Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 4472174, 15 pages
https://doi.org/10.1155/2021/4472174

Research Article

An Implicit Memory-Based Method for Supervised Pattern Recognition

Yu Ma,[1,2,3] **Shafei Wang**,[1,4] **Junan Yang**,[5] **Yanfei Bao**,[1] **and Jian Yang**[1]

[1]*Academy of Military Science of the People's Liberation Army, Beijing 100000, China*
[2]*Naval Aviation University, Yantai 264000, China*
[3]*Peng Cheng Laboratory, Shenzhen 518000, China*
[4]*School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*
[5]*Institution of Electronic Countermeasure, National University of Defense Technology, Hefei 230037, China*

Correspondence should be addressed to Shafei Wang; wangshafei1964@126.com

Received 8 May 2021; Revised 24 June 2021; Accepted 9 July 2021; Published 17 July 2021

Academic Editor: Zi-Peng Wang

How the human brain does recognition is still an open question. No physical or biological experiment can fully reveal this process. Psychological evidence is more about describing phenomena and laws than explaining the physiological processes behind them. The need for interpretability is well recognized. This paper proposes a new method for supervised pattern recognition based on the working pattern of implicit memory. The artificial neural network (ANN) is trained to simulate implicit memory. When an input vector is not in the training set, the ANN can treat the input as a "*do not care*" term. The ANN may output any value when the input is a "*do not care*" term since the training process needs to use as few neurons as possible. The trained ANN can be expressed as a function to design a pattern recognition algorithm. Using the Mixed National Institute of Standards and Technology database, the experiments show the efficiency of the pattern recognition method.

1. Introduction

Pattern recognition methods can be divided into two categories: two-stage and end-to-end. Most traditional pattern recognition methods are two-stage: feature extraction and pattern classification [1]. Feature extraction reduces the number of resources that are required to describe a large amount of raw data. The first step is to identify the measurable quantities that make these training sets $\mathbf{X}_1, \ldots, \mathbf{X}_l$ distinct from each other. The measurements used for the classification, such as mean value and the standard deviation, are known as features. In general, some features constitute a feature vector. Some information gets lost since feature extraction is not a lossless compression approach. The lost information cannot be used for pattern recognition. Therefore, how to generate features is a fundamental issue. In the feature vector selection, two crucial issues are the best number of features and the classifier design [2]. For example, feature selection plays an important role in text classification [3]. The complexity of the practice data makes it arduous to use two-stage methods.

Nowadays, deep neural networks can be trained end-to-end [4]. Raw data can reserve all information of the pattern. Inspired by the biological neural networks that constitute animal brains, the artificial neural network (ANN) is applied to do image classification [5], speech separation [6], forest fire prediction [7], etc. However, a major drawback of neural networks is the black-box character. Explaining why the neural networks make a particular decision is formidable. The knowledge representation of neural network is unreadable to humans [8]. The exact reason why trained deep neural networks can implement recognition remains an open question [9]. The training algorithm does not specify the way to recognize. A causal model is formidable to build or acquire. End-to-end learning relies on data for the cognitive task [10, 11], where the data cannot tell the reasons [12].

In a supervised pattern recognition task, a set of training data (training set) is used to train a learning procedure. A training set is a set of instances that are properly labeled by hand with the corrected labels. The learning procedure attempts to recognize the instances as accurately as possible. The goal of the learning procedure is to minimize the error rate on a test set. The question arising in the recognition task is why a new data instance can be classified as a particular category. The problem of supervised pattern recognition can be stated as follows. Assume that the training set $\mathbf{X}_i$ only contains instances with label i where $i \in \mathscr{L} \triangleq \{1, \ldots, l\}$, and $\mathbf{X}_i \cap \mathbf{X}_i = \varnothing$ for any $i \neq j$. Given l training sets $\mathbf{X}_1, \ldots, \mathbf{X}_l$, the question is how to label a new instance $\tilde{\mathbf{x}}$.

A memory system is involved in the process of recognition. Jacoby and Kelly posit that memory can serve as both storage and a tool [13]. Memory is treated as storage in a recall. In this case, the focus is on the past, and memory is used as computer storage. Meanwhile, memory (from experience) can be used as a tool to perceive and interpret present events.

The implicit memory is acquired and used unconsciously and can serve as a tool/function [14–16]. In one experiment, two groups of people are asked several times to solve a Tower of Hanoi puzzle. One group is amnesic patients with heavily impaired long-term memory, and the other is composed of healthy subjects. The first group shows the same improvements over time as the second one, even if some participants claim that they do not even remember seeing the puzzle before. These findings strongly suggest that procedural memory is completely independent of declarative memory [17]. Given a game state, implicit memory is trained to output an operation. Memorizing the solving steps is not necessary. When the state appears again, the input can evoke the trained operator [18].

Usually, humans implement recognition processes unconsciously, and the focus is on current given images. So the memory works as a tool while implementing the recognition processes. The recognition process depends on the similarity comparison between the current input and the labeled instances. The more similar, the more likely they are in the same class and have the same label. However, the way to compare similarities without memorizing any labeled instances is not evident.

This paper proposes an implicit memory-based method for supervised pattern recognition. The method does not memorize or recall any labeled instances and is not in the two-stage or end-to-end categories. The proposed method has interpretability since similarity criteria are used in the process of recognition. A new instance is recognized as a particular class because the instance appears similar enough to the training data of the class. Compared with the k-nearest neighbors algorithm [19], the proposed method does not need to recall and iterate through the training sets. The process is consistent with the human ability of pattern recognition. People may forget most of the training instances, but they can recognize a new instance. The Mixed National Institute of Standards and Technology (MNIST) database (General site for the MNIST database: http://yann.lecun.com/exdb/mnist) is used to verify the proposed method.

The rest of this paper is organized as follows. First, a model is built to describe implicit memory. Second, with the implicit memory model, a recognition algorithm is proposed. Then an application and the analysis of experimental results are given.

Notations: $|\mathbf{X}|$ expresses the cardinality of the set $\mathbf{X}$; $\{0,1\}^n$ is an n-dimensional space constructed by 0 and 1; d is a metric on $\{0,1\}^n$; define a distance function between any point $\mathbf{z}$ of $\{0,1\}^n$ and any nonempty set $\mathbf{X}$ of $\{0,1\}^n$ by

$$d(\mathbf{z}, \mathbf{X}) = \inf\{d(\mathbf{z}, \mathbf{x}) | \mathbf{x} \in \mathbf{X}\}. \tag{1}$$

Given two n-dimensional $\mathbf{a}$ and $\mathbf{b}$, the element-wise product of $\mathbf{a}$ and $\mathbf{b}$, written $\mathbf{a}^{\circ}\mathbf{b}$, is the vector $\mathbf{c}$ with elements given by $c_i = a_i b_i$; $|\mathbf{x}|_{l_1}$ expresses the number of 1 in the binary vector $\mathbf{x}$; the probability of $X = x$ is written as $\Pr\{X = x\}$; $\mathscr{L} \triangleq \{1, \ldots, l\}$; $\mathscr{N} \triangleq \{1, \ldots, n\}$; $\mathscr{P} \triangleq \{1, \ldots, p\}$.

The expression of an inverter (NOT circuit) can be expressed as follows: If the input variable is called Z and the output variable is called Y, then $Y = \overline{Z}$. If $Z = 0$, then $Y = 1$ and if $Z = 1$, then $Y = 0$. The expression of a 2-input AND gate can be expressed in equation form as follows: $Y = Z_1 Z_2$. The output Y of an AND gate is 1 only when both inputs are 1 s. The expression of a 2-input OR gate can be expressed as follows: $Y = Z_1 + Z_2$. The output Y of an OR gate is 1 when any one or more of the inputs are 1 s.

2. Model of Implicit Memory

In this section, a model is built to describe implicit memory. From one given input, implicit memory can give an output. Both the input and output of implicit memory are actual signals. The signals have explicit physical meaning in the real world, such as sound, light, electricity, etc. The types of input-output signals can be the same or different. Meanwhile, the input-output signals can be measured and represented as a binary vector.

The action of implicit memory can be represented as a function ψ: $\mathbf{Z} \longrightarrow \mathbf{Y}$. The domain $\mathbf{Z}$ is a set of binary vectors. The element of $\mathbf{Z}$ represents the input signal of implicit memory. The codomain $\mathbf{Y}$ is also a set of binary vectors. The element of $\mathbf{Y}$ represents the output signal of implicit memory. The set of all ordered pairs $(\mathbf{z}, \mathbf{y})$ represents the training result of implicit memory where $\mathbf{z}$ is an element of $\mathbf{Z}$ and $\mathbf{y}$ is an element of $\mathbf{Y}$. An example of the function is given as follows: the domain is chosen to be the set $\{000, 001, 100, 110\}$, and the codomain is the set $\{01, 10\}$. Figure 1 shows these mapping relationships.

A computer can store all ordered pairs $(\mathbf{z}, \mathbf{y})$ in a database. Given an input, the computer can search the database for output. With the database, the computer can simulate the external behaviors of implicit memory. However, this method needs to recall the ordered pairs stored in the database. This internal process is different from the process of implicit memory. The implicit memory does not need to execute memorizing or searching. The implicit memory is similar in operation to a high-speed logic circuit. From one given input, the implicit memory would not have a second's hesitation of giving output.

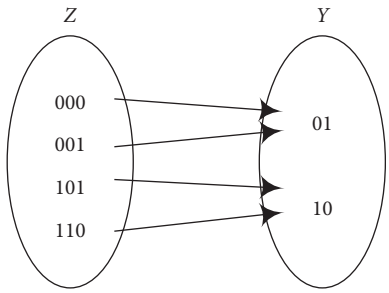

Figure 1: Diagram represents a function with domain $\{000, 001, 101, 110\}$, codomain $\{01, 10\}$, and set of ordered pairs $\{(000, 01), (001, 01), (101, 10), (110, 10)\}$.

A logic circuit and a database have different implementations. In the preparation phase, the database stores the input-output pairs on a hard disk. However, the logic circuit connects the logic gates to implement the input-output maps. At run time, the database searches the given input on the hard disk to find an output. In the logic circuit, nevertheless, the input goes through the logic gates to get the output. The complexity of the practice data also makes it arduous to manually design logic circuits. However, ANN can simulate the actions of implicit memory automatically.

The capacities of the implicit memory can be represented as a set $\mathbf{X} = \{(\mathbf{z}_1, \mathbf{y}_1), \ldots, (\mathbf{z}_n, \mathbf{y}_n)\}$ where $\mathbf{z}_i$ and $\mathbf{y}_i$ are binary vectors. Without loss of generality, assume that all of $\mathbf{z}_1, \ldots, \mathbf{z}_n$ have the same length and all of $\mathbf{y}_1, \ldots, \mathbf{y}_n$ have the same length. And then, the operation of an implicit memory can be expressed with a table. The table lists all allowed input vectors with the corresponding output, as illustrated in Table 1 for an example. The table shows the output for each allowed input. Only the first bit's implementation of the output signal is considered since the other implementations can operate in the same way. The table can serve as a *truth table*.

Truth tables are widely used to describe the operation of logic circuits. With a truth table, the sum-of-products (SOP) expression can be written. The Boolean SOP expression can be obtained from the truth table by ORing the product terms, for which $Y_1 = 1$ is

$$Y_1 = Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3. \tag{2}$$

The first term in the expression is formed by ANDing the three variables Z_1, $\overline{Z}_2$, and Z_3. The second term is formed by ANDing the three variables Z_1, Z_2, and $\overline{Z}_3$. The logic gates can be used to implement the expression. The logic gates are constructed as follows: two inverters to form the $\overline{Z}_2$ and $\overline{Z}_3$ variables; two 3-input AND gates to form the terms $Z_1\overline{Z}_2Z_3$ and $Z_1Z_2\overline{Z}_3$; and one 2-input OR gate to form the final output function, $Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3$. Figure 2 shows the logic diagram. Therefore, a logic circuit can simulate the actions of the implicit memory.

2.1. ANN-Based Boolean Operation. NAND gate is a universal gate because it can be used to produce the NOT, the AND, the OR, and the NOR functions [20]. With NAND gates and appropriate connections, all logic circuits can be built.

Table 1: Example of listing the operation of an implicit memory.

Input			Output	Product term
Z_1	Z_2	Z_3	Y_1	
0	0	0	0	
0	0	1	0	
1	0	1	1	$Z_1\overline{Z}_2Z_3$
1	1	0	1	$Z_1Z_2\overline{Z}_3$

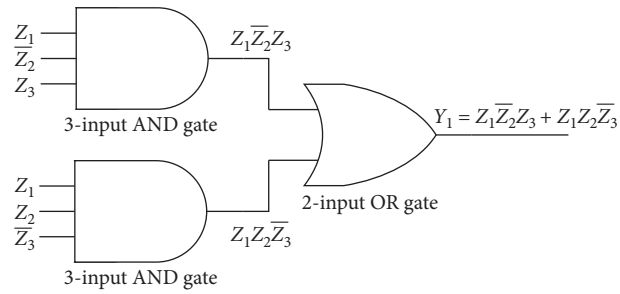

Figure 2: Logic diagram represents $Y_1 = Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3$.

Suppose there is a sigmoid neuron with two inputs, x_1 and x_2. The sigmoid neuron has weights for each input, w_1, w_2, and an overall bias, b. The output of the sigmoid neuron is $\sigma(w_1x_1 + w_2x_2 + b)$, where σ is called the sigmoid function and is defined by

$$\sigma(y) = \frac{1}{1 + e^{y}}. \tag{3}$$

When $w_1 = w_2 = -20$ and $b = 30$, the sigmoid neuron is shown in Figure 3. Then the input 00 produces output 1 since $\sigma(-20 * 0 - 20 * 0 + 30) = \sigma(30) \approx 1$. Similar calculations show that the inputs 01 and 10 produce output 1. But input 11 produces output 0 since $\sigma(-20 * 1 - 20 * 1 + 30) = \sigma(-10) \approx 0$. Therefore, the sigmoid neuron can implement a 2-input NAND gate.

Let the value of w_2 be 0. The output is not affected by the input x_2. The sigmoid neuron equals a 1-input neuron. When $w_1 = -20$ and $b = 10$, the sigmoid neuron is shown in Figure 4. Then input 0 produces output 1 since $\sigma(-20 * 0 + 10) = \sigma(10) \approx 1$. But input 1 produces output 0 since $\sigma(-20 * 1 + 10) = \sigma(-10) \approx 0$. The sigmoid neuron can implement a 1-input NOT gate.

When $w_1 = 20$ and $b = -10$, the sigmoid neuron is shown in Figure 5. Then input 0 produces output 0 since $\sigma(20 * 0 - 10) = \sigma(-10) \approx 0$. But input 1 produces output 1 since $\sigma(20 * 1 - 10) = \sigma(10) \approx 1$. The sigmoid neuron can implement a connecting line.

2.2. Simulation of Implicit Memory. According to De Morgan's laws, Boolean expression can be resolved into 2-input NAND and 1-input NOT. For example,

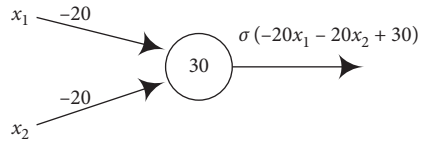

Figure 3: Sigmoid neuron with $w_1 = w_2 = -20$ and $b = 30$.

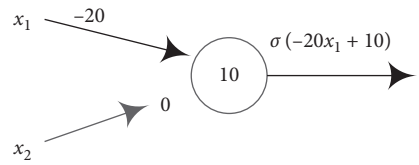

Figure 4: Sigmoid neuron with one input, with weight −20, and an overall bias of 10.

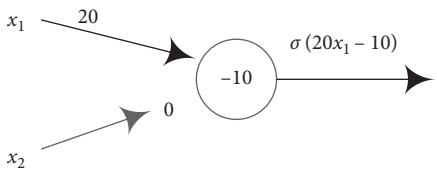

Figure 5: Sigmoid neuron with one input, with weight 20, and an overall bias of −10.

$$\begin{aligned} Y_1 &= Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3 \\ &= \overline{\overline{Z_1\overline{Z}_2Z_3} + \overline{Z_1Z_2\overline{Z}_3}} \\ &= \overline{Z_1\overline{Z}_2Z_3}\;\overline{Z_1Z_2\overline{Z}_3}. \end{aligned} \tag{4}$$

A neural network of five layers can implement expression (4), as is shown in Figure 6. This method of constructing the network has generality. ANN can be configured to execute an arbitrary map. Therefore, ANN can simulate the actions of the implicit memory.

Sometimes a situation arises in which some input variable combinations are not allowed. Because these unallowed states never occur in an application, they can be treated as "*do not care*" terms. For these "*do not care*" terms, either a 1 or a 0 may be assigned to the output. The "*do not care*" terms can be used to simplify an expression. Table 2 shows that, for each "*do not care*" term, a χ is placed in the output. As indicated in Figure 7, when grouping the 1s on the Karnaugh map, the χs can be treated as 1 s to make a larger grouping or as 0 s if they cannot have the advantage. The larger the group, the simpler the resulting term [20]. With "do not care", the neural network of five layers can be simplified to a connection line between Y_1 and Z_1.

Using the ordered pair set $\mathbf{X}$ as a training set, the ANN can be trained to simulate the implicit memory. The optimization process of training the ANN has some similar properties to a Boolean expression simplification process. Their purpose is to implement the required functionality with minimal resources, such as logic gates and activated neurons. The brain might have less than 1% neurons active at any given time [21–23]. If an input vector is not included in the training set, then the ANN can treat the input as a "*do not care*" term. The ANN can output either a 1 or a 0 when the input is a "*do not care*" term.

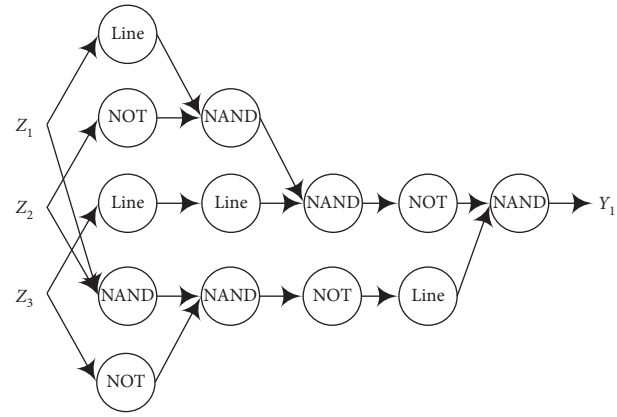

Figure 6: ANN can implement the Boolean expression $Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3$. The neurons are configured to implement the NOT, the NAND, and the connecting line. In general, the neurons have more inputs. While the weight of inputs is 0, the outputs are not affected, which are not drawn.

Table 2: Example of a truth table.

Input			Output	
Z_1	Z_2	Z_3	Y_1	
0	0	0	0	
0	0	1	0	
1	0	1	1	
1	1	0	1	
0	1	0	χ	Do not care
0	1	1	χ	Do not care
1	0	0	χ	Do not care
1	1	1	χ	Do not care

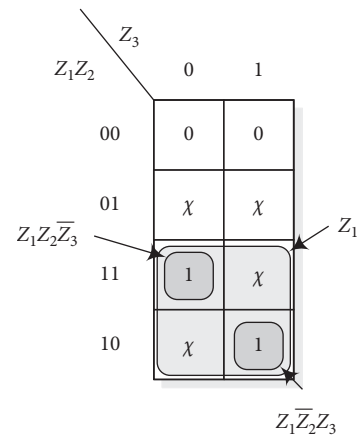

Figure 7: Example of the use of "do not care" conditions to simplify an expression. Without "do not care" $Y_1 = Z_1\overline{Z}_2Z_3 + Z_1Z_2\overline{Z}_3$. With "do not care" $Y_1 = Z_1$. So you can see the advantage of using "do not care" terms to get the simplest expression.

Suppose the supervised learning algorithm trains an ANN. While the given input is in the training set, the output of the trained ANN can be expected. Otherwise, the output of the trained ANN cannot be expected. Only by actual measuring can the output be known. The measurement process is like sampling from a statistical population. The trained ANN model can be expressed as a function f. The function f has the following properties:

(i) The output of the function is a specified value when the input is in the training set. That is, if $(\mathbf{z}, \mathbf{y})$ is in the training set, then $f(\mathbf{z}) = \mathbf{y}$.

(ii) Otherwise, the output can be assigned by generating a sample from a statistical population.

The following section proposes a pattern recognition algorithm with the above function f.

3. Recognition Based on the Implicit Memory Model

The principle of recognition is based on the intuitive assumption that examples in the same class are closer/similar [24, 25]. Therefore, a recognition algorithm is proposed to estimate the similarity via the implicit memory model.

Let's denote a signal by $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, where $x_i \in \{0, 1\}$, $i \in \mathcal{N} \triangleq \{1, \ldots, n\}$. When a function f can precisely predict any masked part of the signal $\mathbf{x}$ and $\tilde{\mathbf{x}}$ is not the same as $\mathbf{x}$, distinguishing between the signal $\mathbf{x}$ and any other signal $\tilde{\mathbf{x}}$ is feasible. The following theorem describes how to recognize the signal $\mathbf{x}$.

Theorem 1. *Consider the signal* $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ *is a constant. For an arbitrary mask* $\mathbf{m} = [m_1, \ldots, m_n] \in \{0, 1\}^n$, *a function* g *satisfies* $g(\mathbf{m}) = \mathbf{x} \circ (1 - \mathbf{m})$ *where* $(1 - \mathbf{m}) = [1 - m_1, \ldots, 1 - m_n]$. *Construct a function* $f\colon \{0, 1\}^n \longrightarrow \{0, 1\}^n$ *of the form* $f(g(\mathbf{m})) = \mathbf{x} \circ \mathbf{m}$. *If an input* $\mathbf{z} \in \{0, 1\}^n$ *is not included in the codomain of the function* g, *then the output of the function* f *is assigned by generating a signal,* $\mathbf{v} = [v_1, v_2, \ldots, v_n]$, *from a random number generator. The generator can produce binary digits, 0, 1, through equal probability sampling.* $\Pr\{v_i = 0\} = \Pr\{v_i = 1\} = (1/2)$, *where* $i \in \mathcal{N}$. *Let* $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]$ *be a new given signal, and following theorem describes how to recognize the signal* $\mathbf{x}$.

$$\zeta = \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} Z(f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})), \tilde{\mathbf{x}} \circ \mathbf{m}), \tag{5}$$

where

$$Z(\mathbf{a}, \mathbf{b}) = \begin{cases} 1, & \text{if } \mathbf{a} = \mathbf{b}, \\ 0, & \text{if } \mathbf{a} \neq \mathbf{b}. \end{cases} \tag{6}$$

If $\tilde{\mathbf{x}} \neq \mathbf{x}$, *then*

$$\Pr\{\zeta = 1\} = 0, \tag{7}$$

$$\Pr\{\zeta = 0\} > \left(1 - \frac{1}{2^n}\right)^{2^n} \longrightarrow \frac{1}{e} \approx 0.3679. \tag{8}$$

If $\tilde{\mathbf{x}} = \mathbf{x}$, *then*

$$\begin{aligned} \zeta &= \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} Z(f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})), \tilde{\mathbf{x}} \circ \mathbf{m}) = \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} Z(f(\mathbf{x} \circ (1 - \mathbf{m})), \mathbf{x} \circ \mathbf{m}) \\ &= \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} Z(f(g(\mathbf{m})), \mathbf{x} \circ \mathbf{m}) = \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} Z(\mathbf{x} \circ \mathbf{m}, \mathbf{x} \circ \mathbf{m}) \\ &= 1. \end{aligned} \tag{9}$$

The proof of Theorem 1 is given in Appendix.

According to equations (7)–(9), recognizing the signal $\mathbf{x}$ with the function f is feasible. When a point can help the function f to retrieve the signal $\mathbf{x}$, the point is called an *evoked point*. Let

$$\mathbf{D} \triangleq \{\mathbf{x} \circ (1 - \mathbf{m})\colon \mathbf{m} \in \{0, 1\}^n\}. \tag{10}$$

There are $|\mathbf{D}|$ *evoked points*, where $|\mathbf{D}| = 2^{|\mathbf{x}|_{l_1}} = 2^{x_1 + x_2 + \cdots + x_n}$. If $\Pr\{x_i = 1\} = (1/2)$, then the expected value of $|\mathbf{D}|$ is $2^{(n/2)}$.

The function f can retrieve the signal $\mathbf{x}$ from the *evoked point*, $\mathbf{h} = \tilde{\mathbf{x}} \circ (1 - \mathbf{m})$, in the current input signal $\tilde{\mathbf{x}}$ since the signal $\mathbf{x}$ can be expressed as $\mathbf{h} + f(\mathbf{h})$. By comparing $f(\mathbf{h})$ with $\tilde{\mathbf{x}} \circ \mathbf{m}$, identifying whether $\tilde{\mathbf{x}}$ is the same as $\mathbf{x}$ or not is feasible. If there exists a mask $\mathbf{m} \in \{0, 1\}^n$ that can make $f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) = \tilde{\mathbf{x}} \circ \mathbf{m}$, then $\Pr\{\tilde{\mathbf{x}} = \mathbf{x}\} = 1 - (1/2^n)$. If $\tilde{\mathbf{x}} = \mathbf{x}$, then each mask $\mathbf{m} \in \{0, 1\}^n$ can make $f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) = \tilde{\mathbf{x}} \circ \mathbf{m}$.

In a similar way of recognizing the signal $\mathbf{x}$ (Theorem 1), identifying whether the new given signal $\tilde{\mathbf{x}}$ is in a set $\mathbf{X} \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p\}$ or not is also feasible.

Lemma 1. *Suppose* $\mathbf{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,n}] \in \{0, 1\}^n$. *If the signals,* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$, *satisfy* $\mathbf{x}_i \neq \mathbf{x}_j$ *for any* $i \neq j$, *then there exists* $p - 1$ *positive integer,* $\alpha_1, \alpha_2, \ldots, \alpha_{p-1}$, *s.t.,* $[x_{i,\alpha_1}, x_{i,\alpha_2}, \ldots, x_{i,\alpha_{p-1}}] \neq [x_{j,\alpha_1}, x_{j,\alpha_2}, \ldots, x_{j,\alpha_{p-1}}]$ *for any* $i \neq j$.

The proof of Lemma 1 is given in the Appendix.

Theorem 2. *Suppose the signals,* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$, *satisfy* $\mathbf{x}_i \neq \mathbf{x}_j$ *for any* $i \neq j$ *where* $\mathbf{x}_i$ *is a constant. Without loss of generality,*

assume that $[x_{i,1}, x_{i,2}, \ldots, x_{i,p-1}] \neq [x_{j,1}, x_{j,2}, \ldots, x_{j,p-1}]$ *for any* $i \neq j$. *Let*

$$\mathbf{B}_i = \mathbf{D}_i \setminus \bigcup_{j \in \mathscr{P}, j \neq i} \mathbf{D}_i \cap \mathbf{D}_j, \tag{11}$$

where

$$\mathbf{D}_i \triangleq \{\mathbf{x}_i \circ (1 - \mathbf{m}) \colon \mathbf{m} \in \{0,1\}^n\}, \quad i \in \mathscr{P} \triangleq \{1, \ldots, p\}. \tag{12}$$

If $[x_{i,1}, x_{i,2}, \ldots, x_{i,p-1}] \neq [0, 0, \ldots, 0]$, then the cardinality of $\mathbf{B}_i$ satisfies that

$$|\mathbf{B}_i| \geq 2^{x_{i,p} + x_{i,p+1} + \cdots + x_{i,n}}. \tag{13}$$

If $[x_{i,1}, x_{i,2}, \ldots, x_{i,p-1}] = [0, 0, \ldots, 0]$, *then*

$$|\mathbf{B}_i| = 2^{x_{i,p} + x_{i,p+1} + \cdots + x_{i,n}} - 1. \tag{14}$$

The proof of Theorem 2 is given in Appendix.

Assume that any signal in set $\mathbf{X}$ does not equal $[0, 0, \ldots, 0]$, i.e., $|\mathbf{x}_i|_{l_1} = x_{i,1} + x_{i,2} + \cdots + x_{i,n} \geq 1$ for each $i \in \mathscr{P}$. Then, $|\mathbf{B}_i| \geq 1$ where $i \in \mathscr{P}$. If $\Pr\{x_i = 1\} = (1/2)$, then the expected value of $|\mathbf{B}_i|$ is $2^{((n-p+1)/2)}$.

When a new signal $\tilde{\mathbf{x}}$ appears, it is possible to identify whether $\tilde{\mathbf{x}}$ is in the set $\mathbf{X}$ or not with a function $f \colon \{0,1\}^n \longrightarrow \{0,1\}^n$. Construct the function f of the form

$$f(\mathbf{x}_i \circ (1 - \mathbf{m})) = \mathbf{x}_i \circ \mathbf{m}, \tag{15}$$

for any $\mathbf{x}_i \circ (1 - \mathbf{m}) \in \mathbf{B}_i$, $i \in \mathscr{P}$. If an input $\mathbf{z} \in \{0,1\}^n$ is not included $\mathbf{B}_1 \cup \cdots \cup \mathbf{B}_p$, then the output of the function f is assigned by generating a signal from a random number generator. If there exists a mask $\mathbf{m} \in \{0,1\}^n$ that can make $f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) = \tilde{\mathbf{x}} \circ \mathbf{m}$, then $\Pr\{\tilde{\mathbf{x}} \in \mathbf{X}\} = 1 - (1/2^n)$. If $\tilde{\mathbf{x}} \in \mathbf{X}$ (for example $\tilde{\mathbf{x}} = \mathbf{x}_j$), then there exists at least one mask $\mathbf{m} \in \{0,1\}^n$ that can make $f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) = \tilde{\mathbf{x}} \circ \mathbf{m}$ since $|\mathbf{B}_j| \geq 1$.

To identify whether a new given signal $\tilde{\mathbf{x}}$ has the same label as $\mathbf{X}$, the distance/similarity between $\tilde{\mathbf{x}}$ and $\mathbf{X}$ can be used, such as

$$d(\tilde{\mathbf{x}}, \mathbf{X}) = \inf\{d(\tilde{\mathbf{x}}, \mathbf{x}) | \mathbf{x} \in \mathbf{X}\}. \tag{16}$$

Without memorizing any element in $\mathbf{X}$, it is also possible to estimate the similarity by using the function f.

Let

$$\begin{aligned} \tilde{\mathbf{D}} &= \{\tilde{\mathbf{x}} \circ (1 - \mathbf{m}) \colon \mathbf{m} \in \{0,1\}^n\}, \\ \mathbf{B} &= \mathbf{B}_1 \cup \cdots \cup \mathbf{B}_p. \end{aligned} \tag{17}$$

If $\tilde{\mathbf{D}} \cap \mathbf{B}_i \neq \varnothing$, then there exist $|\tilde{\mathbf{D}} \cap \mathbf{B}_i|$ evoked points. Let $q_i = |\tilde{\mathbf{D}} \cap \mathbf{B}_i|$. Each evoked point, $\mathbf{h}_{i,j}$, is corresponding to a mask, $\mathbf{m}_{i,j}$.

$$\mathbf{h}_{i,j} = \tilde{\mathbf{x}} \circ (1 - \mathbf{m}_{\mathrm{i,j}}) = \mathbf{x}_{\mathrm{i}} \circ (1 - \mathbf{m}_{\mathrm{i,j}}), \tag{18}$$

where $j \in \{1, 2, \ldots, p_i\}$. The evoked points and the masks can be used to retrieve $\mathbf{x}_i$. With the function f, $\mathbf{x}_i$ can be gotten by

$$\begin{aligned} &\tilde{\mathbf{x}} \circ (1 - \mathbf{m}_{\mathrm{i,j}}) + \mathrm{f}(\tilde{\mathbf{x}} \circ (1 - \mathbf{m}_{\mathrm{i,j}})) \\ &= \mathbf{x}_i \circ (1 - \mathbf{m}_{\mathrm{i,j}}) + \mathrm{f}(\mathbf{x}_{\mathrm{i}} \circ (1 - \mathbf{m}_{\mathrm{i,j}})) \\ &= \mathbf{x}_i \circ (1 - \mathbf{m}_{\mathrm{i,j}}) + \mathbf{x}_{\mathrm{i}} \circ \mathbf{m}_{\mathrm{i,j}} \\ &= \mathbf{x}_i. \end{aligned} \tag{19}$$

The similarity between $\tilde{\mathbf{x}}$ and $\mathbf{X}$ can be estimated by

$$\inf\{d(\tilde{\mathbf{x}}, \tilde{\mathbf{x}} \circ (1 - \mathbf{m}) + f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})))) | \mathbf{m} \in \{0,1\}^n\}. \tag{20}$$

The process of similarity estimation is influenced by $\tilde{\mathbf{x}} \circ (1 - \mathbf{m}) \notin \mathbf{B}$. If $\tilde{\mathbf{x}}$ has the same meaning as element in $\mathbf{X}$, then $\tilde{\mathbf{x}}$ has to overcome the influence. The drawback of this estimation is that the algorithm might not traverse all elements in $\mathbf{X}$. If $\tilde{\mathbf{D}} \cap \mathbf{B}_i = \varnothing$, then there are no evoked points that can help us to retrieve $\mathbf{x}_i$. However, the advantage is that intentionally recollecting all elements in $\mathbf{X}$ is not necessary.

By training l functions $f_1, \ldots, f_l$ to predict the masked part of elements in their respective instance sets, i.e., $\mathbf{X}_1, \ldots, \mathbf{X}_l$, recognizing a new given signal $\tilde{\mathbf{x}}$ as one category is feasible. Let $\mathbf{X}_i \triangleq \{\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,p_i}\}$, $\mathbf{D}_{i,j} \triangleq \{\mathbf{x}_{i,j} \circ (1 - \mathbf{m}) \colon \mathbf{m} \in \{0,1\}^n\}$, and

$$\mathbf{B}_{i,j} = \mathbf{D}_{i,j} \setminus \left(\bigcup_{k \in \{1, \ldots, p_i\}, k \neq j} \mathbf{D}_{i,j} \cap \mathbf{D}_{i,k} \right), \tag{21}$$

where $j \in \{1, \ldots, p_i\}$, $i \in \{1, \ldots, l\}$. Function f_i satisfies

$$f_i(\mathbf{x}_{i,j} \circ (1 - \mathbf{m})) = \mathbf{x}_{i,j} \circ \mathbf{m}, \tag{22}$$

for any $\mathbf{x}_{i,j} \circ (1 - \mathbf{m}) \in \mathbf{B}_{\mathrm{i,j}}$, $j \in \{1, \ldots, p_i\}$. If an input $\mathbf{z} \in \{0,1\}^n$ is not included $\mathbf{B}_{i,1} \cup \cdots \cup \mathbf{B}_{i,p_i}$, then the output of the function f_i is assigned by generating a signal from a random number generator.

When a signal $\tilde{\mathbf{x}}$ is to be labeled, the similarity rule is a natural choice. To reduce the influence on the analysis of similarity, replace the infimum with average. The similarity between $\tilde{\mathbf{x}}$ and $\mathbf{X}_i$ can be estimated by

$$s_i(\tilde{\mathbf{x}}) \triangleq \frac{1}{2^n} \sum_{\mathbf{m} \in \{0,1\}^n} d(\tilde{\mathbf{x}}, \tilde{\mathbf{x}} \circ (1 - \mathbf{m}) + \mathrm{f}_{\mathrm{i}}(\tilde{\mathbf{x}} \circ (1 - \mathbf{m}))). \tag{23}$$

The smaller the similarity is, the more likely they are in the same category and have the same label. The above recognition model is presented in Algorithm 1, which is called the Implicit Recognition Model.

In summary, the Implicit Recognition Model uses similarity comparison to do the recognition. But intentionally recalling any labeled signals is not necessary. The focus is not on the past, but on the current input. The evoked points are objective features, which can help us to retrieve labeled signals from the current input. Both *evoked points* and retrieved signals have explicit physical meaning in the real world, which are objective pieces of evidence to support the judgment.

4. Experiment

In the application, the Mixed National Institute of Standards and Technology (MNIST) database is used to verify the

input: $\mathbf{X}_i = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,p_i}\}$ is a training instance set and $\mathbf{y}_i$ is the known label of $\mathbf{X}_i$; f_i, labeled $\mathbf{y}_i$, is an approximator and can be trained; $i \in \mathscr{L}$; $\tilde{\mathbf{x}}$ is a testing instance.

* Cognitive Process *

(1) **for** each $i \in \mathscr{L}$ **do**
(2) **repeat**
(3) Observing a signal in $\mathbf{X}_i$;
(4) Training the prediction function f_i to predict the masked parts of the signal;
(5) **until** For each signal $\mathbf{x}_{i,k} \in \mathbf{X}_i$ and any $j \neq i$, $s_i(\mathbf{x}_{i,k}) < s_j(\mathbf{x}_{i,k})$.
(6) **end for**
(7) **return** Prediction functions $f_1, \ldots, f_l$ labeled with $\mathbf{y}_1, \ldots, \mathbf{y}_l$ respectively.

* Recognition Process *

(1) **for** each $i \in \mathscr{L}$ **do**
(2) Estimating the similarity between testing signal $\tilde{\mathbf{x}}$ and instance set $\mathbf{X}_i$;
(3) Let $\varsigma_i = s_i(\tilde{\mathbf{x}})$;
(4) **end for**
(5) Assigning the image $\tilde{\mathbf{x}}$ to the class of its highest similarity;
(6) $\hat{l} = \operatorname{argmin}_i \varsigma_i$;
(7) **return** $\mathbf{y}_{\hat{l}}$.

Algorithm 1: Implicit Recognition Model.

Implicit Recognition Model. The MNIST database is one of the famous image classification benchmarks [26]. There are 60,000 instances in the training set and 10,000 instances in the testing set. The database is created by remixing the samples of NIST's original data sets. The black and white images from NIST are normalized to fit into a 28×28 pixel bounding box and antialiased, which introduces gray-scale levels [27]. The first 500 elements of the training set are used for training. All of the testing instances are used for testing.

$\mathbf{X}_i$ is labeled i and only contains training images with label i where $i = 0, 1, \ldots, 9$. Among the first 500 training images, there are $p_0 = 62$ training images labeled 0, $p_1 = 51$ training images labeled 1, and so on. $|\mathbf{X}_0| + \cdots + |\mathbf{X}_9| = 500$. All the 500 images are different. $\mathbf{X}_i \cap \mathbf{X}_j = \varnothing$ for any $i \neq j$.

In the process of recognition, an image in the testing set is recognized as the class of the highest similarity. To execute Algorithm 1, a distance function d is redefined first. Cosine distance is a usual measure of similarity in machine learning [28, 29]. Given two vectors $\mathbf{a}$ and $\mathbf{b}$, the cosine similarity $\cos(\theta)$ is represented by

$$\cos(\theta) = \frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}, \tag{24}$$

where a_i and b_i are components of the vectors $\mathbf{a}$ and $\mathbf{b}$, respectively. The angle $\theta \in [0, \pi]$ is used as the distance between two images in this application. Therefore,

$$d(\mathbf{a}, \mathbf{b}) = \arg\cos\left(\frac{\mathbf{ab}}{\|\mathbf{a}\|\|\mathbf{b}\|}\right). \tag{25}$$

With the artificial neural network, the implicit memory is simulated to execute the Implicit Recognition Model (Algorithm 1). The main process of Algorithm 1 is to train 10 approximators of the functions $f_0, f_1, \ldots, f_9$. Then, f_i can be used to retrieve images in $\mathbf{X}_i$ from the evoked points of the current input image $\tilde{\mathbf{x}}$ and estimate the similarity between $\tilde{\mathbf{x}}$ and $\mathbf{X}_i$.

To imitate the human recognition process, 10 ANNs are trained in two steps. The first step is to train a base neural network to complete all training images by filling in missing regions of a rectangular shape. This step does not use labels. In order to finally generate the specific function f_i, the second step is to add *routing layers* to the trained base neural network. With supervision, the *routing layers* are trained on the instance set $\mathbf{X}_i$. Routing neural networks are used to approximate the specific functions.

4.1. Architecture of ANN. A fully convolutional neural network is modified to be the base neural network. Refer to [30] for more details of the original fully convolutional neural network. The modified network parameters are given in Tables 3–6. Behind each convolution layer, except the last one, there is a Rectified Linear Unit (ReLU) layer. The output layer consists of a convolutional layer with a sigmoid function instead of a ReLU layer to normalize the output to the [0, 1] range. "Outputs" refers to the number of output channels for the output of the layer (Tables 3–6). Fully connected (FC) layers refer to the standard neural network layers. The output layer of discriminator consists of a fully connected layer with a sigmoid transfer layer. The discriminator outputs the probability that an input image came from real images rather than the completion network.

Routing layers are added to the trained base neural network to generate an approximator of the specific function f_i. Figure 8 shows the architecture of a routing neural network. Three routing layers are inserted in front of each network layer. The routing layer performs element-wise multiplication between the input $\mathbf{x} = [x_1, \ldots, x_n]$ and the routing weights $\mathbf{w} = [w_1, \ldots, w_n]$, which can be represented as $\mathbf{x}^{\circ}\mathbf{w} = [\mathrm{x}_1\mathrm{w}_1, \ldots, \mathrm{x}_\mathrm{n}\mathrm{w}_\mathrm{n}]$. The initial value of routing weight $\mathbf{w}$ is set to $1 = [1, \ldots, 1]$. The role of routing layers is to keep $f_i(\mathbf{x}_{i,k}{}^{\circ}(1 - \mathbf{m}))^{\circ}\mathbf{m} + \mathbf{x}_{\mathrm{i,k}}{}^{\circ}(1 - \mathbf{m})$ similar to $\mathbf{x}_{i,k}$, but the distance between $f_j(\mathbf{x}_{i,k}{}^{\circ}(1 - \mathbf{m}))^{\circ}\mathbf{m} + \mathbf{x}_{\mathrm{i,k}}{}^{\circ}(1 - \mathbf{m})$ and $\mathbf{x}_{i,k}$ is becoming far with the training, for any $j \neq i$, $\mathbf{x}_{i,k} \in \mathbf{X}_i$.

Table 3: Architecture of base neural network.

Type	Kernel	Dilation (μ)	Stride	Outputs
Conv.	3×3	1	1×1	14
Conv.	2×2	1	1×1	28
Conv.	2×2	1	1×1	28
Conv.	2×2	1	1×1	56
Conv.	2×2	1	1×1	56
Conv.	2×2	1	1×1	56
Dilated conv.	2×2	2	1×1	56
Dilated conv.	2×2	2	1×1	56
Dilated conv.	2×2	2	1×1	56
Dilated conv.	2×2	2	1×1	56
Conv.	2×2	1	1×1	56
Conv.	2×2	1	1×1	56
Deconv.	2×2	1	1×1	28
Conv.	2×2	1	1×1	28
Deconv.	2×2	1	1×1	14
Conv.	2×2	1	1×1	7
Output	2×2	1	1×1	1

Table 4: Architecture of local discriminator.

Type	Kernel	Stride	Outputs
Conv.	2×2	1×1	14
Conv.	2×2	1×1	28
Conv.	2×2	1×1	56
Conv.	2×2	1×1	112
FC	—	—	112

Table 5: Architecture of global discriminator.

Type	Kernel	Stride	Outputs
Conv.	2×2	1×1	14
Conv.	2×2	1×1	28
Conv.	2×2	1×1	56
Conv.	2×2	1×1	112
Conv.	2×2	1×1	112
FC	—	—	112

Table 6: Concatenation layer of discriminator.

Type	Kernel	Stride	Outputs
Concat	—	—	224
FC	—	—	1

4.2. Training Method of ANN. While the base neural network is training, global and local context discriminators are trained to distinguish real images from completed ones. The base neural network is trained to complete images by filling in masked regions of a rectangle. The training operation on the mask set $\mathbf{M} = \{\mathbf{m}: \mathbf{m} \in \{0,1\}^{28\times 28}\}$ is too large to calculate. A subset (denoted $\mathbf{M}'$) of $\mathbf{M}$ is selected with a rectangle masked part of 7 to 14 for the width and height.

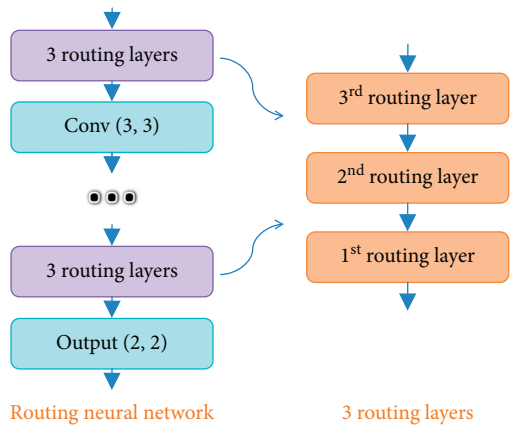

Figure 8: Network architectures of the routing neural network and 3 routing layers.

$$M' = \left\{ \mathbf{m} | \mathbf{m} = \underbrace{0^{(28)}; \ldots; 0^{(28)}}_{a_0}; \begin{matrix} 0^{(b_0)}, 1^{(b_1)}, 0^{(b_2)}; \\ \vdots \\ 0^{(b_0)}, 1^{(b_1)}, 0^{(b_2)}; \end{matrix} a_1 \underbrace{0^{(28)}; \ldots; 0^{(28)}}_{a_2}, a_0 + a_1 + a_2 = 28, b_0 + b_1 + b_2 = 28, a_1, b_1 \in \{7, \ldots, 14\}, \right. \tag{26}$$

where

$$0^{(k)} = \underbrace{[0, \ldots, 0]}_{k}, \quad 1^{(k)} = \underbrace{[1, \ldots, 1]}_{k}. \tag{27}$$

All the 500 training images (denoted $\mathbf{X}$) are used to train the base neural network.

When the training of the base neural network is finished, routing layers are added to generate an approximator of specific prediction function f_i. While the routing layers are training, the parameters of the base neural network remain unchanged. Ten digit labels/categories correspond to 10 approximators of prediction functions. Each prediction function f_i has a unique set of routing parameters.

The training method for routing layers is the same as the base neural network. The routing neural network is also trained to complete images by filling in masked regions of a rectangle. However, only the instance set $\mathbf{X}_i$ is used to train routing parameters of approximator of f_i. The three routing layers are trained one after another. While one routing layer is training, the other two routing layers remain unchanged. While $s_i(\mathbf{x}_{i,k})$ is becoming less than $s_j(\mathbf{x}_{i,k})$ for each signal $\mathbf{x}_{i,k} \in \mathbf{X}_i$ and any $j \neq i$, the cognitive process finishes. The trained routing neural network is an approximator of the specific prediction tool/function f_i.

4.3. Experimental Results. The neural network models are created by TensorFlow 1.8.0. The learning rate of the Adam optimizer is 10^{-3}. A batch size of one image is used for training. First, the base neural network is trained for 100 iterations. Then both the discriminator and base neural network are jointly trained. In each iteration, all 500 training images are shuffled and traversed. For each image, a masked rectangle region is randomly selected to train the neural networks. By 1000 iterations, the base neural network can predict the masked part (Figure 9).

Then, keep the base neural network unchanged, and add routing layers. The first routing layers are trained for 260 iterations and then remain unchanged. The second routing layers are also trained for 260 iterations and then remain unchanged. The approximation of a specific prediction function finishes when the third routing layers are also trained for 260 iterations.

A set of routing parameters correspond to a specific prediction function. Each approximator of functions $f_0, \ldots, f_9$ is corresponding to a routing neural network with a particular set of routing parameters. The approximator of function f_i can only work on the training set $\mathbf{X}_i$. Figure 10 gives an example. The mean distance $\overline{d_{i,f_i}}$ is less than $\overline{d_{i,f_j}}$ for each $j \neq i$ where $\overline{d_{i,f_j}} \triangleq (1/p_i) \sum_{k=1}^{p_i} s_j(\mathbf{x}_{i,k})$ (Figure 11).

With the approximator of the prediction function f_i, the algorithm can estimate the similarity between a testing image $\tilde{\mathbf{x}}$ and the instance set $\mathbf{X}_i$. A subset of M' (denoted $\mathbf{M}_r$) is used to estimate the similarity. Equations (23) and (25) are modified by

$$s_i(\tilde{\mathbf{x}}) \triangleq \frac{1}{|\mathbf{M}_r|} \sum_{\mathbf{m} \in \mathbf{M}_r} d\left(\tilde{\mathbf{x}}, \tilde{\mathbf{x}} \circ (1 - \mathbf{m}) + \mathrm{f}_{\mathrm{i}}(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) \circ \mathbf{m}\right). \tag{28}$$

30 masks are randomly generated, and $|\mathbf{M}_r| = 30$. The masks remain unchanged when the algorithm calculates $\varsigma_0, \ldots, \varsigma_9$. Finally, the image $\tilde{\mathbf{x}}$ can be recognized as the digit $\hat{l}$ satisfying that $\varsigma_{\hat{l}} \leq \varsigma_i$ for any $i \in \{0, 1, \ldots, 9\}$.

The masked part is the key for doing the similarity comparison. If the mask's size is too small, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}} \circ (1 - \mathbf{m}) + \mathrm{f}(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) \circ \mathbf{m}$ are mostly the same. There is no enough data to demonstrate the similarity. The main cause of high similarity should be from the accuracy of prediction (i.e., $f(\tilde{\mathbf{x}} \circ (1 - \mathbf{m})) \circ \mathbf{m}$ is similar to $\tilde{\mathbf{x}} \circ \mathbf{m}$), not the small mask

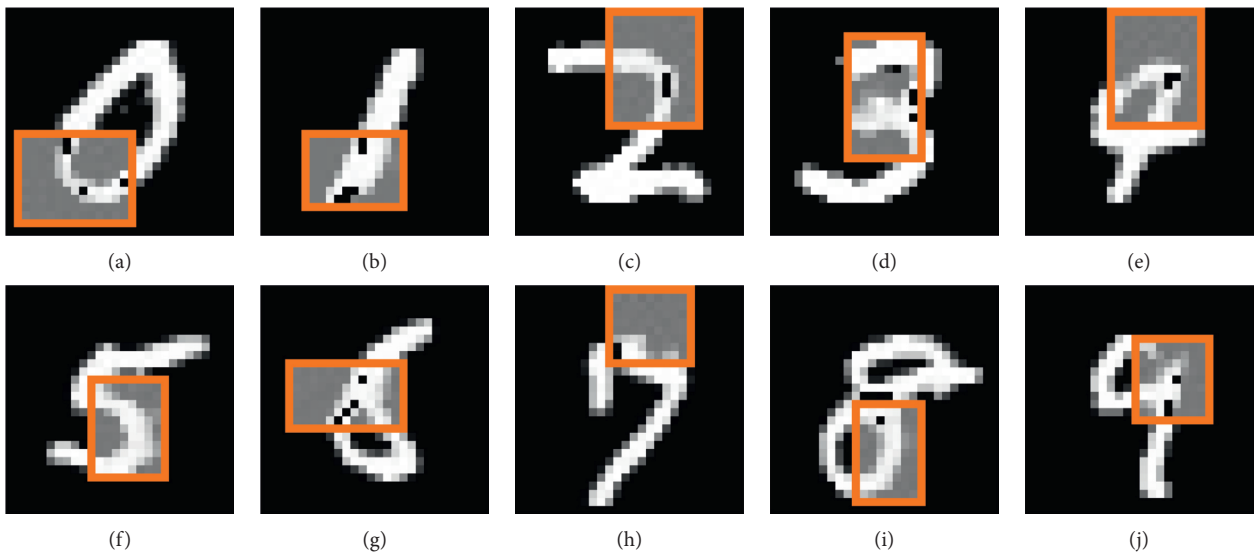

Figure 9: Prediction result by a base neural network on a training image with a label of (a) 0, (b) 1, (c) 2, (d) 3, (e) 4, (f) 5, (g) 6, (h) 7, (i) 8, and (j) 9 using randomly generated masks. The mask covers up a rectangular part of the image, and the predicted result is in the box.

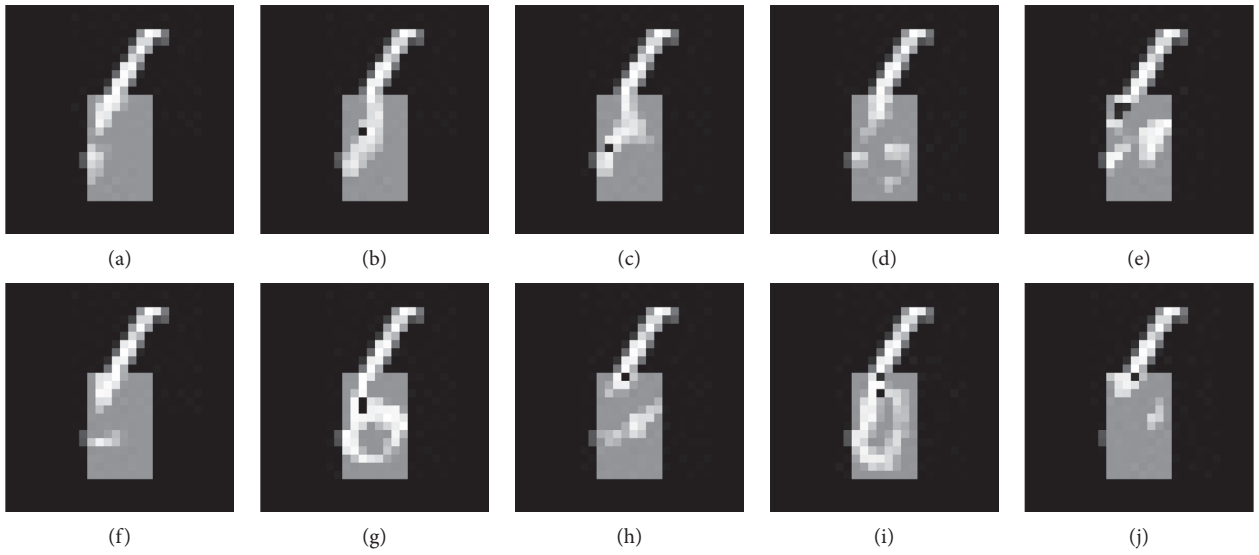

Figure 10: Prediction result by a routing neural network of (a) f_0, (b) f_1, (c) f_2, (d) f_3, (e) f_4, (f) f_5, (g) f_6, (h) f_7, (i) f_8, and (j) f_9 on a training image in $\mathbf{X}_6$ using a randomly generated mask.

window (i.e., $|\mathbf{m}|_{l_1}$ is small). An appropriate size can ensure the estimation results would be more comparable and credible. For a similarity comparison, the mask's window size cannot be too small. Meanwhile, the size cannot be too large to train ANN. The bigger the size is, the harder the masked parts are to be predicted.

Using the Implicit Recognition Model (Algorithm 1), the correct recognition probability approaches 100% on the training set (contains 500 training images). "Digit 4" and "digit 9" are easily to be confused. 80.44% recognition results of the Implicit Recognition Model are the same as the labels, and the confusion matrix is shown in Figure 12. Experimental results show the efficiency of the proposed Implicit Recognition Model.

The proposed model recognizes image $\tilde{\mathbf{x}}$ as digit $\hat{l}$, because $\tilde{\mathbf{x}}$ is most similar to the instances in $\mathbf{X}_{\hat{l}}$. As the similarity becomes lower, both the recognition precision of the Implicit Recognition Model and the one nearest neighbor algorithm (Explicit Recognition Model) decrease while rotating the testing images (Figure 13). Even using the testing image $\tilde{\mathbf{x}}$ itself to do the rotation similarity comparison, the distance $d(\tilde{\mathbf{x}}, r(\tilde{\mathbf{x}}, \alpha))$ becomes further still (Figure 14), where $r(\tilde{\mathbf{x}}, \alpha)$ represents that image $\tilde{\mathbf{x}}$ rotates α degrees. While the rotation angle is around 90 and 270 degrees, the distance approaches 1.22 radians, which is the mean distance between a noise image and the testing image. The pixel of testing image takes integer values between 0 and 255. The pixel of noise image obeys discrete uniform distribution on

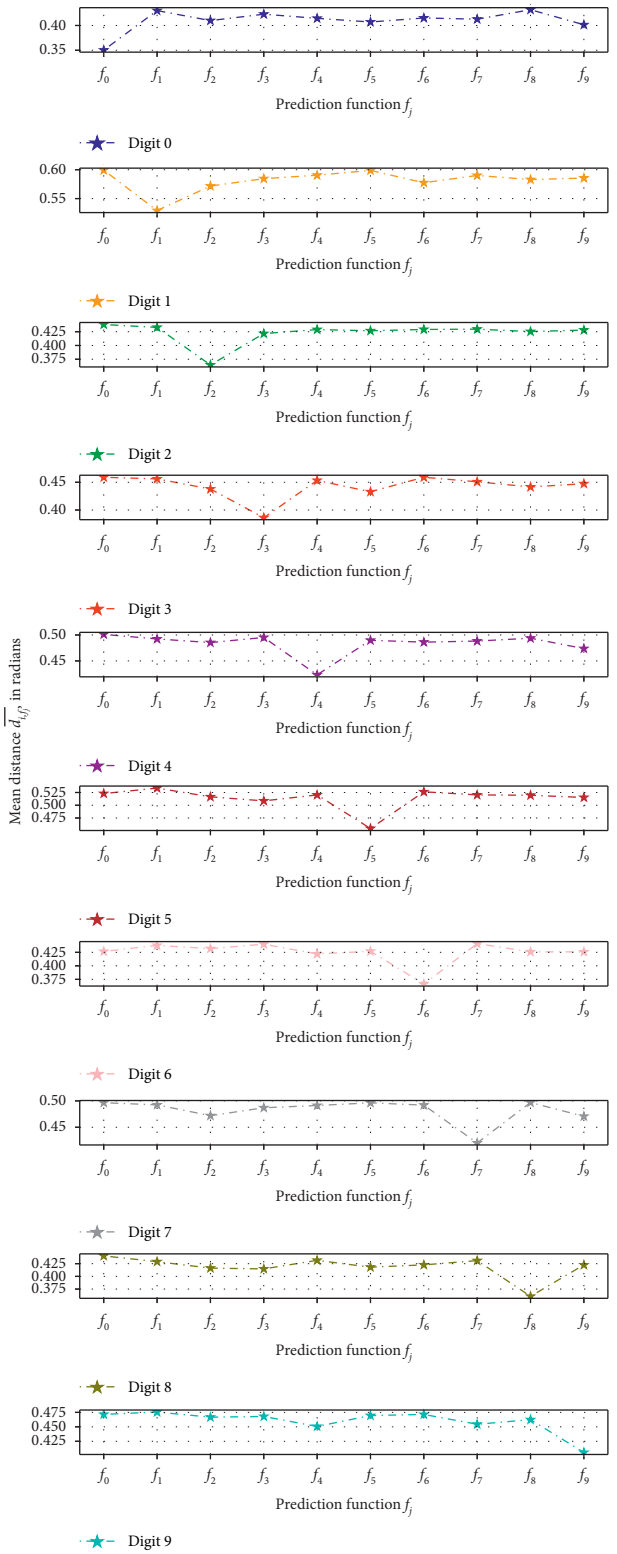

Figure 11: Mean distance $\overline{d_{i,f_j}}$ for each f_j on the training set.

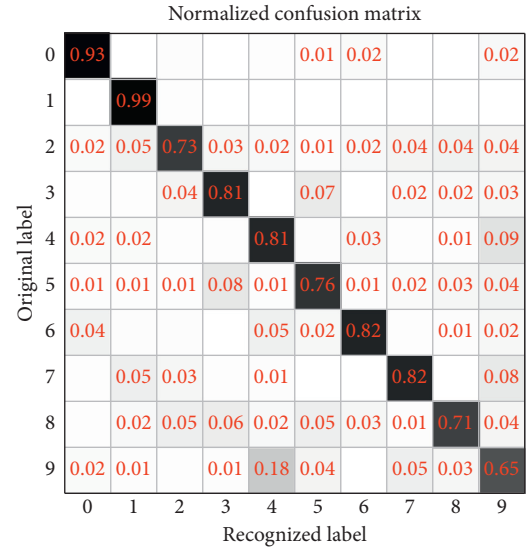

Figure 12: Performance of Implicit Recognition Model. Each column of the matrix represents the instances in an original class while each row represents the instances in a recognized class. The figure is accurate to two decimal places.

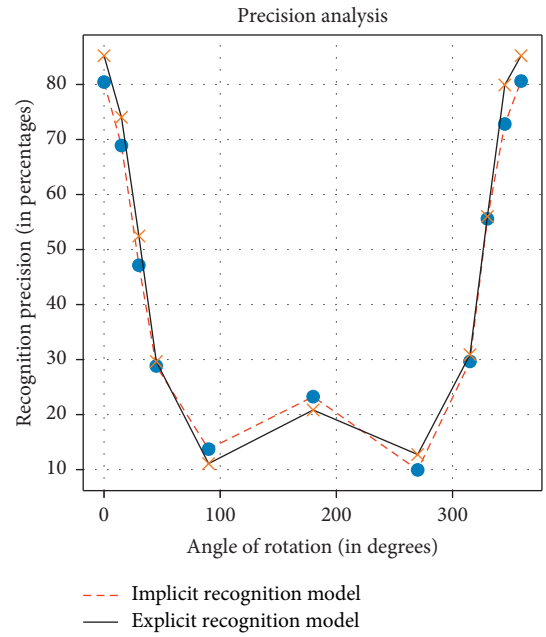

Figure 13: Recognition precision when the testing images rotate 0, 15, 30, 45, 90, 180, 270, 315, 330, 345, and 359 degrees, respectively.

the integers $0, 1, 2, \ldots, 255$. In Figures 13 and 14, the curves show W-shape and M-shape, because the images labeled 1 are similar to their rotated images when the rotation angle is around 180 degrees. Rotation of image can cause similarity changes. Therefore, a rotated image can also be used to define a new category if the similarity change is big enough. For example, the main difference between "W" and "M" is the direction of the opening (Figures 13 and 14).

Furthermore, the proposed algorithm is robust while adding random noise $\mathbf{n}$ to the testing images. Suppose that $\mathbf{n} = [n_1, \ldots, n_{784}]$ where n_i is an independent random variable. Noise point n_i obeys discrete uniform distribution on the integers $\beta, \beta + 1, \beta + 2, \ldots, \gamma$, where β and γ restrict the value range $(0 \le \beta \le \gamma \le 255)$. $\{\beta, \gamma\}$ represents the noise level. The bigger the values, the higher the noise level. The

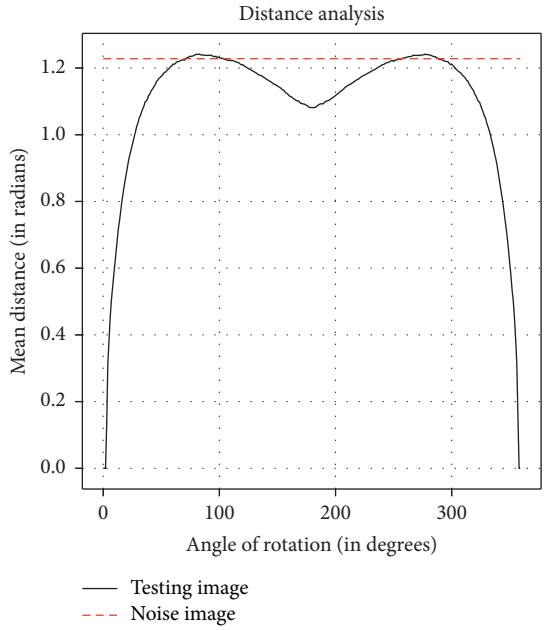

Figure 14: Rotation similarity comparison. Solid line represents the mean distance between a rotated image and its original image in the testing set. Dashed line represents the mean distance between a noise image and the testing image.

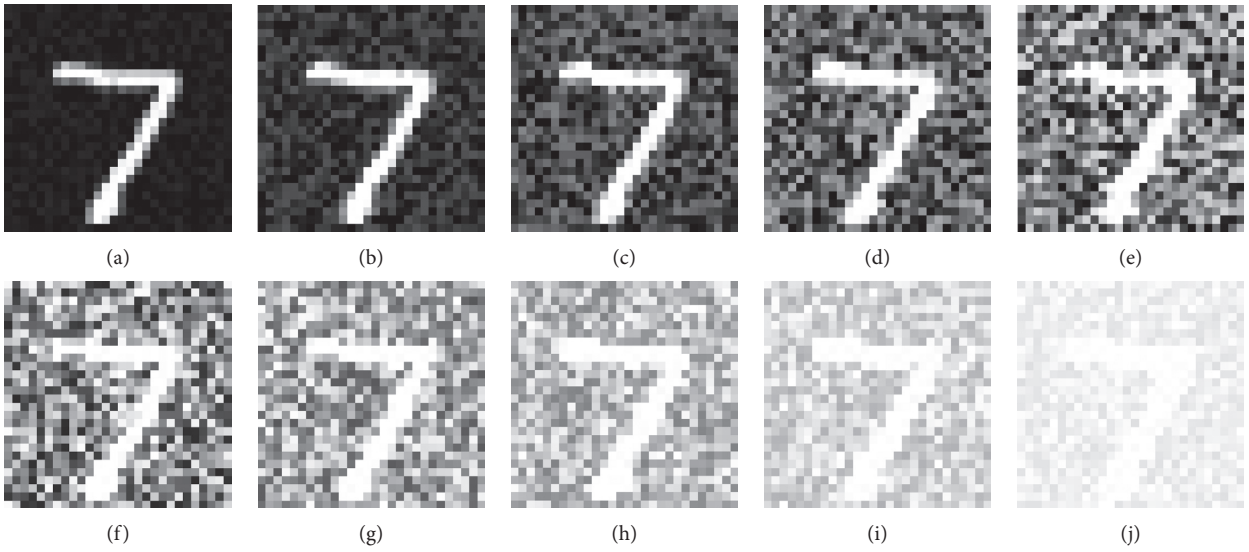

Figure 15: Testing image is disturbed at a noise level of (a) {0, 25}, (b) {0, 76}, (c) {0, 127}, (d) {0, 178}, (e) {0, 229}, (f) {25, 255}, (g) {76, 255}, (h) {127, 255}, (i) {178, 255}, and (j) {229, 255}.

testing images $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots, \tilde{x}_{784}]$ are disturbed by noise $\mathbf{n}$ (Figure 15) to test the robustness of algorithms. The image in noise is denoted as $\tilde{\mathbf{x}}_{\text{noise}} = [T(\tilde{x}_1 + n_1), \ldots, T(\tilde{x}_{784} + n_{784})]$ where

$$T(x_0 + x_1) = \begin{cases} x_0 + x_1, & \text{if } x_0 + x_1 \leq 255, \\ 255, & \text{if } x_0 + x_1 > 255. \end{cases} \tag{29}$$

Using PyTorch 1.5.1, the traditional neural network models are modified to do the recognition, too. Two linear layers after the output layer are added. In front of each linear layer, there is a Rectified Linear Unit (ReLU) layer. The testing images are resized and repeated along the channel. Then, the stochastic gradient descent optimizer and a batch size of 32 images are used for training.

By training 1000 iterations, the traditional neural network models can work when the noise level is lower than $\{0, 127\}$. However, the performance is not stable with increase of the noise level. The recognition precision of many traditional neural networks drops sharply while the change in the noise level is not big, as shown in Figure 16. However, the performance curve of Alexnet and VGG is smooth, as shown in Figure 17. *Overfitting* cannot provide the reasons behind the phenomenon since the neural networks are black-boxes. The

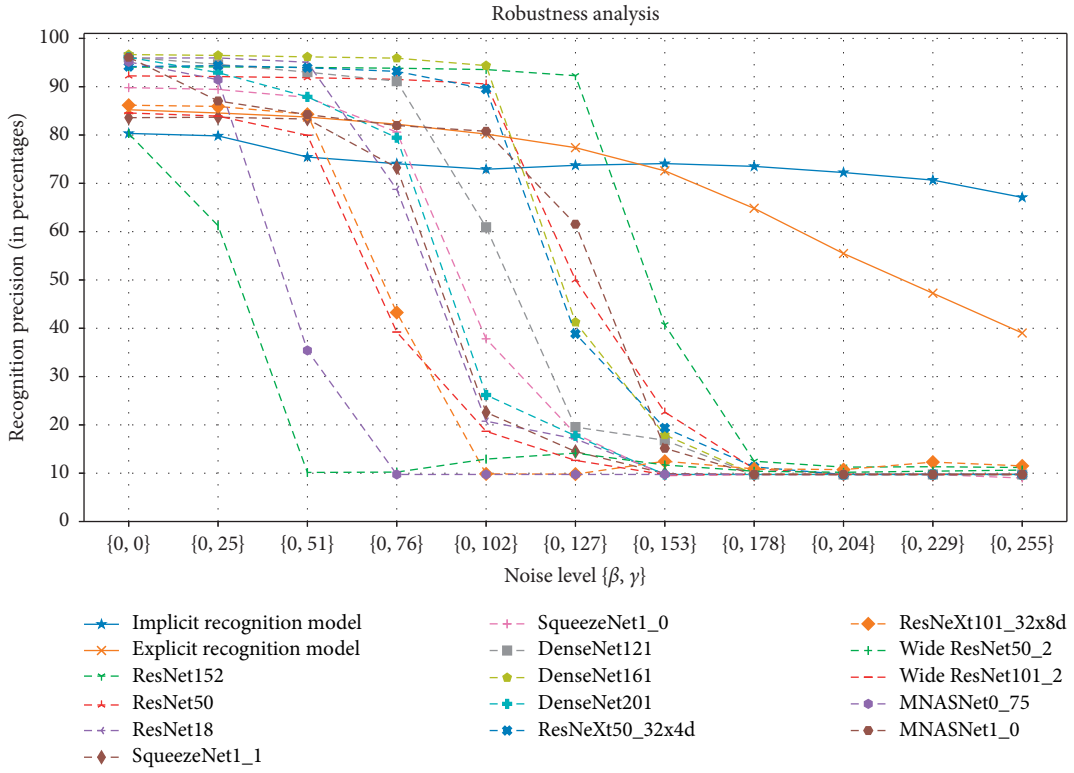

Figure 16: Recognition precision of Implicit Recognition Model, Explicit Recognition Model, ResNet, SqueezeNet, DenseNet, ResNeXt, Wide ResNet, and MNASNet at noise levels {0, 0}, {0, 25}, {0, 51}, {0, 76}, {0, 102}, {0, 127}, {0, 153}, {0, 178}, {0, 204}, {0, 229}, and {0, 255}.

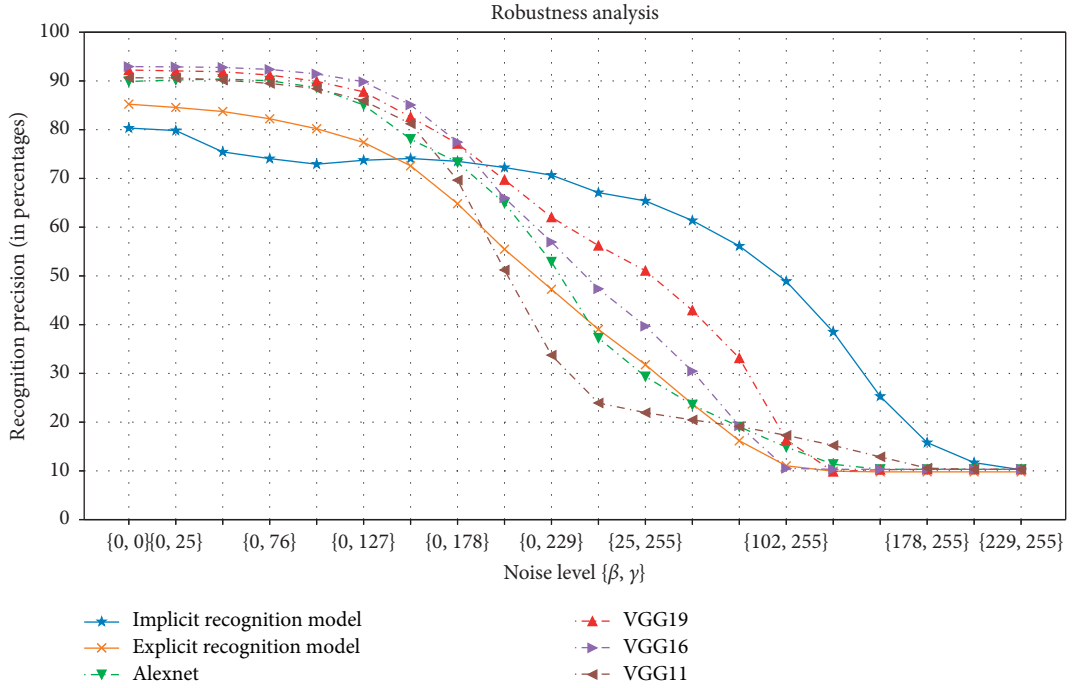

Figure 17: Recognition precision of Implicit Recognition Model, Explicit Recognition Model, Alexnet, and VGG at noise levels {0, 0}, {0, 25}, {0, 51}, {0, 76}, {0, 102}, {0, 127}, {0, 153}, {0, 178}, {0, 204}, {0, 229}, {0, 255}, {25, 255}, {51, 255}, {76, 255}, **{102, 255}**, {127, 255}, {153, 255}, {178, 255}, {204, 255}, and {229, 255}.

human can recognize the digits even when the noise level is {229, 255}. Therefore, the more straight the performance curve, the closer it is to the human capability. When the noise level is {102, 255}, the recognition accuracy of the traditional neural networks is less than 20%, but the accuracy of the Implicit Recognition Model is about 50%. The Implicit Recognition Model improves the robustness significantly.

5. Conclusion

Scientists early know the existence of implicit memory. This paper establishes a mathematical model of implicit memory and explains ANN can simulate the model from the view of digital logic circuit design. A trained ANN can be expressed as a function. When a given input is not in the training set, the output of the ANN is hard to control. With the function, this paper proposes a new pattern recognition method, the Implicit Recognition Model. The Implicit Recognition Model works under the similarity rule and has interpretability. Compared to the one nearest neighbor algorithm (Explicit Recognition Model), the Implicit Recognition Model makes similarity comparisons without recalling any instances. The experiment results show the efficiency of the Implicit Recognition Model.

Appendix

A. Proof of Theorem 1

Proof. Construct a function $\tilde{g}$ of the form $\tilde{g}(\mathbf{m}) = \tilde{\mathbf{x}}^{\circ}(1 - \mathbf{m})$. Let

$$\begin{aligned} \mathbf{S} &\triangleq \{g(\mathbf{m}):\ \mathbf{m} \in \{0,1\}^n\}, \\ \mathbf{T} &\triangleq \{\tilde{g}(\mathbf{m}):\ \mathbf{m} \in \{0,1\}^n\}. \end{aligned} \tag{A.1}$$

When $\mathbf{m} = [1, 1, \ldots, 1]$, $f(g(\mathbf{m})) = f(\tilde{g}(\mathbf{m})) = f([0, 0, \ldots, 0]) = \mathbf{x}^{\circ}[1, 1, \ldots, 1] = \mathbf{x}$. If $\tilde{\mathbf{x}} \neq \mathbf{x}$, then $Z(f([0, 0, \ldots, 0]), \tilde{\mathbf{x}}) = Z(\mathbf{x}, \tilde{\mathbf{x}}) = 0$. Therefore, $\Pr\{\zeta = 1\} = 0$.

Let $\mathbf{U} = \mathbf{T} \cap \mathbf{S}$ and $\mathbf{V} = \mathbf{T} \setminus \mathbf{S}$. For an arbitrary mask $\mathbf{m}$, $\tilde{g}(\mathbf{m})$ belongs to either $\mathbf{U}$ or $\mathbf{V}$. If $\tilde{g}(\mathbf{m}) \in \mathbf{U}$, then $Z(f(\tilde{g}(\mathbf{m})), \tilde{\mathbf{x}}^{\circ}\mathbf{m}) = 0$. If $\tilde{g}(\mathbf{m}) \in \mathbf{V}$, then

$$\Pr\{Z(f(\tilde{g}(\mathbf{m})), \tilde{\mathbf{x}}^{\circ}\mathbf{m}) = 0\} = 1 - \frac{1}{2^n}. \tag{A.2}$$

If $\tilde{\mathbf{x}} \neq \mathbf{x}$, then $1 \leq |\mathbf{U}| < 2^n$. Because $|\mathbf{U}| + |\mathbf{V}| \leq 2^n$,

$$\begin{aligned} \Pr\{\zeta = 0\} &= \left(1 - \frac{1}{2^n}\right)^{|\mathbf{V}|} \\ &> \left(1 - \frac{1}{2^n}\right)^{|\mathbf{U}|+|\mathbf{V}|} \\ &\geq \left(1 - \frac{1}{2^n}\right)^{2^n} \\ &\longrightarrow \frac{1}{e} \approx 0.3679. \end{aligned} \tag{A.3}$$

□

B. Proof of Lemma 1

Proof. The recurrence method is used to prove this lemma. When $p = 2$, there exists a positive integer, α_1, s.t., $x_{1,\alpha_1} \neq x_{2,\alpha_1}$ since $\mathbf{x}_1 \neq \mathbf{x}_2$.

Suppose the lemma is true when $p = m$. Let $[x_{i,\alpha_1}, \ldots, x_{i,\alpha_{m-1}}] \neq [x_{j,\alpha_1}, \ldots, x_{j,\alpha_{m-1}}]$ for any $i \neq j \in \{1, 2, \ldots, m\}$.

If $[x_{m+1,\alpha_1}, \ldots, x_{m+1,\alpha_{m-1}}] \neq [x_{j,\alpha_1}, \ldots, x_{j,\alpha_{m-1}}]$ for any $j \in \{1, 2, \ldots, m\}$, then α_m can be any positive integer $k \leq n$ where $k \notin \{\alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$. If $[x_{m+1,\alpha_1}, \ldots, x_{m+1,\alpha_{m-1}}] = [x_{j,\alpha_1}, \ldots, x_{j,\alpha_{m-1}}]$, then there exists a positive integer, α_m, s.t., $x_{m+1,\alpha_m} \neq x_{j,\alpha_m}$ since $\mathbf{x}_{m+1} \neq \mathbf{x}_j$. □

C. Proof of Theorem 2

Proof. Let $\mathbf{S}_i \triangleq \{\mathbf{x}_i^{\circ}(1 - [0^{(p-1)}, \mathbf{m}]):\ \mathbf{m} \in \{0,1\}^{n-p+1}\}$, where

$$0^{(p-1)} \triangleq \underbrace{[0, \ldots, 0]}_{p-1}. \tag{C.1}$$

$\mathbf{S}_i \in \mathbf{B}_i$ since $[x_{i,1}, \ldots, x_{i,p-1}] \neq [x_{j,1}, \ldots, x_{j,p-1}]$ for any $i \neq j$. If $[x_{i,1}, \ldots, x_{i,p-1}] \neq [0, \ldots, 0]$, then $|\mathbf{B}_i| \geq |\mathbf{S}_i| = 2^{x_{i,p}+x_{i,p+1}+\cdots+x_{i,n}}$. If $[x_{i,1}, \ldots, x_{i,p-1}] = [0, \ldots, 0]$, then $|\mathbf{B}_i| = |\mathbf{S}_i| = 2^{x_{i,p}+x_{i,p+1}+\cdots+x_{i,n}} - 1$ since $[0, \ldots, 0] \notin \mathbf{B}_i$. □

Data Availability

The handwritten digits data supporting this study are from previously reported studies and datasets, which have been cited. The processed data are available in the MNIST database.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] X. Zhang, C. Liu, and C. Y. Suen, "Towards robust pattern recognition: a review," *Proceedings of the IEEE*, vol. 108, no. 6, pp. 894–922, 2020.

[2] S. Theodoridis and K. Koutroumbas, "Pattern recognition," *Features, Feature Vectors, and Classifiers*, Elsevier, Amsterdam, Netherlands, Chapter 1.2, 4th edition, 2010.

[3] L.-J. Cai, S. Lv, and K.-b. Shi, "Application of an improved chi feature selection algorithm," *Discrete Dynamics in Nature and Society*, vol. 2021, Article ID 9963382, 8 pages, 2021.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] Q. Zheng, M. Yang, X. Tian, N. Jiang, and D. Wang, "A full stage data augmentation method in deep convolutional neural network for natural image classification," *Discrete Dynamics in Nature and Society*, vol. 2020, Article ID 4706576, 11 pages, 2020.

[6] C.-M. Yuan, X.-M. Sun, and Z. Hu, "Speech separation using convolutional neural network and attention mechanism," *Discrete Dynamics in Nature and Society*, vol. 2020, Article ID 2196893, 10 pages, 2020.

[7] Y. Li, Z. Feng, S. Chen, Z. Zhao, and F. Wang, "Application of the artificial neural network and support vector machines in forest fire prediction in the guangxi autonomous region, China," *Discrete Dynamics in Nature and Society*, vol. 2020, Article ID 5612650, 14 pages, 2020.

[8] Yu Liang, S. Li, C. Yan, M. Li, and C. Jiang, "Explaining the black-box model: a survey of local interpretation methods for deep neural networks," *Neurocomputing*, vol. 419, pp. 168–182, 2020.

[9] J. Ukita, "Causal importance of low-level feature selectivity for generalization in image recognition," *Neural Networks*, vol. 125, pp. 185–193, 2020.

[10] D. Li, X. Hu, C.-jie Jin, and J. Zhou, "Learning to detect traffic incidents from data based on tree augmented naive bayesian classifiers," *Discrete Dynamics in Nature and Society*, vol. 2017, Article ID 8523495, 9 pages, 2017.

[11] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-end learning from spectrum data: a deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18484–18501, 2018.

[12] J. Pearl and D. Mackenzie, *The Book of Why, Chapter Introduction, Mind Over Data*, Penguin Books, London, UK, 2018.

[13] L. L. Jacoby and C. M. Kelley, "Unconscious influences of memory for a prior event," *Personality and Social Psychology Bulletin*, vol. 13, no. 3, pp. 314–336, 1987.

[14] D. L. Schacter, "Implicit memory: history and current status," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 13, no. 3, p. 501, 1987.

[15] E. L. Glisky, *Implicit Memory*, Springer, Berlin, Germany, 2018.

[16] I. A. García-León, N. Pereda Beltrán, A. Alanis, and J. S. Moreno, "Intelligent system for the evaluation of implicit memory with semantic emotional stimuli (is-eimses)," in *Agents and Multi-Agent Systems: Technologies and Applications 2021*, G. JezicJ. Chen-Burger et al., Eds., p. 337, Article ID 347, Springer, Berlin, Germany, 2021.

[17] N. J. Cohen, E. Howard, B. S. Deacedo, and S. Corkin, "Different memory systems underlying acquisition of procedural and declarative knowledge," *Annals of the New York Academy of Sciences*, vol. 444, no. 1, pp. 54–71, 1985.

[18] J. Lin, Y. Meng, and W. Lin, "Conditional automaticity: interference effects on the implicit memory retrieval process," *Psychological Research*, vol. 85, no. 1, pp. 223–237, 2021.

[19] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[20] T. L. Floyd, "Digital fundamentals," *The Universal Property of NAND and NOR Gates and 4.9 Karnaugh Map SOP Minimization*, Publishing House of Electronics Industry, Beijing, China, Chapter 5.3, 11th edition, 2017.

[21] D. Attwell and S. B. Laughlin, "An energy budget for signaling in the grey matter of the brain," *Journal of Cerebral Blood Flow & Metabolism*, vol. 21, no. 10, pp. 1133–1145, 2001.

[22] L. Peter, "The cost of cortical computation," *Current Biology*, vol. 13, no. 6, pp. 493–497, 2003.

[23] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current Opinion in Neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[24] U. Lall and A. Sharma, "A nearest neighbor bootstrap for resampling hydrologic time series," *Water Resources Research*, vol. 32, no. 3, pp. 679–693, 1996.

[25] S. E. Taabat, "A novel multicriteria decision-making method based on distance, similarity, and correlation: Dsc topsis," *Mathematical Problems in Engineering*, vol. 2019, Article ID 9125754, 20 pages, 2019.

[26] L. Zhou, Q. Li, G. Huo, and Y. Zhou, "Image classification using biomimetic pattern recognition with convolutional neural networks features," *Computational Intelligence and Neuroscience*, vol. 2017, Article ID 3792805, 12 pages, 2017.

[27] Y. Lecun, *MNIST Handwritten Digit Database, Yann Lecun, Corinna Cortes and Chris burges*, BibSonomy, 2013.

[28] H. J. Seo and P. Milanfar, "Training-free, generic object detection using locally adaptive regression kernels," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1688–1704, 2010.

[29] J. Xu, P. Xu, Z. Wei, Xu Ding, and L. Shi, "Dc-nnmn: across components fault diagnosis based on deep few-shot learning," *Shock and Vibration*, vol. 2020, Article ID 3152174, 11 pages, 2020.

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–14, 2017.