Handwritten Digit Recognition using Convolutional Neural Networks

Jean-Sébastien Grondin
McGill Id:260345519
jean-sebastien.grondin@mail.mcgill.ca

Jonathan El Baze
McGill Id:260893222
jonathan.elbaze@mail.mcgill.ca

Abstract

The goal of this project was to develop machine learning methods for identifying and classifying handwritten the biggest digits from a modified MNIST Dataset, which is composed of 64x64 grayscale images containing two or three handwritten digits. The digit (from 0 to 9) that occupied the largest space was first identified and extracted using the well known computer vision library OpenCV [1]. A simple K-Nearest-Neighbor classifier was implemented and used as a baseline for evaluating more advanced models. Different convolutional neural network architectures were investigated in order get the best possible results. Several preprocessing techniques were also used and compared, in order to improve the extracted digit images that feed the different models.The best performing model was a convolutional neural network with two convolutions followed by two fully connected layers, which used ReLu activations, batch normalization, and image binarization (i.e. thresholding). It obtained an accuracy of **93.95%** on the validation set and % **94.4%** on the test set on Kaggle.

1 Introduction

The goal of this project was to develop machine learning methods for identifying and classifying handwritten digits in a modified MNIST Dataset, which is composed of 64x64 grayscale images containing more than one handwritten digits. The training dataset was composed of 40 000 images, with the corresponding labels for the biggest digit in each image. The test set was composed of 10 000 images for which it was needed to predict the class. Different image preprocessing techniques were used and tested to help the machine learning algorithm with the classification task. By giving the machine learning algorithm better input images, the models were found to perform better and improve in accuracy.

A simple K-Nearest Neighbor (KNN) model was also implemented and used as a baseline to evaluate the more advanced models.

K Nearest Neighbors (KNN) is a supervised learning approach working by similarity. When predicting a test sample, the KNN method finds the K most similar observation amongst the training set. Then a voting system is used relying on the true value of those observations (e.g the most frequent label) to make a prediction for the new sample.

Convolutional Neural Networks (CNNs) are supervised deep learning algorithms that have been found to perform very well on image recognition tasks [2] [4]. The convolutional layers in the network are used to extract high-level features that can more easily be processed in subsequent layers. They also offer the advandage of capturing spatial dependencies through the use of filters (i.e. kernels and max-pooling steps) that are learned as part of the training phase. CNNs are also trained through backpropagation of the gradient of the loss function.

One of the interesting findings of this project was related to data preprocessing. Removing the noise in the background, normalizing the pixel intensity values, properly extracting the biggest digit images and resizing them to an adequate size were all found to be key contributor to improving the models performances. Feeding good input data to the models for training purpose and subsequently for classification of unseen images does not guarantee top performance results, but surely is a key element for enabling large gains in accuracy.

2 Related Work

The research work of LeCun & al. [2] has been very useful with developing and benchmarking various classification algorithms for the detection of documents with handwritten digits. The MNIST database [3] , i.e. a large collection of handwritten digits, was created and updated for this work and has been for a long time a very popular dataset for learning techniques and pattern recognition methods. Back then, many classifiers were already tested and compared on this dataset. A small representative subset of those and their performances is reported in Table 1.

Table 1: Performance of various classifiers on MNIST digit recognition

Classifiers	Result	Ref.
Linear		
Linear classifier (1-layer NN)	88.0%	[2]
Pairwise linear classifier	93.0%	[2]
KNN		
K-Nearest-neighbors, Euclidean (L2)	95.0%	[2]
K-NN, Tangent Distance with subsampling to 16x16 pixels	98.9%	[2]
CNN		
Convolutional net LeNet-4	98.9%	[2]
Convolutional net Boosted LeNet	99.3%	[2]
Committee of 35 conv. net, 1-20-P-40-P-150-10	99.77%	[4]

It is interesting to look at the cross-section of classification models listed in this table to guide this project. Simple KNN models can be implemented easily and used for baseline comparisons. The current best performing model is the one at the bottom of the table by Ciresan et al., which obtained an accuracy of 99,77%. It is a much heavier model which requires large computing resources that unfortunately were not available for this project. However, simpler convolutional neural networks can be developed and trained on simple machines. This will be the focus of this study, i.e. to explore and investigate how convolutional neural networks can be used and improved for this particular task.

3 Dataset and setup

The dataset used for this study is a modified version of the MNIST Dataset. It is composed of 40 000 training images with corresponding labels and 10 000 testing images. Each instance corresponds to a grayscale 64x64 pixels image which is composed of two or three handwritten digits. The training dataset was randomly split into a 32 000 images training set and 8 000 images validation set. This 80%-20% split allowed the comparison of different CNN architectures in terms of their accuracy.

Different preprocessing methods and techniques were implemented using the OpenCV library [1]. They are summarized below:

- **Thresholds:** Pixel intensity thresholding was used to isolate the digits from the noise in the background, resulting in binarized images. Various threshold levels were investigated.
- **Contours:** All contours were identified in binarized images.
- **Bounding Square:** A bounding square was drawn around each contour and the one with the largest area was selected.

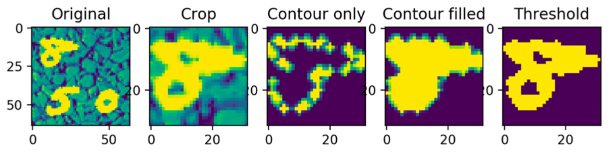

Figure 1: Comparison between different digit extraction methods, True Value = 8

- **Cropping and resizing:** The largest digit was cropped from the original image and resized to the desired aspect.

Several variations of digit extraction methods were also investigated in order to obtain the best quality possible of digit extraction and label match to train the models. Some of these are highlighted in Figure 1. Also, different criterion for identifying the largest digit were also tested, namely the bounding square with the largest area and the largest contour area. The former method was found to yield the best results and was used for all models developed and discussed below.

4 Proposed approach

The convolutional neural networks were configured and implemented using the Pytorch library [5]. A starting CNN architecture was obtained from a pytorch tutorial found online [6]. This CNN demonstrated top performance on the original MNIST dataset and was thought to be a good starting point to classify the handwritten single digits extracted from the modified MNIST dataset. This CNN was configured to take 28x28 pixel handwritten digit as inputs. Thus, all pre-processed images were resized to 28x28 pixels in order to be compatible with this initial architecture. The initial CNN architecture included the following features:

- **Two convolutions** using 5x5 kernels with strides of 1 pixel, without padding, with ReLu activation. The first convolution extracted 10 channels and the second returned 20 channels.
- **Max-pooling** following each convolution of size 2x2.
- **Dropout** after each convolution for regularization.
- **Two fully connected layers**, the first with 50 nodes and ReLu activation and the last with 10 nodes and log softmax.

From this starting point, different improvements have been tested out. One of the first experimentation was to increase the capacity of the network via the number of channels in the convolution network and via the number of nodes in the fully connected layers. Normalization was also added on the extracted digits prior to being fed to the CNN. Batch normalization was also tested instead of using dropouts. Both the binarized (with threshold) and non-binarized images were used and the impact on the performance was verified. For interest, the CNN architecture was also modified to take the original 64x64

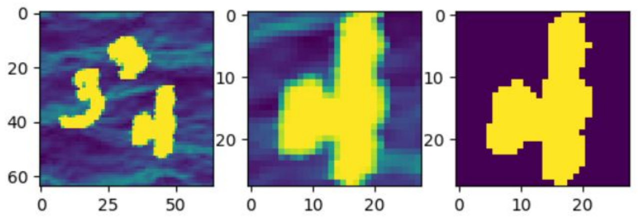

Figure 2: Correct extraction of digit. [left] original image (64x64) [middle] resized (28x28) [right] resized and binarized (28x28)

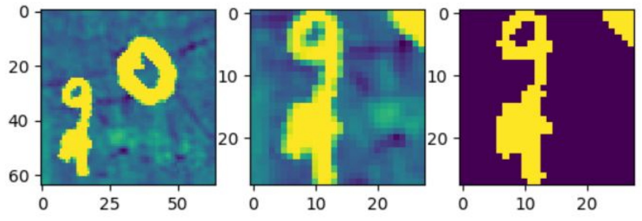

Figure 3: Incorrect extraction of digit. [left] original image (64x64) [middle] resized (28x28) [right] resized and binarized (28x28)

image as an input. As a result, the task of identifying the largest digit was given back to the deep learning algorithm. For this, the CNN had to be modified to include another convoluted layer. The model took much longer to train and the results were less promissing and thus further investigation was not pursued.

5 Results

5.1 Digits Extraction

For each original 64x64 grayscale image, the digit occupying the largest bounging square was extracted and resampled in 28x28 arrays. The method described above yielded good results and it is estimated that no more than 2.7% of the extracted digits were incorrect, i.e. not matching their labels. This was estimated from verifying whether the extracted digit matched its label on a sample of 300 extracted images. Examples of adequate and inadequate digit extractions are provided below. Because the digit extraction functionality is based on thresholding and finding contours, it was found to behave poorly when digits are overlapping and when the background patterns or noise have high pixel intensity values. This was found to be the main cause for incorrect digit extraction.

Originally, a threshold of 220 was used for binarizing the 28x28 single digit images. Later in the study, various thresholds were tested and the optimal threshold was found to be 235. However, varying the threshold between 210 and 250 was found to have very little effect on the overall classification accuracy. Both binarized and non-binarized images were used as inputs to the baseline KNN model as well as the CNNs and results are discussed below.

5.2 Baseline Model: KNN

A simple K-Nearest-Neighbor classification model was implemented as a baseline for comparing the CNNs. The SKLearn library's *KNeighborsClassifier* [7] was used for this purpose and a starting script was obtained from Kaggle [8]. The number of neighbors was varied between k=1 and k=13 and both the binarized (threshold = 220) and non-binarized 28x28 images were used for training and validating this classifier. Results are shown below:

Table 2: Validation accuracy of KNN classifier with different k values

k (neighbors)	Non-Binarized accuracy (%)	Binarized accuracy (%)
1	78.70	85.90
3	80.05	87.34
5	81.40	**88.15**
7	**81.46**	87.72
9	81.31	87.41
11	80.94	87.32
13	80.59	87.17

The best peforming KNN model uses k = 5 neighbors and binarized resized images and has a validation score of **88.15%**. The latter will be used as a reference when evaluating the CNNs' performances.

5.3 Convolutional Neural Networks

Table 3 shows the baseline KNN model performance, as well as the starting CNN and the subsequent incremental changes made to it to improve the classification accuracy.

Table 3: Incremental improvements made to the original CNN architecture

ID	Description	Computing time (%)	Validation accuracy (%)
00	KNN baseline	4m52s	88.2
0	Starting CNN	6m56s	90.0
1a	Added Capacity	12m24s	92.1
2	With normalization pre-processing	11m17s	93.6
3	Using batch normalization	7m42s	93.8
4	Using threshold = 235	7m44s	**94.0**
1b	64x64 image, 3 convolutions	39m57s	86.5

In step 1a, the two layers of convolutions were modified to return 20 and 50 channels respectively. In addition, the first fully connected layer was increased in size from 50 nodes to 500 nodes. An improvement of more than 2% accuracy was obtained as a result. Subsequently, in step 2, the input 28x28 image was normalized by dividing all pixels by 255 so as to have values in the range from 0 to 1 for non-binarized images and either 0 or 1 for binarized images. This also had a positive impact on accuracy, bringing the performance to 93.6%.

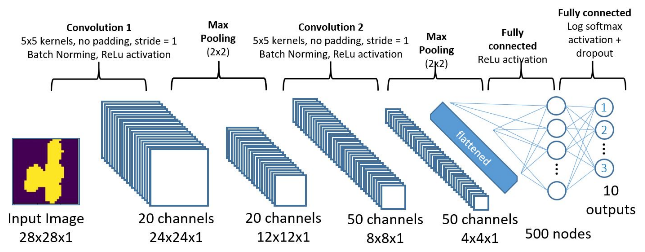

Figure 4: Architecture of final CNN, composed of two convolutions with max pooling and two fully connected layers.

In light of another CNN implementation tutorial in PyTorch [9], it was decided to replace the dropout regularization for batch normalization instead. Batch normalization is another method of regularization for convolutional network, which can help reduce training time and result in better performance. This was found to be the case when this was implemented in step 3. The model accuracy improved by 0.2 % but more importantly, the network was able to proceed through the training phase more quickly, and attain high accuracy with very few epochs. It is of interest to point out that only the accuracy for binarized images is reported since it was found to always yield better results, although both options were tested at each step. Finally, in step 4, different thresholds for binarization were tested and the optimal threshold of 235 was found, which gave another 0.2% accuracy improvement.

As indicated previously, it was of interest to test whether a modified version of this CNN could take the original image as an input and identify by itself the digit that occupies the largest area and classify it. Reasonably good accuracy was obtained, which seems to indicate that it is possible to achieve very high accuracy, given more complex networks and more computing power available. Since this model was found to be a lot more time consuming to train, no more investigation was pursued in this direction.

The optimal CNN architecture tested in this project is displayed in Figure 4. This complete algorithm can be used for classifying handwritten digits with an accuracy of **93.95%** on the validation set and % **94.4% on the test set**, which provides an improvement of +5.8% over the baseline, i.e. the simple KNN model.Two examples of correct classifications are shown in Figures 5 and 6.

6 Discussion and Conclusion

The best CNN model implemented in this study was found to achieve an accuracy of **93.95%** on the validation set and % **94.4% on the test set**. One of the interesting findings of this project was related to data preprocessing. Removing the noise in the background, normalizing the pixel intensity values, properly extracting the biggest digit images and resizing

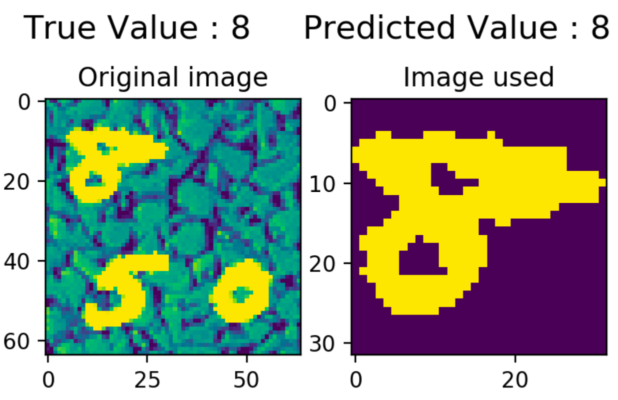

Figure 5: Original image [left] and extracted digit with threshold [right] leading to correct prediction.

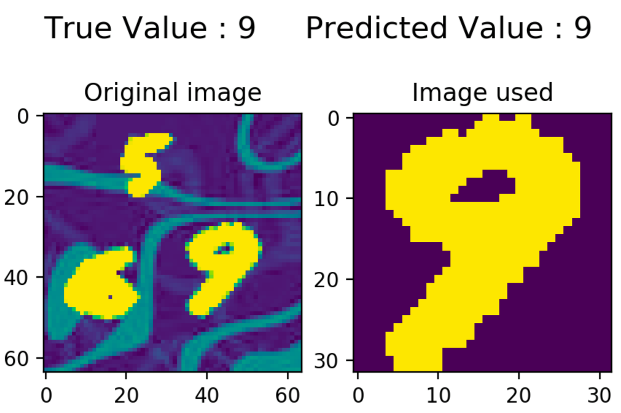

Figure 6: Original image [left] and extracted digit with threshold [right] leading to correct prediction.

them to an adequate size were all found to be key contributor to improving the models performances. Feeding good input data to the models for training purpose and subsequently for classification of unseen images does not guarantee top performance results, but surely is a key element for enabling large gains in accuracy. There are many ways in which the work described in this paper could be improved. The goal of this study was principally to explore the use of CNNs for handwritten digit classification and develop a good intuition for how CNN architectures can be modified to better address certain tasks. In this mindset, the goal of this study was accomplished.

Despite the fact that PyTorch is reasonably straighforward to use and adapt, the learning curve for adapting convolutions to different image sizes and increase the capacity was quite steep. After a lot of experimentation, there is now a strong sentiment of curiosity and appetite for going further and testing more features and functionalities, as well as exploring other image datasets. Here are some steps that would be interesting to investigate and that could help improve further the results:

- **A)** Deskewing digits. A significant propotion of extracted digits have strong skews, which doesn't prevent the CNN from achieving near 95% accuracy but using deskewing could help improve results.
- **B)** Add zero-padding around the 28x28 images to allow the application of the kernel near the boundary and pickup features that are in its vicinity could also help improve accuracy.
- **C)** Different kernel size could also be investigated (e.g. 3x3 and 7x7)
- **D)** Test out different stride steps.

7 Statement of Contributions

- **J-S Grondin** Primary role was to implement the main digit extraction function, KNN baseline model. Was also responsible for building and testing various convolutional neural network architectures. Finally, contributed to the writing of the report.
- **J El Baze** Primary role was to construct the image dataset and implement various digit extraction options. Also responsible for researching related work, running experiments and writing the report.

References

[1] Iseez, *Open Source Computer Vision Librairy*, 2015, `https://github.com/itseez/opencv`

[2] LeCun & al. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11):2278-2324, November 1998

[3] LeCun & al. *MNIST handritten digit database.* 2010 `http://yann.lecun.com/exdb/mnist/`

[4] Ciresan & al. *Multi-column Deep Neural Networks for Image Classification*, CVPR 2012

[5] Paszke & al. *Automatic differentiation in PyTorch*, Conference on Neural Information Processing Systems (NIPS), 2017

[6] Gregor Koehler *MNIST Handwritten Digit Recognition in PyTorch*, 2018 `https://nextjournal.com/gkoehler/pytorch-mnist`

[7] Pedregosa & al. *Scikit-learn: Machine Learning in Python.* JMLR 12, pp. 2825-2830, 2011.

[8] Marwa Saied *Handwritten digits Classification (Using KNN)*, `https://www.kaggle.com/marwaf/handwritten-digits-classification-using-knn/log`

[9] Yunjey, *pytorch-tutorial GitHub*, 2018 `https://github.com/yunjey/pytorch-tutorial/`