Classifying the largest digit on a modified MNIST

Gabriele Dragotto[a], Yixing Hu[b], Zhiyue Jiang[b]

[a]CERC Data Science for Real-Time Decision-Making, Polytechnique Montreal.
gabriele.dragotto,@polymtl.ca
[b]McGill University
{yixing.hu,zhiyue.jiang}@mail.mcgill.ca

COMP551 (Winter 2019) - Project 3

Abstract

In this project, we investigated the performance of different machine learning models on a modified MNIST dataset. The goal is to identify the digit with the largest bounding box in each image. In the first instance, we implemented a model that uses contour approximation methods provided by OpenCV to extract the largest digit in each image. A convolutional neural network was then trained on the processed data. This $LeNet$ model achieved a validation accuracy of 94.2%. In the second instance, we applied a deep convolutional neural network model, $mlCNN$, on the unprocessed image data and achieved a 95.5% accuracy on the validation set. Lastly, we implemented a residual neural network model ($ResNN$). This model achieved a validation accuracy of 98.5%. This model was tested with the test set, and obtained a 98.4% accuracy in the Kaggle competition [5].

1. Introduction

Image recognition tasks in machine learning have significantly improved over the past decade. In particular, neural networks played an important role in complex image classification tasks[3]. In this project, we aimed to build machine learning models for image classification on a modified MNIST dataset. Each image in the modified MNIST dataset can contain multiple digits, and the task is to identify the digit with the largest bounding box.

We investigated 3 different neural network models on this dataset. Our first model, $LeNet$, uses the python OpenCV library to approximate the largest digit via contour approximation methods. The estimated largest object was extracted and fed into a neural network model adapted from [9].

In the second instance, we implemented a convolutional neural network model ($mlCNN$), which does not require OpenCV pre-processing. This network implicitly learns how to recognize the largest digit, and eventually outperformed $LeNet$.

Our final model implements a deep Residual Neural Network ($ResNN$) adapted from [6]. This model also does not require OpenCV preprocessing, and was able to achieve the highest validation accuracy of 98.5%. Its outputs were submitted for evaluation at the Kaggle competition.

2. Related work

The original MINIST dataset [9] contains more than 60000 28×28 images of handwritten digits. Machine learning models such as Support Vector Machines (SVM), K-nearest-neighbor classifiers (KNN), and Convolutional Neural Networks (CNN) all showed successful performances for this classification task [2].

[4] tackled this problem with SVMs using Virtual Support Vectors, implementing jittering to add tolerance to rotation and translations between images. With this method, they were able to achieve an error rate of 0.56% using a polynomial kernel of degree 9.

[13] achieved a 97.17% accuracy with KNN classifiers using a sliding windows approach. To correct for spatial misalignment issues between different versions of the same digit, [13] first extend the 28×28 image to 30×30 via zero padding. Then, a 28×28 sliding window was applied to 9 possible positions. The distance metric between images was then computed as the minimum distance between sliding window pairs.

[1] achieved an accuracy of 99.5% on the original MINIST dataset using CNNs. Their CNN model required less pre-processing steps

compared to the SVM and KNN models. [10] also showed that an ensemble of CNNs can further improve their model accuracy.

In a more recent study, [6] introduced the concept of deep residual neural networks. Their models proved to be effective for image classification tasks, as well as being able to generalize to more complicated tasks such as object detection [6].

3. Dataset and setup

The modified MNIST dataset contains grey scale images (64×64) of handwritten digits. There could be multiple digits within each image, and the task is to identify the digit with the largest bounding box. This dataset contains 40000 labelled images, and 10000 unlabelled ones. We partitioned the labelled images into 90% for training and 10% for validation. The unlabelled images were used as the test set, and evaluated through the Kaggle competition [5].

In this project, we investigated the performance of 3 different machine learning models: $LeNet$, $mlCNN$, and $ResNN$.

In $LeNet$, we first set all pixel values lower than 250/255 to 0 in aims to reduce background noise. Then, the data was processed by OpenCV to identify objects of interest and their corresponding bounding box. The estimated largest digit was extracted, reducing the problem to a single digit classification task.

For $mlCNN$ and $ResNN$, the image data undergoes a max scaling process where the pixel values were divided by a factor of 255. The scaled data was then used as input for the two models.

During the training process of both $mlCNN$ and $ResNN$, the training data was split into batches of size 128. To prevent overfitting, we implemented a form of regularization by distorting images using the $ImageDataGenerator$ function from the $Keras$ library. The distortions include:

- *Rotations*: random image rotations between 0 to 40 degrees.
- *Shifts*: random vertical or horizontal shift within a fraction of 0.1 of image's width.
- *Zoom*: random zoom within a fraction of 0.1 of image's size.

Our $ResNN$ model also included batch normalization between convolution layers, as shown in Figure 2. All of our models were trained on 2 NVIDIA $TITAN\ Xp$ GPUs.

4. Proposed approach

$LeNet$

The largest digit was retrieved via contour approximation methods, reducing the task to a single digit classification task similar to that of the original MNIST dataset. During the feature processing step, we exploited $OpenCV$ to identify the object with the largest bounding box in each image. Only the area selected by OpenCv was retained, and used as input for a CNN model adapted from [9].

$mlCNN$

This model aimed to tackle the classification task with a reduced number of feature processing steps. We adapted a model from [12], which was a CNN model applied to the original MNIST dataset. Figure 1 shows the architecture of this model. With this approach, we aimed to investigate the ability of CNNs to generalize to different classification tasks.

$ResNN$

The task of identifying the largest digit mimics an object detection task, where classifiers have to recognize and locate objects along with their corresponding smallest bounding rectangle. [6] reported good performances of such paradigms on $MSCOCO$ and $PASCALVOC$ datasets. We adapted a simplified version of $ResNet$ from [11], and optimized it for the modified MNIST dataset. For our $ResNN$ model, we added batch normalization [7] between convolution layers, as shown in Figure 2.

5. Results

Our first model, $LeNet$, was trained for 30 epochs using the $Adam$ optimizer [8], with a batch size of 256. This model achieved a validation accuracy of 94.2%. By looking at the missclassified images, we found that the majority of errors were made during the feature processing step. In particular, OpenCV was unable to separate overlapping digits (Figure 3). Meanwhile, we also found incorrect digit size estimates during the feature processing step (Figure 4), resulting in the wrong digits to be extracted. To

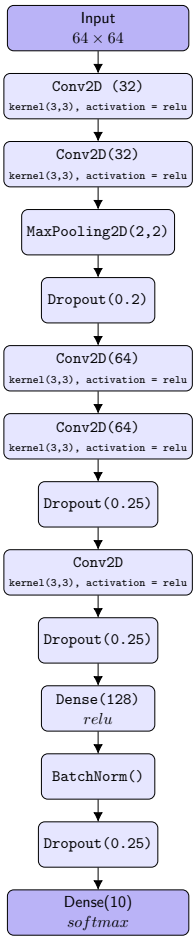


Figure 1: *mlCNN Architecture*

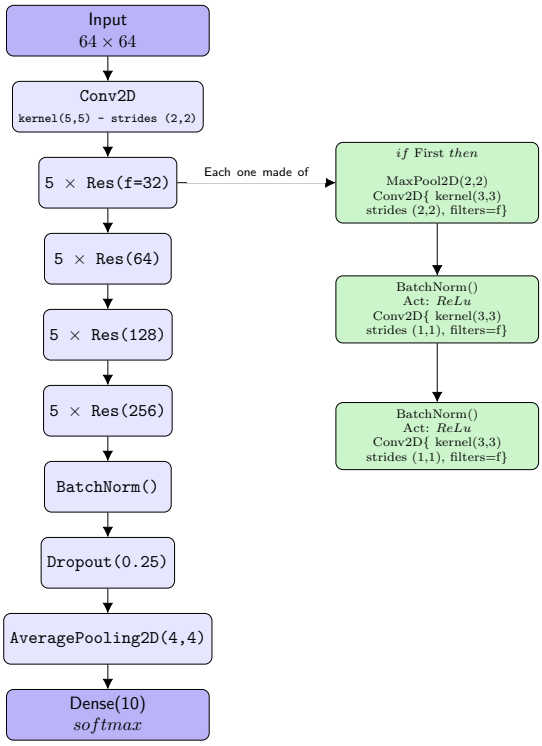


Figure 2: *ResNN Architecture*

Model	Val. Acc.
LeNet	94.20%
mlCNN	95.5%
ResNN	98.50%

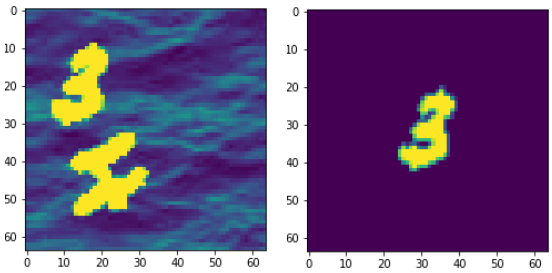
Table 1: *Comparison of validation accuracy between models*

overcome these limits, we investigated alternative models that can perform digit identification and size comparisons simultaneously.

Our *mlCNN* model was able to achieve a validation accuracy of 95.5% without extensive feature processing. This model was trained for 30 epochs, using the *Adam* optimizer [8], with a batch size of 128.

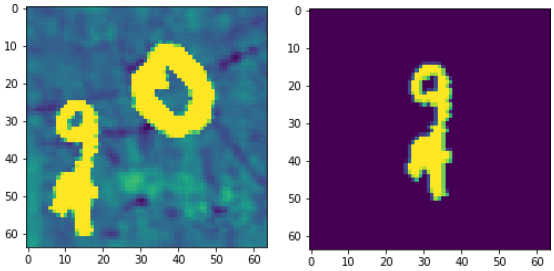
Motivated by the performance of *mlCNN*, we investigated a modified version of *ResNet* adapted from [11]. Our *ResNN* model achieved the highest validation accuracy of 98.5% among the 3 models. This model was used as our final model, and achieved a 98.4% accuracy in the Kaggle competition. The training of this model uses the *Adam* optimizer [8], with a constant learning rate of 10^{-3}.

To determine the optimal training time for *ResNN*, the loss and accuracy were plotted against the number of epochs (figure 5). Each epoch was trained on 329 batches with a batch size of 128. We observed that most of the learning happened within the first 20 epochs, and the performance plateaued afterwards. This



(a) *Not processed* **(b)** *processed*

Figure 3: *The feature processing step of LeNet shows incorrect extraction of the largest digit.*



(a) *Not processed* **(b)** *processed*

Figure 4: *The feature processing step of LeNet was unable to separate overlapping digits.*

model did not show overfitting as the number of epochs increased.

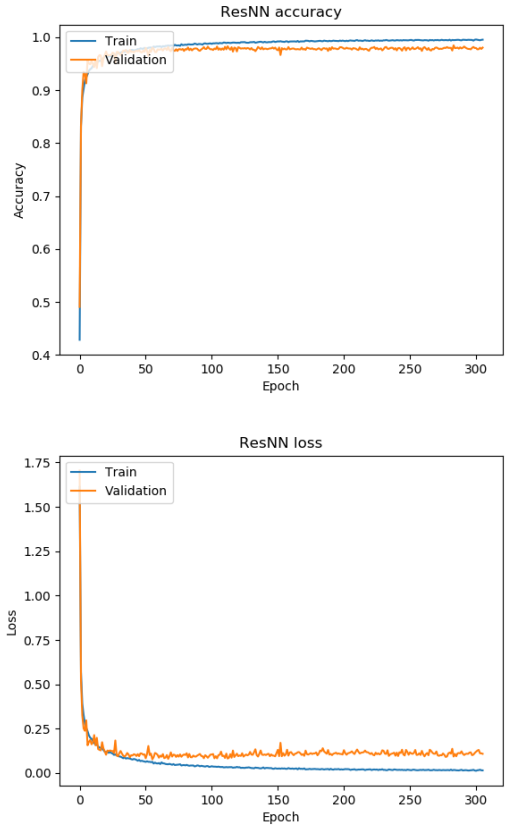


Figure 5: *Accuracy and loss over epochs.*

6. Discussion and conclusion

LeNet, had the lowest validation accuracy amount the 3 models. This model first undergoes a digit selection step with OpenCV prior to the classification process. Due to the sequential nature of this model, mistakes made in first step becomes irreversible, resulting in the stacking or errors. We speculate an erosion process combined with a blob-detector can potentially separate overlapping digits for better digit extraction. On the other hand, the superior performances of $ResNN$, and $mlCNN$ pushed our work towards a fully integrated model for both size comparisons and digit identification. Both $mlCNN$ and $ResNN$ were able to outperform $LeNet$, which demonstrates the ability of CNNs to implicitly learn high level abstractions of the data such as digit size. $mlCNN$, adapted from [12], was designed for the original MNIST dataset. The performance of $mlCNN$ on the modified MNIST dataset shows the generalizability CNNs to different classification tasks. Thus, we speculate a transfer learning approach, tunning a pre-trained network, can potentially achieve better performance.

7. Statement of contributions

1. Gabriele: (33.3%) Coding, Report, Testing
2. Yixing: (33.3%) Coding, Report, Testing
3. Zhiyue: (33.3%) Coding, Testing

References

[1] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.

[2] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, abs/1202.2745, 2012.

[3] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew. Deep learning with cots hpc systems. In *International conference on machine learning*, pages 1337–1345, 2013.

[4] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine learning*, 46(1-3):161–190, 2002.

[5] W. Hamilton. Comp 551 - w2019 - project 3 - modified mnist. https://www.kaggle.com/c/comp-551-w2019-project-3-modified-mnist/leaderboard, 2019. Accessed: 2019-03-10.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[8] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec 2014.

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

[10] U. Meier, D. C. Ciresan, L. M. Gambardella, and J. Schmidhuber. Better digit recognition with a committee of simple neural nets. In *ICDAR*, pages 1135–1139, 2011.

[11] J. Olafenwa. Understanding residual networks. https://towardsdatascience.com/understanding-residual-networks-9add4b664b03, note = Accessed: 2019-03-08.

[12] A. Soni. Mnist with keras for beginners.

[13] B. Toghi and D. Grover. Mnist dataset classification utilizing k-nn classifier with modified sliding window metric. *CoRR*, abs/1809.06846, 2018.