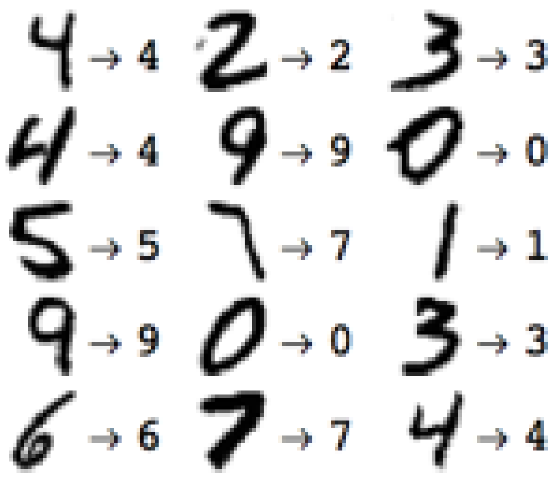


Figure 1.1: number recognition

Pattern recognition is used to give human recognition intelligence to machine which is required in image processing. Pattern recognition is used to extract meaningful features from given image/video samples and is used in computer vision for various applications like biological, biomedical imaging or plate number recognition. In my project, i am going to explore using the same technique for detecting number in an image. The report is organized as follows. In section 2 we are exploring pixel arrays whereby in section 3, we are graphing our images in Matplotlib for visualization purposes. In section 4, we are using a thresholding logic approach whereby in section 5, we are saving data for training and testing.

Image recognition goes a lot further, in any case. It can enable PCs to translate composed content on paper into computerized content, it can help the field of machine vision, where robots and different gadgets can recognize people and objects.

In my case, I'll use numbers, but this could translate to all letters of the alphabet, words, faces etc. . .

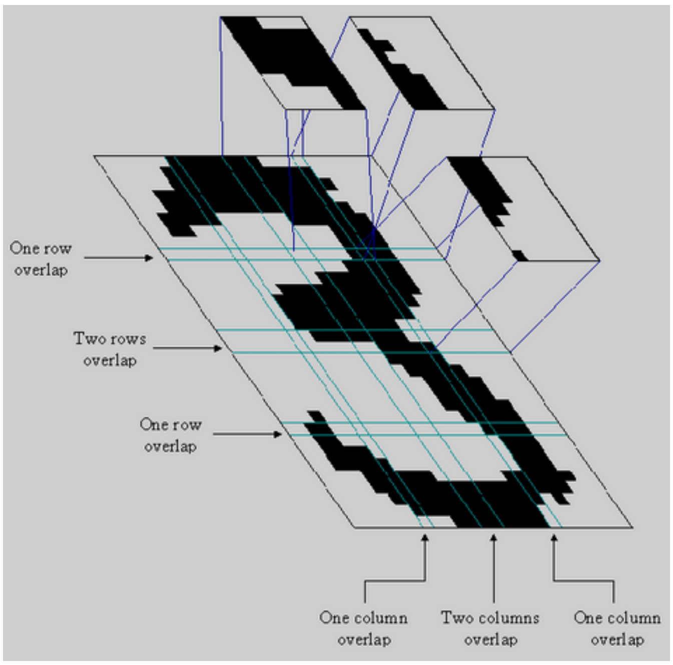


Figure 1.2: Patterns in an image

1.1 Motivation: Plate number recognition

Automatic number-plate recognition is a technology that uses optical character recognition on images to read vehicle registration plates to create vehicle location data. It can use existing closed-circuit television, road-rule enforcement cameras, or cameras specifically designed for the task.

1.2 Approach:Using 4 stages of machine learning

The integration of Machine Learning in our daily businesses and tasks offers a host of benefits: higher productivity, actionable data, and more. Some companies may be able to do this quickly. I chose to take a staged approach. Here are some stages that i will be following to achieve a great accuracy rate.

[Stage 1: Collect and prepare data]
[Stage 2: Make sense of data]
[Stage 3: Use data to answer questions]
[Stage 4: Create predictive applications]

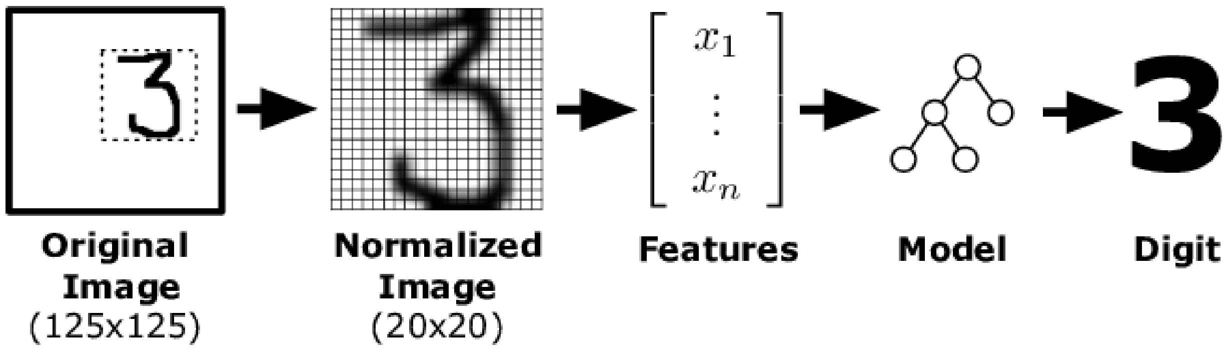


Figure 1.3: Stages of my experiment

1.3 Exploring pixel arrays

In programmer's daily life, the initial stage of solving a problem is pre-processing stage which means collecting data and structuring them for it to fit given set of instructions written in any programming language. Now we're ready to dig into what makes an image in numbers. For this, we use PIL or Pillow, python libraries for images. We are then just importing numpy as np. Numpy is used for number-crunching. we're exploring a 256-color image, since programming starts with a 0 rather than a 1. This color means 255 red, 255 green, 255 blue, and then 255 Alpha. Alpha is a measure of how opaque an image is. The higher the number, the more solid the color is, the lower the number, the more transparent it is. Each pixel is measured in RBGA, so an example row is [255, 255, 255, 255].

1.4 Graphing our images in Matplotlib

In this section,for visualization purposes; i am going to explore how to see the image array visually as well manipulating the images. For this, I will be using Matplotlib (Python library for plotting graphs etc. . .).

1.5 Thresholding logic approach

The idea of thresholding is to simplify the image. Some people particularly like the visual effect as well, but we're interested in the simplifying aspect. An issue arises when we're trying to identify characters, shapes, objects, whatever, because there is a massive list of colors. Anything complex, to be analyzed, needs to be broken down to the most basic parts. With thresholding, we can look at an image, analyze the "average" color, and turn that "average" into the threshold between white or black. Some thresholding wont go all of the way to either full black or white, there will be some gradient, but, for our basic purposes, we want to go all of the way! Next step is to create a function that will take the images we feed it, and threshold it. The way i am going to do this is by taking the "average" color value, and then thresholding any pixel as black if it is any darker or white if it is lighter.

1.6 Saving data for training and testing

Next up, i am going to start using given test example numbers data downloaded from Kaggle. My purpose is to create image arrays out of our numbers data, saving them, so that we can reference them later for pattern recognition. Running the function Examplesfunc() should now create the numArrayEx.txt file and populate it with number arrays. With these, we can then take new numbers, threshold if necessary, then compare the current number array with our known number patterns, making an educated guess on what the number we're looking at is.

Chapter 2

Methodology

I will download datasets of image numbers from Kaggle.com. I will be using Python programming language version 2.7 with following dependencies as follow:

Matplotlib: It is a plotting library for python and it's numerical mathematics extension Numpy

Pillow/PIL: It is a python image library which support opening ,manipulating as well as saving many different image file formats.

Numpy: It is a library for python which adds a support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these given arrays.

2.1 Sequence of my project tasks

- 1st step: Graphing my images in Matplotlib library.
- 2nd step: Using Thresholding function and logic.
- 3rd step: Saving my data for training and testing.
- Last step: Testing and visualizing the outcome.

Chapter 3

Experimental Results

README: The source code must be run in Python 2. To run the file use the following command ”python TestingFile.py” or ”python2 TestingFile.py”.

Test image directory: ”images/test.png”

To test using different images: Go to images folder and add your image and modify the file name on the last line of the source code file named ”TestingFile.py”.

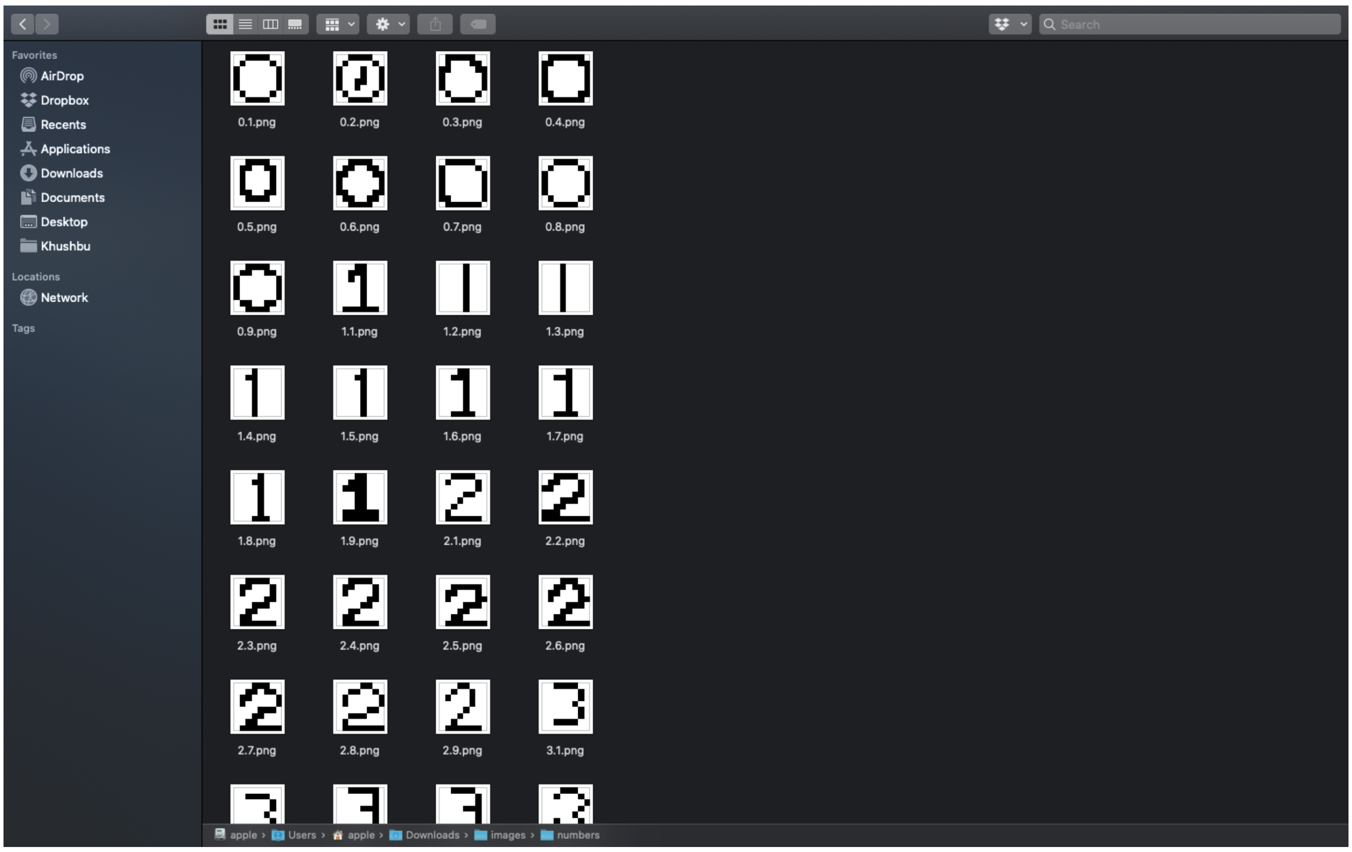
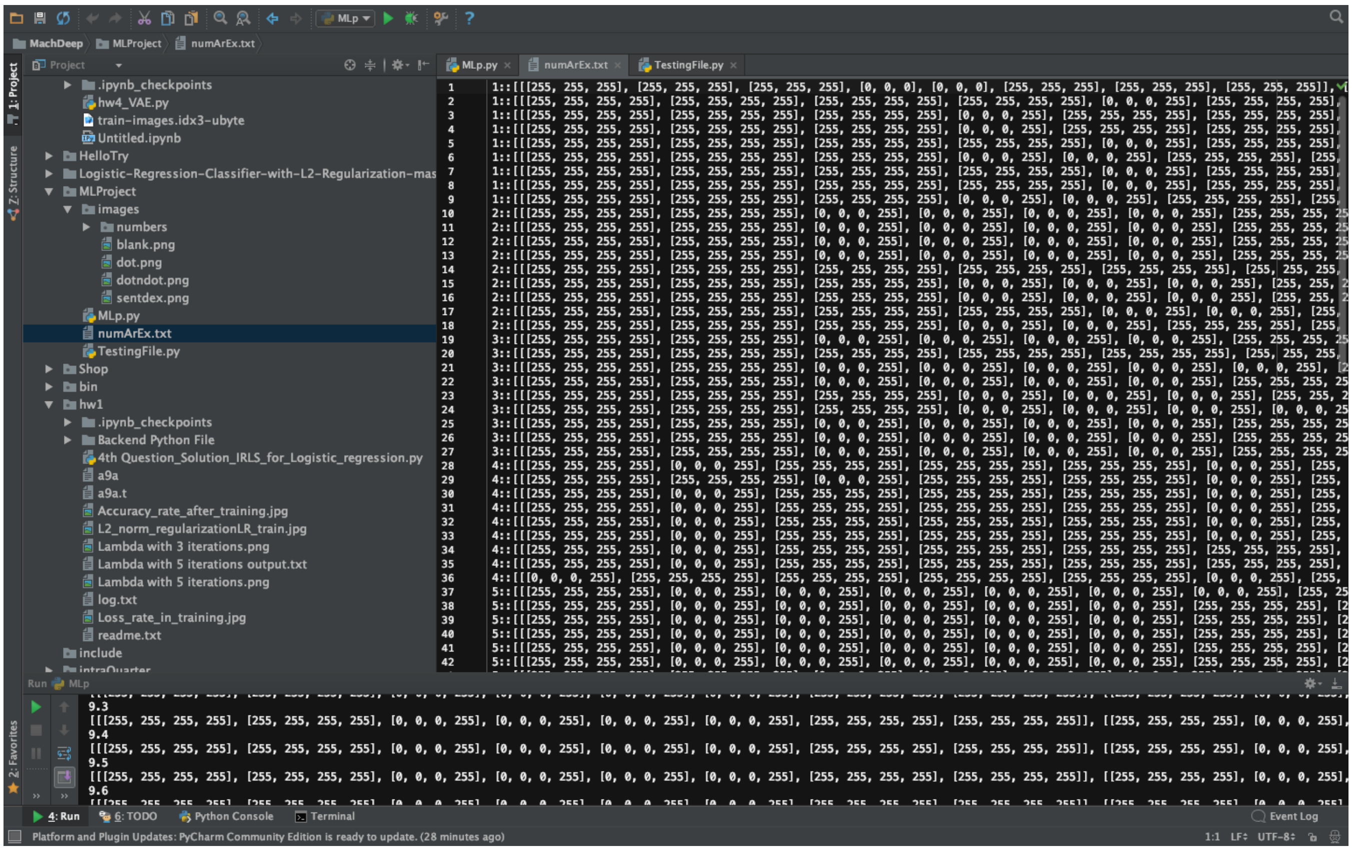


Figure 3.1: Datasets from Kaggle



```
1::[[[255, 255, 255], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255]],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255,
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255],
1::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255,
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255,
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 2
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 2
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255,
2::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255,
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255,
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [2
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 2
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 25
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
3::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 25
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255],
4::[[[255, 255, 255, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255],
4::[[[0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [255,
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 25
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [2
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [2
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [2
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [2
5::[[[255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [2
```

```
9.3
[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255]], [[255, 255, 255, 255], [0, 0, 0, 255],
9.4
[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255]], [[255, 255, 255, 255], [0, 0, 0, 255],
9.5
[[[255, 255, 255, 255], [255, 255, 255, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [0, 0, 0, 255], [255, 255, 255, 255], [255, 255, 255, 255]], [[255, 255, 255, 255], [0, 0, 0, 255],
9.6
```

Platform and Plugin Updates: PyCharm Community Edition is ready to update. (28 minutes ago)

Figure 3.2: NumArEx.txt Creating Number Examples

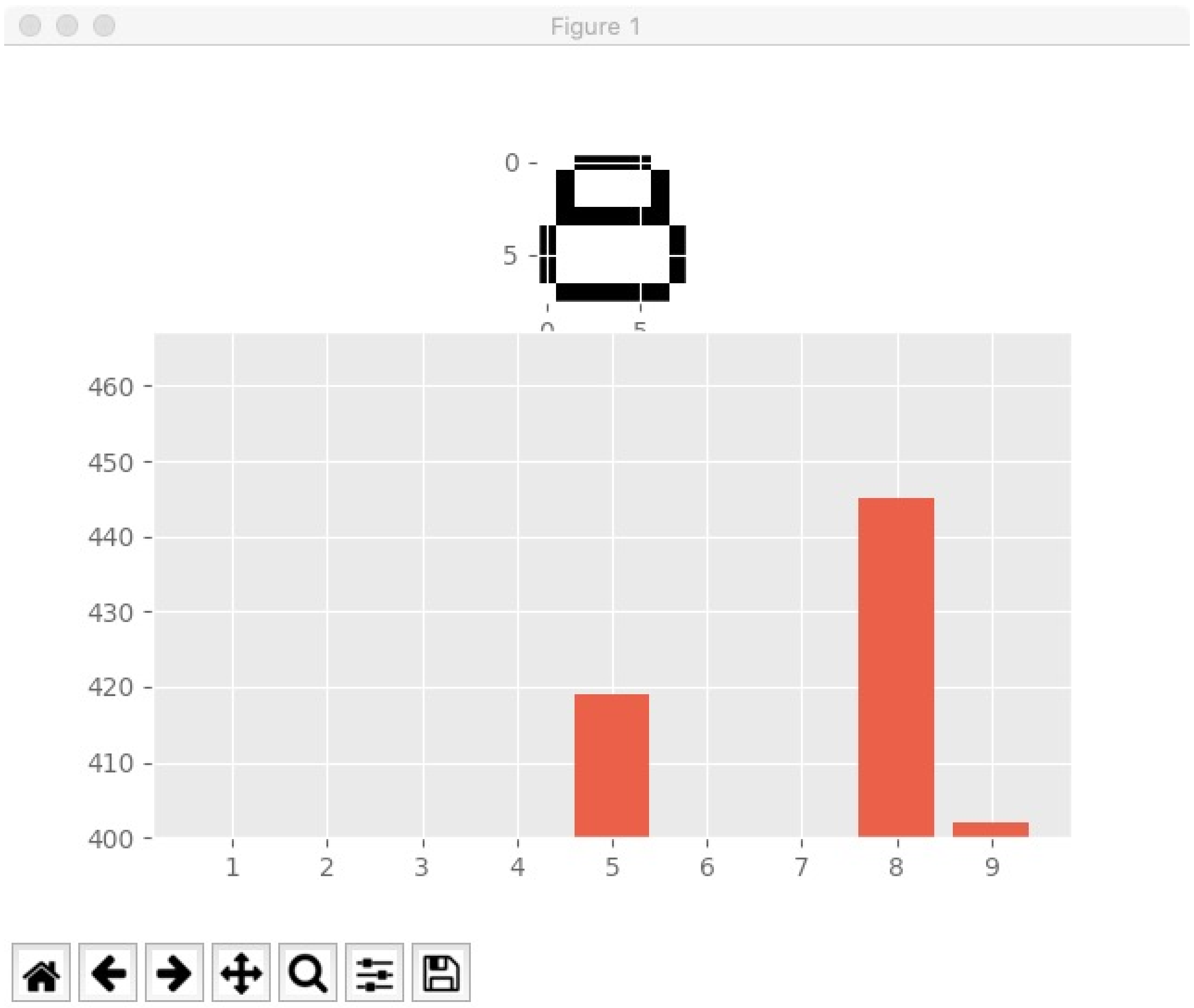


Figure 3.3: output result

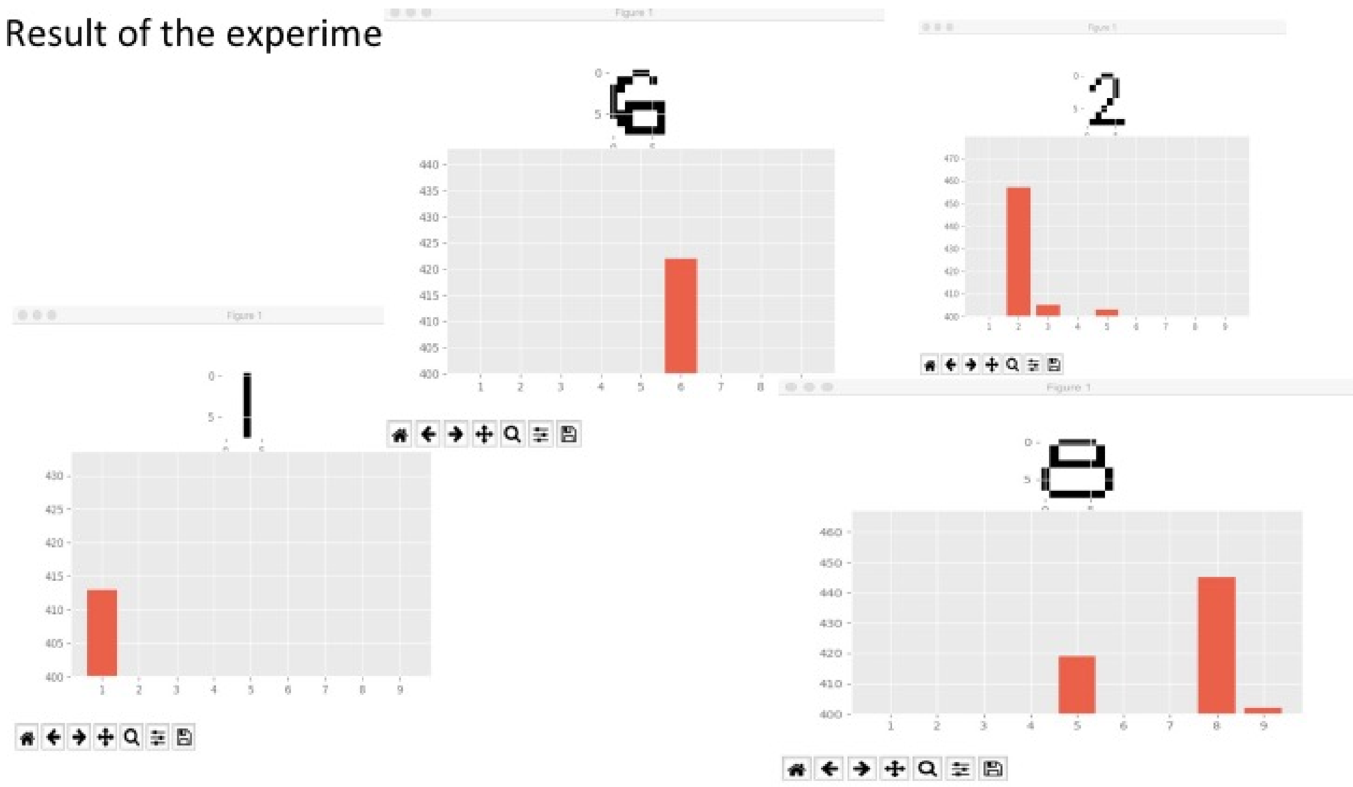


Figure 3.4: output result