Character Structure Analysis by Adding and Pruning Neural Networks in Handwritten Kanji Recognition

Keiji Gyohten(✉), Hidehiro Ohki, and Toshiya Takami

Faculty of Science and Technology, Oita University, Dannnoharu 700, Oita 870-1192, Japan
gyohten@oita-u.ac.jp

Abstract. In recent years, there has been a growing need for techniques that enable users to explain the basis for decisions made by neural networks in image recognition problems. While many conventional methods have focused on presenting the spatial basis for pattern recognition judgments by identifying the regions in the image that significantly influence the recognition results, our proposed method focuses on presenting a structural basis for recognizing handwritten Kanji characters. This is accomplished by pruning neural networks to acquire detectors for simple patterns that are commonly found in Kanji characters. During the process of sequentially adding these detectors, we also apply pruning to the network connecting the detectors, thereby aiming to precisely acquire simple hierarchical connections among the detectors as a structural recognition process in pattern recognition. We successfully applied this method to simple handwritten Kanji images, and achieved Kanji recognition by combining detectors for simple patterns without significantly affecting the recognition rate.

Keywords: Character Recognition · Deep Learning · Neural Network Pruning · Pattern Detector

1 Introduction

Currently, various studies on deep learning are being conducted worldwide to improve its performance by devising the structure of neural networks and learning from more training data. With such performance improvements, deep learning has been able to penetrate various application fields, and is highly likely to become the foundation of social systems in the near future.

To facilitate the integration of deep learning into diverse social systems, it is imperative that society willingly accept machine-generated judgments. For example, in the medical field [1], a technology that can not only make judgments, but also explain the rationale behind such judgments is needed. The field of deep learning explainability has emerged to address this imperative [2]. Consequently, extensive research efforts have been directed towards the domain of object recognition, with particular emphasis on methodologies that spatially elucidate the regions within an image that significantly influence decision-making processes.

H. Lu et al. (Eds.): ACPR 2023, LNCS 14407, pp. 129–142, 2023.
https://doi.org/10.1007/978-3-031-47637-2_10

On the other hand, techniques that enable machines to provide explanations for their decisions based on the structure of patterns in images have been relatively scarce in the existing research literature. In a typical object recognition application, the recognized objects are composed of complex and hierarchically arranged combinations of different components. For example, a car can be described as a combination of wheels, windows, and a body; decomposing it further reveals that a wheel comprises a tire and rim. The structural representations generated by machines may not always align with human intuition. Nonetheless, empowering machines to explain the rationale behind their decisions by leveraging the pattern is crucial to establishing human confidence in the machine's judgments. To accomplish this goal of structural explanation, it is necessary to not only address the given pattern recognition problem, but also develop methodologies for automatically acquiring the underlying structure of the pattern.

In this study, we proposed an approach to Kanji character recognition that utilizes a neural network to automatically and precisely acquire the underlying structure of patterns, and explain the judgment rationale based on a pattern structure. The motivation for addressing the Kanji recognition problem is that Kanji characters are composed of a limited set of components, which renders them suitable to understand the pattern structure. The proposed method consists of a combination of simple detectors based on a neural network. During the training process, edge pruning techniques are employed within the neural network to acquire detectors that respond to simple patterns present in Kanji characters. Furthermore, by sequentially connecting these detectors and pruning edges within the network of connections among the detectors, we can achieve Kanji character recognition. The sparse connections among the detectors in the final neural network provide a representation of the pattern as a tree structure; this constitutes the foundation for judgments. Through experimentation, we confirmed that the character recognition based on detector connections was feasible without significantly reducing the recognition accuracy. Additionally, we demonstrated the ability to analyze the connection relationships of each detector, thereby allowing us to acquire detectors that respond to multiple Kanji characters in common. Through hierarchical connections of these detectors, we could represent the process of pattern detection as a tree structure.

2 Related Works

2.1 Investigating Explainability in Neural Networks

In recent years, there have been various studies on explainability in deep learning [2]. Many of these methods have been based on approaches that identify important factors that contribute significantly to recognition results. For example, the integrated gradients (IG) method finds the elements by integrating the gradient of the activity value of the output corresponding to the recognition target on a pixel-by-pixel basis [3]. This method provides a pixel-by-pixel representation of which parts of the image contribute significantly to an increase in the activity value of the output corresponding to the recognition target. Local interpretable model-agnostic explanations (LIME) finds which small regions in the input image contribute significantly to the recognition result [4]. This method achieves this by randomly generating a group of small regions with the

same features of the image, thus obtaining the activity value of the output corresponding to the recognition target, and expressing the relationship between the two by linear regression. Activation maximization (AM) obtains the input image that mostly activates a certain neuron in a neural network by finding the gradient of its activity value relative to the input image [5, 6]. These methods have received a lot of attention and are at the forefront of explainability in deep learning. Other studies on explainability based on various approaches have also been conducted. Most of these methods represent important elements that contribute significantly to recognition results as units of pixels or regions in an image. Most of them adopt an approach that visually presents the spatial location of the elements in the image.

On the other hand, approaches that explicitly present the structure of recognition targets, such as part-whole relationships, as important elements contributing to the recognition results are not widely observed. In a research effort, based on such an approach, Zhang et al. have achieved highly interesting Kanji character recognition, including zero-shot learning, by utilizing the ideographic description sequence (IDS) dictionary to represent the structure of characters [7]. However, in this method, the structure of the recognition targets must be pre-defined in the dictionary. Generally, it is challenging to pre-define the constituent elements that can represent the part-whole relationships of recognition targets. For example, in the problem of face recognition, assuming that we want to explain a face is recognized based on the fact that it consists of eyes, nose, and mouth, pre-defining the constituent elements of a face and preparing separate recognizers for each of them is not practical. In our research, instead of manually preparing recognizers for the constituent elements of a recognition target, we have attempted to automatically acquire detectors by pruning the edges of a neural network.

2.2 Applying Edge Pruning to Neural Networks

Pruning edges in neural networks, along with quantization, is commonly applied to compress the network size [8]. Various approaches have been attempted to eliminate redundant computations by pruning edges and quantizing the parameters in the network, while preventing degradation of the recognition performance to the extent possible. Our study does not attempt to compress the network size; instead it acquires detectors of simple patterns that commonly exist in the recognition target by means of edge pruning. It is assumed that the performance of a neural network that has been pruned excessively will deteriorate, and distinguishing similar patterns will be difficult. In our study, the neural network that can no longer distinguish similar patterns is used as a detector of patterns that commonly exist in the recognition target.

3 Proposed Method

3.1 Outline

Figure 1 illustrates an outline of the neural network structure employed in the proposed method. Firstly, convolutional neural network (CNN) is utilized to train character images and obtain a character recognition network. After extracting a feature extractor consisting

of convolutional layers, pattern detectors composed of dense layers are sequentially added, and the learning process is repeated. During the training, edge pruning is applied to the dense layers to create detectors that capture simple patterns. Moreover, by applying edge pruning to the dense layers, the connectivity of the pattern detectors is simplified. By analyzing this connectivity of the pattern detectors, it becomes possible to acquire the structure of the underlying patterns that form the basis of recognition.

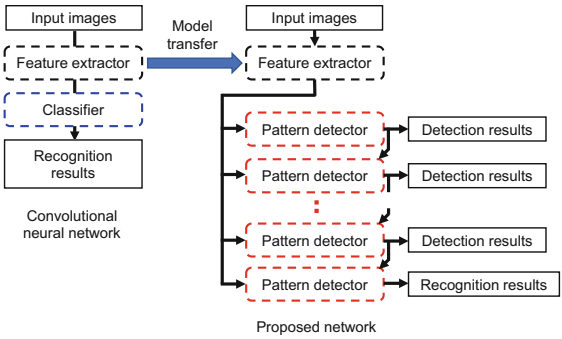


Fig. 1. Outline of the proposed method.

3.2 Obtaining Feature Extractor

First, our method obtains a feature extractor that extracts local features of the input character image. This feature extractor is obtained by transferring the convolution layers from a trained character recognition network based on CNN, which is widely used in pattern recognition. As these filters have a significant impact on character recognition performance, it is necessary to obtain good filters at this stage.

Figure 2 shows the CNN used in this method to obtain the feature extractor. The network is trained using the training data, which consists of binary images of characters as input data and one-hot vectors representing the character types of the input characters as teacher data. Using the training data, the weights in the network are updated to reduce the value of the loss function using an optimization algorithm. The loss function is the categorical cross entropy. During the training process, the weights of the network with the best recognition results for the validation data are used as the final training set. After training, the classifier is discarded and the feature extractor is used as a set of filters to extract the local features of the character images.

3.3 Adding and Pruning Pattern Detectors

Adding Pattern Detectors. Our method adds the 1st pattern detector to the feature extractor obtained in the previous step (see Sect. 3.2), as shown in Fig. 3 and trains the network in the manner described shortly. In the training process, edge pruning is applied to the dense layers in the pattern detector to make it respond to simple patterns in the

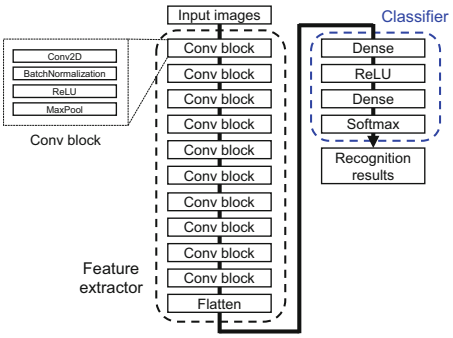


Fig. 2. Convolutional Neural Network for character recognition.

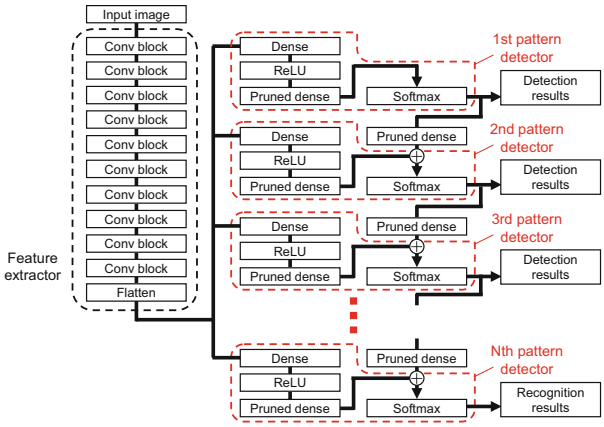


Fig. 3. Adding pattern detectors to the feature extractor.

characters. It may be noted that the weights in the feature extractor are frozen and used in this training process.

After the training of the 1st pattern detector is completed, the 2nd pattern detector shown in Fig. 3 is added and the network is further trained. By adding the 2nd pattern detector, the overall network structure becomes more complex, and a detector that can respond to complex patterns can be obtained. The weights in the 1st pattern detector as well as those in the feature extractor are frozen, and the training with edge pruning described later is applied to the 2nd pattern extractor. By adding pattern detectors sequentially as described above, our method finally obtains the detector that can output the desired Kanji recognition results.

The pruned dense layer within each pattern detector comprises highly sparse edges resulting from the pruning process. As a result, the nodes in the output layer of each pattern detector no longer respond to only one character, but to multiple characters. This signifies that the parameter count representing each pattern detector has been reduced through pruning, thereby enabling it to respond to simple patterns present in multiple characters. The values of these nodes are input to the next pattern detector through the

pruned dense layer that connects them. By revealing how pattern detectors are interconnected via the pruned dense layer, it will be possible to explain to the machine the combination of structural patterns present in the input image. Additionally, in the event of a failure in recognizing handwritten Kanji images, it may be possible to explain the cause of recognition failure by observing the output values of each pattern detector and identifying which patterns exhibited weak responses.

Training and Pruning Pattern Detectors. As shown in Fig. 4, the proposed method applies pruning to the dense located just before the output layer of the pattern detector and the dense layer connecting the pattern detectors. The pruning is achieved by fixing the weights of the edges between the nodes in the dense layer to zero. Here, we define sparsity, s_f as the ratio of the number of edges, whose weights are fixed to 0, to the total number of edges in the dense layer. In this method, in the process of training, the weights of the edges in the dense layer are fixed to 0 for edges, whose weights are close to 0. This operation is repeated until s_f reaches a pre-defined value.

This training process is divided into two stages: In Stage 1, training for character recognition with edge pruning, shown in Fig. 4(a), takes place, and in Stage 2, training for a detector that responds to simple patterns, shown in Fig. 4(b), takes place.

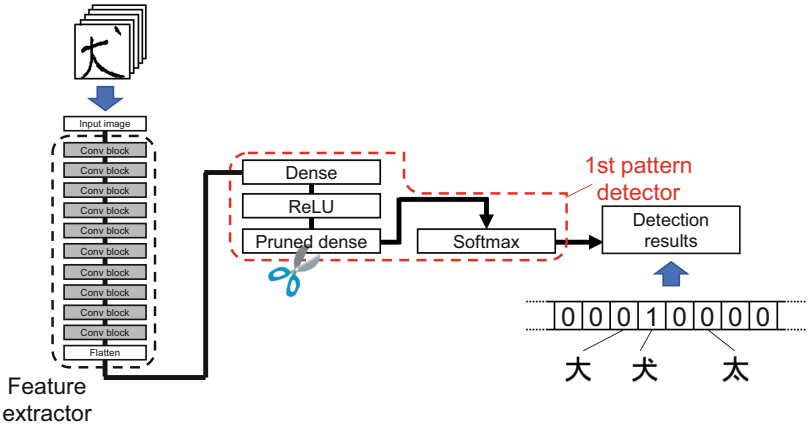
In Stage 1, we use the training data described in Sect. 3.2 to update the weights of the dense layer in the pattern detector until the loss function converges. The teacher data in the training data is represented as a one-hot vector. The character type corresponding to the node with the largest value in the output layer is used as the recognition result. In this process, the edge pruning is repeatedly applied to the dense layer just before the output layer of the pattern detector until s_f reaches a pre-specified value.

The training method with edge pruning, as proposed by us, is described below. First, the final value of s_f and the number of sparsity updates I, are determined in advance. Then, using the training data described in Sect. 3.2, the weights in the dense layers are updated until the loss function converges. Then, for the weights within the dense layers, this method fixes the weights whose value is close to 0 to 0 so that s_f can be calculated by the following equation:

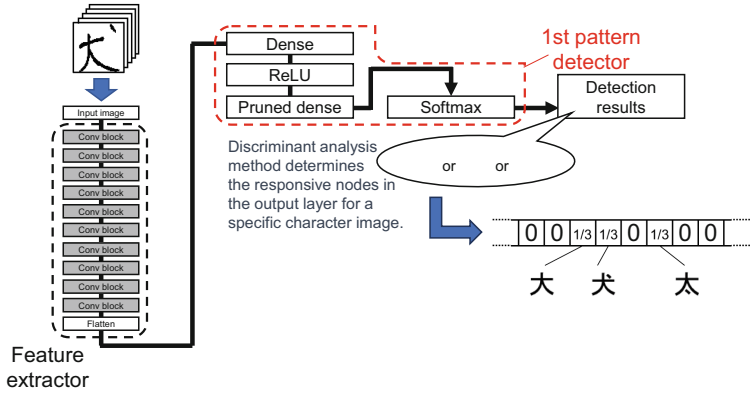
$$\text{sparsity} = s_f i / I,$$

where i is the number of updates of the current sparsity. By repeating the above training and pruning I times, s_f within the dense layers becomes the final s_f specified.

In the Stage 2, we aim to obtain detectors that respond to simple patterns using the network obtained after applying edge pruning. The pruning of the dense layers results in a decrease in the network's recognition performance, limiting it to recognizing only simple patterns. In other words, this pruning yields a network that exclusively responds to various simple patterns of kanji characters. We input the character images from the training data into the network and calculate the average values of the nodes in the output layer for each character type. As the pruned network only responds to simple patterns, even when a character image belonging to a particular character type is inputted, multiple nodes in the output layer may exhibit high values. In this approach, as depicted in Fig. 4(b), we apply discriminant analysis [9] to binarize the average values of the nodes in the output layer, thereby determining multiple nodes that respond to character images of a specific character type. Subsequently, we obtain teacher vectors,



(a) Training for character region with edge pruning



(b) Training detectors that respond to simple patterns

Fig. 4. Training for pattern detectors.

as shown in Fig. 4(b), where non-responsive nodes have a value of 0, and responsive nodes possess probabilities indicating the correct character type as their element values. Using this training data, we retrain the pruned network to obtain detectors that respond to simple patterns. The loss function used in this stage is also a categorical cross entropy.

Once the training of a detector is complete, the next detector is added, and the same training process is performed. As shown in Fig. 5, the weights within the feature extractor, weights of the learned pattern detectors, and weights of the connections between the pattern detectors are all frozen. Only the weights within the newly added pattern detector and weights of its connections with the previous pattern detector are updated through training. Regarding the weights of the connections with the preceding pattern detector, they are primarily initialized randomly, except for the weights of edges connecting the nodes corresponding to the same character type, which are initialized to 1. This

initialization is done to actively utilize the output of the previous pattern detector. By completing the training of the last pattern detector at the stage depicted in Fig. 4(a), we obtain the final character recognition network. Furthermore, as illustrated in Fig. 5, by tracing the sparse connections between detectors from the nodes of the final output layer to the feature extractor, we can understand the simple patterns on which each character is recognized.

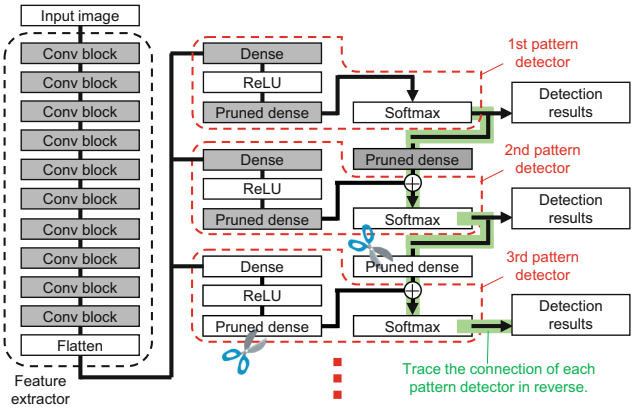


Fig. 5. Adding pattern detectors (gray boxes indicate that weights are frozen).

4 Experimental Results

This section describes the experimental results of the proposed method, which was implemented on a computer with an Intel Xeon E5-1603 v4 2.80 GHz CPU, with 16.0 GB RAM, and an NVIDIA Quadro K620 GPU chip. All the processing was coded in Python. Tensorflow and OpenCV were used to implement the neural network and text image input/output and transformation processes, respectively. The Tensorflow model optimization toolkit was also used for pruning the edges of the neural network. The analysis of the connectivity between pattern detectors was conducted using the anytree library.

The handwritten Japanese character image data set ETL9B was used in this experiment. This set consists of binary images of 201 64 × 63 pixel sized characters for each of the 3036 character types, including Japanese Kanji. In this experiment, 80 types of Kanji learned in the first grade of elementary school, which are relatively simple in structure, were used as the target characters for recognition to confirm whether the basic hierarchical structure within Kanji could be acquired. For each character type, 120, 21, and 60 images were used for training, validation, and test, respectively. For the training images, a random projective transformation was applied to generate 300 character images for each character type. Of the generated images, 270 and 30 were used as training and validation images, respectively. As a result, 390, 51, and 60 images were prepared for training, validation, and test for each character type.

The parameters of the network are shown in Fig. 6. First, as described in Sect. 3.2, we trained the CNN consisting of a feature extractor and a classifier for 1000 epochs using the training images to obtain the model with the best recognition rate for the validation image. The recognition rate of this model for the test image was 97.8%.

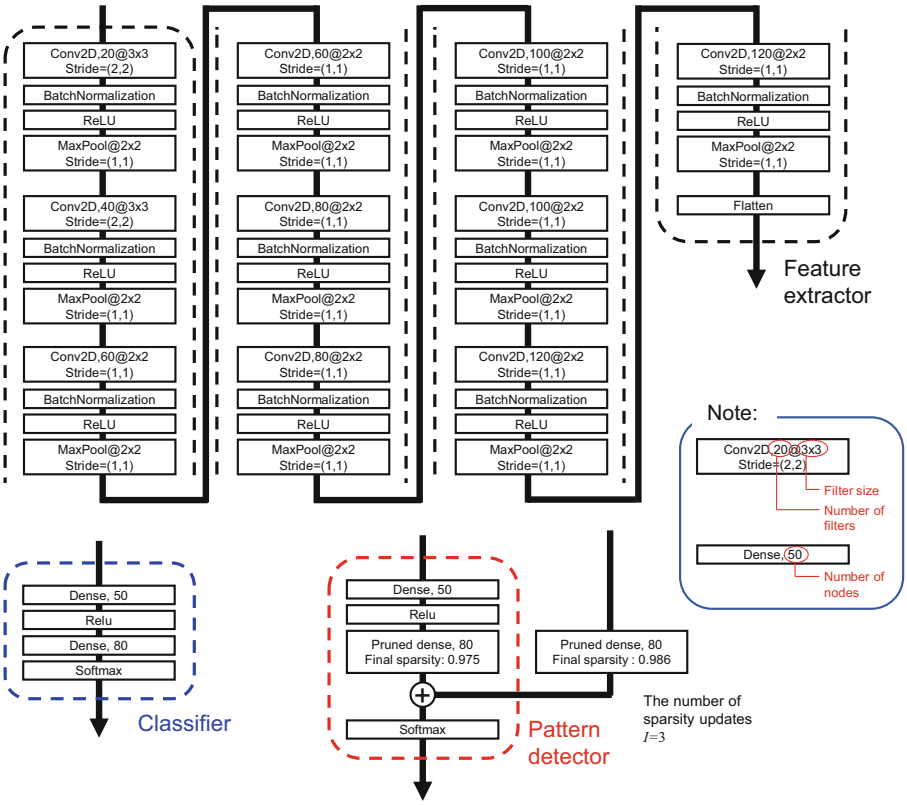
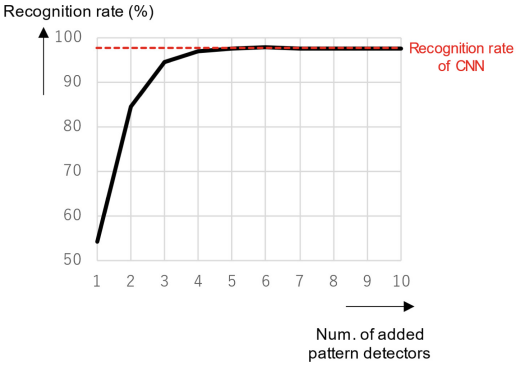


Fig. 6. Parameters in the proposed neural network.

Then, the feature extractor was extracted, and as described in Sect. 3.3, the pattern detectors were sequentially added. By utilizing the learning method described in Sect. 3.3, the weights of the edges in the pruned dense layer within the pattern detector and the one connecting them were acquired while pruning the unnecessary edges. As shown in Fig. 6, the values of final s_f used in the pattern detector were extremely high. This indicates that each node on the output side of the pruned dense layers used in the pattern detectors was connected to at most a few nodes on the input side and, in some cases, might not be connected at all. Therefore, the pruned dense layers used in the pattern detectors were composed of highly sparse edges. During training, early stopping was applied using the validation data to shorten the training time. Figure 7 depicts the recognition rate at the end of character recognition training using edge pruning, where pattern detectors were sequentially added. Despite the pruned dense layers

involved in the output of the pattern detector being composed of highly sparse edges, it was observed that by connecting only a few pattern detectors, a performance comparable to a conventional CNN could be achieved. The highest recognition rate was obtained when six pattern detectors were connected, yielding a recognition rate of 98.0%.



	Num. of added pattern detectors	Recognition rate
CNN		97.8%
Proposed method	1	54.4%
	2	84.5%
	3	94.7%
	4	97.3%
	5	97.9%
	6	**98.0%**
	7	97.7%
	8	97.8%
	9	97.9%
	10	97.8%

Fig. 7. Recognition rates of the proposed method.

Next, in the network, where the six pattern detectors with the highest recognition rates from Fig. 7 were connected, we examined to which character types the output nodes of each pattern detector's output layer responded. Figure 8 lists the nodes in the output layers of each pattern detector that responded to multiple character types and maintained connections with the output layer of the sixth pattern detector. Clearly, the initially added pattern detector outputted recognition results through a pruned dense layer with highly sparse edge connections, causing many nodes in the output layer to respond to multiple character types. However, as the pattern detectors were sequentially added, it was observed that by utilizing the outputs from the previous pattern detector and the feature extractor, each node in the output layer narrowed down the character types to which it responded. Notably, starting from the fourth pattern detector, each node in the output layer began responding to only one character type. In this validation, we expected some nodes in the output layer of the pattern detector to respond to similar character types. However, as shown in Fig. 8, most of the nodes that responded to multiple character types in the output layer were not responding to character types that appeared very similar. It was unclear whether these were detectors that were mistakenly obtained or if there were indeed common patterns that humans cannot intuitively grasp. Additionally, as mentioned in [4], it was possible that they were also responding to regions that humans did not anticipate. In the future, it will be necessary to combine existing methods, such as activation maximization, as described in Sect. 2.1 to visually present to which parts of characters each node in the output layer of each pattern detector was responding.

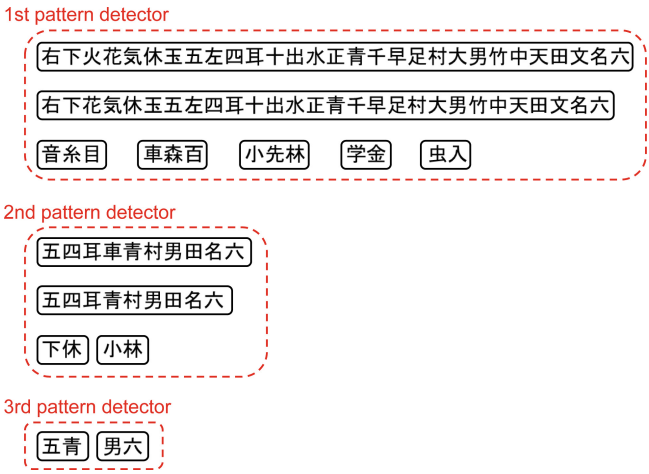


Fig. 8. Nodes in the output layers of pattern detectors responding to multiple characters.

Additionally, the connectivity of the pattern detectors was examined after applying edge pruning to the 6th pattern detector. Several examples are shown in Fig. 9. The solid round rectangles represent nodes in the output layer of the pattern detectors, and the characters inside the rectangles indicate the character types to which the nodes responded in the corresponding character image. The solid round rectangles represent nodes in the output layer of the pattern detector, and the characters in the rectangle indicate that the node responded to character images of that character type. Arrows connecting the nodes represent the remaining edges in the pruned dense layers that connected the pattern detectors. The red round rectangles indicate that when the value of this node increased, the value of the corresponding node in the 6th pattern detector also increased. The blue round rectangles indicate that when the value of this node increased, the value of the corresponding node in the 6th pattern detector decreased. As shown in the upper left tree in Fig. 9, in 51 out of 80 cases, a node responding to a certain character type in the output layer of each pattern detector was serially connected to a node responding to the same character type in the output layer of the sixth pattern detector. Additionally, in the other trees in Fig. 9, where nodes in the output layer of the pattern detectors were interconnected in a more complex manner, it can be observed that this sequential connection was still present. Furthermore, nodes in the output layer of a particular pattern detector tended to suppress nodes that responded to visually similar, but different character types in the output layer of the 6th pattern detector. This hierarchical structure allowed for the acquisition of the similarity relationships among

the target character types. Additionally, in Fig. 9, subtrees within the shaded regions of the same color indicate that they shared the same structure. This implies that the nodes indicated by the red arrows contributed to the final recognition results of multiple character types. In other words, acquiring these nodes meant obtaining detectors that responded to simple patterns, which was the goal of this study. Representing these connectivity relationships in a tree structure signifies the description of the underlying patterns governing the decision-making process. However, it should be noted that these considerations are subjective and difficult to evaluate objectively. Future work should explore quantitative evaluation methods.

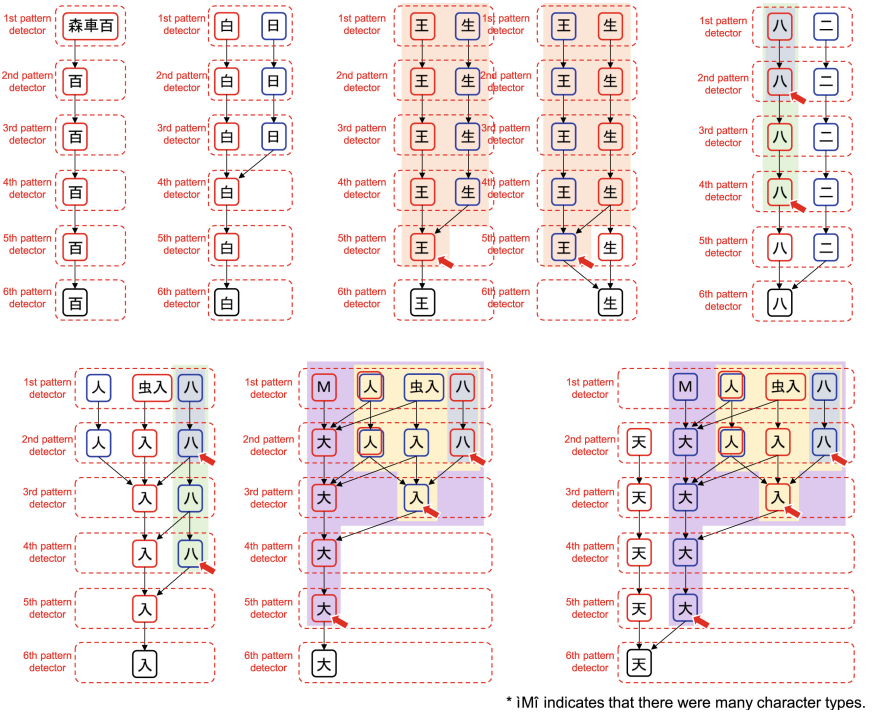


Fig. 9. Obtained connection relationships of pattern detectors.

Figure 10 illustrates examples of node values in the output layer of the pattern detectors for the character images that the proposed method failed to recognize. On the left are the pattern detectors corresponding to the correct character types, while on the right are the pattern detectors corresponding to the misrecognized character types. It can be observed that when the input character was corrupted, nodes corresponding to similar characters also responded and tended to suppress the final output. However, these observations are subjective. In the future, it is necessary to visualize how each node responds to specific patterns using techniques, such as activation maximization. Furthermore, it is essential to establish quantitative measures and conduct more objective evaluations.

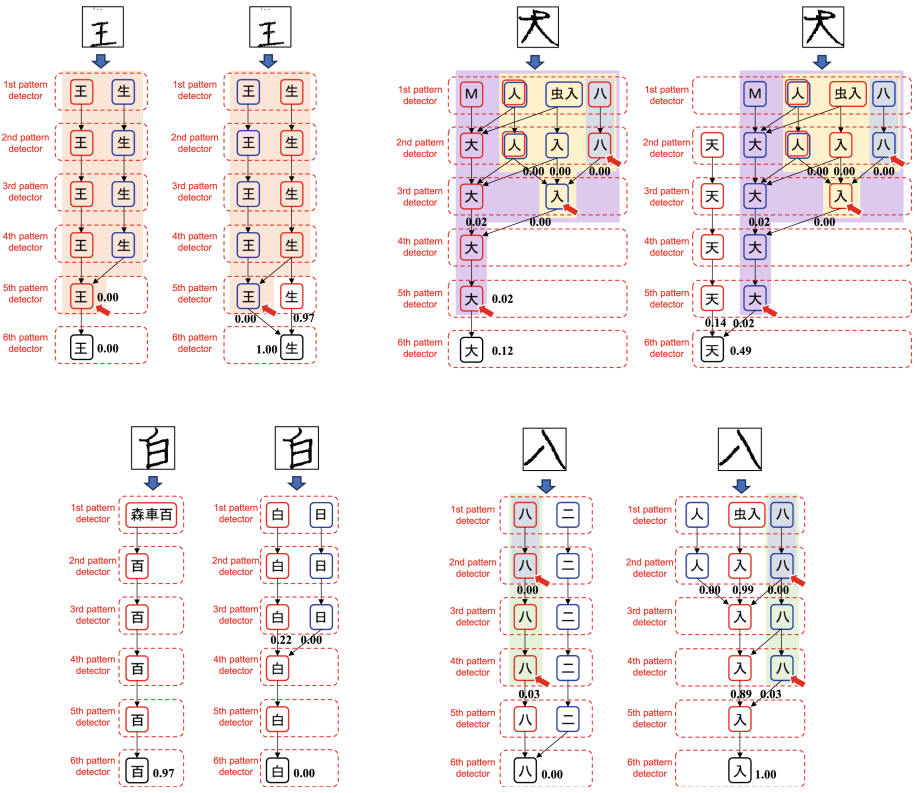


Fig. 10. Examples of node values in the output layers of the pattern detectors for erroneously recognized character images.

5 Conclusions

In this study, we proposed a method to explain the rationale behind judgments based on pattern structure using a neural network for handwritten Kanji character recognition. This approach involved sequentially connecting pattern detectors to feature extractors obtained from a CNN designed for handwritten Kanji character recognition. During the training process, pruning the network enabled each pattern detector to respond to simple patterns shared among similar characters. Furthermore, pruning the network connecting the pattern detectors allowed us to represent the relationships among detectors as tree structures, depicting the foundational patterns for judgments. Through experiments, we confirmed that the proposed method achieved a character recognition performance comparable to that of CNN despite being composed of a sparse neural network. Additionally, we confirmed that the connection relationships among pattern detectors allowed us to understand the relationships among simple patterns shared among similar characters, represented as tree structures.

While we succeeded in acquiring detectors that responded to multiple Kanji characters in common, we were unable to visualize to which patterns in the character images

these detectors were responding. In the future, introducing the conventional method described in Sect. 2.1, which visually presents judgment rationale, is necessary. Furthermore, owing to the computational limitations in this study, we froze the weights of the pre-trained feature extractors and pattern detectors, applying learning only to the newly added pattern detectors. Moving forward, we plan to introduce fine-tuning and explore learning methods that tune all the weights within the network.

Acknowledgements. This work was supported by JSPS KAKENHI (Grant Number JP 19K12045).

References

1. Singh, A., Sengupta, S., Lakshminarayanan, V.: Explainable deep learning models in medical image analysis. J. Imaging **6**(6), 52 (2020)
2. Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., Zhu, J.: Explainable AI: A brief survey on history, research areas, approaches and challenges. In: Tang, J., Kan, M.-Y., Zhao, D., Li, S., Zan, H. (eds.) NLPCC 2019. LNCS (LNAI), vol. 11839, pp. 563–574. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32236-6_51
3. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning, pp. 3319–3328 (2017)
4. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?" Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)
5. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Univ. Montreal **1341**(3), 1 (2009)
6. Nguyen, A., Yosinski, J., Clune, J.: Understanding neural networks via feature visualization: a survey. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.-R. (eds.) Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. LNCS (LNAI), vol. 11700, pp. 55–76. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28954-6_4
7. Zhang, J., Du, J., Dai, L.: Radical analysis network for learning hierarchies of Chinese characters. Pattern Recogn. **103**, 107305 (2020)
8. Liang, T., Glossner, J., Wang, L., Shi, S., Zhang, X.: Pruning and quantization for deep neural network acceleration: a survey. Neurocomputing **461**, 370–403 (2021)
9. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybernet. **9**(1), 62–66 (1979)