Modified MNIST Image Analysis

COMP 551 - Group 75

Sarah Fernandez	260663523
Otman Benchekroun	260663493

Abstract

Containing 70,000 images of handwritten digits, the MNIST database has been a fundamental resource for those working in machine learning. Through utilizing this public database, scientists can create and test object recognition and image analysis models and alter these images to create new data for models to come. Given a modified MNIST dataset with images that contain more than one digit, we were tasked with finding which number occupies the most space within the image. To accomplish this task, we first preprocessed the data to determine which digit within each image had the largest bounding square by pixel area. Afterward, we trained several convolutional neural networks and implemented the bagging ensemble method to achieve our best performance. We found that our best performing model reported an accuracy score of 90.8 on the final test set.

1 Introduction

With 60,000 training images and 10,000 testing images, the MNIST database has provided researchers with an excellent public resource for implementing and testing machine learning models[1]. Its various images of handwritten digits and accompanying labels provides a foundational dataset for machine learning scientists working in the fields of image recognition and computer vision, whose work only becomes increasingly important to technological advancements.

This project utilized a modified MNIST database of 40,000 images where each image contained more than one digit, the goal being to find which number occupies the most space in the image. This ability to differentiate objects in an image and measure their relative locations is tremendously important in today's technology. From traffic monitoring and license plate detection systems to fingerprint and iris pattern recognition programs, the use of object recognition and differentiation is only increasing in everyday life.[2]

For the this task, we trained several models using a training set of 32,000 modified images and a validation set of 8,000 images. All of the models were convolutional neural networks with varying layers and densities. These models were constructed using Keras layering, pooling, and flattening methods.[3] We implemented and manipulated several public algorithms, including LeNet[4], using different combinations of densities, layers, and batch normalization, as well as testing the effects of using convolutional 2D layers against the effects of depthwise separable 2D convolution.[5] The process of preprocessing the images involved identifying the numbers in the image, determining the bounding squares of the numbers, and comparing them their sizes to return a new image of the number with the largest bounding square; this was implemented with methods from the SciPy library.[6] Using the bagging ensemble method, we produced our best performing model; by bagging fifteen convolutional neural networks, we achieved an accuracy score of 90.800 on the test set on Kaggle.

2 Related Work

Image recognition and computer vision are central topics in machine learning. In particular, the recognition of digits is a focus for many machine learning scientists. Below we will explore some relevant papers that touch on digit recognition techniques that were considered in our project development.

[1] https://www.sciencedirect.com/science/article/pii/S0262885604000721?via%3Dihub
[2] https://www.sciencedirect.com/science/article/pii/S107731421300091X
[3] https://keras.io/layers/convolutional/
[4] http://yann.lecun.com/exdb/lenet/
[5] https://keras.rstudio.com/reference/layer_depthwise_conv_2d.html
[6]http://scipy-lectures.org/advanced/image_processing/auto_examples/plot_find_object.html?fbclid=IwAR3OFz9mYXmduoxDfHWZFU18Yk53H7BG-TUFRy9agJWYCcHYgOzRmTPj9CE

[LeCun et al., 1998a] created a convolutional neural network called LeNet-5 to "recognize patterns with extreme variability...and with robustness to distortions and simple geometric transformations."[7] LeNet-5 contains 3 convolutional layers, 2 pooling layers and 1 fully connected layer, followed by the output layer.[8] When tested on the MNIST database this CNN was able to achieve an error rate below 1%.

For their task of multi-digit recognition, [Yang, 2015] preprocessed their images by using canny edge features and a contour finder to draw the bounding box of each feature.[9] In this way, they were able to segment the digit patches to feed into their CNN. Using this approach, they achieved a 99.07% validation accuracy on the MNIST dataset.

Sijmilarly [Goodfellow et al., 2013] used bounding boxes and convolutional neural networks to recognize multi-digit numbers From Google Street View imagery.[10] They found that the performance of their CNN increased with the depth of their convolutional network, achieving an accuracy of 97.84% on the SVHN dataset from a CNN with eleven hidden layers. They found that their model is comparable to that of human operators, and their system has extracted almost 100 million street numbers from Google Street View imagery.

Consistently, we see in research papers on computer vision that convolutional neural networks are the most effective method for digit recognition. For images with multiple digits, researchers used bounding boxes around the digits to create images of these digits to feed into the neural network. Following these examples, we created several neural networks and utilized bounding boxes in our preprocessing stage.

3 Dataset and Setup

The training set contains 32,000 images, with each image represented as a 64x64 matrix of pixel intensity values. Each image is labeled with the number that is largest in the image, with no other features. The test set contains 10,000 unlabeled images. For evaluation and model selection, we used a validation set of 8,000 images.

For pre-processing, we roughly followed the approach outlined by Yang[11]. The first step of the preprocessing involves finding the largest number in each image. The largest number was determined by finding the largest square bounding box using a method outlined in this SciPy lecture[12]. First, a pixel mask is applied to the image, such that any pixel less than a certain

[7] http://yann.lecun.com/exdb/lenet/

[8] https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4

[9] https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdfhttps://web.stanford.edu/class/cs231m/projects/final-report-yang-pu.pdf?fbclid=IwAR1-BsnRjM6VuobsCpYcsuT0IrZI84ah0MLka5C8R2Y_q2TrnRjFGk5JUys

[10]

[11]https://web.stanford.edu/class/cs231m/projects/final-report-yang-pu.pdf?fbclid=IwAR1A_RP8oheRIVw1u7vyNKGEXlWbA_-gnsVGK9dtGls5PbE0286qwCJ0KB8

[12]http://scipy-lectures.org/advanced/image_processing/auto_examples/plot_find_object.html?fbclid=IwAR3OFz9mYXmduoxDfHWZFU18Yk53H7BG-TUFRy9agJWYCcHYgOzRmTPj9CE

threshold is simply set to 0. Then, using one of SciPy's labeling functions and object detection function, we detect the largest rectangular bounding box in the image. From the rectangular box, a square bounding box can be derived by using the largest of the rectangular bounding box's length and width. Once each object has a square bounding box determined, we compare the area of each bounding box and return the largest one. Furthermore, once we have determined the largest square bounding box, we crop the original image to only include the desired number, then we pad the image to a size of 40x40 pixels. If a number were to have a bounding box be larger than 40x40, we would first scale down the cropped image to a size of 32x32, and then we pad the image to 40x40 once again.

4 Proposed Approach

As convolution neural networks have been proven to be some of the best machine learning models for image recognition,[13] we decided to implement different variations of convolutional neural networks to achieve the best performance accuracy. Using public CNN models found online as a guide, we constructed two neural networks: babyNet1 and babyNet2.

BabyNet1 is a convolutional neural network based on a previous model created by Orhan Gazi Yalçın.[14] It contains two 2D convolutional layers, both using a kernel size of 3x3 pixels and a stride of 1x1 pixels. The first convolutional layer contains 64 nodes and the second layer contains 16 nodes. These convolutional layers allow us to preserve the relationship between different parts of an image by filtering the image with a smaller pixel filter (the kernel) to decrease the size of the image without losing the relationship between pixels. Between each layer is a 2x2 max pooling layer, which reduces the spatial size and parameter counts for decreased computational complexity. We include a flatten layer to convert the 2D tensors into 1D tensors for the fully connected Dense layers. babyNet1 has 2 Dense layers; the first Dense layer has 64 nodes, while the second contains 10 nodes. Between our 2 Dense layers is a Dropout layer, which disregards 20 percent of the neurons while training to combat overfitting. The model was compiled using the adam optimizer and using the sparse categorical cross-entropy loss function.

BabyNet2 is a convolutional neural network based on the CIFAR-10, a CNN built by data scientist Rohan Chopra.[15] BabyNet2 has four convolutional layers, each with 96 nodes, kernel size 3x3 pixels, and stride of 1x1 pixels. Between each layer was a 2x2 max pooling layer. Similar to Babynet1, Babynet2 had 2 Dense layers with 64 nodes in the first and 10 nodes in the second with a 0.2 dropout layer between. Babynet2 was also compiled using the adam optimizer and using the sparse categorical cross-entropy loss function.

Once we determined which one of our two models performed best, we aimed to further improve its performance by "bagging" the network. Bagging or "boot-strapping" has been shown

[13] https://medium.com/zylapp/review-of-deep-learning-algorithms-for-image-classification-5fdbca4a05e2
[14]https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d?fbclid=IwAR0Xjlc-6ogZCiSZI8xOTc0R9HZ9iXx3t2nU6VZIT6W8s8czNLSyv4_25yk
[15] https://github.com/09rohanchopra/cifar10/blob/master/cifar10-improved-cnn.ipynb

to improve the accuracies of estimators and classifiers[16]. It does so by taking a number of classifiers, and training each classifier on a different subset of the training data. The subset of the data is acquired by randomly sampling the training data without replacement. In our case, we sampled 30 000 of the 40 000 images without replacement 15 times, and trained 15 BabyNet1 architectures on each of the subsets. Once these classifiers were trained, we made each one classify the test set, and submit a "vote" of it's predicted output. The most popular vote for each output was then cast as the final prediction.

5 Results

Table 1 shows the reported validation accuracies after 10 epochs, using a .2 validation split. BabyNet1 and BabyNet2 did not have a major difference in runtime, however, as expected, our bagging model took 15x the amount of time required for BabyNet1. In general, all models gave similar results. We found that the bagging model performed the best on this dataset, providing an accuracy of 90.5%

Method	**Accuracy (%)**
Bagging Model	**90.5**
BabyNet1	89.21
BabyNet2	89.15

Table 1: Top result for each algorithm

In addition, we tested the effects of batch normalization and depthwise separable convolutional layers on BabyNet1 to see if these features would affect performance accuracy. Depthwise separable convolutions split a kernel into 2 separate kernels that do two different convolutions: a depthwise convolution, which works with the number of channels in an image, and the pointwise convolution, that works with the spatial dimensions of an image.[17] Table 2 shows the accuracy results of training BabyNet1 using convolutional layers and depthwise separable convolutional layers on 10 epochs. We found the difference between the two layer types to be insignificant, as the difference in accuracy was less than .5%.

Method	**Accuracy (%)**
Conv2D	88.99
SeparableConv2D	88.80

Table 2: Result for each convolutional layer type

[16] https://www.stat.berkeley.edu/~breiman/bagging.pdf

[17]

Because the images are in greyscale, it makes sense that the difference between the simple convolutional and separable convolutional layers is minimal; because separable convolutional layers work also with color channels and the images only contain variations of grey, the benefit of the depth dimension is negligible.

We attempted to test the effect of batch normalization on performance accuracy, however its addition into our models greatly increased the runtime and seemed to provide minimal difference in our training accuracy. Therefore, we chose to exclude it from our models.

6 Discussion and Conclusion

In this project, we classified which number was larger in a multi-digit image. We used methods from the SciPy library to find which number in an image had the largest bounding square and created a collection of images of those digits. Feeding these images into two modified public CNNs, we determined which model had the highest performance accuracy. After choosing this model, we implemented the bagging ensemble method to increase our accuracy. We achieved an accuracy percent of 90.8 in the Kaggle leaderboard.

We learned that the manipulation of images by changing their colors and sizes, and the creation of bounding squares in images greatly impacts the performance accuracy of a model. Before adding bounding boxes, BabyNet1 could only achieve a training accuracy of around 12 percent. With simple bounding boxes, BabyNet1 could achieve a 70 percent accuracy; finally, with bounding squares, BabyNet1 could consistently provide an 89% accuracy.

For future work, we would like to further correct our preprocessing method and further explore the effects of varying layers in CNNs. While our preprocessing method generally did well to create bounding squares, it had difficulty determining bounding squares for numbers that connected to other numbers. Though this only comprised 2 percent of the training data, it was still an obstacle that could be addressed in future work.

Additionally, having colored images instead of greyscale could possibly correlate to better performance. With colored images, using depthwise separable convolution layers could provide greater input from the depth dimension, which could increase the accuracy of our model.

7 Statement of Contributions

Both members of the group contributed equally in different respects. Sarah implemented the convolutional neural networks babyNet1 and babyNet2, while Otman created the preprocessing and bagging methods. Both partners trained and tested the models together. Regarding the report, Sarah researched and summarized relevant research papers, and both partners contributed to writing the report.

References

[Andreopoulos & Tsotsos, 2013] Andreopoulos, Alexander, & John K.Tsotsos. (2013). “50 Years of object recognition: Directions forward.” *Computer Vision and Image Understanding*. 117(8): 827-891.

[Breiman, 1994] L, Breinman. (1994). “Bagging Predictors” , *Technical Report No. 421, University of Berkeley*, September 1994.

[Chopra, 2017] Chopra, Rohan (2017). CIFAR - 10. GitHub Repository. https://github.com/09rohanchopra/cifar10/blob/master/cifar10-improved-cnn.ipynb.

[Goodfellow et al., 2013] Goodfellow, Ian J., Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. (2013). “Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.” Google Inc.

[Kussul & Baidyk, 2004] Kussul, Ernst, & Tatiana Baidyk (2004). "Improved method of handwritten digit recognition tested on MNIST database". *Image and Vision Computing*. 22 (12): 971–981.

[LeCun et al., 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. (1998). "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324

[Ouaknine, 2018] Ouaknine, Arthur. (2018). “Review of Deep Learning Algorithms for Image Classification.” *A Medium Corporation.*

[Wang, 2018] Wang, Chi-Feng. (2018). “A Basic Introduction to Separable Convolutions.” *Towards Data Science*, August 13, 2018.

[Yalçın, 2018] Yalçın, Orhan Gazi. (2018). “Image Classification in 10 Minutes with MNIST Dataset.” *Towards Data Science*, August 19 2018.

[Yang, 2015] X.J. Yang (2015). “MDig: Multi-digit Recognition using Convolutional Neural Network on Mobile”, University of Stanford, 2015.