Project Elective

Map Analytics - Midterm Report

Nilay Kamat, Vineet Priyedarshi

March 2024

Contents

Chapter 1

Extraction of relevant images

1.1 Introduction

In this project, we aim to automate the extraction of relevant choropleth map images from PDF research papers. Choropleth maps are commonly used in research to visualize data related to geographic regions. However, manually extracting these maps from PDF documents can be time-consuming and tedious. Our solution utilizes a combination of Python libraries, image processing techniques, and machine learning models to streamline this process.

1.2 Problem Statement

Given a PDF file containing research papers, the goal is to extract choropleth map images that are relevant and suitable for annotation and segmentation tasks. The challenges include identifying relevant maps among other images in the document, classifying maps as either colorbar or discrete, and selecting high-resolution and well-defined maps for further analysis.

1.3 Methodology

1.3.1 Data Extraction

We start by using Python library functions to extract all images from the input PDF document and store them in a designated folder. This step ensures that we have access to all the images contained within the research papers.

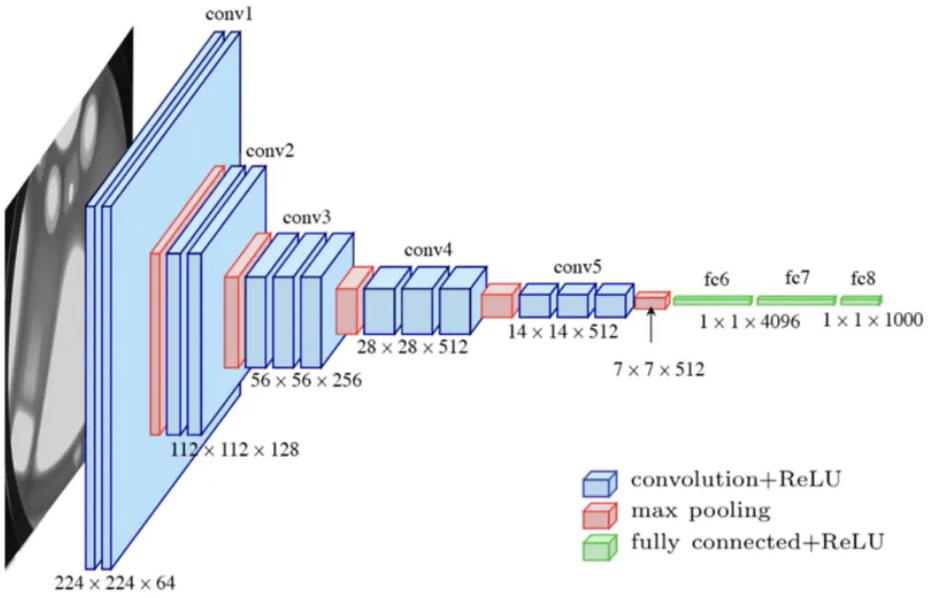
1.3.2 Image Classification

Next, we utilize a pre-trained object detection model from Roboflow to classify the extracted images into two categories: colorbar maps and discrete maps. For this project, we focus on processing discrete maps. For now we just focus on discrete map images. Images classified as discrete maps with a confidence level above 0.8 are selected and moved to a separate folder for further processing.

1.3.3 Map Selection

In the map selection phase, we aim to identify high-quality and relevant maps, specifically focusing on maps related to the United States. To achieve this, we employ a similarity metric approach.

1.3.4 vgg16 Image similarity



VGG16 is a powerful pretrained model that can be used for identifying similarities between images. By using this model, we can extract high-level features from different images and compare them to identify similarities. This technique has a wide range of applications, from image search and recommendation systems to security and surveillance.

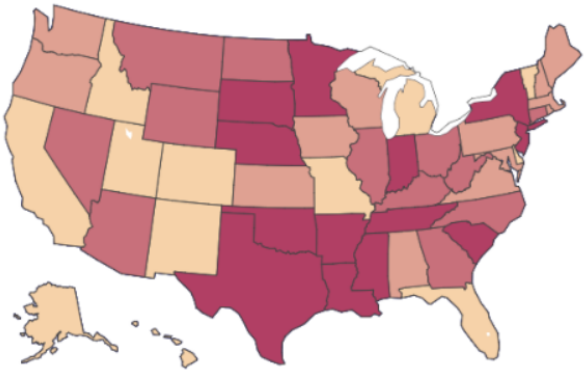
This function takes an image object as input and converts the image to a 3D array with the img_to_array method from the Keras image module.

The resulting array is then expanded to have an additional dimension using the np.expand_dims()

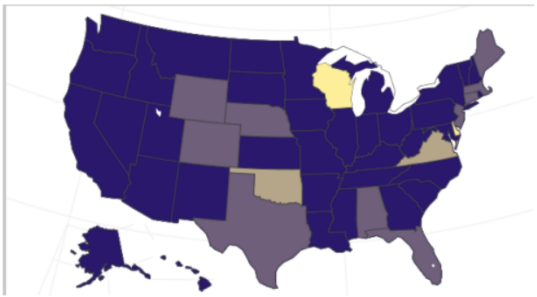
method, which is required for the VGG16 deep learning model input. The expanded array represents a single image with shape (1, height, width, channels), where height, width, and channels correspond to the dimensions of the image.

The function then calls the predict() method on the VGG16 model, which has been previously defined in the code. This method takes the expanded numpyarray as input and generates an embedding for the image using the pre-trained weights of the VGG16 model.

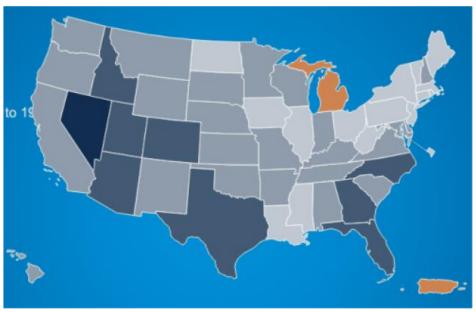
Finally, the function returns the image embedding as a numpyarray. This embedding can be used as a feature representation of the input image, which can be used for tasks such as image retrieval, similarity search, or classification



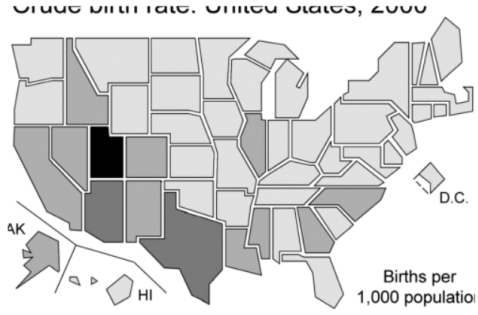
Below are the images and their similarity value (in squared brackets) with respect to the image above if taken as reference.



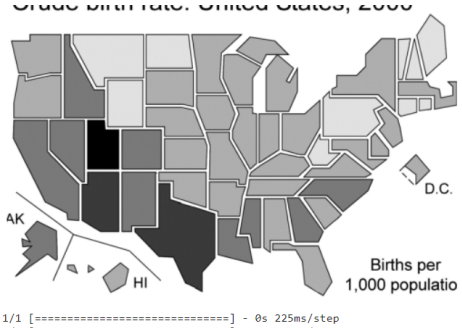
```
1/1 [==============================] - 0s 223ms/step
1/1 [==============================] - 0s 225ms/step
0.9524277448654175
[0.82067287]
```



```
1/1 [==============================] - 0s 217ms/step
1/1 [==============================] - 0s 207ms/step
0.9641153216362
[0.77171445]
```




```
1/1 [==============================] - 0s 217ms/step
1/1 [==============================] - 0s 217ms/step
0.929362416267395
[0.6519327]
```




```
1/1 [==============================] - 0s 225ms/step
1/1 [==============================] - 0s 218ms/step
0.9412789940834045
[0.72379935]
Predictions saved to predictions3.txt
```

1.4 Results and Evaluation

The automated extraction and selection process have been successful in identifying and isolating relevant choropleth maps from PDF research papers. The accuracy of map classification and selection depends on the performance of the object detection model and the chosen similarity metric.

1.4.1 Performance Metrics

We evaluate the performance of our system using the following metrics:

- Classification Accuracy: Percentage of correctly classified maps as discrete or colorbar.
- Similarity Threshold Effectiveness: Impact of varying similarity thresholds on the selection of high-quality US maps.

1.4.2 Limitations and Future Enhancements

Model Fine-Tuning: Fine tuning the similarity metric by creating a object classification model.

Chapter 2

Extraction of Numerical Data

2.1 Introduction

The task proposed in the project was to retrieve the numerical data present in each map image and evaluate it for the states in the USA. The extraction of the numerical data is being done with the help of OCR techniques.

2.2 Problem Statement

Given any map image containing certain data about the USA and represented using Choropleth or Isocline maps, our task is to extract the numerical data that was used to create the thematic map. The challenges include working with different types of data depictions - Colour Bar or Discrete Legends, leveraging OCR techniques to obtain decent results of the numerical data and to extract the data in a format such that an analysis can be made using the same.

2.3 Approach

2.3.1 TesseractOCR

Extending from the KerasOCR attempt that had been made previously, we utilised another model, "TesseractOCR", to perform the task of extracting the relevant numerical data. While this model did provide more accuracy in obtaining the numerical values, it still faltered in being highly accurate.

Example: A specific map image contained a colour bar having numerical values 31-36 at certain positions. TesseractOCR was able to identify certain values such as 31, 32, 33 and 36. But for the values 34 and 35, it reported them as 4 and 5 which would cause a large inaccuracy in our final analysis.

Due to this drawback, we decided to approach the problem by considering the following:

1. Enhancing the image using Super Resolution.
2. Masking the other contents present in the image except for the relevant numerical data region. We would leverage the annotation model from Roboflow used to obtain the bounding boxes to perform the masking.
3. File format issue? Check if there were certain types of file formats wherein the image failed to provide decent results.

2.3.2 Super Resolution(ESRGAN) followed by PaddleOCR

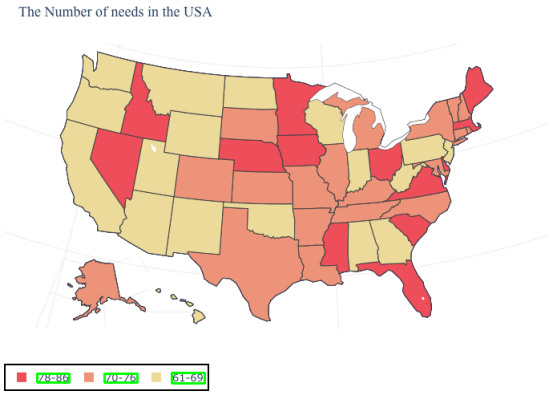
We decided to enhance the images provided using ESRGAN(Enhanced Super-Resolution Generative Adversarial Networks), a method for single image super-resolution that aims to generate realistic textures and details while enhancing the perceptual quality.

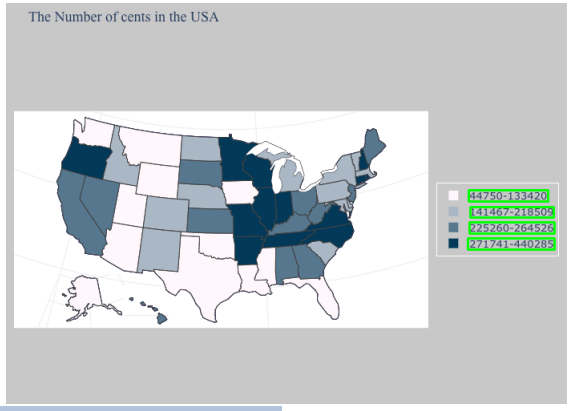
Followed by image enhancement using ESRGAN, we leveraged the PaddleOCR model to obtain OCR results from the map image on the relevant numerical data. Similar to the code used during the usage of the TesseractOCR model, we used a Jupyter Notebook/Python script to obtain output of the average value, the unit, the colour associated with the value and the map number(present in the image title). There were slight changes that had to be made in the code due to the change in the model. The values in the legends are given in the format (a−b) followed by some units. Hence, using some post-processing of the value detected using the OCR model, we stored the average value and the unit along with the colour corresponding to this obtained average value.

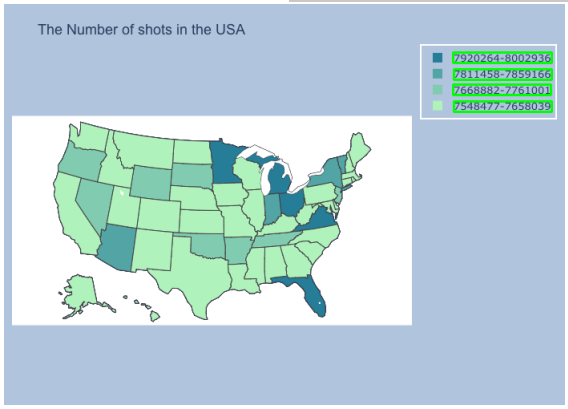
2.4 Results

We used a couple of map images from the MapQA dataset to verify the working of our script and to check if the output obtained aligned with the expectations and requirements. Images obtained on using PaddleOCR model after enhancing the images using ESRGAN were as follows:

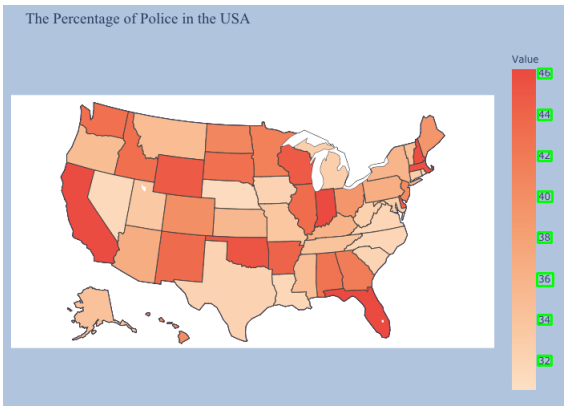
1. Discrete Legends

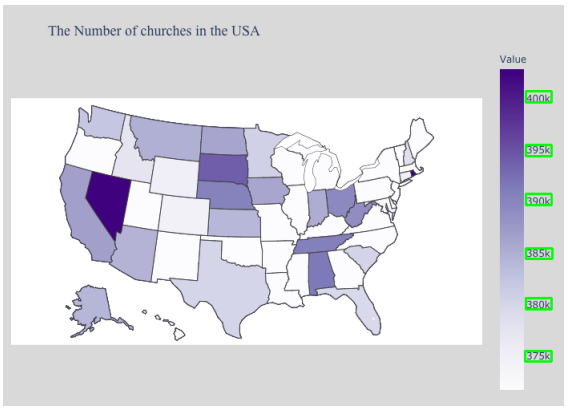






2. Colour Bar





Formatted output of 1 map image

1. ('map_7', 1, 84, 31, 63, 7.0, 'M')
2. ('map_7', 2, 118, 49, 78, 6.5, 'M')
3. ('map_7', 3, 148, 69, 90, 6.0, 'M')
4. ('map_7', 4, 173, 93, 100, 5.5, 'M')
5. ('map_7', 5, 194, 120, 112, 5.0, 'M')

6. ('map_7', 6, 207, 148, 126, 4.5, 'M')

7. ('map_7', 7, 218, 178, 149, 4.0, 'M')

8. ('map_7', 8, 228, 204, 176, 3.5, 'M')

Here, the output comprises of (mapName, ID, R, G, B, value, unit) where ID is just a number to store the number of detected values in the image and R,G,B represent the associated colour. The value stored here was simple due to it being a Colour Bar map, but in Discrete Legends, we evaluate the average of the 2 values mentioned as a−b and append it to the tuple accordingly. Such outputs were stored in an output file to be used for the Colour Matching/Association task of the project.

The codes and results can be found at the following repository.

Chapter 3

Colour Association/Matching

3.1 Objective

The task is to retrieve the numerical value corresponding to the states present in the USA using the output of the segmentation(which provides the colour representation of the state) and the extracted numerical data(using OCR techniques).

3.2 Approach

We will be using the output obtained from the modified segmentation task along with the extracted numerical data to perform an analysis on the map image. Our input sources provide information regarding the RGB value of colour of the states in the map and the extracted values present in the image along with their corresponding value. There are certain different approaches that we experimented with in obtaining the similarity between the RGB values provided in the inputs in order to make an association/matching of sorts to evaluate the value.

In order to make an association, the logic/work flow used was to obtain the state's(under observation) colour as given in the output of the segmentation task. Using this colour representation and the colour representations along with their values present in the extracted values(via OCR), an association/matching could be made using some form of similarity metric.

We coded a simple Python script to use the 2 input text files and make a similarity comparison for each state present in segmentation task output. Choosing the colour with the most similarity, we then assigned the colour's corresponding value to that state, writing it an output file that

shall serve as the output for the Colour Matching task.

3.3 Colour Similarity Metrics

3.3.1 Euclidean Distance

The most naive method of obtaining the similarity between 2 different colours is using the Euclidean distance between the 2 colours in the RGB space.

This approach, although gives an accurate representation of the distance between 2 colours in the RGB space is quite different from the actual real world perception of colours by the human eye. This approach will work in the case of discrete legends and choropleth maps as the colours present in the map and legends is exactly same or very closely similar. Hence, the RGB values will not be far apart enough to cause a substantial change. Hence, Euclidean distance can be used as a metric to evaluate the values present in some images. In order to understand for exactly which images to use Euclidean distance as a metric can be obtained from the "Image Classification" task.

3.3.2 Improved Euclidean Distance

A modified approach to the similarity metric by giving pre-defined weights to the RGB values. Here, the weights for each are as follows:

1. Red(R) = 0.3
2. Green(G) = 0.59
3. Blue(B) = 0.11

Although this does give a better estimate, it still is not able to distinguish between colours as per the human perception.

3.3.3 CIELAB colour space

CIELAB colour representation represents colours using (L*, a*, b*) where

1. L* = perceptual lightness
2. a* and b* for the four unique colors of human vision: red, green, blue and yellow.

We are yet to try this metric to compare colours but this format of colour representation is best suited for the Euclidean distance metric of evaluation of the similarity between colours.

3.4 Results

Currently, through the usage of the simple Euclidean distance metric, we obtained the output using a fabricated segmentation output.

Input provided(Segmentation)

1. ('State 1', 84, 31, 63)
2. ('State 2', 118, 49, 78)
3. ('State 3', 173, 93, 100)
4. ('State 4', 207, 148, 126)
5. ('State 5', 118, 49, 78)
6. ('State 6', 148, 69, 90)

Input provided(Extracted Numerical Data using OCR)

1. ('map_7', 1, 84, 31, 63, 7.0, 'M')
2. ('map_7', 2, 118, 49, 78, 6.5, 'M')
3. ('map_7', 3, 148, 69, 90, 6.0, 'M')
4. ('map_7', 4, 173, 93, 100, 5.5, 'M')
5. ('map_7', 5, 194, 120, 112, 5.0, 'M')
6. ('map_7', 6, 207, 148, 126, 4.5, 'M')
7. ('map_7', 7, 218, 178, 149, 4.0, 'M')
8. ('map_7', 8, 228, 204, 176, 3.5, 'M')

Output(Colour Matching Analysis)

1. 'State 1': 7.0, 'M'
2. 'State 2': 6.5, 'M'

3. 'State 3': 5.5, 'M'
4. 'State 4': 4.5, 'M'
5. 'State 5': 6.5, 'M'
6. 'State 6': 6.0, 'M'

As can be seen from the output obtained, the colour matching is giving results as expected. The only issue here is that the similarity metrics used is not an accurate metric. Another issue is that this would only work accurately with map images containing Discrete Legends.

3.5 Future Plan

Work on using CIELAB colour space to establish an accurate similarity measure for colour matching. Pertaining to map images containing Colour Bar, implementation of Piecewise Linear Interpolation should give better results.