COMP 551: Applied Machine Learning
Project 3: Image Classification
Team: ZSM

Zeyad Saleh
Computer Engineering
McGill University
(260556530)
zeyad.saleh@mail.mcgill.ca

Matthew Cooke
Computer Science
McGill University
(260553365)
matthew.cooke2@mail.mcgill.ca

Sacha Perry-Fagant
Physics & Computer Scien
McGill University
(260571927)
sacha.perry-fagant@mail.mc

I. INTRODUCTION

Image classification is currently a popular topic in Machine Learning. In this project, the task is to extract information about a two digit operation from an image, where the operator (addition or multiplication) and the operands (digits 0 to 9) are hand written on top of a non-uniform background image. This project differs from the simple image classification task as it adds an extra semantic layer on top of it, which is subject to further interpretation. To classify one example correctly, all three symbols of the image must be classified correctly. The authors decided to tackle this problem by creating a pipeline through which the data undergoes a preprocessing stage where the background is removed and the symbols are isolated, followed by training a machine learning model on a modified EMNIST dataset, which contains only the twelve symbols of interest (0-9, a and A, m and M) and their rotated versions, and finally a prediction and semantic labeling stage from which a final result is computed.

II. RELATED WORK

KNN was chosen as there have been previous applications of KNN to optical character recognition. In particular, there exists a program called Gamera uses KNN to classify characters as well as optical music recognition, OMR. This program uses KNN with a list of features as well as user classified data to classify a given set. With similar results to Gamera as a goal, KNN was chosen as a classifier [1]. A classic example of an application of CNNs is the application to image classification. Because of this, many tutorials use a CNN to classify the MNIST test set. Since CNN is such a typical example of image classification, it makes sense to use it for this project. One such example uses Keras to create a CNN and analyze the MNIST. [5]

III. PROBLEM REPRESENTATION

The dataset provided is composed of a training set of 50,000 images and their corresponding classes, as well as a test set of 1000 images which are subject for classification. Each line of the training and the test sets is composed of 4096 comma-separated entries, and represents a 64 by 64 pixel gray-scale image. Each one of these entries represents a pixel of the image, and can have a value between 0 and 255 that represents the intensity of light of the corresponding pixel, where 0 represents black and 255 represents white pixels (see Fig. 1).

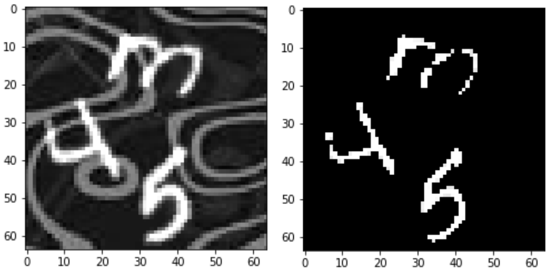


Fig. 1. Image before and after background removal

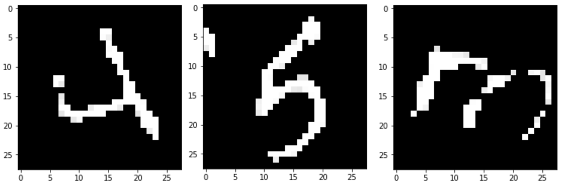


Fig. 2. The Segmented Images

Since the use of the existing EMNIST datasets for training was interesting for this project, it was important to understand their representation too. These datasets are formed by a collection of 28 by 28 pixel gray-scale images, where each pixel can take a float value between 0 and 1 and represents a single digit or letter. Thus the symbols of the images in the test set need to be identified, and each symbol needs reformatted to match the EMNIST data format.

IV. ALGORITHMS AND IMPLEMENTATION

Since the data has to go through several steps, it was decided to create a pipeline through which the data goes through several stages to be transformed from a raw input to a meaningful output. The following sections describe the different stages of the pipeline.

A. *Preprocessing*

The first step in preprocessing was to remove the background image. For this task, multiple methods were explored.

1) Brightness Threshold: Since the symbols are usually brighter than the background, the simplest way to remove the background is to set a brightness threshold under which the brightness of the pixels is set to zero. After trying several ranges of thresholds visually, we found that some thresholds do not apply to different images, since their global brightness might be less than the others.

2) Average Filter: Another simple approach to removing the background is to truncate the brightness of all pixels whose brightness is smaller than the average brightness of the entire image. However a single pass of the average filter does not perform well as it leaves a big portion of bright pixels belonging to the background on. Multiple passes of the average filter however gives good results as can be seen in Fig. 1 with 3 passes. It is important to note that in order to improve the performance of the average filter on each iteration, the average brightness is calculated based only on the bright pixels of the image.

3) KNN: It would have been interesting to explore a KNN approach to the remove the background image, however, due to time constraints that was not achieved.

B. *Symbol Extraction*

The next step was to group the remaining pixels into 3 symbols, and get their images. This is tackled first by getting the centers of the symbols, then cropping the image with certain dimensions around it. One difficultly with this step was the fact that sometimes one symbol was disconnected from itself, or two symbols are connected together.

1) Connected Components: One way to get the symbols is by tracing the connected components and calculating their center of mass. This was done using the openCV library, but it caused many problems since a big portion of the symbols in the dataset were connected to each other, resulting in the detection of only one or two symbols.

C. *K-Means clustering*

K-means was used to get around this problem by separating them into 3 symbols to be classified. However, the issue with this method, is the fact that the classes for the individual symbols of the training data is not known. The training data only provides the result of operation applied to the two digits. In order to get around this, a modified EMNIST dataset was used to train the classifiers. Since this set has individual digits and letters labeled, it was simple to incorporate it into the classifiers. The modified EMNIST dataset is created from the original MNIST dataset, but with an addition of rotated versions of the symbol as well (30 and -30 degrees). In order to make this data compatible with the EMNIST data, the size of each symbol had to be converted to match the EMNIST standard. Since rather than classifying operators, individual symbols are being classified, the number of classes is reduced from 40 to only 12 (10 digits and 2 operators).

D. *Classifiers*

1) Logistic Regression: Logistic regression is a classifier that takes in multiple feature input, with each feature set to either 0 or 1, and has outputs encoded as a discrete set of classes. In the case of image classification, once the images were binarized, it was simple to transform the image into one long array of pixels with values of either 0 or 1. The logistic regression function from the Python module scikit-learn was used. This module uses L2 regularization, or least squares error, as its loss function. When classifying the individual digits, an abysmal result of 6% correct examples was achieved. However, this was an expected result. As the logistic regression classifies each pixel as a feature, it focuses on whether or not a pixel in a certain area is black or white. If a letter is slightly rotated or offset, logistic regression will not be able to recognize it as the same symbol.

2) Feed Forward Neural Network: A feed forward neural network was used in order to classify the segmented figures produced during our preprocessing steps. Using these figures, 3 per original image, the figures could be classified upon an MNIST letters and digits trained classifier. This classifier is first fitted to an MNIST dataset for digit and letter recognition. Once trained the network will predict the sub-figures from the preprocessed data. The network would learn through back-propagation. Although functional, this approach is not optimal. The training and predicting steps using this network are slow for large datasets. The network used has a topology of 784 input nodes, 784 hidden nodes (1 layer), and 12 output nodes representing the digits 0 through 9 and the letters ”M/m” and ”A/a”. The output of the network is not one value, but an array. This array contains 12 entries, one for each symbol class, containing the probability the symbol belongs to each of the classes. The symbol is assigned to the class with the maximum value. Once the three symbols are acquired, it must be checked that a valid operation exists between them. There must be exactly 2 digits and one operator. If this result is not achieved, the next most probable values are considered until a valid operation exists. The operation is applied and the result is saved. The neural network successfully classified only about 11% of correct examples. Therefore, the network was then applied to the original given images, to predict an output from 40 classes. This method provided even worse results, at about 5%. Although better than both logistic regression and our K nearest neighbours approaches, this method provided disappointing accuracy. We also attempted to run the network with more, smaller, hidden layers, however the accuracy remained poor, between 5% and 10%.

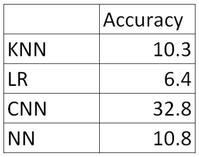
3) K Nearest Neighbors: The K nearest neighbors classifier, or KNN, works by classifying an example based on the distance to the closest k labeled examples. The ’K’ in KNN signifies the number of neighbors we consider. The k neighbors are considered, and the class that comprises the most neighbors is what the example will be classified as. The KNN function of OpenCV was used. [4] Firstly, as a test, the kNN was trained using the entire image. Taking this route provided a very inaccurate classifier, the results generated having only a 8 % accuracy. Next, the individuals symbols were used to classify the data. This time, the MNIST dataset was used as the training data. The number of neighbors parameter, k, was chosen to be 5. For each example, the labels of each of the symbols were found. If there were not exactly two digits and one operators, the neighbors were searched until the combination with the least total distance was found. In the case of three digits, if an operator was not found, then the operator was randomly chosen to be either A or M. The operator was then applied to the two digits and the result was stored. Unfortunately, this method also gave us a poor accuracy of 10%. Similar results to logistic regression were expected and achieved. Like logistic regression, each pixel is considered as a feature and rotations or displacements are not understood by the KNN. Unlike Gamera, there were no encoded character features to look at. If these features had been incorporated, the KNN might have done better.

E. Convolutional Neural Network

A convolutional neural network, or CNN, is a type of neural network. What makes a CNN different from a regular NN, is the fact that it allows some connections to go backwards. One advantage of using a CNN is the fact that, in the case of image classification, translations of images don't faze the CNN as much as other classifiers. Our network has the following topology. We start with a 28*28 input layer accepting our symbols, 3 per image. We feed this into another convolutional layer that feeds into 2 dense layers. Finally, this is fed into an output layer with 12 nodes: representing digits 0-9 and letters a,m.

V. TESTING AND VALIDATION

Our testing showed that when predicting digits, our CNN worked well. However, once letters were introduced, our accuracy fell from 90% to 30%. This can be explained by the fact that it only takes one wrong prediction out of the 3 symbols to result in an incorrect classification.



	Accuracy
KNN	10.3
LR	6.4
CNN	32.8
NN	10.8

Fig. 3. The accuracy of each classifier.

VI. CONCLUSION

As one can see in table 3, most of the classifiers had inaccurate results, with the CNN outperforming them all. As expected, logistic regression and KNN performed equally poorly, having no way of deciphering information about the overall shapes of the digits. The CNN, having the ability to ignore these translations, did three times better than any of the other classifiers. As one can see in figure 4, the CNN does three times better than any of the other classifiers. The NN scores second place in our list of classifiers, but surprisingly, it didn't outperform the lower two by a lot. It is possible that not enough layers and nodes were used to make a big improvement over the other two classifiers. KNN did perform slightly better then logistic regression. Unlike KNN, logistic regression has no sense of neighbors. If there were cases where the wrong number of operators and digits were found, it had no choice but to randomly assign, since it had no information about the other possible classes.

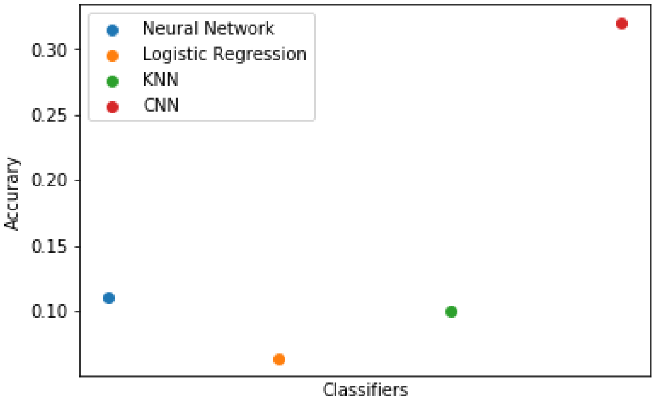


Fig. 4. A plot showing the accuracy of each method

A. Improvements

There was a lot of training data available for this project but it was disregarded in favour of the MNIST dataset. Although this dataset is well classified and contains many examples, its format is different than that of the symbols requiring classification. It is possible that comparisons with the given training set would have increased accuracy of the classifiers. As mentioned previously, the KNN was inspired by a KNN that used specific character feature selection. Having similar feature selection, rather than using only the pixel data, might have improved the data. This is also true for logistic regression. Having a neural network with an increased number of layers and nodes could have found more informative features to classify the data. The CNN was only run with 10 epochs, but running with 100 might have provided better results.

VII. STATEMENT OF CONTRIBUTION

We hereby state that all the work presented in this report is that of the authors. Matthew Cooke implemented the feed forward neural network. Sacha Perry-Fagant implemented the Logistic Regression as well as the k nearest neighbors algorithm. Zeyad Saleh implemented the preprocessing as well as the Convolutional Neural Network.

REFERENCES

[1] The Gamera Project. (n.d.). Retrieved from http://gamera.informatik.hsnr.de/

[2] Werbos, P. J. (1994). The roots of backpropagation: from ordered derivatives to neural networks and political forecasting. New York: J. Wiley & Sons.

[3] Differences between L1 and L2 as Loss Function and Regularization. (2013, December). Retrieved from http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

[4] Mordvintsev, Alexander, and Abid K. OCR of Hand-Written Data using kNN. OpenCV, 2013, `opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_opencv/py_knn_opencv.html`

[5] J. B. (2016, June). Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras. Retrieved from https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/