SVD Implementation on MNIST Image Classification Based on CNN

Satrya Budi Pratama
Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Bandung, Indonesia
satrya@zeroinside.net

***Abstract*—In recent years, many development has been made in the field of image classification. Deep learning has proven to be powerful for classficiation, one of them is convolutional neural networks (CNN). In image classification one of the challange is to recognized the handwritten digit. CNN architecture which has high precision for that problem is Lenet-5.**

In this paper, we recognized the handwritten of MNIST dataset and implement SVD for feature extraction as preprocessing method. We build LeNet-5 improvement model and evaluate the model. Based on experiment, the SVD improves the performance of the model with the 10 number of components and gives 99.03% accuracy. Also the compression ratio with 2 number of components can reduce about 7.87 times of the file size.

***Index Terms*—CNN, LeNet-5, SVD, MNIST**

I. Introduction

Handwriting recognition is the recognition of handwritten letters, numbers and symbols by computer systems [1]. The problem is every handwritten has different style. Every person has their style of writting a character, word and symbol. The computer try to assign the digitilzed character to its symbolic class. In the case of a print image, this is referred to as optical character recognition (OCR) meanwhile in the case of handprint, it is loosely referred to as intelligent character recognition (ICR) [2]. This has been a topic of research for decades. Some of the research areas include signature verification, bank check processing, postal address interpretation from envelopes, and get the digits on test letter for recuirement.

In recent years, the rise of artificial intelligent (AI) and big data especially in machine learning has brought a rapid development in the handwriting recognition application. Google has been developed Vision AI [1] that can be used for reading printed and handwritten text. Some mobile application also has the OCR features such as Pen to Print [2], Notes Plus [3], and Photomath [4]. The web application such as i2OCR [5], OnlineOCR [6], and Convertio[7]. All of these technologies are use machine learning for the recognition or classificaion model.

[1]https://developers.google.com/ml-kit/vision
[2]https://www.pen-to-print.com
[3]https://new.notesplusapp.com/
[4]https://photomath.net
[5]http://www.i2ocr.com/about
[6]https://www.onlineocr.net/
[7]https://convertio.co/ocr/

Deep learning is one of the popular method on machine learning. It has been highly and widely implemented in various research fields especially in image recognition. The most important property of deep learning methods is that it can automatically learn feature representations thus avoiding a lot of time-consuming engineering [3]. Betterchip processing abilities, considerable advances in the machine learning algorithms, and affordable cost of computing hardware are primarily crucial reasons for the booming of deep learning [4].

One of them is Convolutional Neural Network (CNN). CNN was firstly introduced by Kunihiko Fukushima [5]. It was later proposed by Yann LeCun. He combined CNN with back-propagation theory to recognize handwritten digits and document recognition [6], [7]. His system called LeNet-5, it was eventually used to read hand-written checks and zip codes. CNN is a powerful technique for image processing as well as natural language processing. The main advantage of CNN is the high accuracy in its results. However, it requires high computational cost. In addition, it needs a lot of data to be trained. The complexity of the CNN slows down the training process thus it is necessary to use a good GPU to overcome this problem [8]. Beside the hardware, another approach to reduce computational cost by eliminating the features as input to the model and use low-rank approximation. Liu et al. [9] claim that training classifiers using MNIST dataset without using feature extraction methods shows inferior performance. This motivates us to use SVD as feature extraction and compare the performance to its original features from MNIST dataset using LeNet-5 model.

II. Dataset and SVD

A. MNIST Dataset

In this paper, we use Modified National Institute of Standards and Technology (MNIST) dataset. The MNIST is a dataset developed by LeCun, Cortes, and Burges for evaluating machine learning models on the handwritten digit classification problem [7]. It has been widely used in research and to design novel handwritten digit recognition systems. It contains $60,000$ training cases and $10,000$ test cases of handwritten digits (0 to 9). Each digit is normalized and centered in a gray-scale (0 - 255) image with dimension 28×28. Each image consists of 784 pixels that represent the features of the digits.

An example of a digit can be shown in 1 and a example of dataset in 2.

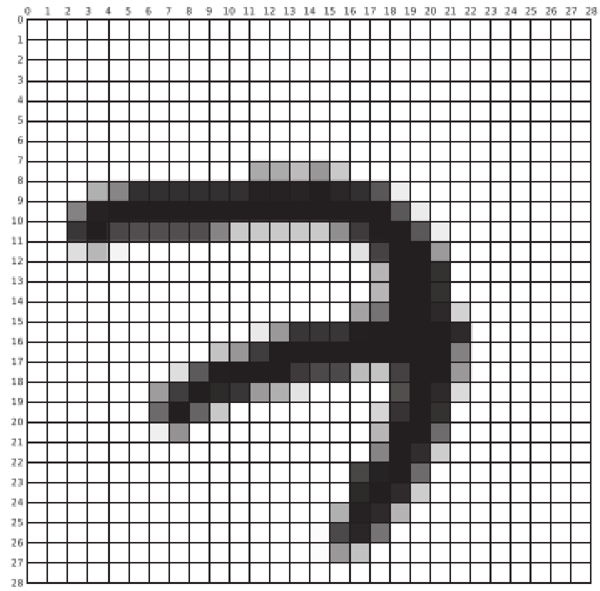

Fig. 1. MNIST sample belonging to the digit 7 adopted from [10]

Fig. 2. 100 digit example of a MNIST dataset adopted from [10]

B. SVD

Singular Value Decomposition or SVD intimately related to the familiar theory of diagonalizing a symmetric matrix [11]. SVD says that rectangular matrix A can be broken down into the product of three matrices : an orthogonal matrix U, a diagonal matrix Σ, and the transpose of an orthogonal matrix V as equation 1 below.

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n} \tag{1}$$

where :

U^TU : Identity Matrix (I)
V^TV : Identity Matrix (I)
U : orthonormal eigenvectors of AA^T
V : orthonormal eigenvectors of A^TA
Σ : diagonal matrix containing the square roots of eigen values from U or V in descending order.

SVD is a method for identifying and ordering the dimensions along which data points exhibit the most variation. This ties in to the third way of viewing SVD, which is that once we have identified where the most variation is, its possible to find the best approximation of the original data points using fewer dimensions. Hence, SVD can be seen as a method for data reduction. Also the SVD applied on linear squares optimization and data compression with reduced rank approximation on [11].

To construct matrix with the low-rank approximation obtain from SVD, with the number of components as k in 2.

$$A_{m\times n} = U_{m\times k}\Sigma_{k\times k}V^T_{k\times n} \tag{2}$$

The approximately compressed size can be calculated using this formula

$$S = (x \times k) + k + (k \times y) \tag{3}$$

where :

S : Compression Size
x : Width of image
k : Number of components
y : Height of images

III. Implementation

We use LeNet-5 architecture to train model on MNIST dataset using TensorFlow. The architecture of the networks shown in 3. It has 7 layers exclude the input layer, it's

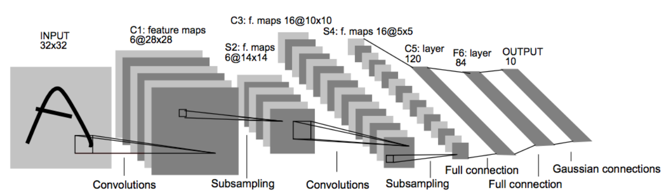

Fig. 3. LeNet-5 Architecture adopted from [7]

basicaly consist of convolutional layer, pooling layer, and fully connection layer.

1) Convolution
   This layer has input dimension $32\times32\times1$ and the output is $28\times28\times6$. We will use ReLu activation function since it's better than sigmoid [12].
2) Subsampling or Pooling
   This layer has input dimension $28\times28\times6$ and the output is $14 \times 14 \times 6$.
3) Convolution
   This layer has input dimension $14\times14\times6$ and the output is $10 \times 10 \times 16$. Also has ReLu activation function.

4) Subsampling or Pooling
   This layer has input dimension $10 \times 10 \times 16$ and the output is $5 \times 5 \times 16$.
5) Fully Connected
   This layer has input dimension $5 \times 5 \times 16$ and the output is 120.
6) Fully Connected
   This layer has input dimension 120 and the output is 84.
7) Output
   This layer has input dimension 84 and the output is 10.

As suggestion by Wang et al [12], to imporve the LeNet-5 model on MNIST dataset is to use 0.0002 learning rate and add drop out layer before output layer. So we use Adam as optimizer with 0.0002 learning rate, and add 0.2 dropout layer before output layer.

First step was downloaded the MNIST dataset [8]. There are 4 main file for training data and testing data. The train data contains $60,000$ examples, and the test data $10,000$ examples. However, the LeNet-5 architecture only accepts 32x32xC images, where C is the number of color channels. In order to reformat the MNIST data into a shape that LeNet-5 will accept, we pad the data with n rows of zeros on the top and bottom, and n columns of zeros on the left and right. The number of n is a half of substraction between 32 and feature extraction dimension. The original MNIST has 28×28 so we pad 2 rows at top and bottom, then 2 columns on left and right.

Before train the model, we did several preprocessing method. We shuffled the dataset and splited train data into 2 sets: 20% train (48,000 images) and 80% validation (12,000 images). We applied SVD as feature extraction to all data sets, with the k number of components should less than the minimal of dimension. We will choose k in range $2, 4, 6, ..., 30$ based on the highest accuracy of evaluate model.

We trained the model with 12 epoch with SVD and without SVD using train data and validation data. We tested the model using testing data and evalutate both model then compared the performance between two method.

IV. Result and Analysis

A. Preprocessing with SVD

We compared the sample images from MNIST and the result of the preprocessing data. The result between the original features of the images can be shown in 5, and the reconstructed image with 2 number of components can be shown in 4. It gives the blurry images with lower number of components, and more detail with higher number of components. It's just has reduced rank of the matrix, so the value of the pixels is lower than before. The image dimension also not reduced, it stays with 32×32 since we don't use principal components analysis (PCA) approach but low-rank approximation.

The compression size is about $(32 \times 2) + 2 + (2 \times 32)$ or 130. We can compare to actual images that has 32×32 or 1024. So, the compression ratio is $\frac{1024}{130}$ or 7.87. It shows that with 2 number of components we can reduce about 7.87 times of the file size.

[8]http://yann.lecun.com/exdb/mnist/

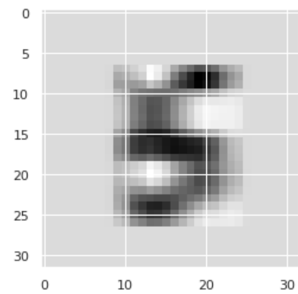

Fig. 4. Reconstructed Sample Image

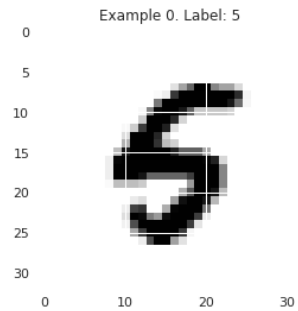

Fig. 5. Original MNIST Sample Image

B. Training Model

We trained the model with original data and SVD data with 50 epoch. As we used the validation data, we can test the trained model on every epoch and give the performance. The accuracy without SVD can be shown in 6 and the accuracy with SVD and use 6 number of components can be shown in 7

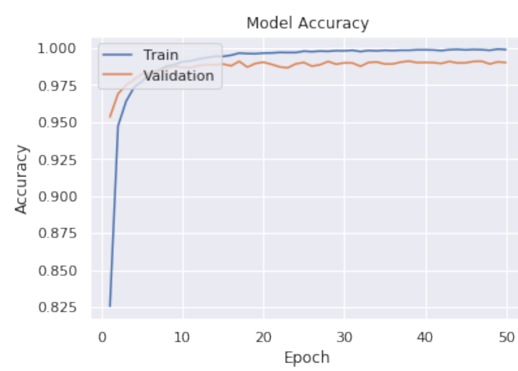

Fig. 6. Accuracy of Trained Model without SVD

We focused on validation data not on the train data itself, since train data had been seen by our model. The highest validation accuracy is 99.12% until 38 epoch on model without SVD, and 99.12% until 38 epoch on model with SVD. So, we can choose the best model according to the epoch is 38 epoch.

Fig. 7. Accuracy of Trained Model with SVD

From the result, we can conclude that SVD model can reach same accuracy on validation data even with lower-rank data.

C. Testing Model

We tested the model with data test without SVD and with SVD that use 6 number of components. The model without SVD got 98.99% meanwhile 98.76% with SVD, it was quite different accuracy.

We want to know also the performance with different number of components, so we observed the 2,4,6 .. 30 number of components and did trained model then evaluate it on every number of components. In figure 8 shows the performance of the model with different number of components. With 10 number of components, it gives the 99.03% accuracy.

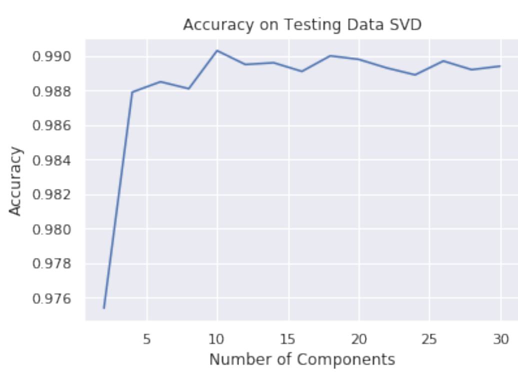

Fig. 8. Accuracy of Trained Model with SVD on Test Data

V. Conlusion

In this paper, we proposed the preprocessing method of SVD with lower-rank approximation on MNIST dataset using LeNet-5 improvement model. Based on experiment, the SVD improves the performance of the model with the 10 number of components and gives 99.03% accuracy. Also the compression ratio with 2 number of components can reduce about 7.87 times of the file size.

References

[1] M. Mijwil, “Handwriting recognition methods,” 09 2015.
[2] R. Plamondon and S. N. Srihari, “Online and off-line handwriting recognition: a comprehensive survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.
[3] M. Cokun, . yldrm, A. Uar, and Y. Demir, “An overview of popular deep learning methods,” *European Journal of Technic*, vol. 7, pp. 165–176, 12 2017.
[4] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[5] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
[6] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *Advances in neural information processing systems*, 1990, pp. 396–404.
[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[8] A. Alsaafin and A. Elnagar, “A minimal subset of features using feature selection for handwritten digit recognition,” *Journal of Intelligent Learning Systems and Applications*, vol. 9, no. 4, pp. 55–68, 2017.
[9] C.-L. Liu and H. Fujisawa, “Classification and learning methods for character recognition: Advances and remaining problems,” in *Machine learning in document analysis and recognition*. Springer, 2008, pp. 139–161.
[10] A. Baldominos, Y. Saez, and P. Isasi, “A survey of handwritten character recognition with mnist and emnist,” *Applied Sciences*, vol. 9, no. 15, p. 3169, 2019.
[11] D. Kalman, “A singularly valuable decomposition: the svd of a matrix,” *The college mathematics journal*, vol. 27, no. 1, pp. 2–23, 1996.
[12] Y. Wang, F. Li, H. Sun, W. Li, C. Zhong, X. Wu, H. Wang, and P. Wang, “Improvement of mnist image recognition based on cnn,” in *IOP Conference Series: Earth and Environmental Science*, vol. 428, no. 1. IOP Publishing, 2020, p. 012097.