Character Recognition using Adjustment Convolutional Network with Dropout Layer

Wahyudi Setiawan
Department of Informatics, Faculty of Engineering, University of Trunojoyo Madura, Bangkalan, Jawa Timur, Indonesia

corresponding author: wsetiawan@trunojoyo.ac.id

Abstract. Character Recognition is the machine's ability to receive and translate certain character input. It can be handwriting, computer-fonts, and images. In this article, character recognition is done using the Adjustment Convolution Network (ACN). The ACN architecture used consists of 28 layers (input, 5 convolutional, 5 Batch Normalization, 5 Rectified Linear Units, 4 max-pooling, 5 dropouts, Fully Connected Layer, Softmax, Output). Dropout is located in each convolution block. The values used are 0.25, 0.3 and 0.4. Dropout keeps away from the overfitting incident. The network is carried out with the optimization of Adaptive Moment Estimation (Adam) which is a combination of momentum and adaptive sub-gradient variable. Data contain digits 0-9, letters A-Z, and letters a-z that amounted to 171,562. Data is taken from MNIST, notMNIST and Char74K public data. A comparison between training vs. testing data is 70:30. The results showed accuracy of up to 99.04% (MNIST), 94.90% (notMNIST), 98.9% (font-Char74K), 79.4% (Handwritten-Char74K), and 86.80% (Image-Char74K) respectively.

Keyword : Adjustment Convolutional Network, Character Recognition, Dropout Layer, Handwritten, MNIST dataset.

1. Introduction
Character recognition (CR) can translate an object character from handwritten, computer-fonts or images into text through a computer. CR is included in the fields of pattern recognition, computer vision, and machine learning. CR technology is generally carried out through the stages of preprocessing, feature extraction, classification, and recognition. CR applications include passport recognition documents [1], banking documentation [2], motor vehicle number recognition [3], traffic sign recognition [4], captcha for anti-boot system [5], historical documents [6], digitizing handwriting document [7], and text-to-speech [8].

In this decade, deep learning used spread widely especially in classification and recognition. One of the methods of deep learning is Convolutional Neural Network (CNN). Research on character recognition that is handwritten recognition using neural network backpropagation with gradient descent, test data using MNIST dataset, the number of neurons is 65. The test produces an accuracy of 96.08% [9]. Furthermore, character recognition uses Support Vector Machines (SVM), Artificial Neural Networks (ANN), Convolutional Neural Networks. The data used is Arabic handwritten. Accuracy of up to 33% (SVM), 37% (ANN) and 71% (CNN) respectively [10]. Next, the study used spiking neural networks with MNIST data. Accuracy is 78.92% [11]. Hereafter, the research used CNN and MNIST dataset. CNN architecture not detail explained, accuracy up to 99.25%. Then, the
study of Wang et al. using the MNIST dataset, the architecture uses modified LeNet5 with gradient
descent, accuracy up to 99.32% [12]. Wen et al. using AlexNet, the dataset uses MNIST with an
accuracy of 99.56% for the training set and 99.23% for the testing set [13].

Previous research was limited to using digit handwritten data with machine learning and deep learning
methods including SVM, ANN, and CNN, so it is necessary to do variations in the form of character
recognition. Addition it is necessary to do adjustment on CNN architecture. The novelty of this
research is first to use variations of datasets such as handwritten digits, computer font characters, and
images. Second, the ACN architecture that uses 28 layers with dropouts on each convolution block.

2. The Material and Method

2.1 Methodology of Research

Generally, the CNN architecture consists of a feature extraction layer (convolutional layer,
activation of ReLU, pooling) and classification layer (fully connected layer, softmax). For
the feature extraction layer, batch normalization and dropout can be included.

As is known, the first dropout was delivered by Hinton et al. and Srivastava et al. to avoid
overfitting on CNN [14] [15]. The concept of dropout is quite simple, it does not reuse randomly
selected neurons. Neurons that are not dropped will continue the next process. Hinton paper explains
the dropout value of 0.5 used at the end of the convolution block before the FCL or dense layer [14].
But there are also use dropouts at the end of each convolution block even with a small value of 0.1 or
0.2. Dropout is placed after the activation function on each convolution block. So the sequence starts
from the convolution layer, ReLU, and dropout [16].

In this research, a stage consisting of input data, then the data is processed on the CNN 28 layer
architecture, as well as an optimization of the weight and bias using Adam. The architecture consists
of an input layer, and 5 convolution blocks (convolutional layer, batch normalization, ReLU
activation, max-pooling, and dropout). In the fifth convolution block without max-pooling.
Furthermore, the classification layer consists of the Fully Connected Layer, Softmax and
Classification Layer as output. The CNN architecture used in this study has 5 dropouts with different
values. Dropout is applied at the end of each convolution block. In blocks 1 to 4, dropouts are placed
after max-pooling, in the fifth block, dropouts are placed after ReLU. Dropout values vary 0.25 in
blocks 1 and 2, dropout 0.4 in block 3, dropout 0.3 in blocks 4 and 5. We call it Adjustment
Convolutional Network (CAN). The ACN architecture is shown in Figure 1.

Figure 1. The Adjustment Convolutional Network architecture 28 layers for character recognition

Next, optimization is done to improve recognition results. Optimization is done by Adam, gradient
descent method. This method was chosen because it is a combination of momentum and adaptive sub-
gradient variables that are often used in optimization [17]. For the variables in the optimization
initialization values include max-epoch 10, learning rate 0.001, validation frequency 10 and minibatch-size 256.

2.2 Dataset
This study, using 3 datasets: MNIST, notMNIST, and Char74K. MNIST is a dataset of handwritten digits containing 60,000 examples with 10 classes. The data has been normalized to a size of 28x28 (a total of 784 pixels) and the object is in the center of the image. The second dataset is notMNIST is a character dataset from letters A to J (10 classes) with a total of 18,725 data. The size of each image in notMNIST is equal to MNIST. While the third dataset, Char74K, has 74,000 examples divided into 62 classes (0-9, A-Z, a-z). A total of 7,410 characters are drawn from natural images, 3,410 are handwritten using tablet PCs and 62,992 characters are computer fonts. An example of the data used for a trial is shown in Figure 2.

![Visualisasi dataset dari Mnist, NonMnist dan Char74K](image)

The testing is divided into 11 scenarios (each of scenario has 10 character class): testing using MNIST data digits 0-9, notMNIST letters AJ, Char74K: Computer font digits 0-9, Computer font AJ, Computer font aj, Handwritten digits 0-9, Handwritten letters A-J, Handwritten letters a-j, Image 0-9, Image A-J, and Image a-j.

3. Result and Discussion

3.1. Results
Performance Measures using MNIST, notMNIST and Char74K data are shown in Table 1, while the result for Char74K data shows in Table 2. The Graphics of Performance Measure shown in Figure 3.

| Dataset | MNIST Digit 0-9 | NotMNIST Huruf A-J | Char74K Font Computer | Char74K Huruf | Char74K Huruf a-j |
|---------|-----------------|---------------------|-----------------------|--------------|------------------|
| P       | R               | A                   | P                     | R            | A                |
| 1       | 99.5            | 99.2                | 99.35                 | 98.4         | 96.75            |
| 2       | 99.4            | 99.7                | 99.55                 | 96.1         | 95.4             |
| 3       | 98.6            | 99.2                | 98.90                 | 94.1         | 95.45            |
| 4       | 99.2            | 99.1                | 99.15                 | 96.6         | 93.3             |
| 5       | 99.6            | 98.5                | 99.05                 | 94.9         | 94.95            |
| 6       | 98.2            | 99.5                | 98.85                 | 95.9         | 96.4             |
| 7       | 99.2            | 99.1                | 99.15                 | 94.8         | 93.7             |
| 8       | 99.4            | 98.3                | 98.85                 | 94.5         | 96.5             |
| 9       | 99.3            | 98.9                | 99.1                  | 92.9         | 92.9             |
| 10      | 97.9            | 99.45               | 94.8                  | 94.5         | 94.65            |

Note: P (Precision), R(Recall), A(Accuration) in %
Figure 3. Performance Measure Graphics from dataset (a) MNIST, (b) notMNIST dan (c) Digit-Font-Char74K (c) Digit-Handwritten-Char74K (c) Digit-Image-Char74K
The results show that the best accuracy achieved by MNIST data is 99.55%, while for notMNIST, the best accuracy is 99.20%. For Char74K, the accuracy of using data fonts, handwritten and digit images reached 98.7%, 88.55%, and 83.8% respectively. MNIST data shows the best accuracy because the data has been normalized, the image size is equal to others, and the amount of data is relatively large at 60,000. The Char74K data is needed to do preprocessing, normalization of size and even more data. The original data can be augmented. For the highest and lowest recognition results shown in Table 3.

### Table 2. Performance Measure of Handwritten and Image-Char74K

| Dataset          | Handwritten-Char74K | Image-Char74K |
|------------------|---------------------|---------------|
| Digit 0-9        | A                   | A-J           |
| Digit 0-9        | A                   | A-J           |
| P R A            | P R A               | P R A         |
| 1 87.5 77.8 82.65 | 87.5 75 65.65     | 93.4 94.5     |
| 2 87.5 77.8 82.65 | 75 70.6 62.5 76.9 | 79.2 45.2     |
| 3 75 100 87.5 81.3 | 72.2 76.75 71.4 93.8 | 82.6 93.8 91.7 |
| 4 87.5 77.8 82.65 | 84 68.8 68.8 68.8 | 66.7 66.7 66.7 |
| 5 56.3 81.8 69.05 68.8 | 78.6 73.7 74.8 81.3 | 81.3 70.2 64.3 |
| 6 93.8 83.3 88.55 87.5 | 87.5 73.7 83.9 83.8 | 87.5 100 93.75 |
| 7 75 70.6 72.8 62.5 | 100 61.25 56.3 100 | 73.7 82.4 78.05 |
| 8 81.3 92.9 87.1 87.5 | 82.4 84.95 81.3 72.2 | 76.75 42.9 66.7 |
| 9 87.5 73.7 80.6 81.3 | 61.9 71.6 87.5 66.7 | 77.1 10 33.3 21.65 |
| 10 62.5 66.7 64.6 62.5 90.9 | 76.7 68.8 73.3 71.05 20 100 60 | 56.5 68.4 62.45 10 100 55 |

Note : C (Class)

### Table 3. The highest and lowest recognition

| Dataset          | Highest Accuracy Identity | Lowest Accuracy Identity |
|------------------|---------------------------|--------------------------|
|                  | Accuration(%) Class       | Accuration(%) Class      |
| MNIST            | 99.55 2 "I"              | 98.45 10 "B"            |
| notMNIST         | 96.15 6 "F"              | 92.9 9 "I"              |
| Digit Font-Char74K | 98.7 3 "2"            | 93.35 10 "F"            |
| A-J FontComp Char74K | 99.2 1 "A"           | 91.65 9 "I"             |
| a-j FontComp Char74K | 99.85 5 "e"           | 98.1 6 "f"              |
| Digit Handwritten Char74K | 88.5 6 "g"            | 64.6 10 "g"             |
| A-J Handwritten Char74K | 87.5 1 "A"           | 71.6 9 "I"              |
| a-j Handwritten Char74K | 87.5 1 "a"            | 65.6 6 "f"              |
| Digit Image Char74K | 83.8 6 "g"            | 21.65 9 "F"             |
| A-J Image Char74K  | 93.95 1 "A"            | 57.15 2 "B"             |
| a-j ImageChar74K  | 80.9 1 "a"             | 47.55 3 "c"             |

The best recognition result is character "I" in the MNIST dataset which is 99.55%. The character "A" produces the best accuracy (three times) in each dataset, while for the characters "S" and "a" two times. For characters "I", "F", "2", and "e" one time produces the highest accuracy. While the lowest accuracy is the character "9" in the Digit Image Char74K dataset which is 21.65%. Character "9" and "I" get the lowest accuracy as much as three times, while the character "F" has the lowest accuracy two times and the characters "8", "B" and "C" one time get the lowest accuracy. Furthermore, the average Performance Measure is shown in Table 4.

### 3.2. Discussion

In this research, ACN architecture is applied with 28 layers consisting of an input layer, 5 block convolutional layers with dropouts on each block and FCL and softmax on the classification layer. The filter sizes used in the multilevel convolutional layer start at 32, 64, and 128 in the last 3 blocks. The dropout process varies, smaller at the beginning of the convolutional layer and larger at the middle and the end of the convolutional layer.
Now, we have analyzed the dataset. The number of training data in MNIST in each class is more than 4,000 except in the 5th grade the number is 3,795. But this does not affect the results of accuracy, because the amount of data used is relatively large for each class. For notMNIST data, the amount of data in each class is 1,872 to 1,873. The shape of data characters that are diverse and more complex when compared to MNIST makes the classification results lower. Besides, the amount of data used for the training process is also less than MNIST which can lead to lower recognition accuracy.

For Char74K, data divided into 3 types: computer-fonts, handwritten and images. Font data has a size of 128x128 with the number of each class was 1,016. Handwritten data has 55 images in each class. Image data has unbalanced data from 33 to 558. It causes a lack of accuracy in Char74K, the amount of data is relatively small and also the invisible character data is also more complex than the previous two datasets.

The recommendation that can be produced from this research: First, the data used must be increasingly more, if possible use thousands of data. If the amount of data is lacking, data augmentation can be done by modifying data, for example through rotation, translation, flip, scale, or giving noise. Second, use the same data size to compare. The MNIST and notMNIST datasets use the size 28x28, whereas Char74K uses the size 128x128. The larger data size affects the resulting performance measure. Third, use the preprocessing process and image enhancement especially on Image-Char74K data so that the tested data can produce better accuracy.

| Scenario | Dataset  | Data form     | Precision (%) | Recall (%) | Accuracy (%) |
|----------|----------|---------------|---------------|------------|--------------|
| 1        | MNIST    | Digit 0-9     | 99.03         | 99.05      | 99.04        |
| 2        | notMNIST | Font A-J      | 94.87         | 94.91      | 94.90        |
| 3        |          | Computer-Font Digit 0-9 | 96.70 | 96.88 | 96.70 |
| 4        |          | Computer-Font A-J | 96.08 | 96.33 | 96.10 |
| 5        |          | Computer-Font a-j | 98.85 | 98.87 | 98.90 |
| 6        |          | Handwritten Digit 0-9 | 79.39 | 80.24 | 79.40 |
| 7        | Char74K  | Handwritten A-J | 77.52 | 79.70 | 77.50 |
| 8        |          | Handwritten a-j | 73.16 | 75.17 | 73.13 |
| 9        |          | Digit Image 0-9 | 59.22 | 71.79 | 66.70 |
| 10       |          | Image A-J     | 76.81         | 87.12      | 86.80        |
| 11       |          | Image a-j     | 56.33         | 81.16      | 73.10        |

4. Conclusion

The character recognition has been carried out in digits 0-9, letters A-J, letters a-z consisting of handwriting, computer-fonts, and images. The process is carried out using the ACN 28 layer architecture with a dropout on each convolution block. The results showed 99.04% accuracy in MNIST datasets, 94.90% for notMNIST data, 98.9% for font-Char74K data, 79.4% for Handwritten-Char74K and 86.80% for Image-Char74K.

References

[1] K. Bulatov, V. V Arlazarov, T. Chernov, O. Slavin, and D. Nikolaev, “Smart IDReader: Document Recognition in Video Stream,” in 14th IAPR International Conference on Document Analysis and Recognition Smart, 2017, pp. 39–44.
[2] T. J. Saleem and M. A. Chishti, “Assessing the Efficacy of Machine Learning Techniques for Handwritten Digit Recognition,” Int. J. Comput. Digit. Syst., vol. 2, no. 2, pp. 299–308, 2020.
[3] V. Dalarmelina, M. A. Teixeira, and R. I. Meneguette, “A Real-Time Automatic Plate Recognition System Sensor Networks for ITS,” no. 1, 2020.
[4] G. Villalonga and J. Van De Weijer, “Recognizing New Classes with Synthetic Data in the Loop: Application to Traffic Recognition,” pp. 1–21, 2020.
[5] T. Garg, “Novel Framework For Handwritten Digit Recognition Through Neural,” 3C Tecnol. Glosas innovación Apl. a la pyme. Edición Espec. Mayo, no. May, pp. 448–467, 2019.
[6] L. Ma, C. Long, L. Duan, X. Zhang, and Q. Zhao, “Segmentation and Recognition for Historical Tibetan Document Images,” IEEE Access, vol. PP, p. 1, 2020.

[7] A. Kumar et al., “Handwriting Recognition in Low-resource Scripts using Adversarial Learning,” in IEEE Explore, 2019, pp. 4767–4776.

[8] C. Zito, F. Tesser, M. Nicolao, and P. Cosi, “Statistical Context-Dependent Units Boundary Correction For Corpus-Based Unit-Selection,” 2020.

[9] M. Kaur, T. Garg, R. Wason, and V. Jain, “Novel framework for handwritten digit recognition through neural networks,” 3C Tecnol. innovación Apl. a la pyme, no. May, pp. 448–467, 2019.

[10] A. Challa, “Automatic Handwritten Digit Recognition On Document Images Using Machine Learning Methods,” Karlkrona, Sweden, 2019.

[11] B. Li et al., “A Spiking Neural Network Model Mimicking the Olfactory Cortex for Handwritten Digit Recognition,” 2019 9th Int. IEEE/EMBS Conf. Neural Eng., pp. 1167–1170, 2019.

[12] Y. Wang, F. Li, H. Sun, W. Li, C. Zhong, and X. Wu, “Improvement of MNIST Image Recognition Based on CNN,” in IOP Conference Series: Earth and Environmental Science, 2020, vol. 428, pp. 1–8.

[13] Y. Wen, Y. Shao, and D. Zheng, “A Novel Deep Convolutional Neural Network Structure for Off-line Handwritten Digit Recognition,” in ICBDT2019. 2019, pp. 1–5.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” Toronto, Ontario, Canada, 2012.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, and Y. Bengio, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” J. Mach. Learn. Res., vol. 15, no. 2014, pp. 1929–1958, 2014.

[16] S. Park and N. Kwak, “Analysis on the Dropout Effect in Convolutional,” in AACV2016, 2017, pp. 189–204.

[17] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in ICLR, 2015, pp. 1–15.