Deep Recognition on MNIST Double Digits

A CMPUT 328 Final Project Report

Chen Jiang, Jennifer Yuen, Shuyang Li, Yanren Qu, Nilanjan Ray*, Nhat Nguyen
Department of Computing Science
University of Alberta
Edmonton, Canada, T6G 2E4
{cjiang2, jyuen1, shuyang2, yanren, nray1*, nmnguyen}@ualberta.ca

Abstract — Handwritten digit recognition is an important problem in character recognition. Relevant datasets like MNIST database has been used as benchmarks for deep learning and pattern recognition methods for years. In this report, we present a series of comparison experiments on digit recognition and localization with convolutional neural networks(CNNs) from shallow to deep architecture. We perform evaluation on a MNIST-double-digit(MNISTDD) database generated by GAN [1]. Across the network benchmarking, we show that deeper networks present a good robustness over complex handwritten image samples with a degree of performance loss.

Keywords —MNIST-Double-Digit(MNISTDD), Convolutional Neural Networks(CNNs), Deep Learning, Pattern Recognition

1. Introduction

In recent years, CNNs have become the popular architecture choices for various computer vision problems. Combined with successful architectures like VGG [2] and ResNet [3], CNNs achieves remarkable performance over object detection, image recognitions, etc. While many benchmarks for computer vision problems are publicly available, the freely available MNIST database of handwritten digits regarding handwritten digit recognition has been a simple yet classic pattern recognition research problem. The simplicity of this task enables fast evaluations of different machine learning and pattern recognition algorithms. The firsthand knowledge from this project can be easily applied to real-time problems like car-license recognition.

In this report, we present methodology of fast-testing of deep classification and localization with choices of shallow to deep convolutional neural networks. We perform the network evaluation on the MNIST-Double-Digit(MNISTDD) dataset which is generated by a Generative Adversarial Network (GAN) [1]. Our main architecture resembles the design of VGG-16 [2] in convolution layers and we benchmark two shallow architectures in regards of speed and accuracy. We present and analyze the result of our architecture on both classification and localization task.

The rest of the report is organized as follows: In Section 2, the proposed methodology is presented. Relevant results are presented and discussed in Section 3. Finally, concluding remarks are drawn with a brief discussion in Section 4.

2. Methodology

Our methodology aims to test the strength of our architecture in both classification and localization task. The overall pipeline is depicted in Figure 1. The first approach is to perform direct classification on two-digits images. The second approach involves digit localization which we predict both the classes and the bounding boxes of the digits.

Figure 1: MNISTDD recognition pipeline.

2.1 Preprocessing

2.1.1 Data Normalization by Simple Scaling

In practice, normalizing the data dimensions to approximately the same scale helps with the performance of many methods. Every image **I** in the dataset is processed by dividing 255 so that the pixel value lies in the range of [0, 1].

2.1.2 Whitening Transformation

Whitening transformation combined with zero-phase whitening filters (ZCA) is then further applied to all image samples before the network training. The goal of whitening is to make features less correlated with

each other and having identity covariance matrix. In practice, whitening transformation is widely used. By applying such transformation, the performance of CNN networks is improved [4][5].

Suppose the normalized image samples $\boldsymbol{x^i}$ are stored as column vectors, we compute the covariance matrix by:

$$\sum = E(x\mathrm{x}^T) = \frac{1}{m}\sum_{i=1}^{m}(x^i)(x^i)^T \quad (1)$$

where m is the number of the image samples. We perform the SVD decomposition over the covariance matrix and compute the matrix U of eigenvectors $\mathbf{U} = [U_1, U_2, \ldots, U_n]$ and corresponding eigenvalues λ_1, $\lambda_2, \ldots, \lambda_n$. Then we calculate the ZCA whitening matrix $\boldsymbol{W_{ZCA}}$ as follows:

$$W_{ZCA} = \mathrm{U}\begin{bmatrix} \frac{1}{\sqrt{\lambda_1+\varepsilon}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{\lambda_2+\varepsilon}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sqrt{\lambda_n+\varepsilon}} \end{bmatrix} \quad (2)$$

where ε is a small constant to prevent zero-division problem. In our project, we set $\varepsilon = 0.1$. Figure 2 presents a MNISTDD image before and after whitening.

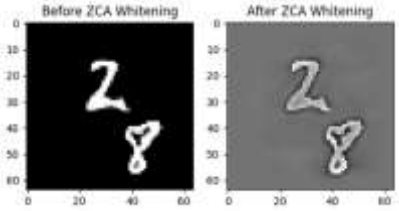

Figure 2: Image before and after ZCA whitening, the edge of the whitened image presents to be sharper.

2.2 Architecture

We experimented with two types of models, one is a VGG-like architecture and the other is a smaller, shallow architecture. The VGG-like architecture is similar to the VGG-16 design, except that we preserve only three blocks of convolutional layers and take out the 4096 fully connected layers. Even so VGG-like architecture is still capable of learning rich features thanks to its large number of parameters. For the shallow model, we mainly use a combination of 64 filters for the convolution layers. Figure 3 presents the architectures for both of the networks.

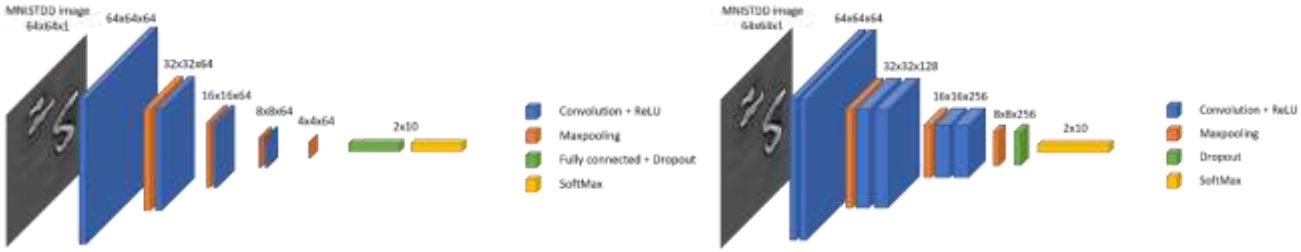

Figure 3: Architectures of the VGG-Like model and the shallow model.

2.3 Training Detail

We further preprocess the labels and the bounding boxes of the dataset to help with the overall procedure. One-hot encodings are performed over the labels of the dataset. Two labels for given double-digit image are encoded into 0-1 vector of length 10, transforming from the shape (2, 2) to the shape (2, 10). For bounding box preprocessing, since the bounding box sizes are fixed to 28 x 28, we extract the first (x, y) coordinate from the bounding boxes data and use them only. Different loss functions are used for training the tasks of the classification and the bounding box localization.

Classification Loss: We use the softmax cross entropy function for the direct image classification task:

$$\mathrm{loss(D)} = -(1-l)\cdot\log(1-p_i) - g\cdot\log(p_i) \quad (3)$$

where p_i is the probability that the image belongs to class i, l = 1 if the image belongs to class I, otherwise 0. The model directly generates the two-digits prediction in this scenario.

Localization Loss: We use a simple Euclidean loss to train the bounding box regression as shown below:

$$\text{loss(D)} = \frac{1}{N}\sum_{k=1}^{N}(\widehat{v_k} - v_k) \qquad (4)$$

where $\widehat{v_k}$ to be predicted i-th bounding box coordinate, v_k to be the ground truth bounding box coordinate. Two approaches are presented for the localization. For the first approach, we first predict the bounding boxes, then we use the predicted bounding boxes to crop the digits out and generate the class predictions on the cropped images. The second approach is to perform the bounding box prediction and the classification at the same time. For localization and classification task at the same time, classification loss and Euclidean loss are summed up as the total loss for training.

3. Experiment

3.1 Experiment Settings

The proposed CNN models are implemented using Tensorflow [6] v1.4 with Python 3.6. The weights for VGG-like network are initialized with methods proposed in He. et. al[7], and the weights for shallow network are initialized using a random normal distribution with a standard deviation of 0.1. Training is performed on a Cybera Rapid Cloud Ubuntu GPU Machine with a NVIDA Tesla K80 graphics card, with a batch size of 128, a weight decay of 0.0005, a Momentum Optimizer with an initial learning rate of 0.01, a momentum of 0.9 and the exponentially decaying learning rate of decay rate 0.95.

For classification, we emphasize on the cases where the model is able predict both digits or one digit correctly, denoted as “all true” and “one true”. For bounding box prediction, we apply the intersection over union(IOU) with a threshold of 0.5 as our evaluation metric. IOU measures the area of overlap over the area of union between predicted and ground truth bounding boxes. If the calculated IOU between the predicted and the ground truth bounding boxes is greater than the threshold, we count the predicted bounding box as a correct prediction.

3.2 Experiment Results

Table 1 presents the results for different models. Figure 4 presents the training loss and accuracy of the VGG-Like classification model. “ZCA” indicates that ZCA whitening is presented before the model training. “L” indicates that the bounding boxes are first predicted before performing the classification model on the cropped digits. “L+C” means the classification and localization tasks are performed at the same time.

Classification: We are able to achieve an all true accuracy of 98.26% and a one true accuracy of 99.96% for the VGG-Like architecture with ZCA whitening transformation. For the smaller shallow architecture with ZCA whitening transformation, we achieved similar results of 97.36% for the all true accuracy and 99.8% for the one true accuracy. Direct classification models stand up as the models having the highest classification accuracies, while classification + localization models have a 1% to 2% of difference to direct classification models. Localization with cropped image classification turns out to be having the worst performance.

Localization: We are able to achieve an all true IOU accuracy of 94.12% and a one true accuracy of 95.08% for the classification + localization model. Localization with cropped image classification turns out to be having the worst performance even on the bounding-box-extraction-only task. Due to the time limits, we did not test the localization task on VGG-Like model. Figure 5 presents some cases of bounding boxes prediction.

Model	All True Acc	One True Acc	All IOU Acc	One IOU Bbox
LeNet	79.76%	96.00%	\	\
LeNet_ZCA	74.08%	95.54%	\	\
Shallow	96.72%	99.74%	\	\
Shallow_ZCA	87.36%	99.8%	\	\
VGG_Like	97.7%	99.88%	\	\
VGG_Like_ZCA	**98.26%**	**100%**	\	\
Shallow_ZCA_L	86.16%	96.76%	91.86%	93.42%
Shallow_ZCA_C+L	95.48%	99.62%	**94.12%**	**95.08%**

Table 1: Validation accuracy with CNN models.

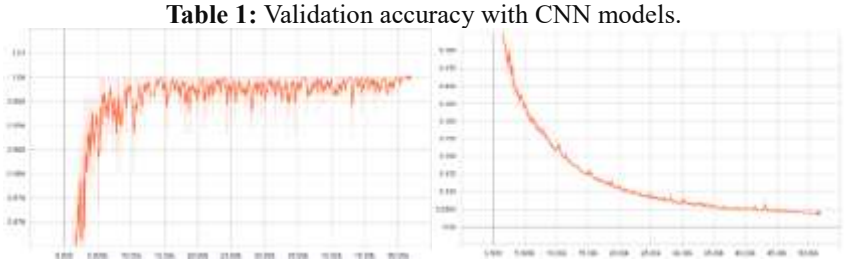

Figure 4: Overall training accuracy and loss of VGG-Like model.

3.3 Experiment Discussion

We present some observations based on our experiments. Firstly, VGG-Like models are able to outperform all types of models presented. This is mainly due to the network's ability to learn more discriminative features with larger number of parameters. Secondly, the classification + localization model presents a 1% to 2% difference compared to classification models only, and the classification after localization models present a huge difference than either of the two types of the models above, suggesting that the localization is more challenging than classification. The classification + localization model presents a higher score for either bounding box extraction or classification than the classification after localization model, suggesting that some feature maps learned can be used to predict different tasks, and separating feature maps may result to irrelevant feature learning.

4. Conclusion

In this report, we present different benchmark for classification and localization task over different CNN network architectures. The experiments performed over the unconstrained MNISTDD datasets demonstrate the effectiveness of our methods. In future, we will evaluate the performance of our methodology over more challenging tasks, from simple tasks like k-digits-MNIST, to more real-world related problems like car license recognition.

Acknowledgement

This is the final project report for Group Tanh of MNISTDD recognition task in CMPUT 328, instructed by Prof. Nilanjan Ray, University of Alberta.

GitHub address: https://github.com/cjiang2/CMPUT328_MNISTDD

Work distribution of each group member is listed as follows:

- Chen Jiang: Overall project design, architecture design and coding (13 hours, 7 days)
- Jennifer Yuen: Bounding box prediction design and coding (12 hours, 7 days)
- Shuyang Li: Architecture design and coding (10 hours, about 5 days)
- Yanren Qu: Data preprocessing, training and validation data collection, Savor and Tensorboard implementation. (10 hours, about 5 days)

The implementation references are listed below:

- IOU implementation: https://blog.mturk.com/tutorial-measuring-the-accuracy-of-bounding-box-image-annotations-from-mturk-ad3dfcdf8aa0?gi=6d9668cda4f4
- Main project codings based on an old Tensorflow project repo from Chen Jiang: https://github.com/zonetrooper32/AgeEstimateAdience
- Tensorboard implementation based on LeNet Jupyter Notebook provided by CMPUT 328 during class.

Reference

[1] Odena, Augustus, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans." arXiv preprint arXiv:1610.09585 (2016).

[2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[3] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[4] Li, Zuhe, Yangyu Fan, and Weihua Liu. "The effect of whitening transformation on pooling operations in convolutional autoencoders." EURASIP Journal on Advances in Signal Processing 2015.1 (2015): 37.

[5] Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009).

[6] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

[7] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.