Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 21(2): 167-173, Jun. 2023 **Regular paper**

Handwritten Hangul Graphemes Classification Using Three Artificial Neural Networks

Aaron Daniel Snowberger[1] and Choong Ho Lee[1]*, *Member, KIICE*

[1]Department of Information and Communication Engineering, Hanbat National University, Daejeon 34158, Republic of Korea

Abstract

Hangul is unique compared to other Asian languages because of its simple letter forms that combine to create syllabic shapes. There are 24 basic letters that can be combined to form 27 additional complex letters. This produces 51 graphemes. Hangul optical character recognition has been a research topic for some time; however, handwritten Hangul recognition continues to be challenging owing to the various writing styles, slants, and cursive-like nature of the handwriting. In this study, a dataset containing thousands of samples of 51 Hangul graphemes was gathered from 110 freshmen university students to create a robust dataset with high variance for training an artificial neural network. The collected dataset included 2200 samples for each consonant grapheme and 1100 samples for each vowel grapheme. The dataset was normalized to the MNIST digits dataset, trained in three neural networks, and the obtained results were compared.

Index Terms: Handwriting Recognition, Hangul Graphemes, Hangul Recognition, Neural Networks

I. INTRODUCTION

Hangul was developed in 1443 to help improve the literacy of the Korean people, who until then had used classical Chinese characters. The modern Korean alphabet consists of 24 basic letter shapes: 14 consonants and 10 vowels. These letters can be combined to create 27 additional graphemes. Table 1 displays the 51 Hangul graphemes.

Table 1. Modern Hangul alphabet

Type	Graphemes
Basic consonants	ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
Double consonants	ㄲ ㄸ ㅃ ㅆ ㅉ
Complex consonants	ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ
Basic vowels	ㅏ ㅑ ㅓ ㅕ ㅗ ㅛ ㅜ ㅠ ㅡ ㅣ
Complex vowels	ㅐ ㅒ ㅔ ㅖ ㅘ ㅙ ㅚ ㅝ ㅞ ㅟ ㅢ

Hangul graphemes are stacked within syllabic blocks to form characters in up to nine combinations that always begin with an initial consonant. A vowel grapheme is second to the

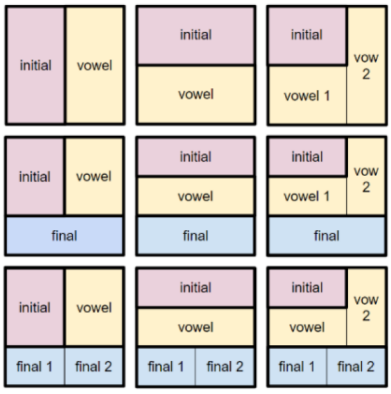

Fig. 1. Hangul grapheme combinations to create characters.

Received 23 February 2023, **Revised** 08 May 2023, **Accepted** 09 May 2023
***Corresponding Author** Choong Ho Lee (E-mail: chlee@hanbat.ac.kr)
Department of Information and Communication Engineering, Hanbat National University, Daejeon 34158, Republic of Korea

Open Access https://doi.org/10.56977/jicce.2023.21.2.167 print ISSN: 2234-8255 online ISSN: 2234-8883

right or below the initial consonant. Optional ending consonant graphemes may also be included. Fig. 1 shows how Hangul graphemes form syllabic blocks.

The combination of the 51 Hangul graphemes in these nine arrangements results in 11,172 characters. However, so many combinations are problematic when performing machine learning classifications [1]. Therefore, many Hangul datasets include only samples of the 2350 most common characters [2-4].

However, using only the most basic 24 Hangul letters for handwritten OCR is difficult. This is because some of the double and complex letters may be written in such a way that individual letters cannot be distinguished.

Therefore, in this study, we did not use the most basic 24 Hangul letters nor the 2350 most common pre-combined Hangul characters. Instead, we gathered and used all 51 Hangul graphemes to train three different neural networks for classification. The three neural networks included a multilayer perceptron (MLP), a convolutional neural network (CNN), and a recurrent neural network (RNN).

II. RELATED WORK

Research on handwritten Hangul recognition has been conducted since the early 1990s [5], beginning with a focus on stroke recognition [6-7]. Later works included the use of Hidden Markov Models [8] and, more recently, artificial neural networks [9] for character classification.

However, because there are 11,172 possible Hangul characters, previous studies focused on the pre-combined 2350 Hangul letters [9-10] of the Korean Industrial Standard Codes KSC5601 [11]. This approach was used to create three Hangul datasets: PE20 [2], SERI95 [3], and PDH08 [4].

However, performing OCR on handwritten Hangul using only approximately 20% of all the possible characters is a very limited approach. Therefore, it is reasonable to segment the Hangul characters into graphemes for OCR. Two studies have shed light on possible methods. Dziubliuk et al. [12] discussed syllable recognition in two dimensions using separable blocks and self-attention along horizontal and vertical image directions. Kim et al. [13] suggested using 51 Hangul consonant and vowel graphemes, rather than only 24 basic consonants and vowel letters, for classification.

III. SYSTEM MODEL AND METHODS

A. Dataset Creation

Handwritten Hangul samples were collected from 110 freshmen at J University. Overall, 2200 samples were collected for each consonant grapheme and 1100 samples were collected for each vowel grapheme. This produced a dataset of 89,100 Hangul graphemes. Handwritten samples were collected from students using the following procedure.

1) Data Collection

Each student was given a grid paper with rows designated to write different graphemes. Consonant rows contained 20 boxes and vowel rows contained 10 boxes. Initially, complex consonants that serve as character endings, called batchim, were not collected. These data were collected later with a second paper. After the students filled out the papers, they were scanned as PDF files and exported to individual JPG

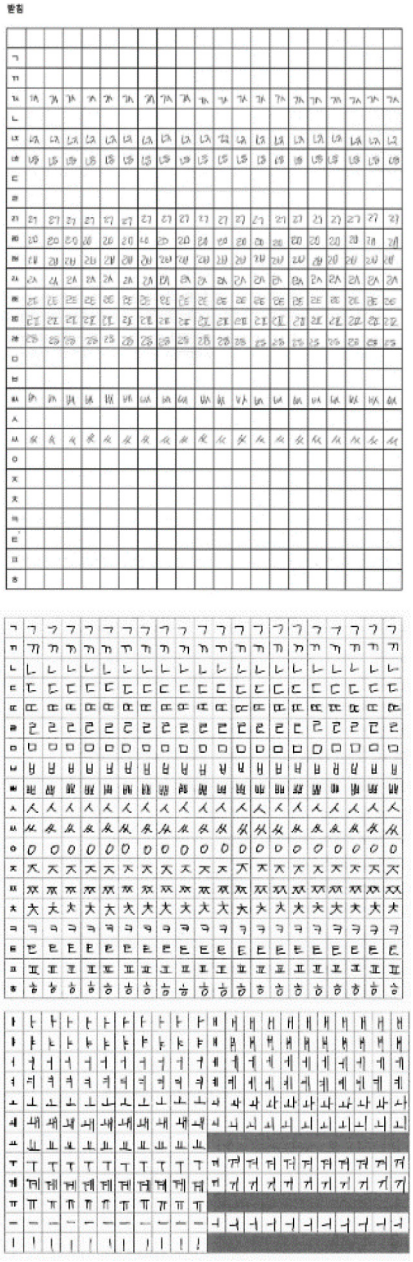

Fig. 2. Example of papers used to collect handwritten Hangul data. Batchim (ending consonant graphemes) are present in the second page.

files. Fig. 2 shows a sample of the papers used to collect data from the students.

2) Data Preprocessing

After handwritten samples were gathered from the students, the grid images were cropped into individual graphemes using a modified table cell detection algorithm [14]. The images were loaded with OpenCV, converted to grayscale, and binarized with THRESH_BINARY using a threshold value between 250 and 255. The binarized images were inverted, and vertical and horizontal kernels were applied to the inversed binary images using OpenCV's morphologyEx function. The results were dilated to isolate the grid lines surrounding each grapheme. Finally, the connectedWithStats function was used to save each grapheme into an identifying folder for each page.

3) Data Reorganization and Cleanup

After saving all the graphemes per page, the images were mathematically sorted into folders based on their names. In other words, for every consonant grapheme, groups of 20 images were moved into their numerically determined folder. For every vowel grapheme, groups of 10 images were moved into their numerically determined folder. Individual grapheme folders were inspected manually for errors.

If any image was incorrectly moved into a folder, it was manually relocated to the correct folder. In some instances, the Python cropping algorithm incorrectly cropped the images. Sometimes graphemes overlapped the grid lines, causing a single grapheme to be cropped two or three times. Partial graphemes were combined in Photoshop to recreate the original handwritten graphemes. Other images containing overlapping strokes or other smudges and marks were cleaned using Photoshop to remove noise.

4) Dataset Normalization

Individual graphemes, distributed into grapheme folders, were normalized to the MNIST digits dataset [15]. A Python program was developed to loop through each image in a folder and process all folder images. Each image was loaded in grayscale and then enhanced by five using the Python Image Library's (PIL) ImageEnhance function. Pixel values less than 80 (on a 0-255 scale) were binarized to black (0) and all pixel values greater than 80 were binarized to white (1). The binarized images were inverted, and the Python Image Library getbbox method was used to isolate the bounding box containing the white pixels of each Hangul grapheme. The images were then cropped according to the bounding box dimensions. The cropped image was resized to 20 pixels on its long side and padding was added to create square 28 × 28-pixel images. Fig. 3 shows the procedure for normalizing the dataset. Fig. 4 shows a sample of the original dataset above the normalized dataset.

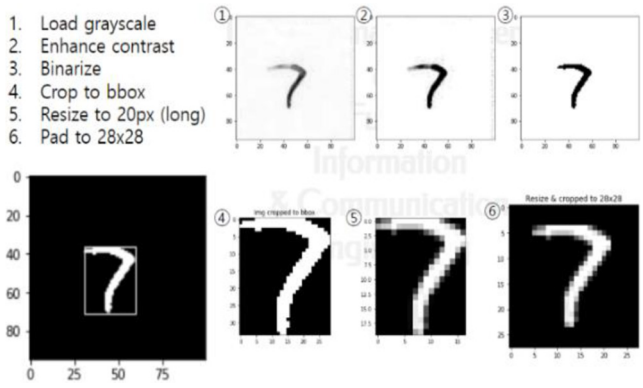

Fig. 3. Dataset normalization steps.

Fig. 4. Original handwritten Hangul dataset (top) and normalized Hangul graphemes dataset (bottom).

The resulting Normalized Hangul Graphemes Dataset was then randomly split into 'Train' and 'Test' folders. The 'Train' folder contained 1800 of each consonant grapheme and 900 of each vowel grapheme, i.e., approximately 82% of the data. The 'Test' folder contained 400 of each consonant grapheme and 200 of each vowel grapheme, i.e., approximately 18% of the data.

B. Artificial Neural Network Training

A Python algorithm was developed to load the Normalized Hangul Graphemes Dataset images and folder labels into two NumPy arrays. This was performed to produce the same data structure as the MNIST digits dataset provided by the Keras

API [16]. The images were loaded as inverted grayscale images (Fig. 4), and the pixel values were normalized between 0 and 1. The labels were integer-encoded (0-50). Additionally, an array of label strings was created for visualization.

Finally, the datasets were trained using three simple artificial neural networks as presented by Atienza [17], i.e., an MLP, a CNN, and an RNN. The neural networks were intentionally kept small to reduce the training time and serve as guidelines for future research. Each network was trained for 20, 50, and 100 epochs and the prediction results were compared. The details of each network model, hyperparameters, and training are provided below.

1) MLP

The MLP uses three dense layers with rectified linear unit activation and a dropout of 0.45 between each dense layer. A softmax activation function is used for the third dense output layer. The network was trained with a batch size of 128 and 256 hidden units. The Adam optimizer was used with a categorical cross-entropy loss function and accuracy metrics. The MLP has 279,859 trainable parameters. Fig. 5 summarizes the MLP model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               200960

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 256)               65792

 activation_1 (Activation)   (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 51)                13107

 activation_2 (Activation)   (None, 51)                0

=================================================================
Total params: 279,859
Trainable params: 279,859
Non-trainable params: 0
_________________________________________________________________
```

Fig. 5. MLP network model.

2) CNN

The CNN uses three convolutional layers with rectified linear unit activation and max pooling between each layer. The output of the final convolutional layer is flattened, and a dropout of 0.2 is applied before data is sent to the final dense layer. The softmax activation function is used for the dense layer. The network was trained with a filter of 64, batch size of 128, kernel size of 3, and pool size of 2. The CNN used the Adam optimizer with categorical cross-entropy loss and accuracy metrics. The CNN has 103,923 trainable parameters, making it less than 40% the size of the MLP. Fig. 6 summarizes the CNN model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 max_pooling2d (MaxPooling2D  (None, 13, 13, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dropout (Dropout)           (None, 576)               0

 dense (Dense)               (None, 51)                29427

 activation (Activation)     (None, 51)                0

=================================================================
Total params: 103,923
Trainable params: 103,923
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. CNN network model.

3) RNN

The RNN uses a SimpleRNN layer with 256 units and a dropout rate of 0.2. The input was a 28-dimension vector with 28 timestamps for the 28 × 28-pixel image inputs. The output from the SimpleRNN layer is sent to the final dense layer with softmax activation. The model was trained using the categorical cross-entropy loss function, stochastic gradient descent as the optimizer, and accuracy metrics. The RNN has only 86,067 trainable parameters, making it the smallest network. Fig. 7 summaries the RNN model.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_1 (SimpleRNN)    (None, 256)               72960

 dense_1 (Dense)             (None, 51)                13107

 activation_1 (Activation)   (None, 51)                0

=================================================================
Total params: 86,067
Trainable params: 86,067
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. RNN network model.

IV. RESULTS

Each neural network model was trained for 20, 50, and 100 epochs, and evaluated using the test dataset. Table 2 lists the prediction accuracy of each model after each training session.

Table 2. Neural network prediction accuracy for MLP, CNN, and RNN

Network	20 epochs	50 epochs	100 epochs
MLP	87.7%	89.3%	89.6%
CNN	96.2%	96.1%	96.5%
RNN	82.9%	89.7%	90.5%

The predictions of the Test set were also visualized for each model. Figs. 8-10 show the prediction results of each model after training for 100 epochs. One image per class is displayed, with the predicted classes on the left and actual classes in parentheses on the right. Incorrectly classified image classes are marked in red.

The classification report of SKLearn was used to examine the prediction precision for each label. Table 3 lists the results for each network after training for 100 epochs. High precision scores (>0.95) are in bold, and the lowest scores (<0.80) are underlined.

The precision scores in Table 3 help distinguish the easiest or most difficult graphemes to classify.

Generally, consonant graphemes with complex endings (batchim) were classified most successfully. For example, classes 2 (ㄲ), 4 (ㄴㅈ), 5 (ㄴㅎ), 9 (ㄹㄱ), and 19 (ㅄ) all had high precision scores over 0.95 for all three networks. However, if the individual consonant graphemes that comprise these ending consonants are considered separately, none have very high precision scores. One possible reason is that

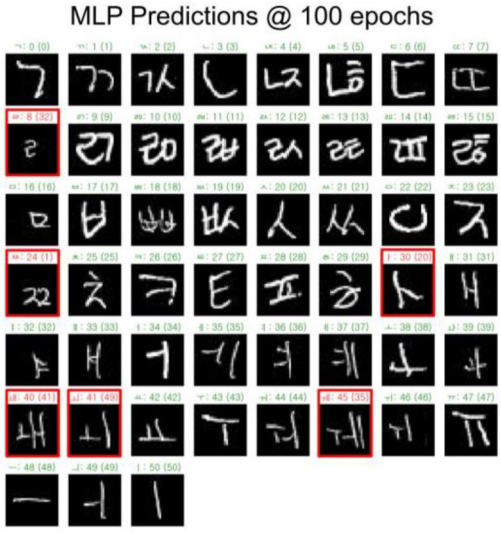

Fig. 8. MLP network predictions after training for 100 epochs.

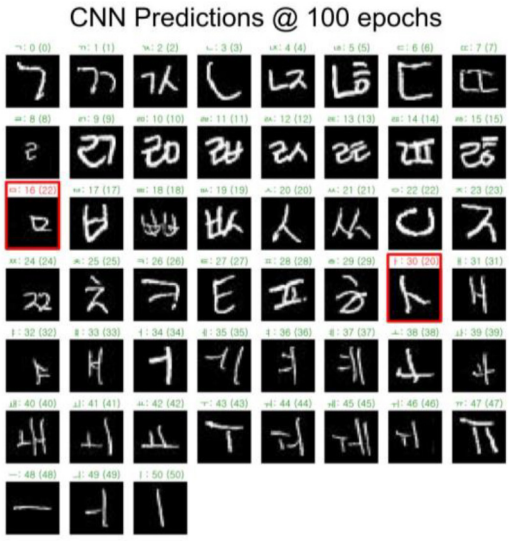

Fig. 9. CNN network predictions after training for 100 epochs.

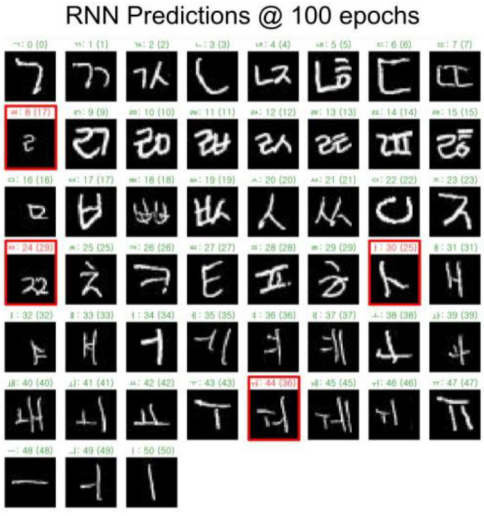

Fig. 10. RNN network predictions after training for 100 epochs.

because complex-ending graphemes contain more lines and shapes than simpler graphemes, neural networks are better able to learn the distinguishing features.

However, the simplest graphemes, with the smallest number of lines and shapes, were more difficult to classify. For example, class 34 (ㅕ) had precision scores of less than 0.80 in all three networks. Additionally, sloppily written graphemes were misclassified. For example, in class 30 (ㅏ), the horizontal line is written such that it was misclassified as class 20 (ㅅ) by all three networks. However, sloppily written letters are always possible in handwriting.

Other misclassified graphemes fall into one of two categories. First, visually similar graphemes, such as class 44 (ㅝ) and class 46 (ㅟ), with only a single stroke difference, were

Table 3. Hangul handwriting neural network training precision for MLP, CNN, and RNN

Class	MLP	CNN	RNN
ㄱ (0)	0.90	**0.97**	**0.99**
ㄲ (1)	0.88	**0.97**	0.91
ㄳ (2)	**0.95**	**0.98**	**0.96**
ㄴ (3)	0.93	**0.98**	**0.98**
ㄵ (4)	**0.97**	**0.99**	**0.98**
ㄶ (5)	**0.95**	**0.98**	**0.95**
ㄷ (6)	0.86	0.94	**0.99**
ㄸ (7)	0.89	**0.98**	0.89
ㄹ (8)	0.90	0.94	0.92
ㄺ (9)	**0.98**	**0.99**	**0.97**
ㄻ (10)	0.90	**0.97**	0.90
ㄼ (11)	**0.97**	**0.98**	0.93
ㄽ (12)	**0.96**	**0.99**	0.93
ㄾ (13)	0.91	**0.98**	0.91
ㄿ (14)	0.86	0.94	0.89
ㅀ (15)	0.90	**0.96**	0.91
ㅁ (16)	0.85	**0.96**	0.91
ㅂ (17)	0.89	0.91	0.92
ㅃ (18)	0.94	**0.99**	0.88
ㅄ (19)	**0.96**	**0.99**	**0.96**
ㅅ (20)	0.91	**0.96**	0.94
ㅆ (21)	0.90	**0.97**	0.88
ㅇ (22)	0.88	**0.95**	0.90
ㅈ (23)	0.82	0.94	0.94
ㅉ (24)	0.85	**0.98**	0.84
ㅊ (25)	0.94	**0.97**	0.89
ㅋ (26)	0.87	**0.96**	0.91
ㅌ (27)	0.85	**0.95**	0.88
ㅍ (28)	0.84	**0.98**	0.89
ㅎ (29)	0.82	0.93	0.81
ㅏ (30)	0.81	0.94	**0.98**
ㅐ (31)	0.76	0.91	0.77
ㅑ (32)	0.91	**1.00**	0.93
ㅒ (33)	0.86	**0.98**	0.85
ㅓ (34)	0.74	0.79	0.74
ㅔ (35)	0.75	**0.95**	0.81
ㅕ (36)	0.80	**0.96**	0.81
ㅖ (37)	0.92	**0.99**	0.85
ㅗ (38)	0.87	0.92	0.90
ㅘ (39)	0.85	0.94	0.90
ㅙ (40)	0.93	**0.98**	0.94
ㅚ (41)	0.86	0.91	0.90
ㅛ (42)	0.91	0.91	0.87
ㅜ (43)	0.89	**0.95**	0.92
ㅝ (44)	0.89	**0.95**	0.93
ㅞ (45)	0.95	**0.99**	0.86
ㅟ (46)	0.84	**0.96**	0.88
ㅠ (47)	0.88	0.93	0.79
ㅡ (48)	0.94	**0.99**	**0.99**
ㅢ (49)	0.79	0.75	0.81
ㅣ (50)	0.92	**0.95**	**0.97**
Overall	**0.88**	**0.96**	**0.77**

sometimes misclassified. However, a larger network may be better able to distinguish between the two. Second, incorrectly normalized graphemes, such as classes 8 (ㄹ) and 24 (ㅉ) in Figs. 8 and 10, respectively, were also misclassified. This incorrect normalization sometimes occurs with graphemes that contain extra marks or noise after the binarization phase of preprocessing. Fig. 11 shows an example of this error.

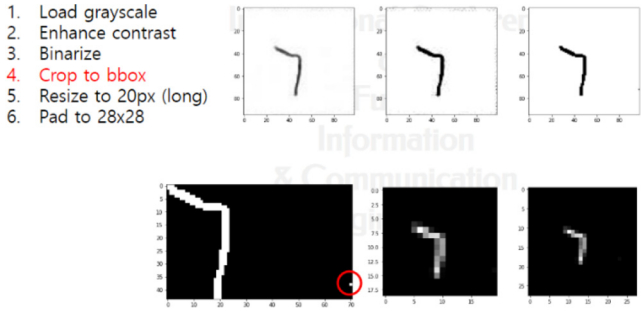

Fig. 11. Example of an incorrectly cropped and centered grapheme.

V. DISCUSSION AND CONCLUSIONS

This study demonstrates a process for gathering handwritten Hangul data and normalizing a dataset for classification using artificial neural networks. Overall, 89,100 grapheme images were collected across 51 classes. Each consonant grapheme had 2200 samples, and each vowel grapheme had 1100 samples. The collected data were normalized to the MNIST digit dataset and trained using three artificial neural networks. The MLP, CNN, and RNN were trained for 20, 50, and 100 epochs. Three important points regarding the training should be noted.

First, the prediction accuracy of MLP and CNN on the test dataset did not increase significantly beyond the first 20 epochs of training. Second, although the accuracy of the RNN was initially the worst and improved the fastest, its rate of improvement slowed at 100 epochs. This indicates that there is a limit to its accuracy, even with longer training. Larger networks may be better able to learn the features of graphemes and achieve higher accuracy. Third, the CNN performed much better overall than any of the other networks, even with only 10 epochs of training, resulting in 95.5% accuracy. Therefore, deep CNNs are the best networks for training in future studies.

REFERENCES

[1] P. D. Moral, S. Nowaczyk, and S. Pashami, “Why is multiclass classification hard?,” *IEEE* Access, vol. 10, pp. 80448-80462, 2022. DOI: 10.1109/ACCESS.2022.3192514.

[2] D. H. Kim, Y. S. Hwang, S. T. Park, E. J. Kim, S. H. Paek, and S. Y.

Bang, "Handwritten Korean character image database PE92," *Proceedings of 2nd International Conference on Document Analysis and Recognition* (ICDAR '93), Tsukuba, Japan, pp. 470-473, 1993. DOI: 10.1109/ICDAR.1993.395693.

[3] I. J. Kim, "HangulDB," GitHub repository, 2019, [Online] Available: https://github.com/callee2006/HangulDB.

[4] D. S. Ham, D. R. Lee, I. S. Jung, and I. S. Oh, "Construction of printed hangul character database PHD08," *The Journal of the Korea Contents Association*, vol. 8, no. 11, pp. 33-40, Nov. 2008. DOI: 10.5392/jkca.2008.8.11.033.

[5] E. J. Kim and Y. Lee, "Handwritten hangul recognition using a modified neocognitron," *Neural Networks,* vol. 4, no. 6, pp. 743-750, Jan. 1991. DOI: 10.1016/0893-6080(91)90054-9.

[6] P. K. Kim, J. K. Lee, and H. J. Kim, "Handwritten Korean character recognition by stroke extraction and representation," in *TENCON '93. IEEE Region 10 International Conference on Computers, Communications and Automation*, Beijing, China, pp. 1098-1101, 1993. DOI: 10.1109/TENCON.1993.320070

[7] P. K. Kim and H. J. Kim, "Off-Line handwritten Korean character recognition based on stroke extraction and representation," *Pattern Recognition Letters,* vol. 15, no. 12, pp. 1245-1253, Dec. 1994. DOI: 10.1016/0167-8655(94)90115-5.

[8] W.S. Kim and R. H. Park, "Off-line recognition of handwritten Korean and alphanumeric characters using hidden markov models," *Pattern Recognition,* vol. 29, no. 5, pp. 845-858, May. 1996. DOI: 10.1016/0031-3203(95)00124-7.

[9] I. J. Kim and X. Xie, "Handwritten Hangul recognition using deep convolutional neural networks," *International Journal on Document Analysis and Recognition*, vol. 18, no. 1, pp 1-13, Mar. 2015. DOI: 10.1007/s10032-014-0229-4.

[10] S. W. Park, "A study on the OCR of Korean sentence using deep learning," in *Annual Conference on Human and Language Technology*, Daejeon, South Korea, pp. 470-474, 2019.

[11] Korea Industrial Standards Association, "Code for Information Interchange (Hangul and Hanja)," *Korean Industrial Standard*, 1987, Ref. No. KS C 5601-1987.

[12] V. Dziubliuk, M. Zlotnyk, and O. Viatchaninov, "Sequence learning model for syllables recognition arranged in two dimensions," in *Document Analysis and Recognition - ICDAR 2021*, vol. 12823, pp. 100-111, DOI: 10.1007/978-3-030-86334-0_7.

[13] G. U. Kim, Son J. M., K. H. Lee, and J. S. Min, "Character decomposition to resolve class imbalance problem in Hangul OCR," *arXiv preprint arXiv*: 2208.06079, Aug. 2022. DOI: 10.48550/arXiv.2208.06079.

[14] A. R. Sreekiran, *Checkbox/Table Cell Detection Using OpenCV-Python*, 22 Nov. 2022, [Online] Available: https://towardsdatascience.com/checkbox-table-cell-detection-using-opencv-python-332c57d25171.

[15] Y. LeCun, C. Cortes, *MNIST handwritten digit database*, ATT Labs, 2010, [Online] Available: http://yann.lecun.com/exdb/mnist/.

[16] "MNIST digits classification dataset", Keras API, 2022, [Online] Available: https://keras.io/api/datasets/mnist/.

[17] R. Atienza, *Advanced Deep Learning with TensorFlow 2 and Keras*, 2nd ed., Packt Publishing Ltd., Birmingham, Feb. 2020, [Online] Available: https://www.packtpub.com/book/programming/9781838821654/.

Aaron Daniel Snowberger

Aaron Daniel Snowberger is a Ph. D. candidate majoring in Information Communication Engineering at Hanbat National University, Korea. He received his B. S. degree in Computer Science from the College of Engineering, University of Wyoming, USA in 2006 and his M. FA. degree in Media Design from Full Sail University, USA in 2011. He has been at Jeonju University since 2010 and is presently a professor in the Department of Liberal Arts. His research interests include computer vision, natural language processing, image processing, signal processing, and machine learning.

Choong Ho Lee

Choong Ho Lee received his Ph.D. in Information Science from Tohoku University, Sendai, Japan in 1998, and his M.S. and B.S. degrees from Yonsei University, Seoul, Korea, in 1985 and 1987, respectively. He has worked for KT from 1987 to 2000 as a researcher. He has been at Hanbat National University, Daejeon, Korea since 2000. Presently, he is a Professor of the Department of Information Communication Engineering. His research interests include digital image processing, computer vision, machine learning, big data analysis, and software education.