Using Principal Component Analysis for Optical Character Recognition

C. Colizzi & C. Ramiro

November 16, 2020

1. Summary

The following report analyses the creation of a computer software capable of recognising characters from pictures and translating them into strings. The code was written to address and solve any problems that frequent travellers may have when travelling to other countries, or any issues that someone may have with translating non-digital text. Previously, in order to translate text, one would have to manually transcribe it into an app, which often became tedious and time-intensive. Using Principal Component Analysis, our system provides a viable alternative to this by automatically detecting text in images. Currently, our system has achieved only 79% accuracy, but this could easily be improved with further iterative development of the algorithm through linear regression and image segmentation.

2. Introduction

Communication is at the base of all things we do in life. When we are in our home country, we almost take this for granted, but when we travel abroad this is rarely the case. Most people that have travelled to another country, or even another region, have found themselves in front of a sign, text or something else that looked completely alien to them. This is why translators have been created, to easily make something we don't understand suddenly understandable. This has worked very well in the past, but with the passing of the years, this concept has struggled to keep up with technological advancements. It quickly became impractical to copy down hundreds of words at a time to just make sense of what is on the menu of a specific restaurant. Therefore, the need for some kind of image-to-translated text software rose very quickly. We have decided to create an algorithm capable of identifying the text inside of an image, "preparing it" to be translated by another algorithm. There are no social issues associated with the use of this algorithm, yet some technological ones might be encountered in its correct implementation.

The algorithm works by using the "nearest neighbour" approach, where an image is analysed and compared to a given dataset to best determine what the most similar image, or the "nearest neighbour," is. This is done using PCA, or principal component analysis, to take each image and convert it into a vector, then taking the x principal eigenvectors to collapse the set into a fewer number of dimensions, and finally "projecting" our source image into this imaginary space, allowing us to easily find its neighbour. We have not, nor are we planning to, found any ethical implication of this new algorithm. It has the sole purpose of making travels and communications easier and effortless.

Looking again at the bigger picture, we are exploring how Principal Component Analysis can be used with technology to recognize words from images. We are verifying this by checking the accuracy with which this kind of optical character recognition occurs.

3. Methods

Our method is made up of a combination of manual and automated processes to prepare the input image for PCA and identify the character in it.

Firstly, we have to define the characteristics of the images that are processed. Images are manually cropped so that the character is at the centre of the image and clearly recognisable. Next, each image is fed into the program, which resizes it as 28x28 and converts it to a binarized (0-1 instead of 0-255) greyscale.

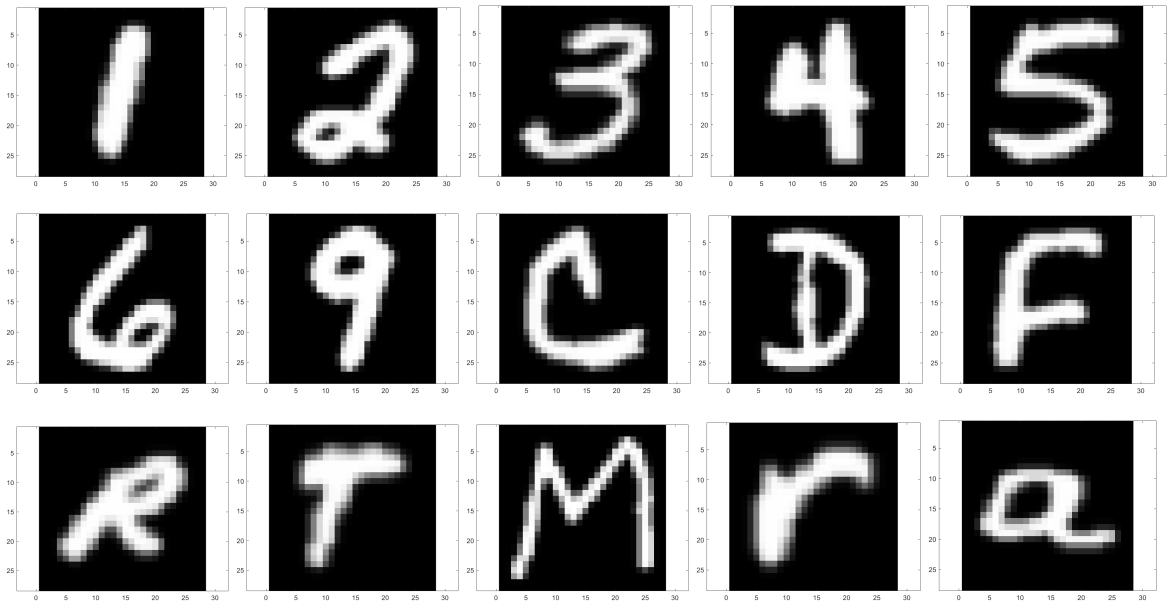

Figure 1: These are some of the 28x28 pixel images used by the EMNIST dataset. They include digits, uppercase letters, and lowercase letters.

After this, these images, which we will refer to as the test images (images that we don't know what they represent), are fed into a function, along with the training dataset (images that were previously analysed and that we know were correctly identified, along with their relative labels) and the number of principal components that we want to use during PCA.

This function first vectorises each test image, reshaping it to an 784 x n matrix, where each column is a single image, n is the number of test images, and 784 is the total number of pixels in each

image (28^2). Secondly, each image is mean centred by column: here, the average pixel brightness of each image is subtracted from all values, so that we now have a spectrum of brightness centred at 0.

$$ATrain = ATrain - \mu_{ATrain}$$
$$ATest = ATest - \mu_{ATrain}$$

A covariance matrix is then calculated. This is done by multiplying our transposed mean-centred matrix with its original counterpart, and then normalising the result.

$$C = \left(\frac{1}{length(ATrain) - 1}\right) \times ATrain^T \times ATrain$$

Afterwards, the eigenvectors, along with their corresponding eigenvalues, are calculated. These represent the main variation trends within the dataset. Each image is dimensionally-reduced using X principal components, in our case 28, as we have determined it to be the number returning the highest accuracy (see Detailed Findings).

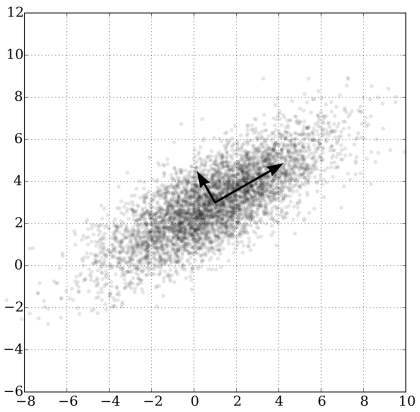

Figure 2: An example of principal component analysis. Here, the arrows represent the principal trends in the data.[1]

After all of these are calculated, each test image is projected into an imaginary space and compared to all training images, and its "nearest neighbour" is found. This is the image that resembles a closest match to our test image. Given that we know the label (what the image contains) of the neighbour image, it is safe to assume that this will be the same as what our test image contains.

4. Detailed Findings

[1] "Principal Component Analysis." *Wikipedia*, Wikimedia Foundation, 5 Nov. 2020, en.wikipedia.org/wiki/Principal_component_analysis.

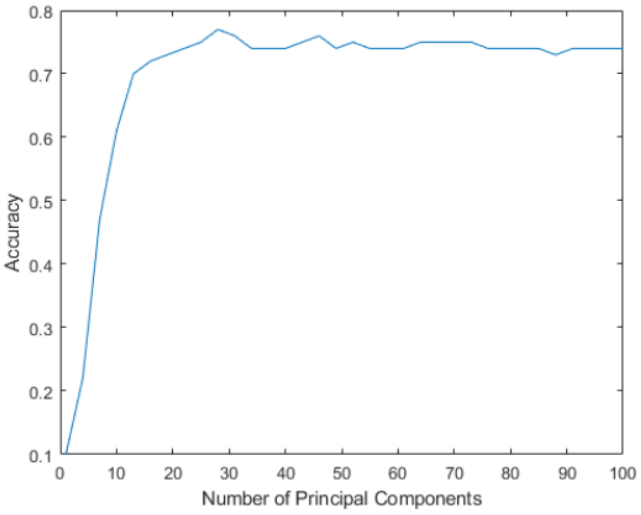

Figure 3: A graph showing how the number of principal components used affects the accuracy (with 50,000 training images).

In order to increase the accuracy of our algorithm, we sweeped the number of principal components to find the best number to use when testing our program. As can be seen in Figure 3, there is a sweet spot at 28 principal components where the accuracy is higher than anywhere else on the graph, yielding 77% accuracy when tested with 50,000 training images. Thus, we set our number of principal components to 28 in our next sweep:

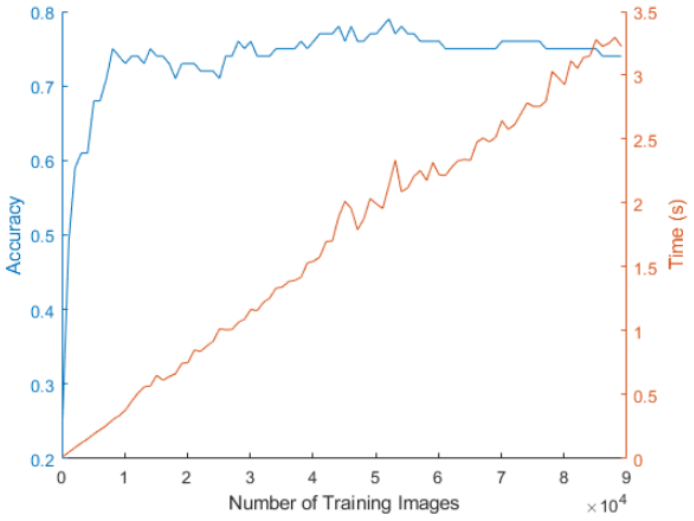

Figure 4: A graph showing how the number of training images affects the accuracy of the character recognition as well as the time it takes for the function to run (using 28 principal components).

Once we had found the best number of principal components to use, we went on to sweep the number of training images versus the accuracy they yielded. We predicted that more training images would give a higher accuracy at the expense of the time it took to run the function. Thus, we decided to plot the number of training images versus both the time it takes to run the function and the accuracy they yield.

As can be seen in Figure 4, the value of 52,100 training images gives the highest accuracy in the graph (79%) while remaining at the reasonable use time of 2.14 seconds. Although higher values could be swept and would potentially give higher accuracies, the use time would continue to increase as is seen in the graph, therefore not making it worthwhile. Thus, once we applied the 28 principal components as well as the 52,100 training images, our algorithm output an accuracy of 79% and a use time of 2.14 s.

This is, for better or for worse, not yet an improvement to existing optical character recognition software, such as CVISION Tech's[2], which yields around 98% accuracy. Although 79% accuracy may be mostly right with individual characters, the probability of recognizing all of the characters in a word correctly go under 50% with any word that has three or more letters. This can become an issue as it could come to a point where most of the words being translated from a page have incorrect characters in them, thus leading to a useless string of almost-words that will result in confusion for the user.

A potential source of error in our method could stem from the fact that most of the pixels in the images we are analyzing are from the background but are being weighed with equal importance to the pixels that make up the characters. Using a different approach of linear regression might help improve this, as that way we can create a "mask" that will weigh the center pixels more than those in the outer corners/backgrounds. Furthermore, our program only currently analyzes images of one character at a time. This problem could be addressed by implementing an image segmentation software that will automatically recognize characters in an image of a word and recognize them each individually.

Even if our program had 100% accuracy, however, it could still lead to mistranslations as literal translation of individual words does not always translate the meaning of an entire sentence. For example, the Spanish saying "me la suda," meaning "I couldn't care less," would literally translate to "I sweat it" or "it sweats me". These translations do not make sense by themselves, and in fact could be interpreted to mean the opposite of what they are actually attempting to express (with "sweating something" meaning to care too much in English). This would therefore have a negative impact and could lead to many misunderstandings. Thus, this character recognition software would have to be

---

[2] "Home." *CVISION Technologies*,

www.cvisiontech.com/library/ocr/accurate-ocr/ocr-accuracy-rates.html.

trained not only to recognize and translate individual words, but also to recognize specific phrases and idioms and translate their meanings accordingly.

5. Recommendations

By using PCA and Nearest Neighbour recognition, we have managed to create a program that recognizes characters from images and translates them into computer-readable text. Our hope for this project was to apply this character recognition software to full words, then plugging them into a pre-existing translator in order to output a translated version of the image. This technology could have a large positive impact by providing a way to bridge language gaps when travelling to a foreign country, such as when ordering food from a restaurant, reading maps and road signs, and reading important documents. All of these uses and more could be helpful in increasing cross-cultural connectivity and making travel more accessible to people who do not speak the native language. The program we have created, however, is still inaccurate and could be improved through training with linear regression, image segmentation, and translation of idioms.

6. References

Cohen, Gregory, et al. "EMNIST: Extending MNIST to Handwritten Letters." *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, doi:10.1109/ijcnn.2017.7966217.

"Home." *CVISION Technologies*, www.cvisiontech.com/library/ocr/accurate-ocr/ocr-accuracy-rates.html.

Patricia.flanagan@nist.gov. "The EMNIST Dataset." *NIST*, 28 Mar. 2019, www.nist.gov/itl/products-and-services/emnist-dataset.

"Principal Component Analysis." *Wikipedia*, Wikimedia Foundation, 5 Nov. 2020, en.wikipedia.org/wiki/Principal_component_analysis.

7. Appendix

Link to code: https://drive.matlab.com/sharing/9339cdab-08e2-417a-9e6d-210d56ab7570

See "Main.mlx" for main document