CS5487 Machine Learning: Prin. & Prac. Project Report
Handwritten Digits Classification using CNN

***Chen, Shiwei**
shiwechen6-c@my.cityu.edu.hk

***Wang, Shixiang**
sxwang6-c@my.cityu.edu.hk

* Indicates Equal Contributions.

City University of Hong Kong
April 20, 2022

1 Introduction

Hand Written Digit Classification is a task that requires the computer to recognize digits, usually the Arabic numeral digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), produced by a human from different sources, including visual images, papers, touch screen from intelligent devices, etc. It has been one of the most essential topics in both statistical machine learning and deep learning and has many interesting applications in the industry, such as automatic bank check processing, vehicle plate recognition, archaic document preserving, etc.

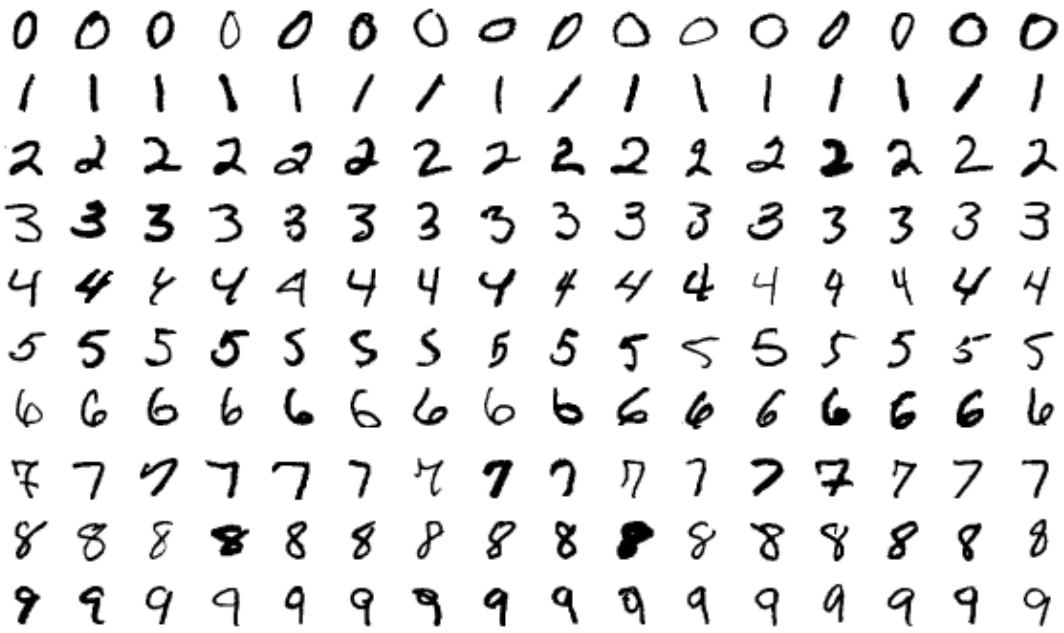

Figure 1: Mnist Dataset Examples [LeCun et al., 1998a]

The task is in fact more challenging than it looks. Although it might be easy for us humans to recognize digits in drastically different calligraphical styles from various input mediums, which is indeed impressive if you think about it, it is still a difficult task for the machine to tackle. One way to simplify the problem is to predefine the inputs into a fixed size and grey-scaled only images. Then the machine can utilize the pixel features to extract patterns for each digit class. This method is implemented in one of the most fundamental datasets in handwritten digit classification, the MNIST (Modified National Institute of Standards and Technology database) [LeCun et al., 1998a], which is also widely used for training and testing in various image processing systems. A slice of the dataset is shown in figure 1.

In this project, we are going to examine the effectiveness of both traditional machine learning

algorithms and deep learning algorithms on the MNIST dataset. We show that the Convolutional Neural Networks (CNN) [LeCun and Bengio, 1998] can achieve very high precision and is on par with human precision. In the subsequent sections of this report, we will discuss our methodology, experiments, and results analysis, respectively.

2 Methodology

The main method we used in this project is CNN, which is most commonly applied to analyze visual imagery [Valueva et al, 2020]. It stimulates the biological process in many visual systems of animals, where individual cortical neuron only responds to stimuli within a specific range. At the same time, the collective group of them can cover the entire visual field, despite possible overlap [Fukushima, 1980]. This is more than mere naturality as it learns to optimize the filters on the way and uses relatively little input preprocessing compared to other image classification algorithms, and thus is also relatively independent from priors and handcrafted/engineered features.

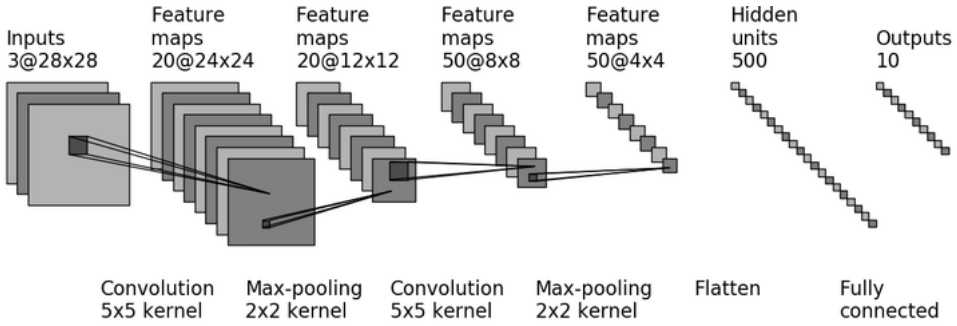

Figure 2: An Example of CNN Model: LeNet-5 [Lecun et al, 1998b]

CNN can be regarded as a regularized multilayer perceptron (MLP) [Haykin, 1994]. However, without relying on traditional regularization methods such as penalizing parameters (e.g., weight decay) and trimming connectivity (e.g., dropout [Srivastava et al, 2014]), CNN focuses on the hierarchy of data and assembles complex patterns using simpler sub-patterns learned by its filters. Although it usually comes along with add-ons such as Pooling layers, Residual Blocks, etc., it is still considered on the lower extreme of the scale of complexity. An example of neural networks using CNNs as the backbone is shown in figure 2.

Notwithstanding, CNN does have some disadvantages in some aspects. The most obvious one is that it completely loses track of the composition and position of the components and the incomplete information is forwarded to a neuron that might not be able to classify the input. Furthermore, due to its way of learning, CNN usually requires a sufficiently large amount of data to process and train. In the next section, we will introduce how we deal with this problem in this project.

3 Experiment Setup

The dataset we use in this project is the *digits4000* provided along with the project guideline. Namely, it has 4000 vectorized greyscale handwritten digit images of size 28 by 28 evenly distributed among the 10 classes, which is just a tiny subset of the original MNIST dataset that has a total of 70,000 images. As mentioned earlier in the previous section, the data is ready to be consumed by CNN. However, some preprocessing is still necessary for it to be effective (i.e., Curse of Dimensionality [Bellman, 1957]) on the baselines we introduced. We first use standard normalization to normalize each pixel feature and then apply Principle Component Analysis [Pearson, 1901] with a 95 percent explained variance threshold to perform dimensionality reduction. After the preprocessing, each digit image is represented by a feature vector of size 284 instead of 784.

Three algorithms are selected as the baselines. From traditional machine learning, the Bernoulli Naive Bayes (BernoulliNB) [Manning et al, 2008] serves as a weak baseline and as a representative of the generative methods, whereas the Support Vector Machine (SVM) [Cortes and Vapnik, 1995] serves as a strong baseline and the representative of the discriminative methods. From neural networks, the MLP will serve as the competitive baseline. Grid Search Cross-Validation [LaValle et al, 2004] is employed to find the optimal hyperparameters and select the best model for these baselines.

For CNN models, since the amount of training data is relatively low, we utilize some data augmentation tricks to boost the variety of training data with the expectation that the model will learn more if it sees more. These tricks include a 15-degree random rotation, a 20-degree random affine with (0.1, 0.1) translation and (0.9, 1.1) scaling, and finally a random gaussian noise generator with 0 mean and 0.05 variance. The augmented data is then standard normalized with training mean/variance and then packed into a batch of size 16. We build three simple CNN models, M3, M5, and M7, to examine the effectiveness of convolutional kernel size. M3 uses a kernel of size 3 and consists of 9 layers, M5 uses a kernel of size 5 and consists of 5 layers, and M7 uses a kernel of size 7 and consists of 4 layers, where each layer is a convolution layer followed by a Bath Normalization Layer [Ioffe and Szegedy, 2015].

All methods including the baselines will be evaluated based on their overall precision, recall, and F1 score on the *challenge* dataset, which consists of 150 images of handwritten digits in the same format, evenly distributed among classes as well. The results will be discussed in the following section.

4 Experiment Result

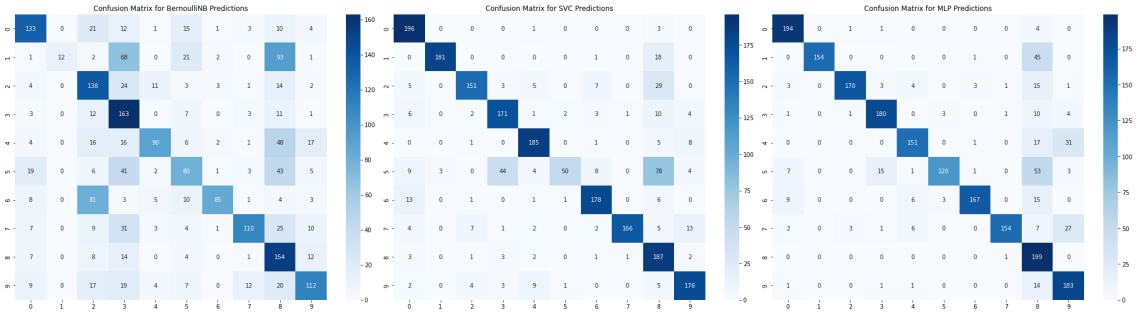

Figure 3: Confusion Matrices for BernoulliNB, SVC, and MLP on training set.

The training results for the baseline models are displayed as a confusion matrix in figure 3 above. The accuracies for them on the training set are 0.54 for BernoulliNB, 0.82 for SVC, and 0.88 for MLP, which are not quite satisfactory. All three models like to classify input as the digit eight, which is quite interesting. We suspect that this is because digit eight encompasses almost all components that are used for writing a digit (e.g., circle, half-circle, line intersection, and angles). Although this algorithm does not recognize pictoric patterns directly by design, they might have learned them statistically.

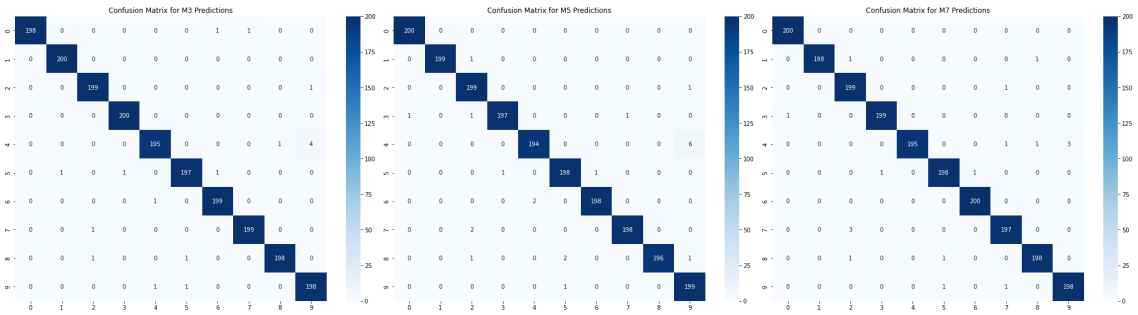

Figure 4: Confusion Matrices for M3, M5, and M7 on training set.

For the CNN models, we use Negative Log-likelihood Loss (plus softmax equals to CrossEntropy Loss) as the scoring metrics, Adam optimizer [Kingma and Ba, 2015] with 0.0001 learning rate for stochastic optimization, and train for 100 epochs with a 10 strikes early stopping strategy. The results are displayed in the form of confusion matrices in figure 4. Their accuracies on the training set are pretty impressive, with 0.992 for M3, 0.989 for M5, and 0.991 for M7, which are significant improvements from the baselines.

The misclassified samples for the CNN models are shown in figure 5. As we can see, most of the misclassified samples are poorly written digits. We believe that it is sufficient to categorize them as outliers because they do not follow standard calligraphical conventions for writing Arabic numerals.

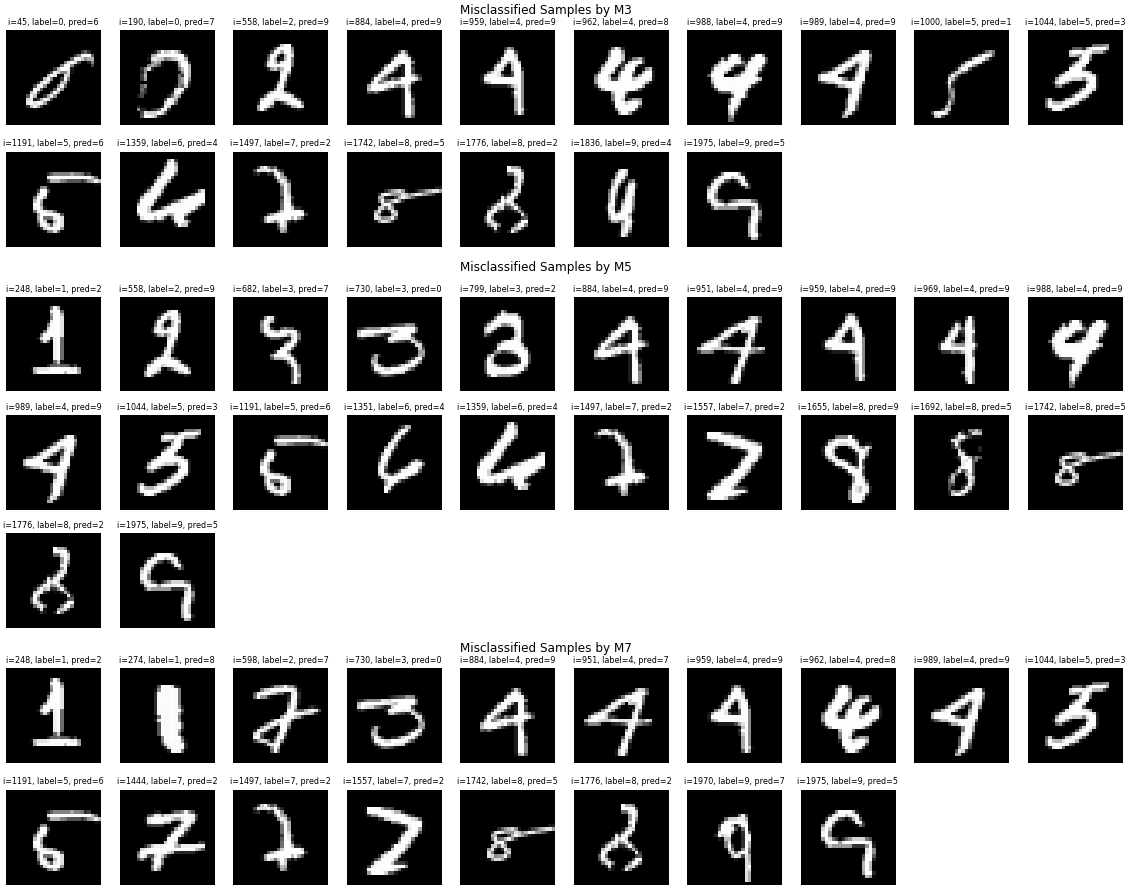

Figure 5: Confusion Matrices for M3, M5, and M7 on training set.

For instance, even a human reader takes a few moments to recognize the 1,176-the sample as digit eight. Therefore, it is better for the models to not remember the patterns from these samples. Otherwise, they would cause overfitting and lose generalizability. The overall statistics for training are displayed in table 1. Note that all scores are Marco averaged, which is the same as weighted averaged as samples are evenly distributed among classes.

Methods	Precision	Recall	F1-score	Support
BernoulliNB	0.66	0.54	0.52	2000
SVC	0.85	0.82	0.81	2000
MLP	0.88	0.84	0.84	2000
CNN-M3	0.992	0.992	0.991	2000
CNN-M5	0.989	0.989	0.989	2000
CNN-M7	0.991	0.991	0.991	2000

Table 1: Training Evaluation Metrics for Baselines and CNNs.

Finally, we test the CNN models on the *Challenge* set, the results are displayed in figure 6 and table 2 below. They achieve accuracies of 0.92, 0.93, and 0.96, respectively. It seems that a larger

convolution kernel size helps the CNN models better generalize the patterns of handwritten digits. We think that this is like observing from a farther distance usually gives one a more comprehensive and robust view of an object or a scene. While a smaller kernel size is like a close-up examination, one gets all the nitty-gritty details but loses some relational information, or to an extreme, like a blind man feeling an elephant.

Methods	**Precision**	**Recall**	**F1-score**	**Support**
CNN-M3	0.921	0.920	0.917	150
CNN-M5	0.924	0.920	0.918	150
CNN-M7	0.966	0.960	0.960	150

Table 2: Challenge Evaluation Metrics for M3, M5, and M7.

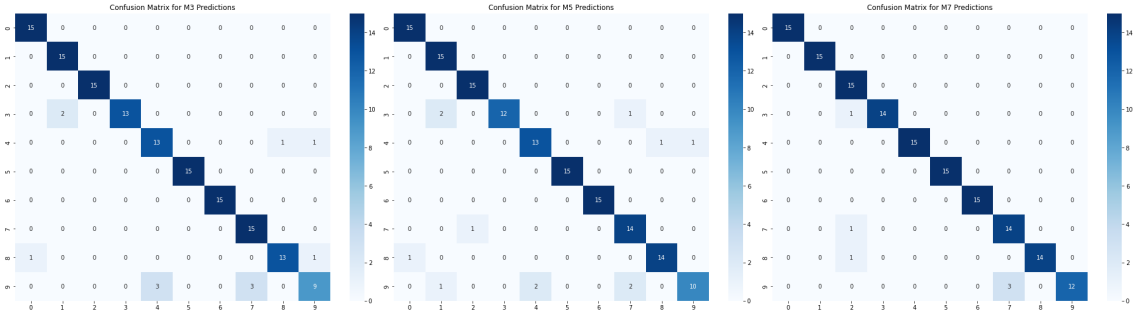

Figure 6: Confusion Matrices for M3, M5, and M7 on Challenge set.

Bibliography

[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, November 1998.

[LeCun and Bengio, 1998] LeCun, Y. and Bengio, Y. (1998). "Convolutional networks for images, speech, and time series".

[Valueva et al, 2020] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. (2020) "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". Mathematics and Computers in Simulation, Volume 177, 2020, Pages 232-243.

[Fukushima, 1980] Fukushima, K. (1980) "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". Biol. Cybernetics 36, 193-202 (1980).

[Haykin, 1994] Haykin, S. (1994). "Neural networks: a comprehensive foundation". Prentice Hall PTR.

[Srivastava et al, 2014] Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya; and Salakhutdinov, Ruslan. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". Journal of Machine Learning Research. 15. 1929-1958.

[Lecun et al, 1998b] Lecun, Yann; Bottou, Leon; Bengio, Y.; and Haffner, Patrick. (1998). "Gradient-Based Learning Applied to Document Recognition". Proceedings of the IEEE. 86. 2278 - 2324.

[Pearson, 1901] Karl Pearson F.R.S. (1901). LIII. "On lines and planes of closest fit to systems of points in space". The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11), 559-572.

[Bellman, 1957] Bellman, Richard Ernest. Rand Corporation (1957). “Dynamic programming”. Princeton University Press. p. ix.

[Manning et al, 2008] C.D. Manning, P. Raghavan, and H. Schuetze (2008). “Introduction to Information Retrieval”. Cambridge University Press, pp. 234-265.

[Cortes and Vapnik, 1995] Cortes, C., Vapnik, V. “Support-vector networks”. Mach Learn 20, 273-297 (1995).

[LaValle et al, 2004] LaValle, S. M., Branicky, M. S., and Lindemann, S. R. (2004). “On the relationship between classical grid search and probabilistic roadmaps”. The International Journal of Robotics Research, 23(7-8), 673-692.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. (2015). “Batch normalization: accelerating deep network training by reducing internal covariate shift”. In Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML’15). JMLR.org, 448-456.

[Kingma and Ba, 2015] Kingma, D.P., and Ba, J. (2015). “Adam: A Method for Stochastic Optimization”. CoRR, 2015.