

Active Learning on Neural Networks through Interactive Generation of Digit Patterns and Visual Representation

Dong H. Jeong

*Department of Computer Science and Information Technology
University of the District of Columbia
Washington D.C., USA
djeong@udc.edu*

Jin-Hee Cho

*Department of Computer Science
Virginia Tech.
Falls Church, USA
jicho@vt.edu*

Feng Chen

*Department of Computer Science
University of Texas at Dallas
Richardson, USA
feng.chen@utdallas.edu*

Audun Jøsang

*Department of Informatics
University of Oslo
Oslo, Norway
audun.josang@mn.uio.no*

Soo-Yeon Ji

*Department of Computer Science
Bowie State University
Bowie, USA
sji@bowiestate.edu*

Abstract—Artificial neural networks (ANNs) have been broadly utilized to analyze various data and solve different domain problems. However, neural networks (NNs) have been considered a black box operation for years because their underlying computation and meaning are hidden. Due to this nature, users often face difficulties in interpreting the underlying mechanism of the NNs and the benefits of using them. In this paper, to improve users' learning and understanding of NNs, an interactive learning system is designed to create digit patterns and recognize them in real time. To help users clearly understand the visual differences of digit patterns (i.e., 0 ~ 9) and their results with an NN, integrating visualization is considered to present all digit patterns in a two-dimensional display space with supporting multiple user interactions. An evaluation with multiple datasets is conducted to determine its usability for active learning. In addition, informal user testing is managed during a summer workshop by asking the workshop participants to use the system.

Index Terms—Active Learning, Neural Networks, Visualization, Digit Patterns

I. INTRODUCTION

Artificial Intelligence (AI) has significant impacts in many disciplines, such as medicine [1], education [2], and earthquake [3], and so forth. AI enables machines to learn data to perform various tasks to function like humans [4]. Examples of AI-enabled technologies include abnormal behavior detection in banking transactions, home protection using camera-based monitoring and object recognition, customized learning mechanisms providing tailored lessons to students, and driving assistance with upgraded and autonomous driving technologies. AI also has received much attention as a tool in education to improve students' learning and knowledge acquisition [5].

Despite the powerful capability of AI and its numerous applications in our daily lives, AI has been known as a black

box tool because it is not easy to understand the logic behind its internal computations [6]. This motivated researchers to design trustworthy and interpretable AI systems [7], [8]. However, it is still difficult to understand AI systems due to the well-known issue of transparency [9] because machine or deep learning algorithms used in AI models are highly complex. For instance, neural networks (NNs) or deep NNs (DNNs) often include thousands of artificial neurons to learn from and process large amounts of data. Because of the numerous neurons and their complex interconnections, it is difficult to determine how decisions are made [10]. Understanding how the AI models work and generate resulting predictions is a critical step in interpreting the meaning of AI's outcomes. But, it remains challenging to understand data using a predictive model that finds patterns from training the data and analyzes the difference between predictions and the patterns.

This paper aims to integrate AI technologies into learning through hands-on practices. Specifically, designing an interactive learning system is considered to assist users in understanding NNs clearly through multiple learning activities. To support an interactive learning environment on NNs, we have designed the system with a pattern generator, where a user can generate digit patterns. In addition, providing a visual representation of the patterns is considered to help the user understand the differences among the generated patterns. Whenever the user creates patterns, the user can experience NN training and recognition in real time. To represent the patterns in a 2D display space, Principal Component Analysis (PCA) was used to reduce the dimensions of the patterns and present them in a lower-dimensional space [11], [12]. By interactively navigating the display space, the user can identify the similarities and differences between the patterns. To evaluate the system's effectiveness for active learning,

2400-digit patterns are generated and used to test the system. A broadly known handwritten digits dataset (called MNIST) is also used to determine the capability of supporting a real-time interactive digits analysis. To understand the usefulness of the system, informal user testing was organized during a summer workshop by asking the workshop participants to use the system. We found that they understood NNs well by initiating active learning with the system.

The rest of this paper is structured in six sections. Section II provides previous studies on designing educational systems for understanding AI. In Section III, the designed system is explained. Section III includes a detailed explanation about the applied neural networks and visual representation. Section V shows the conducted evaluation of interactive learning on recognizing digits in real time. After discussing interesting insights in Section VI, we conclude this paper by providing possible future work in Section VII.

II. RELATED WORK

Due to high interest in AI, most education institutes, including colleges or high schools, have introduced new AI degree programs [13]. AI has become a powerful paradigm in scientific research communities due to its diverse applications in broad and various domains [14]. Due to this popularity, many students have shown a strong interest in understanding AI. In particular, they have exposed their high interest in deep learning (DL) because it has been commonly used to detect complex patterns in high-dimensional data with little or no human interventions. However, understanding the underlying ideas of the output prediction in DL is not trivial due to the black-box nature of the AI models [15]–[17]. Li and et al. [18] explored various visualization techniques to understand the structure of neural loss functions and their effectiveness. Chatzimpampas and et al. [19] emphasized how important information visualization is in understanding machine learning (ML) models and enhancing trust in ML. Although they highlighted the importance of utilizing visualization in ML, their primary considerations fell into addressing specific domain problems instead of helping students understand the internal computation of ML.

Computer science education researchers have developed various tools to improve students' knowledge of AI technologies. Marinescu-Istodor and Jormanainen [20] developed a web-based tool for high school students to enhance their knowledge in recognizing objects using ML. They designed the tool to identify objects using a camera and determine their object classes in real time based on training samples. In this tool, when a student gives a wrong answer, the student sees a question mark rather than a message saying the answer is wrong. If an object has been misclassified (i.e., the student says a wrong answer), the tool could fix such a mistake by correctly training and classifying its class name with additional samples. The authors aimed to design the tool to motivate students by improving their class engagement. You and Yin [15] developed a device (called Omega) to enhance college students' understanding on NNs by addressing the black box nature

and representing their interactions during NN training steps. In particular, the device visually presented the weight changes in hidden layers during the NN training. Lamy and Tsopra [16] introduced a visual translation of simple NNs to prove the visual interpretation using rainbow boxes with adding interactive functionality. Kim and Shim [21] emphasized the need of providing AI education for non-engineering major students by creating a visual solution. Although numerous studies have been conducted to design practical approaches to improve students' learning, most studies mainly aimed to teach users NN training steps.

Unlike the existing approaches explained above, our study differs in that the proposed interactive learning system enables users to create input data patterns and train NNs interactively. This will significantly increase the user's learning and knowledge gained on NNs because it supports real-time computation and recognition of the user-generated digit patterns.

III. SYSTEM DESIGN

We hypothesized that supporting real-time interactive data generation, training, and recognition through NNs could increase users' understanding of the underlying idea of NNs. Based on this hypothesis, we have designed an active learning system (named Neural Network Trainer) to support the user in generating digit patterns and recognizing them with NNs. We also designed an additional system (called Neural Network Tester) to evaluate multiple user-generated patterns simultaneously. For supporting active learning in the system, integrating a graphical user interface (GUI) was considered to address the advancement of users' understanding of NNs through direct interactions with the system.

The Neural Network Trainer system consists of two layouts – Digit Pattern Generator (Figure 1a-left) and Visual Analyzer (Figure 1a-right). Digit Pattern Generator includes a pattern grid (Figure 1a-(i)) with multiple control panels (Figure 1a-(ii) ~ 1a-(v)). The pattern grid allows the user to create digit patterns (i.e., 0 ~ 9) by clicking each cell in the pattern grid. It has 12×8 grid cells representing a digit pattern. Each cell holds binary information as 1 or 0. It shows the size of NNs, including nodes in input, hidden, and output layers (Figure 1a-(ii)). Two list boxes have been added to keep all created digit patterns and the total number of patterns representing each digit in Figure 1a-(iii) and 1a-(iv), respectively. Real-time training and testing of NNs are handled with the control buttons (Figure 1a-(v)). The result of the recognized digit pattern with NN appears with probability distributions (Figure 1a-(vi)). Visual Analyzer represents user-generated digit patterns in a 2D display space by applying PCA computation.

The Neural Network Tester system supports evaluating multiple user-generated NNs with various testing datasets. The primary purpose of having the system was to help the workshop participants understand the performances of their generated NNs in recognizing digits competitively with others. Figure 1b demonstrates the evaluation of three NNs created by three groups of users. Similar to the digit pattern generator, it has a pattern grid (Figure 1b-(i)) with multiple control

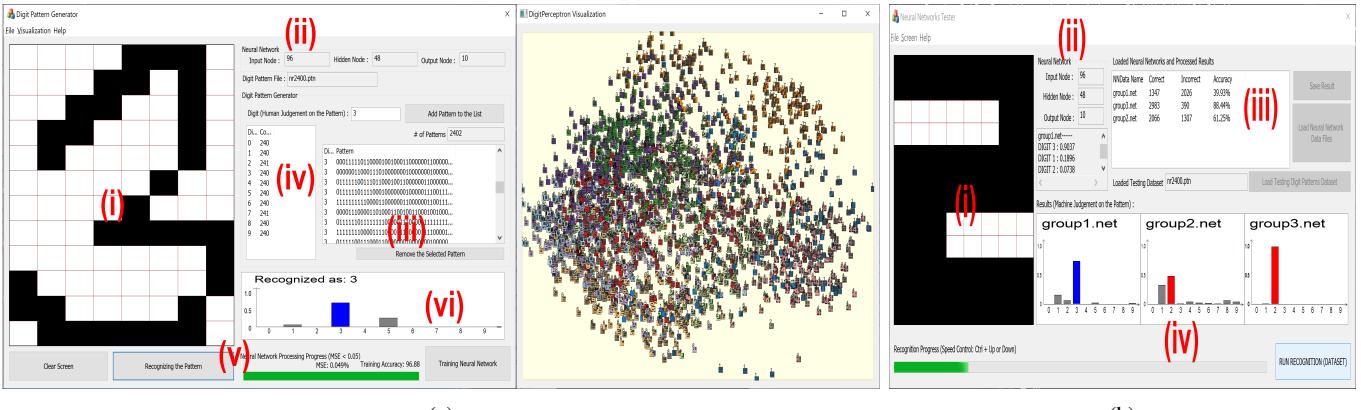


Fig. 1: Two systems are designed as (A) neural network trainer and (B) neural networks tester. The neural network trainer system consists of two layouts – digit pattern generator (left) and visual analyzer(right). The digit pattern generator supports the user in generating digit patterns and training neural networks. The visual analyzer represents user-generated digit patterns on a PCA projection space. The neural networks tester system evaluates multiple user-generated patterns with testing datasets.

panels (Figure 1b-(ii) ~ 1b-(iv)). A list box (Figure 1b-(iii)) shows the loaded user-generated NNs. With a testing dataset, it evaluates the NNs showing overall accuracies (Figure 1b-(iii)) and probability estimation (Figure 1b-(iv)). The probability estimation indicates how each pattern is recognized with each NN. If a digit is recognized correctly, a reddish bar graph is represented. If not, a bluish bar graph is displayed to denote incorrect recognition.

IV. DESIGN OF NEURAL NETWORKS

To support real-time digit pattern generation and recognition, a three-layered NN based on the backpropagation method [22] was used. It feeds error rates back to NNs to optimize weights with optimal values. The input layer has 96 nodes to be matched to the cells of each digit pattern. The output layer has 10 nodes to represent digits 0 through 9. Although one or more hidden layers are often utilized in designing NNs, we have used one hidden layer consisting of 48 nodes in the system for speedy computation. For performance optimization, a gradient descent method was used because it could allow a parameter update of the weights. The sum of the squared error (SSE) was applied as the gradient of loss function L to determine the difference between the predicted (\hat{y}_i) and actual inputs (y_i) by:

$$L = -\frac{1}{N} \sum_{o=0}^N \sum_{j=0}^C (y_{o,j} - \hat{y}_{o,j})^2 \quad (1)$$

where N is the length of digit samples, C is the number of classes, and $y_{o,j}$ is an observation y_o with a class j .

To run NNs, momentum (γ) and learning rate (η) are defined to accelerate the training speed and accuracy of NNs. Momentum is a method that expedites the gradient descent by increasing the step size toward global minima. It is critical to find an optimal momentum value because a too-large value may skip global minima, or a too-small value may face

local minimum issues. The learning rate controls how quickly a model adapts to the problem of training digit patterns. However, similar to tuning the momentum value, using an optimal learning rate is critical because it impacts the speed of the convergence to a solution and whether we can reach global optima. ΔW_{ij} and ΔW_{ij}^{t-1} represents weight changes in current and previous training iterations. They are given by:

$$\Delta W_{ij} = \gamma \Delta W_{ij}^{t-1} - \eta \frac{\sigma L}{\sigma W_{ij}}, \quad (2)$$

where $\frac{\sigma L}{\sigma W_{ij}}$ denotes the partial derivative of the loss function L to decent update weights with learning rate η with a multiplication of -1 to move towards global minima. The values of γ and η are determined based on empirical analysis for performance optimization in training data [23].

To activate nodes in the NNs, various activation functions are available, such as Sigmoid, ReLU (Rectified Linear Unit), Tanh, or hyperbolic tangent Activation Function. ReLU is a broadly used activation function in convolutional neural networks (CNNs) or deep learning because it supports faster training [24]. However, it often causes a dying ReLU Problem [25] that decreases the ability of training data due to negative values becoming zero. Thus, the Sigmoid function, $\sigma = \frac{1}{1+\exp^{-x}}$, is used in our system. It transforms the weighted sum of nodes to represent the probability of a value x that belongs to a certain class. Although the Sigmoid activation function requires more computation than ReLU, it supports well for training a NN model in our designed system because it consists of a single hidden layer NN. To train NNs, termination condition ϵ is defined as $\epsilon < 0.05$, which reduces the cost function L to become below the threshold. OpenMP API [26] is used to speed up the computation of NNs using multi-processors (i.e., multi-core processors).



Fig. 2: Examples of user-generated digit patterns from the 2400-digit pattern dataset. Each pattern is created using the clickable pattern grid in the Neural Network Trainer system.

A. Pattern Generation

To generate digit patterns, the user can enable or disable each cell using a computer mouse or a touch monitor screen (if available). As mentioned earlier, the initial cell region has 12×8 size that supports creating up to $2^{12 \times 8}$ possible digit patterns. However, the overall number of patterns will be less because preprocessing generates duplicates by making each pattern fit into the cell region boundary. The applied preprocessing consists of three steps: (1) Determining the boundary of each pattern to find an object-bounding box; (2) Moving the pattern to the top-left corner; and (3) Applying scaling to make it fit the cell region. For scaling, a Nearest Neighbor Interpolation algorithm [27] is used because it requires very little calculations. Since each digit pattern has a binary color attribute (i.e., 0 or 1), each cell is marked if the interpolation satisfies the condition $I(x) > 0.5$. To help the user understand the internal preprocessing steps, intermediate outcomes become available only if a tracking option is enabled in the system.

The system supports saving user-generated digit patterns to a file and loading previously generated ones. To validate the effectiveness of the system, 2400-digit patterns are generated. Figure 2 shows samples of the 2400-digit patterns. When loading the previously generated digit patterns from a file, the system detects duplicated patterns and removes them if they exist.



Fig. 3: Conversion of the MNIST handwritten digits from the original images (28×28 gray color) to two-tone colored images (12×8 binary color). The converted dataset is named TT-MNIST.

A handwritten digits dataset, MNIST [28], is also used to evaluate the system. It includes 70,000 grayscale images of handwritten digits with 60,000 training images and 10,000 testing images. Each digit sample is centered at a fixed-size image of 28×28 pixels. To make the images usable in our system, color conversion is applied to make them follow a binary color scheme using two-tone colors, black and white. Then, image size conversion is utilized to scale the image size down to 12×8 to make them fit into the clickable pattern grid in the designed system. Figure 3 shows examples before and after applying the image conversion. Both gray color attribute conversion and image scaling are applied by using nearest neighbor interpolation with referencing neighbor color attributes. If the interpolated value meets the condition, i.e., $I(x) < \delta, 0 \leq \delta \leq 255$, the corresponding color attributes are changed to 0 when $I(x) < \delta$ and 1 when $I(x) \geq \delta$. We empirically determined an optimal value ($\delta = 85$) for converting MNIST digits to gray-colored images. For convenience, we call the user-generated 2400-sample and the converted two-tone colored MNIST images DS-2400 and TT-MNIST, respectively, in the rest of this paper. The two datasets are used to conduct a performance evaluation of the designed system. A detailed explanation about the conducted evaluation study is included in the evaluation section.

B. Visual Representation

As mentioned above, a visualization representation is added to show all digit patterns to help users understand the difference between digits (see Figure 1a-right). Since each digit pattern consists of 96 cells, a dimension reduction technique, PCA is applied to project it in a PCA projection space (i.e., 2D display space). By default, the first and second principal components are used to display each digit pattern along the x - and y -axis.

The system supports basic navigational user interactions (i.e., zooming and panning) to help the user can navigate freely

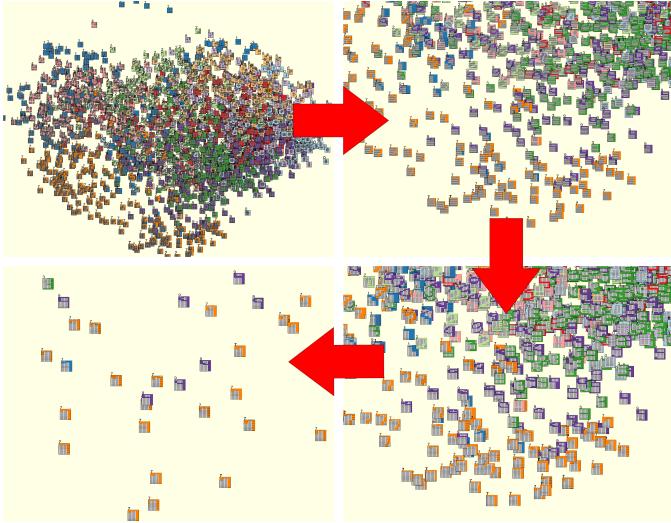


Fig. 4: Zooming and panning user interaction techniques are supported to navigate the PCA projection space. The user performed the zooming user interaction to see the region with the digit “7” patterns (located at the bottom of the space).

within the 2D display space to see the relationship among the digit patterns (see Figure 4). It helps users understand the similarities between the digit patterns and the logic of recognizing them through NNs. Digit patterns maintaining similar cell outlines might appear nearby within the PCA projection space. As shown at the bottom of the visual representation, a digit 7’s all patterns appear in a region. However, some digit 1’s patterns appear near digit 7’s patterns because they maintain similar markers, including vertical down-strokes with distinctive up-strokes on the top. Since digit 1’s patterns do not always include the distinctive up-strokes, they appear in multiple regions. Similarly, different digit’s patterns often appear in the same regions. The hidden layer outputs from trained NNs are used as additional features in PCA computation to create separable projections of digit patterns. Figure 5 shows examples with the DS-2400 and TT-MNIST datasets. To help understand the usefulness of using the hidden layer outputs, randomly selected 940 digit patterns from the DS-2400 dataset are used to generate the projection with and without using the hidden layer outputs as additional features (see Figures 5a and 5b). It shows the benefit of using the hidden layout outputs by forming separated clusters among different digit patterns. With the TT-MNIST dataset, we observed a clear difference in Figures 5c and 5d. For instance, digit “6” patterns were observed in several locations in the PCA space (see Figure 5c). But, with the integration of the hidden layout outputs, the digit patterns were positioned in the same region (see the arrow in Figure 5d).

V. EVALUATION OF THE INTERACTIVE LEARNING SYSTEM

A. Interactive Learning

As discussed earlier, understanding how NNs work is not easy because of the complex nature of computing and updating

its underlying structures continuously. The designed system may help users understand how it trains NNs and recognizes digit patterns. The system does not fully unveil its underlying structure of how the NN model changes its weights over time. However, we can conjecture that it supports interactive learning on NNs. More specifically, interactive learning manages three steps of the learning process: generating digit samples, training a NN model with the samples, and recognizing user-entered digits with the model. Generating digit patterns is essential to understand the effectiveness of NNs because it helps the user to identify how the NNs are trained to recognize digits. However, since it is not easy to create data, most studies have utilized existing datasets (e.g., MNIST, MS-COCO, ImageNet, Fashion-MNIST) to design new NN algorithms and evaluate their performances.

To support the user in creating digit patterns interactively, we used cell-based digit pattern generation to design simplified data digit samples using a computer mouse. The system allows training NNs whenever the user generates digit pattern(s). Unlike conventional approaches using numerous data samples, our system can train NNs with a small number of digit samples (e.g., < 10). For instance, if a digit sample (denoting digit “one”) is applied to train NNs, the same result (resulting digit “one”) will be determined. Instead, if two distinctive digit samples (e.g., “one” and “two”) are used to train NNs, the system correctly recognizes their differences. For example, if the user tries to recognize a new input pattern (similar to “one” or “two” digit patterns), the system correctly recognizes it as either one or two. Figure 6 shows an example of recognizing a new digit pattern with four digit samples. Even though only four-digit patterns are used to train NNs, it correctly recognizes the new pattern with a high probability (0.93). The user can continuously add new digit patterns to improve the performance of recognizing digits interactively. This interactive digit pattern generation and recognition initiate active learning to help the user understand the logic behind NNs. For showing the probability, the probability distribution over all predicted classes is measured using a softmax function. It converts a vector of K values in the output layer to probability values. To show a normalized probability $[0, 1]$ from the output value, the softmax function (σ) applies the exponential function.

$$\sigma(\vec{x}_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (3)$$

where \vec{x}_i indicates the values in the NN output layer, e^{x_i} and e^{x_j} denote standard exponential function for output vector, respectively.

B. Performance Evaluation

To support interactive learning, it is vital to maintain real-time training of NNs with the system while maintaining high accuracy. We performed an evaluation with Intel i9-9980HK Processor, 2.4 GHz, 8 cores. Figure 7 shows that training time with the DS-2400 dataset was about 8.14 ± 1.89 seconds. With the DS-2400 dataset, the average training time was maintained to be less than 0.5 seconds. At the same time,

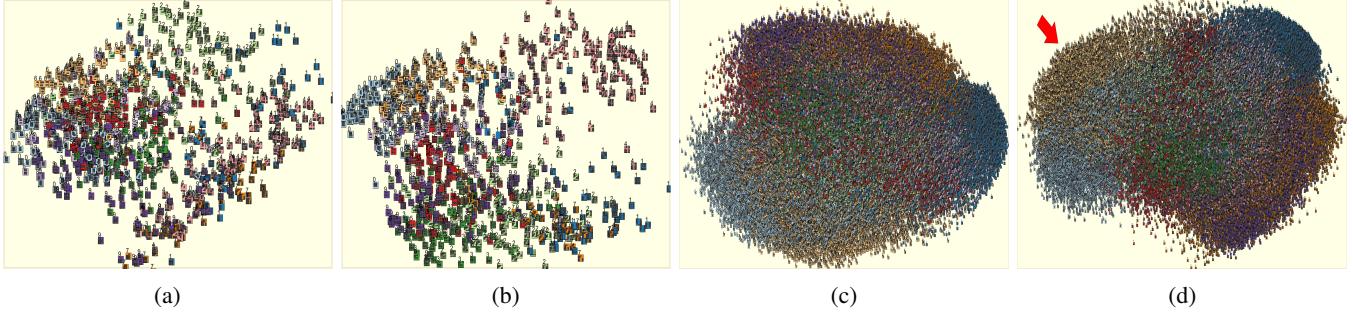


Fig. 5: Visual representations of 940 digit patterns from the DS-2400 dataset in (a) and (b) and TT-MNIST dataset in (c) and (d). (a) and (c) show PCA projections using all digit patterns. (b) and (d) use hidden layer outputs as additional features.

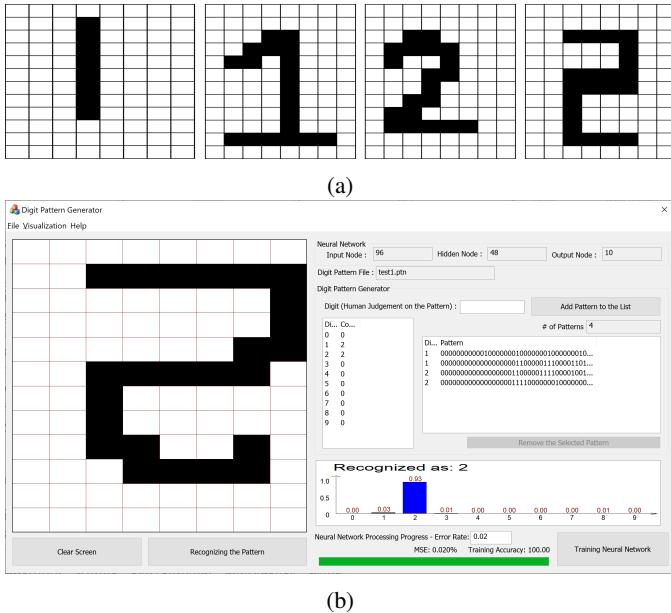


Fig. 6: An example of NN training with four-digit patterns (a) and recognizing a new digit with the trained NN (b).

the training accuracy was above 97%. This indicates that the system supports users in performing real-time interactive analysis on digit recognition by training NNs. With the TT-MNIST dataset, we found that the training accuracy was maintained above 96%. The training time gradually increases as the size of the samples grows. Approximately 70 seconds have been taken to train 60,000-digit patterns. Overall, the proposed system takes less than 2 seconds to train up to 10,000 digit patterns (training with $MSE < 0.05$, resulting in a training accuracy of 0.98). Since the system supports real-time training on user-generated digit patterns, we can consider the system effectively helps the user understand how NNs work to effectively recognize the digits.

VI. DISCUSSION

As we mentioned above, the system is helpful for users to understand the logic behind NNs in recognizing digits. To understand how effective the system is in enhancing users'

learning on NNs, we utilized the system during the workshop¹ for community college students. Most of the students do not have any knowledge or experience using NNs or related applications. They showed high interest in creating digit patterns and training NNs to recognize digits. Three groups were formed. They created patterns spending about 10 minutes (see Table I). To understand the effectiveness of user-generated digit patterns, we tested the user-generated NNs using both the DS-2400 and TT-MNIST datasets. Although the testing accuracy was not high, we found that the trained NN by Group2 (using only 40 digit patterns) showed about 0.5 testing accuracy for the DS-2400 dataset.

TABLE I: Testing the NNs created by workshop participants with the two datasets (i.e., DS-2400 and TT-MNIST).

	Group1	Group2	Group3
Generated Patterns	29	40	13
Testing Accuracy (with DS-2400)	0.29	0.50	0.32
Testing Accuracy (with TT-MNIST)	0.26	0.30	0.20

The students commented that the system was highly interactive and useful for them to understand the underlying idea of NNs. They also reported the importance of utilizing both interactive pattern generation and visualization to upgrade their knowledge of NNs. Since the evaluation through workshop participants is purely informal user testing, it is essential to conduct formal user testing to examine the system's effectiveness in enhancing the user's understanding levels. Therefore, we plan to extend our study to conduct a formal user evaluation to validate the usefulness of the developed system.

Although the system is useful for advancing users' understanding of NNs through interactive learning, it has a limitation of not showing the connection weights in NNs. Although representing the weight changes does not deliver additional information about understanding the internal changes of NNs, many researchers have emphasized the effectiveness of showing them [15], [29]. Thus, designing an effective visual representation technique to show the connection weight changes in NNs is critical for advancing the users' knowledge

¹NSF funded 2022 Artificial Intelligence Awareness summer workshop: <https://csit.udc.edu/mudl/>

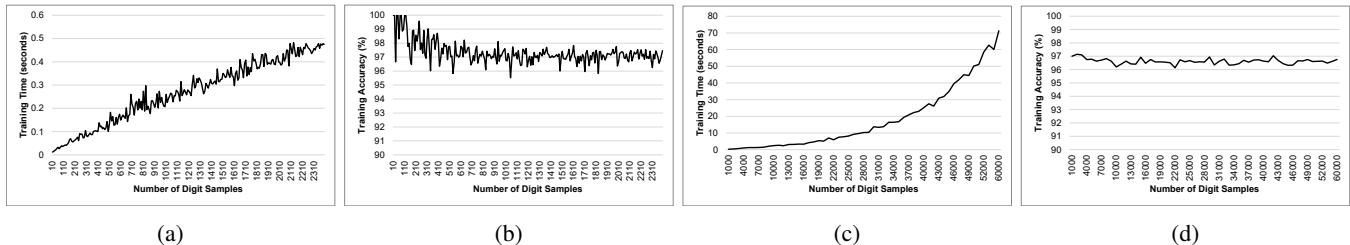


Fig. 7: Training time (seconds) and accuracy with different sizes of training data with (a) and (b) the DS-2400 dataset and (c) and (d) the TT-MNIST dataset. The x -axis shows the training digit sample size. (a) and (c) represent training time (seconds) and (b) and (d) show training accuracy.

of what information is effective in recognizing digits by NNs. It is important to note that PCA has an inherent ambiguity in the signs of resulting principal components. Thus, it generated multiple sign-flipped visual representations (see Figure 8).

VII. CONCLUSION AND FUTURE WORK

In this paper, we designed an interactive learning system to help users understand how neural networks perform digit recognition. The developed interactive learning system allows users to generate digit patterns and train them to recognize them in real time. To support real-time training and recognition, we introduced a simplified neural network with backpropagation to the system. Most importantly, we applied a visualization technique to show the difference among the digit patterns in a PCA projection space. In our experiments, we demonstrated the computational speed of training neural networks to evaluate the system's effectiveness. The key findings from this study are: (1) The developed interactive learning system took a short training time, which is critical for users to learn and understand NNs in real time; (2) The training accuracy was high (e.g., $96 \sim 97\%$) that validates the accuracy of the developed system as a tool to train NNs; and (3) Through our informal user testing based on the responses from community college students participated in the summer AI workshop, we received highly positive feedbacks although the number of the participants was fairly small (about ten participants).

For future work, we plan to conduct formal user testing to determine the system's effectiveness in terms of how much the user can understand the principle of neural networks. Since the system has been designed as a stand-alone application, a conversion of the system to a web-based application will be performed to make it become broadly available and accessible through a web browser. We will also extend the visual representation of digit samples with different dimensional reduction techniques, such as t-distributed stochastic neighbor embedding (t-SNE) [30] or Uniform Manifold Approximation and Projection (UMAP) [31]. The complete codes and a pre-compiled executable are available at <https://github.com/drjeong/DigitPerceptron>

REFERENCES

- [1] D. D. Miller and E. W. Brown, "Artificial intelligence in medical practice: the question to the answer?" *The American journal of medicine*, vol. 131, no. 2, pp. 129–133, 2018.
- [2] B. du Boulay, "Recent meta-reviews and meta-analyses of aid systems," *International Journal of Artificial Intelligence in Education*, vol. 26, no. 1, pp. 536–537, 2016.
- [3] E. Karasözen and B. Karasözen, "Earthquake location methods," *GEM-International Journal on Geomathematics*, vol. 11, no. 1, pp. 1–28, 2020.
- [4] Y. Duan, J. S. Edwards, and Y. K. Dwivedi, "Artificial intelligence for decision making in the era of big data—evolution, challenges and research agenda," *International journal of information management*, vol. 48, pp. 63–71, 2019.
- [5] S. A. D. Popenici and S. Kerr, "Exploring the impact of artificial intelligence on teaching and learning in higher education," *Research and Practice in Technology Enhanced Learning*, vol. 12, no. 1, p. 22, Nov 2017.
- [6] F. Wang, R. Kaushal, and D. Khullar, "Should health care demand interpretable artificial intelligence or accept "black box" medicine?" pp. 59–60, 2020.
- [7] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.
- [8] A. Holzinger, M. Dehmer, F. Emmert-Streib, R. Cucchiara, I. Augenstein, J. D. Ser, W. Samek, I. Jurisica, and N. Díaz-Rodríguez, "Information fusion as an integrative cross-cutting enabler to achieve robust, explainable, and trustworthy medical artificial intelligence," *Information Fusion*, vol. 79, pp. 263–278, 2022.
- [9] K. Haresamudram, S. Larsson, and F. Heintz, "Three levels of ai transparency," *Computer*, vol. 56, no. 2, pp. 93–100, 2023.
- [10] Y. Bathaei, "The artificial intelligence black box and the failure of intent and causation," *Harvard Journal of Law & Technology*, vol. 31, p. 889, 2018.
- [11] S. Roweis, "Em algorithms for pca and spca," *Advances in neural information processing systems*, vol. 10, 1997.
- [12] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang, "ipca: An interactive system for pca-based visual analytics," *Computer Graphics Forum*, vol. 28, no. 3, pp. 767–774, 2009.
- [13] W. Tyson, R. Lee, K. M. Borman, and M. A. Hanson, "Science, technology, engineering, and mathematics (stem) pathways: High school science and math coursework and postsecondary degree attainment," *Journal of Education for Students placed at risk*, vol. 12, no. 3, pp. 243–270, 2007.
- [14] Y. Xu, X. Liu, X. Cao, C. Huang, E. Liu, S. Qian, X. Liu, Y. Wu, F. Dong, C.-W. Qiu, J. Qiu, K. Hua, W. Su, J. Wu, H. Xu, Y. Han, C. Fu, Z. Yin, M. Liu, R. Roepman, S. Dietmann, M. Virta, F. Kengara, Z. Zhang, L. Zhang, T. Zhao, J. Dai, J. Yang, L. Lan, M. Luo, Z. Liu, T. An, B. Zhang, X. He, S. Cong, X. Liu, W. Zhang, J. P. Lewis, J. M. Tiedje, Q. Wang, Z. An, F. Wang, L. Zhang, T. Huang, C. Lu, Z. Cai, F. Wang, and J. Zhang, "Artificial intelligence: A powerful paradigm for scientific research," *The Innovation*, vol. 2, no. 4, p. 100179, 2021.
- [15] M. You and J. Yin, "Real-time visualization of neural network training to supplement machine learning education," in *2019 IEEE Integrated STEM Education Conference (ISEC)*. IEEE, 2019, pp. 371–374.

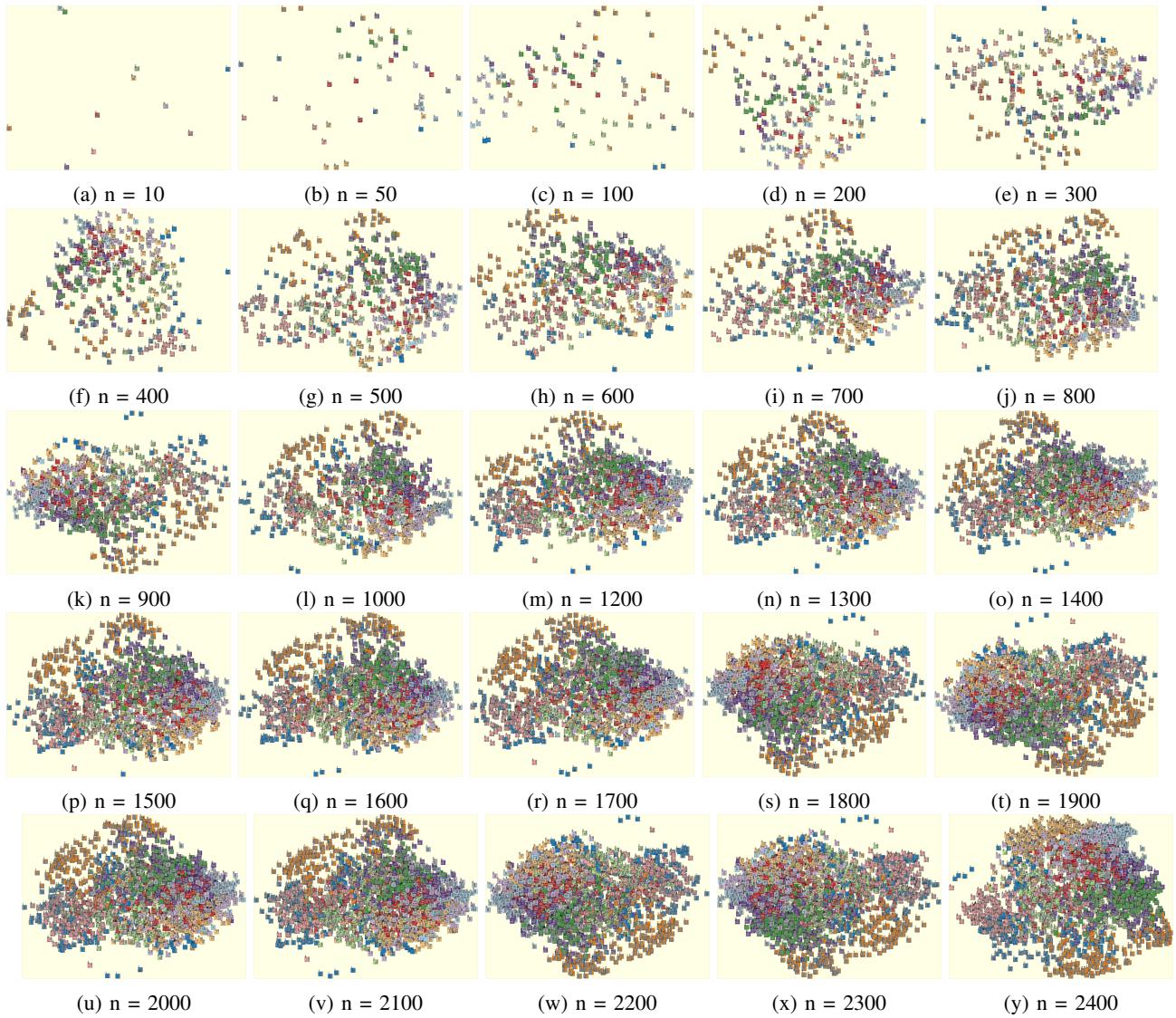


Fig. 8: Visual representations of n numbers of digit patterns from the DS-2400 dataset.

- [16] J.-B. Lamy and R. Tsopra, "Visual explanation of simple neural networks using interactive rainbow boxes," in *2019 23rd International Conference Information Visualisation (IV)*. IEEE, 2019, pp. 50–55.
- [17] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, aug 2018.
- [18] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 6391–6401.
- [19] A. Chatzimparpas, R. M. Martins, I. Jusufi, and A. Kerren, "A survey of surveys on the use of visualization for interpreting machine learning models," *Information Visualization*, vol. 19, no. 3, pp. 207–233, 2020.
- [20] R. Mariescu-Istodor and I. Jormannainen, "Machine learning for high school students," in *Proceedings of the 19th Koli calling international conference on computing education research*, 2019, pp. 1–9.
- [21] J. Kim and J. Shim, "Development of an ar-based ai education app for non-majors," *IEEE Access*, vol. 10, pp. 14 149–14 156, 2022.
- [22] R. Rojas, *Neural Networks - A Systematic Introduction*. Berlin: Springer-Verlag, 1996.
- [23] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478.
- [24] C.-N. Chou, C.-K. Shie, F.-C. Chang, J. Chang, and E. Y. Chang, *Representation Learning on Large and Small Data*. John Wiley & Sons, Ltd, 2019, ch. 1, pp. 1–28.
- [25] L. Lu, Y. Shin, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *Communications in Computational Physics*, vol. 28, no. 5, pp. 1671–1706, 2020.
- [26] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan kaufmann, 2001.
- [27] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008.
- [28] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [29] O. Deperlioglu and U. Kose, "An educational tool for artificial neural networks," *Computers & Electrical Engineering*, vol. 37, no. 3, pp. 392–402, 2011.
- [30] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.
- [31] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.