Enhancing Neural Network Training, Architecture, and Verification

唐, 一平

Enhancing Neural Network Training, Architecture, and Verification

Yiping Tang

February, 2024

Abstract

In recent years, artificial intelligence technology has ushered in tremendous progress, and at its core is the neural network powered by deep learning. However, there are still some challenges when using neural networks for several difficult tasks, especially when it comes to recognizing ancient documents, recognizing rotated objects, and verifying certain properties of the functions represented by neural networks obtained through training. In this thesis, we address these tasks by improving neural network training methods, proposing an architecture that is robust to object rotation, and constructing a concise representation of a given neural network for efficient verification. Below we briefly describe our contributions.

Firstly, as a first step in ancient document recognition, we develop neural network training methods for recognizing Japanese historical handwritten characters called "Kuzushiji" written consecutively. Kuzushiji characters are hard to recognize because a series of characters of different sizes are continuously written without breaks, which makes it hard to segment each character. Moreover, the shape of a character is sometimes significantly deformed depending on the characters before and after it. To overcome the difficulties, we employ an object detection method called Yolo to identify the location of each character as well as its label simultaneously. More precisely, based on the prior knowledge that the detection targets are aligned only vertically, we modify the procedure of integrating a number of candidate bounding boxes into a few final boxes, each of which hopefully contains a single character image. Our method achieves significantly better accuracy than just using the original Yolo, and also outperforms other baselines. However, the object detection based method has the disadvantage that we need to annotate each image not only with character labels but also the correct bounding boxes as its location information before training neural networks. To reduce the cost of bounding box annotation, we develop an automatic bounding box annotation method by using a single Kuzushiji image database provided by the Center for Open Data in the Humanities (CODH). Specifically, we choose three images randomly from the CODH database and arrange them

vertically to create fake training data with bounding box annotation and train the neural network with a number of the fake training data to obtain a low-accuracy recognizer. We then use the recognizer as a bounding box annotator to annotate the true training data with bounding boxes and retrain the recognizer repeatedly. We applied to the 2nd Kuzushiji Recognition Contest held in 2019 sponsored by the IEICE PRMU technical committee and our proposed method won the 4th place.

Secondly, we propose a convolutional neural network (CNN) architecture with a hexagonal lattice structure, called a Hexagonal CNN (HCNN), which has the property of rotation invariance in steps of 60 degrees. That is, for any HCNN f and any input image x to f, it holds that $f(x) = f(\bar{x})$ where $\bar{x}$ is an image obtained by rotating x by 60 degrees. This contrasts with the fact that the typical CNN architectures only provide the translation invariance property. Therefore, when the recognition targets may be rotated arbitrarily such as insects and microscopic images, we often need to use data augmentation technique to generate a number of images rotated at various angles from each of the training image data, resulting in a large amount of training data. To address the problem, several CNN architectures with rotation invariance have been proposed, but they only guarantee the rotation invariance in steps of 90 degrees or lose translation invariance, so the improvement in recognition accuracy seems to be limited. In the HCNN, each convolution channel consists of six sub-channels with hexagonal kernels, each corresponding to the rotation angle 0, 60, 120, 180, 240, and 300 degrees, respectively, with an upper layer that integrates them. This achieves rotation invariance in steps of 60 degrees, without losing translation invariance. Experimental results on the Rotated MNIST dataset, synthetic biomarker images, and microscopic cell images demonstrate that the HCNN, without using data augmentation, outperforms several existing methods.

Thirdly, as a technical contribution to explainable AI (XAI) research, we propose an algorithm that constructs a Boolean circuit representation from a given neural network for later use for efficient verification of certain properties. As a promising approach to give an appropriate interpretation to the classifier represented by a neural network, there is an attempt to construct a Boolean function representation such as a decision diagram (DD) and a Boolean circuit that is equivalent to the classifier when restricting the domain to the binary vectors. However, a blowup in the size of the resultant logical representations is a major problem, besides a large amount of calculation is required to construct them. To tackle the problem, we develop a general Boosting algorithm to construct a concise DD as a final hypothesis. The algorithm has the excellent property that when the target is a linear threshold function, the larger its margin, the smaller

the size of the DD obtained and the smaller the amount of calculation. Applying the algorithm to each of the linear threshold gates that make up a given neural network classifier, we obtain DD representations of all the gates, from which we can easily construct a Boolean circuit that represents the given classifier. To evaluate the effectiveness of our construction, we conducted experiences of verifying the robustness of a classifier from its Boolean circuit representation, where the robustness is widely used to measure the tolerance of a classifier to noise or adversarial attacks. More precisely, the robustness of a classifier to an input image x refers to how many pixels of x need to be flipped to change the prediction of the classifier. Although it is NP-hard to compute the robustness, we can compute it with the aid of a SAT-solver. Experimental results show that our construction generates Boolean circuits of smaller size, thereby allowing more efficient verification of robustness than existing methods.

In conclusion, this thesis contributes to the enhancement of neural networks in three aspects: training, architecture, and verification.

Acknowledgements

Firstly, I would like to show my gratitude to my supervisor Professor Eiji Takimoto. He allowed me at the beginning of 2019 to pursue a Ph.D. and offered me a research assistant position for financial support for my work and life. I learned a lot from the discussion with him not only about mathematics, and online learning but also about the correct attitude towards the academy. For instance, how to read the papers and extract the insights from the papers.

Next, I am extremely grateful to Professor Kohei Hatano. He offered me a part-time job in RIKEN AIP computational learning theory team. Simultaneously, I benefited from discussions with him on both academic issues and my career.

I also would like to thank Professor Yukiko Yamauchi, the chair of the committee, and Associate Professor Hiroto Saigo, the member of the committee, who gave me many valuable comments.

I am also grateful to the technical staff of our laboratory, Ms. Sanae Wakita, Ms. Chiaki Ikechi, Ms. Akiko Ikeuchi, Ms. Junko Umemoto, and Ms. Yuko Niki. They always supported me in many situations not only paperwork. I also express thanks to all of the staff in the Department of Informatics, Kyushu University.

Last, but not least, I thank my family and friends for their support.

The results of this thesis were published in proceedings of JADH 2018, proceedings of HIP 2019, proceedings of DS 2023 and journal of IEICE transaction. I appreciate all editors, committees, anonymous referees, and publishers.

Contents

Chapter 1

Introduction

In the context of the development of artificial intelligence and neural networks, this technology has gradually integrated with ”Humans,” giving rise to numerous intriguing topics. This thesis revolves around convolutional neural networks, starting from basic recognition tasks, exploring various properties from the design of convolutional neural network architecture, and further proposing methods for more efficient validation of network properties. These properties are closely related to recognition tasks and provide a theoretical foundation for improving the recognition accuracy and efficiency of verification of neural networks. Regarding the recognition task, we aim for the recognition task of Japanese handwritten ancient characters called ”Kuzushiji”. ”Kuzushiji” originates from the Edo period in Japan and is distinct from modern Japanese characters. Its most significant feature is that the shapes of individual identical characters often show a large amount of difference. Kuzushiji is a kind of hand-written character whose strokes are omitted and connected. Based on these characters there exists not only the intriguing culture and formal documents of Japan but also the attractive art of handwriting skills. Furthermore, we find that most of the existing documents and books are written in ”Kuzushiji”, which kinds of character shapes are influenced by human factors such as character adhesion, personal writing habits, local customs etc. Additionally, due to the significant passage of time since the Edo period, damage and aging of books and documents are inevitable, further complicating the recognition task. Before the widespread application of convolutional neural network technology in image recognition tasks, the digitalization conversion of ”Kuzushiji” largely depended on the labor-intensive efforts of experts. In 2017, the Center for Open Data in the Humanities (CODH) [1] in Japan organized an online ”Kuzushiji” recognition contest named

[1] http://codh.rois.ac.jp/index.html.en

PRMU [2] and provided single-character "Kuzushiji" image datasets, 3-character "Kuzushiji" image datasets, and multi-character "Kuzushiji" image datasets. Upon examining these datasets, we found various factual errors, including but not limited to discrepancies between character positions and annotations, character labels and annotations, and the presence of more than three characters in 3-character images. Consequently, we organized experts to conduct double-checks on this dataset and construct a new dataset named "Japanese Historical Hand-Written Characters Segmentation Data (Segmentation Data Set)". Through manual checks, we corrected 104 instances in the 3-character dataset and excluded 548 tuples from our dataset since experts could not recognize them. Considering the exceptional performance of convolutional neural network technology in the recognition contest, where the recognition accuracy of 49 single character "Kuzushiji" images has reached 97.33% [11], our focus on the recognition accuracy of the 3-character "Kuzushiji" dataset, which has an accuracy of only 87.12% [46]. In the recognition contest, the winning team Nguyen et al. applied sliding window technology to recognize the characters sampled from each window as a single character from top to bottom, and obtained the confidence rate of each corresponding character. These confidence rates are a sequence of extracted features processed by Bidirectional Long Short-Term Memory (BLSTM) to extract the relation information between sliding windows [46]. The advantage of this method is that it can make full use of mature single-character recognizers to extract character confidence information. However, when adhesion occurs in "Kuzushiji", it is difficult to obtain good recognition results. Therefore, to address this problem, we propose two methods for segmentation and recognition of "Kuzushiji" characters. The first method simultaneously learns segmentation rules and character classifiers from a dataset with character labels and segmentation information. This method is based on the traditional object detection method and proposes to use clustering technology to perform a secondary selection of bounding boxes (BBs) to achieve more accurate segmentation. The second method is for segmentation and can be used alongside any single-character recognizer. This method first uses selective search (SS) to generate a large number of BBs, then uses histograms to identify character areas, and finally passes the accurate BBs to a well-trained single-character classifier for recognition. Both of our proposed methods focus on using BBs to divide character areas; the difference is that one is derived from learning the segmentation information in the existing data set to obtain BBs, and the other depends on the characters and background information in the image. Compared with other traditional character recognition methods, our method has obvious advantages in recognition accuracy and complete

[2] https://sites.google.com/view/alcon2017prmu (in Japanese).

the recognition of 3-character "Kuzushiji" images in real-time. On another front, we realize that manually annotating character coordinate boxes is not only time-consuming but also unnecessary. We design a self-learning approach that utilizes single-character and three-character image data. Through iterations, this method autonomously generates BB for each character, seamlessly integrating with the previously mentioned recognition method. This approach achieves high-precision recognition without the need for explicit character coordinate information.

During the "Kuzushiji" recognition task, we observed that the character could be correctly classified regardless of its placement within the image. However, when the character has a certain rotation degree, recognition often fails. This phenomenon is typically attributed to the inherent translation invariance of traditional convolutional neural networks, which struggle with recognizing rotated objects. This deficiency not only impacts the recognition of handwritten ancient characters but also affects the recognition tasks in normal. To solve this problem, research on rotation invariance is crucial, as it entails enabling neural networks to recognize objects regardless of their orientation. Some related methods based on special convolutional kernels tend to collect more rotation feature information with larger kernels (9×9 [44] or 15×15 [19]), simulating polygons, but this leads to an exponential increase in computational complexity. [8] introduces a special structure for enhancing the rotation invariance of convolutional neural networks. They acquire rotation feature information by rotating and feature maps based on Group CNN (GCNN) [13] and concentrate different rotation feature information onto one feature map using a partial occlusion method. This operation maintains the same number of channels as the original networks and enhances the 45-degree rotation invariance for the network. These related works attempt to enhance rotation invariance by modifying convolutional kernels or using special network structures, but complete rotation invariance has not been achieved [13, 8, 4, 59, 44]. We make use of the terms to define the complete rotation invariance of a function $f(\cdot)$ with respect to a rotation $g(\cdot)$ in the sense $f(g(\cdot)) = f(\cdot)$. The data augmentation technique works but costs more compute time and memory. Based on the principle that hexagons can be rotated losslessly by 60 degrees (have the same structure before and after rotation), we propose a hexagonal convolutional kernel and a network structure that can easily replace the convolutional layers in any convolutional network. Our hexagonal kernel, which only requires 7 pixels, achieves 60-degree rotation invariance (traditional 3x3 convolutional kernels typically cover only 90-degree rotations). We prove that when the input image is hexagonal (composed of hexagonal pixels rather than square pixels) and our proposed method is applied to a fully convolutional network, complete rotation invariance at 60-degree rotation is achieved without any data augmentation.

When the input data is non-hexagonal images, we have to use resampling to obtain hexagonal-like images to make them suitable for our method. It should be noted that adding a non-convolutional layer (such as a pooling layer) to the network will affect the invariance, leading to changes in recognition results for rotated images. Our experimental results show that our method achieves rotation invariance on a homemade simple pattern image data set and exhibits excellent performance on microscopic cell recognition tasks [9], surpassing other related works.

Subsequently, we intend to understand the decision process of neural networks. It is well known that neural networks are often viewed as a black box; this is because the decision-making process within them is agnostic. This means that exploring the properties of neural networks is challenging. Research on opening the "black box" of neural networks is a new field known as neural network interpretability, which aims to make investigating the properties of neural networks more efficient. For this topic, some promising research simplifies the problem: limit the weights or biases of the neural network to binary; limit the input and output of each layer in the neural network to binary; use fixed specific network structure, etc. Then the neural network can be converted into an equivalent Boolean representation, enabling the use of lots of mature verification methods to investigate network properties, such as equivalence, model count, and robustness. Here, we focus on how to compute the robustness of neural networks efficiently. Robustness is a property used to measure the stability of neural networks and is defined as the minimum number of alterations to the network input that can change the network output, i.e., $r_f(x) = \min_{a\in\{-1,1\}^n}\{\|a\|_1 | f(x) \neq f(x \oplus a)\}$, where $x \oplus a = (x_1 \oplus a_1, \ldots, x_n \oplus a_n)$ with $x_i \oplus 0 = x_i$ and $x_i \oplus 1 = -x_i$. In practice, directly calculating robustness from neural networks is not straightforward because it requires multiple changes to all input values followed by observation of whether the results change [37, 41, 60, 63, 66]. Computing robustness from the equivalent conjunctive normal form (CNF) or circuit expression of neural networks seems like an NP-hard problem but can be solved using SAT solvers. In addition, computing robustness from the equivalent decision diagram expression of neural networks requires only polynomial time, but the resulting decision diagrams can be very large. Therefore, we start from the representation of the circuit. Shi et al. provide an idea: (i) Restrict the domain of each neuron in the neural network are fixed to $\{-1, 1\}^n$, it can be regarded as a linear threshold function (LTF), which can be converted into the form of a decision diagram; (ii) the decision diagram can be linearly converted into the form of a circuit; (iii) according to the structure of the neural network, the equivalent circuit of the neural network can be obtained by combining circuits [52] into one. During this process, steps (ii) and (iii) are straightforward and involve

simple transformations. Step (i) is the critical step that causes the decision diagram to grow explosively. Inspired by the work of [42], we construct decision diagrams with the characteristic "Node labels are layered, and each label can appear in different layers", named Aligned Binary Decision Diagram (ABDD). The characteristic of "Node labels are layered" is derived from ordered binary decision diagrams (OBDD), which simplifies the calculation in SAT solvers. The characteristic of "each label can appear in different layers" is specific to binary decision diagrams (BDD) and is looser than the condition that the same label can appear only once in the same layer of an OBDD. This characteristic helps make the resulting decision diagram (DD) smaller. We conduct a sample-based robustness (SR) experiment, using the same neural network setting from [52], which includes a 3-layer neural network with two convolutional layers and one fully connected layer. To compute the robustness of the neural network, we design an additional circuit $d_{k,x}$ that represents the Hamming distance k between sample x and a new sample $x\prime$. We use SAT solvers to solve the satisfiability problem related to the circuit $d_{k,x}$ and the equivalent circuit of the neural network. The experimental results show that our method produces the smallest decision diagrams, and the time required to calculate robustness using our decision diagrams is the least, making it the most efficient. Concerning the representation of neural networks using decision diagrams directly, Shi et al. provide an approach by sentential decision diagram (SDD) [52]. SDD is a special decision diagram in which different nodes can perform $(\vee, \wedge)$ operations through the $apply(SDD-1, SDD-2)$. By converting our ABDD into SDD and connecting them according to the neural network structure, more complex robustness tests, such as model-based robustness, can be achieved at a lower cost.

In conclusion, this thesis covers a wide range of topics related to the integration of recognition, architecture and verification of neural networks, with a specific focus on "Kuzushiji" character recognition, rotation invariance and interpretability. This thesis proposes novel methods for character segmentation, recognition, and improving the rotation invariance of neural networks. Additionally, we explore the field of neural network interpretability and efficient robustness computation. Our experimental results demonstrate the effectiveness of our approaches in various tasks.

1.1 Our main Contributions

In this section, we briefly describe our contributions to each problem.

1.1.1 Construction of Japanese Historical Hand-Written Characters Segmentation Dataset

We construct a more accurate 'Kuzushiji' dataset from the CODH dataset and PRMU dataset. The image data are categorized into single-character images, 3-character images, and multi-character images. Each character has its category label and coordinate information. Mislabeled and erroneous images have been modified or excluded.

1.1.2 3-Character "Kuzushiji" Recognition

In contrast to the recognition of single-character images, multi-character recognition tasks require the prior classification of the location of the target character before recognition. In previous research, the selection of character positions relies on bounding boxes, which means that the quality of recognition results depends on the choice of bounding boxes. However, in the context of "Kuzushiji" recognition tasks, due to the influence of character shapes, writing habits, and character adhesive phenomena, selecting the appropriate bounding boxes is challenging. Our method utilizes a combination of bounding boxes learned from segmentation information by neural networks and clustering algorithms, along with the selective search technique and histogram-based judgment, to accurately select suitable bounding boxes. Our technical contribution is to devise a better aggregation method of possible candidates of bounding boxes and associate labels suited for Kuzushiji character recognition. Additionally, we introduce a method that achieves high-precision text recognition and bounding box output simultaneously without requiring pre-prepared character coordinate information.

1.1.3 Hexagonal Kernel-Based Method to Achieve Rotation Invariance

The work on the rotation invariance of neural networks aims to enhance their generalizability about rotation. Neural networks with complete rotation invariance can produce the same results from rotated images as they do for unrotated images after learning. Previous related works utilize special convolutional kernels or network structures to capture rotation features in unrotated images but these methods have obvious limitations. Our approach utilizes the hexagonal kernel to propose a novel network structure with two kinds of hexagonal convolutional layers to achieve rotation invariance. Our contributions can be summarized as follows: (1)We prove the rotational invariance of HCNN (2)We improve the rotation robustness of the CNNs using proposed methods. (3)Our Ro-block can be easily applied to other CNNs.

1.1.4 Efficiently Investigate the Property of Neural Networks

Exploring the properties of neural networks is a shortcut to understanding their decision-making behavior. However, due to the opacity, exploration often comes with many challenges, which is why they are referred to as black boxes. Neural networks typically consist of numerous layers of neurons, with each neuron performing mathematical operations and passing results to the next layer. The interactions between these layers are complex and difficult to explain, as the function of each neuron depends on various factors. When a neural network receives input and generates output, the intermediate process involves many hidden layers and nodes, where each node may influence the final result. Several related works attempt to transform neural networks into more interpretable representations, such as formulas, circuits, and decision diagrams. In [52], the key issue is framed as how to obtain more efficient equivalent representations for each neuron in a neural network. We improve the boosting algorithm from [42], using full-sample inputs to learn equivalent decision diagrams for neurons with a training error of 0. The main difference between our method and other related works lies in the determination of the variables used to label nodes in the decision diagram. In our decision diagram, "node labels are layered, and each label can appear in different layers." Our modification might look easy but is non-trivial in a theoretical sense. To achieve the same theoretical guarantee with [42], we introduce a new information-theoretic criterion to choose variables that are different from the previous work. Our method is based on the assumption that when a neuron is regarded as an LTF, the larger the margin of the LTF, the smaller the decision diagram we obtain. In experiments, our decision diagrams more efficiently perform investigations of neural network properties while keeping the decision diagram size to a minimum.

1.2 Organization

The rest of this thesis is organized as follows: In Chapter 3, we introduce some tools or methods that are not explicitly defined or explained in the main part. In Chapter 4, we introduce the details of the Japanese historical hand-written characters segmentation dataset. In Chapter 5, we propose two kinds of segmentation methods for the "Kuzushiji" classification task and a self-learning method for generating coordinate information of Kuzushiji characters. Next, we propose a hexagonal kernel-based method to prove and achieve the rotation invariance in Chapter 6. Lastly, in Chapter 7, we consider improving the explainability of neural networks and propose

a Boosting-based method to make the equivalent representation of the network smaller and more efficient.

Chapter 2

Preliminaries

In this chapter, we will supplement some definitions and details that were not specifically described in the main text.

2.1 Neural Network (NN)

In recent years, propelled by advancements in computer hardware technology and performance, neural network [23] techniques have experienced an explosive surge in development. Neural networks emulate the functioning of neurons in the human brain using internal weights, and they achieve complex problem-solving through the transmission of these weights across layers. This paradigm shift has not only been driven by hardware improvements but also by the availability of vast amounts of data and breakthroughs in training algorithms. As a result, neural networks have become the cornerstone of artificial intelligence, demonstrating remarkable capabilities in tasks ranging from image and speech recognition to natural language processing. The adaptability and learning capacity of neural networks continue to push the boundaries of what is achievable in the field of machine learning, making them an integral part of cutting-edge technologies and applications. As shown in figure 2.1, the traditional neural network roughly consisted of the following parts. (1) Input Layer: Receives external information, similar to our sensory organs receiving input. (2) Hidden Layers: Process information, identifying patterns and features, much like our brain in thought and learning. (3) Output Layer: Produces the final result, akin to our body reacting to stimuli. Note that, in each hidden layer, the activation function should be used to make the neural network nonlinear, i.e., Relu-function [40], Step-function,. . . .

In this thesis, the part of the experiment focuses on the recognition of image data or the detection of objects within images. Therefore, we will provide a brief introduction and definition

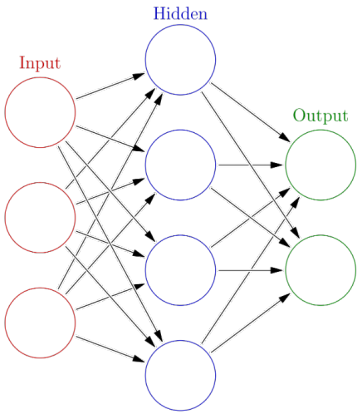


Figure 2.1: A sample of neural network

of the neural networks employed in recognition and detection tasks. The neural networks utilized include Convolutional Neural Networks (CNNs) [36] for the recognition of single Kuzushiji characters and YOLOv3 [48], an object detection tool, for detecting sequences of three consecutive Kuzushiji characters. Specifically, in the context of Kuzushiji recognition, these convolutional layers play a crucial role in capturing and identifying distinctive patterns and features within the input image, enabling the network to make accurate character predictions. Meanwhile, YOLOv3, being an advanced object detection framework, excels in efficiently detecting and localizing multiple objects within an image, making it particularly suitable for the continuous Kuzushiji character detection task.

2.1.1 Convolutional NN (CNN)

In contrast to traditional neural networks, convolutional neural networks exhibit a unique structure known as the convolutional layer, designed to extract feature information (stored in channels) from the input of the preceding layer.

Let $I_t \in R^{\text{input height} \times \text{width} \times C_t}$ to be the the input of t-th layer, where C is the number of channels. And, $K \in R^{C_{t+1}, C_t, \text{kernel size} \times \text{kernel size}}$ to be the convolutional kernel used to extract feature information. (In general, kernel size is set to 3.) As shown in figure 2.1, the calculation of the traditional convolutional layer at channel $c \in C_{t+1}$ at pixel (i, j) is written as follows.

$$I_{t+1,c}[(i,j)] = \sigma\Big(\sum_{1 \leqslant l \leqslant C_t, 0 \leqslant m,n \leqslant 2} I_{t,l}[(i+m, j+n)] \times K_{c,l}[(m,n)]\Big). \tag{2.1}$$

where σ is an activation function.

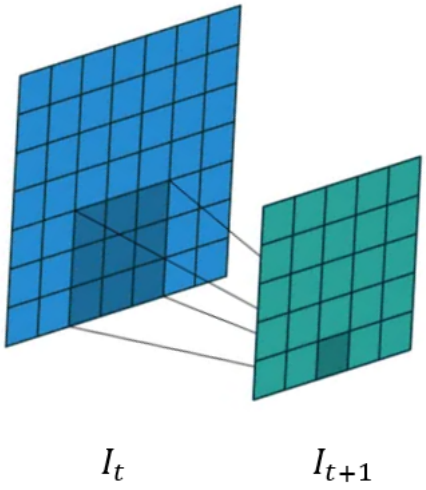


Figure 2.2: A sample of the convolutional layer.

2.1.2 Bianry CNN (BCNN)

In Chapter 5, we introduce more sophisticated variants of convolutional neural networks (CNNs), but they all follow the same convolutional calculations. In Chapter 6, based on related work, we employed a binary convolutional neural network (BCNN) [52]. During the training process, it remains consistent with traditional neural networks. However, during the prediction phase, we manually replace the activation functions with Step functions as follows.

$$\sigma(x)\begin{cases} 1, & \text{if } x \geqslant 0 \\ -1, & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that, the calculation of the convolutional layer will be redefined in Chatper 5

2.2 Object detection tool: YOLOv3

Another approach is YOLOv3, a common object detection tool, which is a framework known for its speed and accuracy. Unlike traditional methods that rely on region proposal networks and multiple stages, YOLOv3 adopts a one-stage architecture, allowing it to simultaneously predict bounding boxes and class probabilities for multiple objects in an image. The key advantage of YOLOv3 lies in its ability to divide the input image into a grid and make predictions at the grid level, enabling efficient and real-time object detection. This grid-based approach streamlines the detection process by directly predicting object parameters, such as bounding box coordinates

and class probabilities, without the need for complex post-processing steps. Specifically, its operation can be divided into two main parts: (i) the backbone neural network learns the location and classification information of the targets, providing multiple predicted bounding boxes and class probabilities simultaneously during inference. (ii) Through Non-Maximum Suppression (NMS) techniques, the bounding boxes are aggregated, precisely identifying the most suitable object positions and class information.

In the first part, the backbone neural network, often a deep CNN, is responsible for processing the input image and extracting features that are relevant for object detection. The network outputs predictions in the form of multiple bounding boxes, each associated with a specific class probability.

In the second part, Non-Maximum Suppression (NMS) is employed to refine the predictions. NMS is a post-processing technique that iteratively selects the bounding box with the highest confidence score (value of class probability) and suppresses other boxes that have significant overlap with it. This ensures that only the most accurate and relevant bounding boxes are retained, eliminating redundancy and enhancing the precision of the final detection results.

2.3 K-means clustering

The k-means algorithm originated as a vector quantization method in signal processing and is now widely employed as a clustering analysis technique [2]. In this thesis, we leverage it to address the aggregation issue in improving the bounding boxes outputted by object detection in Kuzushiji recognition task. Specifically, we apply the k-means algorithm to the centers of the bounding boxes in terms of vertical positions (drawn at the Y-axis). We partition these centers into k clusters, ensuring that each point belongs to the cluster whose mean (cluster center) is the closest to it. Given $C = \{(box_1), \ldots, (box_m)\}$ and the number of clusters $k(\in N)$, where $box_i = (x_{l_i}, y_{l_i}, x_{h_i}, y_{h_i})$ as shown in figure 2.3. Then, we apply K-means clustering to C based

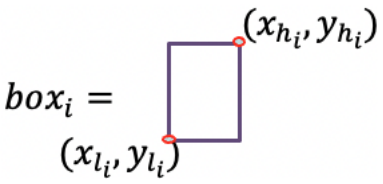


Figure 2.3: An example of box used in K-means calculation.

on $dist((box_i), (box_j)) = |\frac{y_{l_i}+y_{h_i}}{2} - \frac{y_{l_j}+y_{h_j}}{2}|$, and obtain a partition $C = C_1 \vee C_2 \vee, \ldots, \vee C_k$.

Chapter 3

Construction of Japanese Historical Hand-Written Characters Segmentation Data Set

3.1 Introduction

Character recognition techniques play a pivotal role in the realm of digital humanities. In 2017, the advancement of computers and scanners led to a notable increase in the creation of digital images of historical documents. While the demand for digitizing documents is substantial, conventional OCR (optical character recognition) systems designed for contemporary characters face limitations in directly addressing pre-modern character recognition issues. The difficulty of the task varies depending on the language. Specifically, for Japanese, the challenges in recognizing pre-modern handwritten characters using traditional optical recognition techniques stem from (i) documents being written with brushes, resulting in many characters being interconnected rather than separated by spaces, (ii) the use of various symbols (such as Chinese and Japanese characters) to convey the same meaning for a character, and (iii) the simplification or abbreviation of certain characters. Consequently, recognizing Japanese pre-modern characters from their document images remains a formidable challenge.

The recognition task can be divided into two phases, (i) segmentation of sentences to single-character images and (ii) recognition of single-character images. Given an image of a single character, it is easy to recognize by using machine learning techniques such as deep neural networks. Nguyen et al. reported that the recognition accuracy of single characters is up to 97% [46]. On the other hand, segmenting a sentence image into single-character images is a

bottleneck. Nguyen et al. proposed method has an accuracy of three consecutive characters is about 88%. So, it's still a challenge to increase the accuracy of recognition.

The goal of this work is to construct data sets of Japanese pre-modern characters with the segmentation information of each character within sentences, for which developers and researchers could validate their segmentation methods.

3.2 The CODH data sets and their variants

In 2017, the Center for Open Data in the Humanities (CODH) released datasets encompassing Japanese pre-modern characters and literatures [1]. These datasets comprise images of pre-modern Japanese documents books and transcriptions. Additionally, these datasets contain images of 403,242 individual characters belonging to 3,999 distinct types. Importantly, these datasets are made available under the CC-BY-SA license, permitting unrestricted use and modification with proper citation.

Utilizing these datasets, the Technical Committee on Pattern Recognition and Media Understanding (PRMU) of the Institution of Electronics, Information and Communication Engineers (IEICE) in Japan organized an algorithm contest aimed at recognizing Japanese pre-modern handwritten characters, known as "the Kuzujishi Challenge" [2], in 2017.

The contest featured approximately 230,000 images of Japanese kana characters from 15 ancient Japanese books within the CODH datasets. The contest datasets included tuples comprising the original image of specific characters, the characters of interest (ranging from one to about six characters), which served as the ground truth recognition results and positional information for each character expressed through the coordinates of the enclosing rectangle.

The primary objective of the contest was to recognize characters within a given image based on the provided coordinates of the bounding rectangle. Notably, the contest datasets were categorized into three difficulty levels, labeled as level 1, 2, and 3. The level 1 dataset contained 240,000 single Kuzushiji characters, while the level 2 dataset comprised 80,000 sets of three consecutive Kuzushiji characters. The level 3 dataset featured sets with multiple characters, exceeding three. An illustration of these datasets is shown in Figure 3.1.

[1] http://codh.rois.ac.jp/char-shape/
[2] http://codh.rois.ac.jp/old-char-challenge/

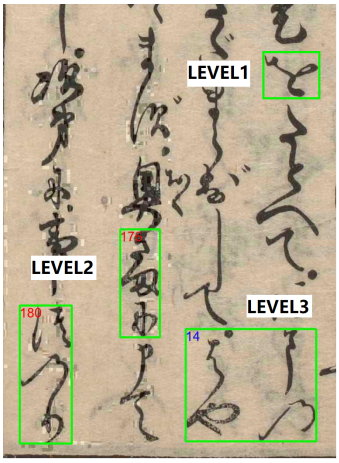


Figure 3.1: An example of contest data used in the PRMU 2017.

3.3 Construction of our data sets

Our technical work involves the creation of segmentation datasets by reorganizing the coordinate information from images in the level 1, 2, and 3 datasets obtained from CODH and the PRMU contest. Upon observation, we noted that all images in level 2 and 3 contain the same single Kuzushiji character found in the level 1 dataset, originating from the same book or document. Exploiting this observation, we can construct segmentation datasets for level 2 and 3 images by incorporating the coordinate information of bounding rectangles (consisting of the X and Y coordinates of the top-left corner, width, and height) for single characters in the corresponding level 1 character.

Consequently, we successfully generated 78,940 segmentation datasets featuring consecutive three Kuzushiji characters and 12,583 datasets containing multiple Kuzushiji characters. To elaborate, each dataset is a tuple encompassing the original image, coordinate information of a large rectangle encapsulating three or more characters, and inside rectangles delineating single characters within the larger bounding rectangle. Figure 3.2 provides illustrative examples of our dataset.

Furthermore, we verified manually all of the constructed data whether character label or coordinate. Through the manual check from experts, we corrected 104 instances from the level 2 data set and 548 tuples of the data were excluded from our data set since experts could not recognize them. Similarly, we corrected 95 instances from the level 3 data set. Examples of difficult instances are shown in Figure 3.3. Several problems such as the four characters in the three-character border image, error character labels and only half of a character in the image

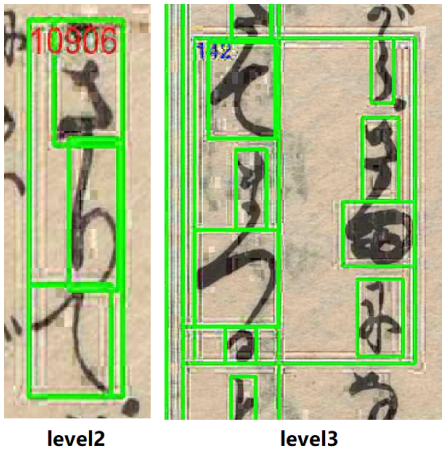


Figure 3.2: An example of our data set.

were found. At the moment, we intend to remove the problem image found in this check-work and only reformed images are used. In the next chapter, we focus on the automatic recognition of Kuzushi characters based on this segmentation data set. Our data set is available under the license of CC-BY-SA on the web site.

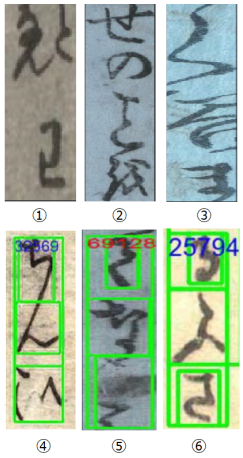


Figure 3.3: Examples of abnormal instances excluded from our data sets.

3.4 Conclusions

In this work, we constructed segmentation data sets of Japanese pre-modern characters named Kuzushiji from the CODH data sets and the PRMU contest data set.

Chapter 4

Recognition of Japanese Historical Hand-Written Characters Based on Object Detection Methods

4.1 Introduction

In this chapter, we focus on enhancing the training of neural networks through a recognition task of pre-modern handwritten characters. In addition, this work is also a contribution to digital humanities literature.

Digital humanities is an interdisciplinary field that explores humanities through the application of information technology. Recently, there has been a surge in interest across various domains, extending beyond the humanities to encompass natural language processing, data mining, image recognition, and machine learning.

Digital humanities have gained considerable traction in Japan, propelled by the development of diverse datasets, particularly through the organizations of the Center for Open Data in the Humanities (CODH) [1]. In the realm of Japanese historical literature, the character written by hand-written known as "Kuzushiji," presents challenges for automated recognition of Kuzushiji despite the growing availability of digitized images of pre-modern Japanese literature even with the state-of-the-art OCR technologies.

In comparison to recognition tasks in more established languages, such as hand-written English [27] and Arabic [1], the challenges posed by Kuzushiji character recognition tasks are pronounced due to several factors. These challenges include (i) characters being connected

[1] http://codh.rois.ac.jp/index.html.en

without explicit intervals, (ii) the use of different symbols to represent the same character, and (iii) frequent simplification or abbreviation of characters. Specifically, the complexity of the Kuzushiji recognition task lies in the difficulty of accurately segmenting sentences into individual characters. Once a sentence is correctly segmented into single characters, the recognition of each character becomes relatively straightforward. For instance, the Center for Open Data in the Humanities (CODH) reported an impressive recognition rate of up to 97.33% for single characters using deep neural network technology on 49 types of Japanese Hiragana characters from the CODH dataset [11].

The high-precision performance in recognizing multiple Kuzushiji characters was achieved by Nguyen et al. [46] in 2017, earning them the best-award in the PRMU algorithm contest [2]. Their method achieved a 12.88% error rate for consecutive three Kuzushiji characters. This approach comprises a feature extractor and recognizer, functioning as an end-to-end solution. Given an image of consecutive three Kuzushiji characters in a column, their feature extractor performs the following steps: (i) generates multiple segmented boxes by employing sliding windows of varying sizes on the input image, (ii) feeds the portion of the image within each segmented box to a single character recognizer, obtaining recognition confidence rates for all possible character classes, (iii) processes the output of the recognizer through a Bidirectional Long Short-Term Memory network (BLSTM) to extract relational information between sliding windows, and (iv) decodes the output sequence of BLSTM using a Connectionist Temporal Classification (CTC)-based transcription layer. While their approach exhibits potential for improvement, it's worth noting that the segmentation candidates they employ are generated using heuristics and lack optimization or learning from data. In this context, we intend to explore a fully image-oriented approach to achieve character recognition.

In this chapter, we introduce two novel methods for recognizing Kuzushiji characters. The first method is designed to simultaneously learn segmentation rules and character classes from datasets containing character labels and segmentation information. To enhance the utilization of coordinate information in segmentation, we leverage a segmentation dataset derived from the CODH dataset and the PRMU contest dataset [62]. This dataset is created by incorporating bounding box information from CODH datasets and segmenting multiple characters in the PRMU contest datasets. Our method draws inspiration from the YOLOv3 object recognition method [48], which, when given an image with labels and bounding boxes as ground truth for target objects, learns a classifier to predict labels and bounding boxes of candidate objects.

[2] https://sites.google.com/view/alcon2017prmu (in Japanese)

However, we observed that a straightforward application of such object recognition methods proves ineffective for Kuzushiji characters, as each character often comprises multiple symbols, and YOLOv3 tends to output only some of these symbols as candidates. One of our key technical contributions involves developing an improved aggregation method for possible candidates of bounding boxes and labels, tailored for effective Kuzushiji character recognition.

The second method proposed is a segmentation technique that can be employed in conjunction with any single character recognizer. Our methods surpass other baseline approaches, achieving state-of-the-art accuracy in both segmentation and recognition tasks on datasets featuring three consecutive Kuzushiji characters.

In addition, we also proposed a supplement for the object-detection-based method. This supplement generates the temporary consecutive three-character Kuzushiji image by stacking single Kuzushiji character images and obtaining the coordinate information of each character in it. From this, our proposal method achieves the prediction of consecutive three-character Kuzushiji images which does not depend on the coordinate information of each character.

In the experiments on the recognition of consecutive three Kuzushiji characters, our first method outperforms baselines including a naive application of YOLOv3 (an illustration of our recognition results is shown in Figure 4.1). The first method shows the benefit of training detectors using datasets containing both labels and segmentation information. In contrast, the second method exhibits inferior recognition performance compared to the first; however, it attains superior segmentation accuracy.

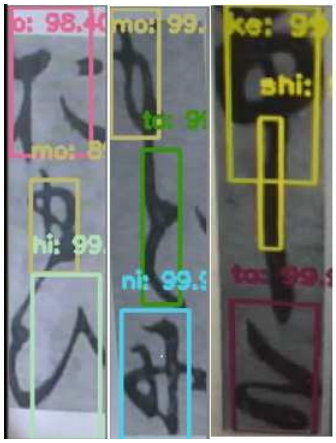


Figure 4.1: Illustration of recognizing and locating characters.

4.2 Related Work

4.2.1 CODH data sets

In 2017, the Center for Open Data in the Humanities (CODH) released open datasets featuring Japanese historical characters and literature [3]. These datasets encompassed images of pre-modern Japanese documents and books, along with their transcriptions. Additionally, the datasets included images of 684,165 individual characters belonging to 4,645 different types.

Furthermore, in 2018, CODH published two additional datasets specifically focused on Kuzushiji characters [12]. One of these datasets is named Kuzushiji-49, comprising 49 different kana characters. The other, referred to as Kuzushiji-Kanji, contains 3,832 Japanese kanji characters. Clanuwat et al. conducted recognition tasks using these datasets and achieved impressive results, with an accuracy rate of 97.33% in Kuzushiji-49 and an accuracy rate of 98.83

4.2.2 PRMU algorithm contest data sets and their Kuzushiji recognition Contest

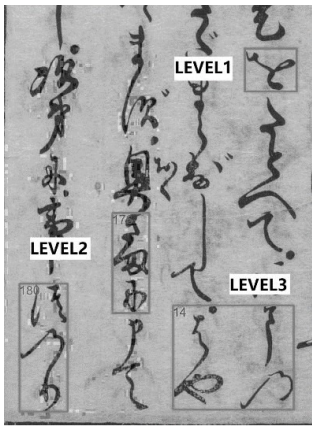


Figure 4.2: Examples of Kuzushiji separated data set.

Pattern Recognition and Media Understanding (PRMU), a Japanese association specializing in pattern recognition, organized a Kuzushiji character recognition contest in 2017. For this contest, PRMU provided a dataset derived from CODH, focusing on excerpts from original documents and book images containing three or more consecutive Japanese Kana characters. The dataset encompasses 46 varieties of Japanese hiragana characters and is categorized into

[3]http://http://codh.rois.ac.jp/pmjt/ (in Japanese)

three levels based on learning difficulty, ranging from level 1 to 3 (see Figure 4.2 for an illustration). Level 1 comprises images containing only a single character, level 2 includes images with consecutive three characters, and level 3 consists of images with more than three characters.

Nguyen et al. emerged as the winners of the contest [46]. They devised a sliding-windows-based method to generate preliminary bounding boxes and employed Convolutional Neural Networks (CNNs) to extract character features from these boxes. Their approach achieved a label error rate (LER, as defined in E.q. 4.5.1) of 12.88% for the recognition task involving consecutive three Kuzushiji characters.

4.2.3 Object detection

Object detection techniques, including Faster-RCNN [50], SSD [38], YOLO [49], and YOLOv3 [48], are commonly used for the automatic recognition and localization of objects in images or videos, such as in applications like autonomous driving and pedestrian detection.

In the realm of object detection, two main types of methods exist: the one-stage method and the two-stage method. The two-stage method, exemplified by RCNN [25] and Fast-RCNN [50], typically involves the use of techniques to identify a large number of bounding boxes enclosing objects before the actual recognition step. Some of these techniques leverage color and texture changes between the target object and the background in the image or employ pixel distributions and heat maps to locate selected boxes.

However, in the case of Kuzushiji character recognition, applying these two-stage methods can be challenging due to the monochrome nature of the images, which contain minimal color or texture information. On the other hand, the one-stage method, such as YOLO [49], learns to segment and recognize objects simultaneously through a neural network. Consequently, we are considering the application of object detection techniques for the recognition and localization of Kuzushiji characters.

4.3 Proposed Methods

4.3.1 Frame Group Decision Method(FGDM)

Our first proposed method, named FGDM (Frame Group Decision Method), is built upon YOLOv3 [48], a newer version of YOLO [49], which is a one-stage object detection method

implemented using a deep neural network as the backbone. YOLOv3 learns a classifier that outputs fixed grids on the input image. Each grid corresponds to a sub-classifier, providing a fixed number of bounding boxes and confidence rates for all possible classes of objects (characters) within each box. YOLOv3 consists of two phases: the training phase and the prediction phase. In the training phase, using images with ground truth (pairs of bounding box coordinates and labels), YOLOv3 optimizes a loss function considering classification errors and location errors, as detailed in [49].

During the prediction phase, when given an image, YOLOv3 applies the trained classifier to the input and obtains bounding boxes with confidence rates from the grids. Due to the presence of numerous bounding boxes with low confidence rates (indicating poor predictions) for a single target object, YOLOv3 employs the Non-Maximal Suppression (NMS) rule to filter out such boxes. In this process, a bounding box is removed if its maximum confidence rate for a given object (character) falls below a predetermined threshold. Subsequently, other bounding boxes that overlap with the highest confident bounding box are successively removed. After repeated iterations of this step, the remaining boxes with maximum confidence rates for certain character labels constitute the output of YOLOv3. While the NMS rule is effective for general object detection, we observed that in Kuzushiji recognition cases, it tends to eliminate correct bounding boxes with low confidence rates for the correct label. This is attributed to the dense arrangement of characters in Kuzushiji recognition tasks, in contrast to the sparser arrangement seen in general object detection tasks.

Given the simplicity of input images in Kuzushiji character recognition tasks, lacking intricate background elements with a plain white background, we propose an alternative bounding box aggregation method. This method is designed to better suit the unique characteristics of Kuzushiji character images compared to general object detection tasks.

Characters in the image are often arranged on lines from the top to the bottom. So, we can easily separate different columns of the text first. The specific implementation is shown as follows. Here, for simplicity, we assume the number K of characters is a given prior knowledge of each input image.

Step1 : Remove bounding boxes with lower confidence with some threshold.

Step2 : Perform clustering with K group for bounding boxes in terms of vertical positions of their centers (Figure 4.3 shows an illustration of the result) .

Step3 : For each group, choose the bounding box with the highest confidence and the associated character.

It's worth noting that our prior assumption about the number of characters is relaxed. Even when the number of characters is unknown, if the distribution of the frames is relatively balanced, the K-means++ method [2] can be employed to estimate the number of groups.

The key distinction from YOLOv3's original aggregation method lies in our consideration of each distinct group as an attention area for the original image. This is achieved by observing the Y coordinate of each bounding box (after the separation into columns) and selecting boxes with high confidence in a local sense. This modification aims to enhance accuracy in Kuzushiji recognition, a result that will be demonstrated in the experiments later.

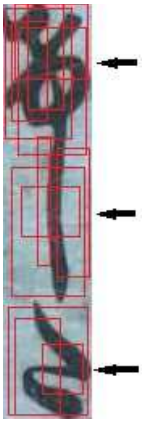


Figure 4.3: Illustration of small groups of bounding boxes formed by the FGDM. The details of the FGDM are shown on page 3 (Steps 1 to 3).

4.3.2 Frame Decision Method Based on Selective Search(FDMBSS)

The Selective Search [55] method is a valuable two-stage object detection technique utilized in various research, including RCNN [25] and Fast-RCNN [50]. This technique efficiently generates numerous candidate bounding boxes for characters. Selective search operates by creating bounding boxes around target objects based on nearby pixels and backgrounds, effectively separating objects from backgrounds through color expression. The process involves approximating the curve separating the object and background with straight lines, from which rectangles are formed. The frame combination method [58] is then applied to obtain bounding boxes for target objects in the image. Figure 4.4 illustrates bounding boxes constructed by Selective Search (marked as A) and by FDMBSS (marked as B).

When using the Selective Search method for Kuzushiji character recognition, we observe that the bounding boxes exhibit a high degree of freedom. This results in overly large or small bounding boxes that may not contain character pixels, prompting us to automatically filter them out. While there are excellent frames among these bounding boxes, they are limited in number. Furthermore, for hand-written characters like Kuzushiji, the histogram method often identifies a nearly suitable bounding box, although obtaining a perfect fit is challenging. To address these issues, we propose a specialized segmentation framework named FDMBSS. This framework constrains Selective Search by utilizing the histogram method to obtain the best-fitting and most suitable bounding box for target objects.

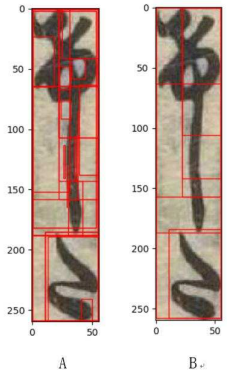


Figure 4.4: Illustration of the bounding boxes forKuzushiji characters constructed by the selective search.

The FDMBSS (Frame Decision Method with Selective Search) is a three-step process designed to enhance bounding box accuracy for Kuzushiji character recognition: (i) Selective Search Candidates: Begin by using the Selective Search method to generate candidates for bounding boxes of characters. Here, we remove bounding boxes that are deemed too small or too large based on a specified threshold for size. This step ensures that only boxes with an appropriate size are retained. (ii) Histogram-Based Method: Utilize the histogram-based method, taking the number of characters K as a parameter. This method counts the black pixels along each horizontal line in the image, constructing a histogram. The image is then divided into $K-1$ horizontal lines, determined by the $K-1$ lowest points in the histogram. (iii) Bounding Box Comparison: Compare the bounding boxes obtained from the Selective Search and the histogram-based method. For each bounding box (X, Y, W, H) from the histogram method, calculate the sum of absolute differences for the four coordinates (where (X, Y) specifies the upper left point, and (W, H) specifies the width and height). If the sum of the error is smaller than a specified threshold, employ the bounding box from Selective Search; otherwise, use

the bounding box from the histogram method. Finally, after applying the FDMBSS method to segment characters individually, single-character recognizers such as CNN, RNN, RNN+LSTM, and complex networks like RESNET-53 [32] are employed to obtain labels for each segmented character. This multi-step method aims to optimize the accuracy of bounding box determination and subsequent character recognition for Kuzushiji.

4.4 Kusushi recognition method based on the generation of data with character segmentation information and repeated training

We also propose an expanded approach that utilizes the method described in chapter 4 for Kuzushiji character recognition. The method in chapter 4 is grounded in the object recognition approach, where the recognizer takes an image as input and outputs bounding boxes enclosing candidate Kuzushiji characters along with their corresponding labels. Its training process of recognizer requires training data augmented not only with labels for each Kuzushiji character but also with information about the bounding boxes that surround them. By learning the provided bounding box information in the training data, simultaneous prediction of character segmentation and recognition becomes feasible. Indeed, on the dataset [57], the approach achieved a low recognition error rate of 2.5%.

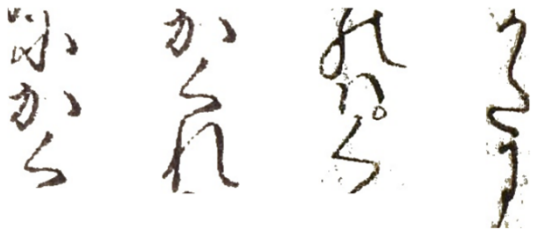


Figure 4.5: Example of three-character Kuzushiji image data.

As an extension, we considered the possibility of achieving similar accuracy in predictions without the need for bounding boxes, specifically when the input data consists solely of the images and labels for single and three-character Kuzushiji. Therefore, in this part, we create training data by augmenting the single and three-character Kuzushiji (as shown at figrue 4.5) with information about bounding boxes surrounding each character. Subsequently, we apply the chapter 4's method to the augmented data. The details are outlined below.

4.4.1 Creation of Training Data with Character Segmentation Information and Training Method for the Recognizer

(1) Creation of initial training data with segmentation information from single-character image data

For each three-character Kuzushiji image, we extract labels for three characters and obtain three corresponding single-character images. These three single-character images are combined to form a three-character image. Based on the size of each single-character image, we add information about the bounding box surrounding each character. Additionally, to generate a more natural three-character image, we randomly insert spaces between characters according to the following rule:

For each pair of characters, select a uniformly distributed random number between 0 and 1, and insert a space equal to 3-character image length * (random value / 6) if the random value is greater than or equal to 0.3. To avoid noise from the background or color of character images, we convert them to binary images (assuming that all input images are already binary).

(2) Apply the method in chapter 4 to the initial training data segmentation information and train the initial recognizer.

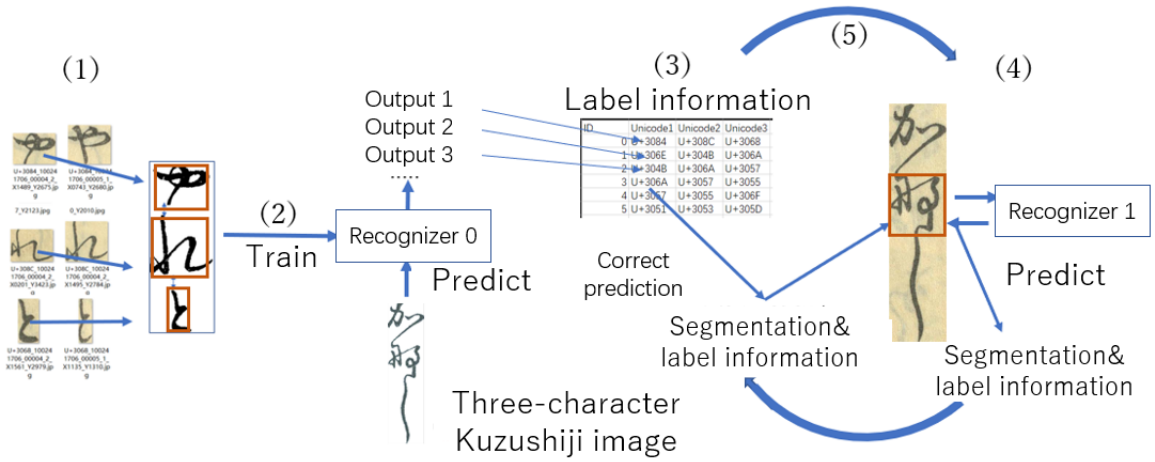


Figure 4.6: The flow chart of steps (1) to (5).

(3) Creation of secondary training data with segmentation information

Apply the initial recognizer to the training data of three-character Kuzushiji images, considering the bounding boxes output by the recognizer as character segmentation information. Create secondary training data segmentation information, using the data where the recognizer correctly predicted labels.

(4) Train the secondary recognizer using chapter 4's method on the secondary training data segmentation information.

(5) Repeat steps (3) and (4) multiple times.

This work achieve a low recognition error rate of 6.33% and get fourth place in the 2019 PRMU (Pattern Recognition and Media Understanding) Japanese historical handwriting recognition algorithm contest.

4.5 Experiments

We conduct experiments to evaluate proposed methods and some past works. The CPU used here is Intel(R) Xeon(R) Gold 2.30GHz with 9 cores and the GPU is NVIDIA Tesla P4000 with 1 node.

4.5.1 Data Sets and Evaluation Critetia

We utilized the dataset constructed by Tang et al. [62]. This dataset is created by merging the CODH datasets and the PRMU contest dataset, comprising 77,000 images of consecutive three Kuzushiji characters (Level 2 dataset of the PRMU contest). Each image in the dataset includes information about the coordinates of the bounding box and the label for each character. The dataset is split into 70,000 training images and 7,000 test images.

To measure accuracy in predicting labels and bounding boxes, we have defined two evaluation criteria as follows. Each bounding box i is specified with four coordinate parameters, the position of the upper left point (X_i, Y_i) and its width W_i and height H_i, respectively. Given the coordinate sequence of predicted bounding boxes $\hat{S} = ((\hat{X}_i, \hat{Y}_i, \hat{W}_i, \hat{H}_i)|i = 1, 2, \ldots, m)$ and ground truth coordinate of bounding boxes $S = ((X_i, Y_i, W_i, H_i)|i = 1, 2, \ldots, m)$ (as shown in Figure 4.7), the consistency rate (CR) of the predicted sequence of boxes is defined as follows:

$$\epsilon\text{-Y-CR} = \frac{\sum_{i=1}^{m} \mathrm{I}(|Y_i - \hat{Y}_i| \leq \epsilon \times \tilde{H})}{m} \times 100\%, \tag{4.1}$$

where $\tilde{H}$ is the height of the input image, and I is the indicator function and outputs 1 if the argument is true and outputs 0, otherwise. The parameter ϵ is for the degree of tolerance. Here, CR only focuses on differences of bounding boxes in the vertical direction, which is sufficient for our purpose.

Note that, when X coordinate is correctly predicted, the X and W coordinates can be

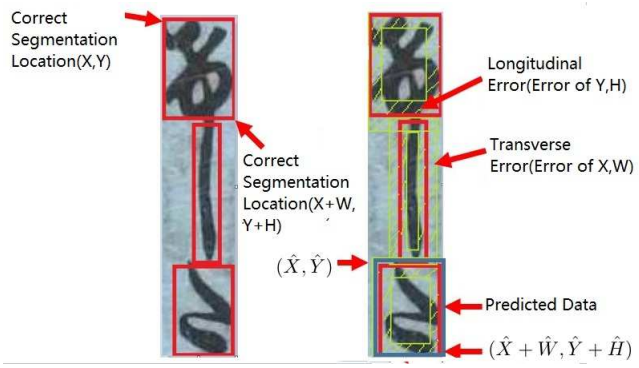


Figure 4.7: Examples of evaluation methods to tolerate coordinate errors, showed an illustration of ground-truth and predicted bounding boxes in the Kuzushiji character image.

calculated easily, similarly, the H coordinate can be determined easily when the Y coordinate has been correctly predicted.

We also use an evaluation criterion for character label recognition. The label error rate(LER) of a pattern classifier h w.r.t. a set Z of labeled instances (x, y) (x corresponds to an image and y corresponds to a sequence of labels) is defined as follows:

$$\mathrm{LER}(h, Z) = \frac{1}{\mathrm{T}} \sum_{(\mathrm{x},\mathrm{y}) \in Z} \mathrm{ED}(h(x), y),$$

where T is the total number of target labels in Z and $\mathrm{ED}(p, q)$ is the edit distance between two sequences p and q.

4.5.2 Results

For the backclone network of YOLOv3, we employ the darknet53 [48] model, which is trained using the training set of 70,000 images. As for the recognizer of FDMBSS, we utilize the RESNET-53 [32] architecture.

As comparators and baselines, we include YOLOv3-darknet53, which utilizes the original aggregation method (NMS) of YOLOv3 for selecting bounding boxes. Additionally, for a naive baseline, we use a segmentation method that segments characters based on histograms along horizontal coordinates, combined with a single-character recognizer implemented with darknet53.

All images are resized to 64*288, and the original image is divided into 2×9 grids to create pre-predicted areas, ensuring that one of the areas must contain an object. The anchor is set as (10,60, 15,20, 20,40, 35,45, 30,60, 40,55, 40,80, 50,90, 60,100), determined based on the shape

of Kuzushiji characters. Table 4.1 is a summary of the results.

Table 4.1: Results of character recognition tasks for three consecutive Kuzushiji characters.

	LER	0.07-Y-CR	0.05-Y-CR
Histogram+RESNET-53[32]	36.6%	63.46%	55.96%
YOLOv3	6.65%	75.8%	66.9%
FGDM-a	5.24%	81%	71.3%
FGDM-b	2.5%	83.18%	73.3%
FDMBSS	13.5%	86.5%	68.6%

Here, FGDM-a refers to the result of FGDM with the same learning rate as YOLOv3, and FGDM-b represents the ones with decreasing learning rates by multiplying 0.1 every 40,000 rounds.

As depicted in Table 4.1, the proposed methods (FGDM-a, b) achieve a lower Label Error Rate (LER), but the prediction accuracy of bounding boxes is still deemed insufficient. On the other hand, our FDMBSS attains the best recognition results at a 0.07-Y-CR for bounding box prediction, although its LER is inferior to the FGDM. For a more detailed evaluation using a 0.05-Y-CR, the FGDM method still appears more promising. However, we believe that the FDMBSS method can serve as a valuable two-stage method for character recognition. It is noteworthy that the improvement of FGDM over YOLOv3 is attributed to the effective choice of the bounding box aggregation method, demonstrating the efficacy of our method.

Finally, we compare our results with the previously reported outcome by Nguyen et al. [46]. Their method achieved a LER of 12.88%. Although there are slight differences in datasets and training/test divisions, our method seems to achieve better accuracy. Since the use of segmentation information appears to enhance recognition accuracy.

4.6 Analysis of the result about different labels of characters

An evaluation metric commonly used in object detection frameworks is mAP (mean Average Precision) [18]. In our proposed method, the mAP value is 68%. This indicates that not only the error rate of recognition but also the prediction of frame coordinates has achieved a high level of accuracy.

As illustrated in Figure 4.8, the character ”ho” in Figure 4.8-A exhibits the best recognition result with a high accuracy rate, despite the curve not being entirely smooth due to uneven sampling. This is attributed to the distinctiveness of the symbols in this character. Conversely,

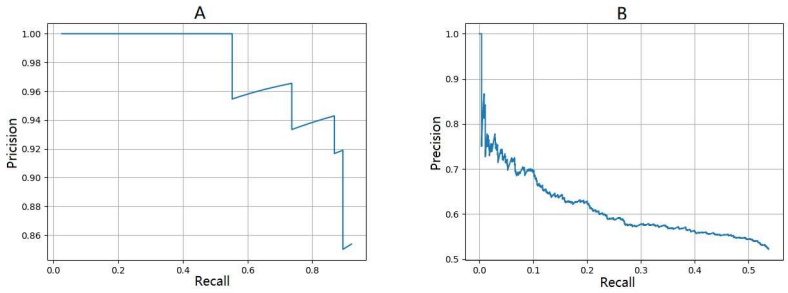


Figure 4.8: The precision rate and recall rate of Kuzushiji character

the recognition of the character ”shi” in Figure 4.8-B proves challenging with an unsatisfactory accuracy rate. The character's shape, resembling a straight line, makes it susceptible to misclassification as an internal part of another character.

Note that pre-processing such as black-and-white binarization of character images to avoid the effect of this background noise will effectively improve the recognition rate and the applicability of the network, and random scrambling of data is also an important step. In addition, we found that in the single character recognition task, the traditional convolutional neural network has certain limitations when processing characters with a certain inclination. We will describe details of this problem in the next chapter.

Figure 4.9: A:The character of Kuzushiji-”ho”.

Figure 4.10: B:The character of Kuzushiji-”shi”.

4.7 Conclusion

In this paper, we propose segmentation and recognition methods for Kuzushiji recognition, and achieving state-of-the-art results on datasets of consecutive three Kuzushiji characters. It is based on the object detection named YOLOv3 [48] under Kuzushiji segmentation character date set[62].

Chapter 5

Rotation-Invariant Convolution Networks with Hexagon-Based Kernels

5.1 Introduction

In this chapter, we focus on the design of the architecture of neural networks to solve the generalization problem in rotation object recognition tasks. The topic about the network that has only learned a single direction object can recognize the same object after the rotation, which is named the rotation invariance problem of the neural network.

Rotational invariance is an essential property in neural networks in computer vision. Convolution Neural Network (CNN) with rotational invariance can recognize the target object from various angles, no matter what he has learned about this object. This property can be well applied to insect recognition and medical image recognition tasks, especially for cell classification. However, it is difficult or impossible to obtain a rotation-invariant property with the classical CNNs. Therefore, we try to enhance the generalization ability of the CNN for the rotating target by doing additional operations of architecture on convolution layers.

In this chapter, we introduce a novel convolutional neural network (CNN) structure featuring hexagonal kernels and a Rotation-channel-merge-block (Ro-block) structure, as illustrated in Figure 5.1. The key innovation lies in achieving rotation invariance, on 60-degree-wise rotations, through the use of hexagonal-based lattices. Unlike the standard square-based lattice, which is invariant to 90-degree-wise rotations, the hexagonal structure is expected to exhibit greater robustness to image rotations. Our Ro-block is designed to preserve and integrate 60-degree-wise rotated feature information, enhancing rotation robustness within complex CNNs. We demonstrate that our structure is indeed rotation-invariant with respect to 60-degree-wise

rotations at the origin. Notably, our proposed structure is provably completely rotation-invariant, (e.g., $f(x) = f(\bar{x})$, where $\bar{x}$ is rotated x.) a distinctive feature not present in previously proposed structures. One notable advantage of our method is that it does not need any data augmentation, eliminating the need for training datasets that include both original and rotated data. This leads to more efficient use of data. Our method can be seamlessly integrated into standard CNNs, trained end-to-end using Stochastic Gradient Descent (SGD) with the same back-propagation and easily implemented using standard libraries.

We validate the effectiveness of our method through experiments involving forward images on various datasets, including a homemade simple pattern image dataset, a rotated version of the MNIST handwritten digits image dataset (MNIST-rot), a synthetic biomarker image dataset, and a microscopic cell image dataset [9]. Particularly, in experiments with synthetic biomarker images and microscopic cell images, we assess the rotation robustness of our trained networks. Additionally, we observe that the well-trained standard CNN exhibits translation invariance, consistent with the known fact that CNNs tend to be robust to translations. Therefore, in conjunction with the rotation invariance around the origin, our trained networks demonstrate robustness to rotations on a global scale. Our method exhibits robust performance on datasets containing both locally and globally rotated objects.

Our main contributions can be summarized as follows: (1) We prove of rotational invariance of HCNN, which is the fully convolutional network with our Ro-block. (2) We improve the rotation robustness of the other complex CNNs with our Ro-block. (3) Our Ro-block can be easily applied to other CNNs.

5.2 Related Work

Traditional convolution kernels in convolution layers are analogous to filters in signal-processing tasks in physics. Several studies, such as [8, 43, 13], have demonstrated that adjusting convolution kernels is a common method to enhance a network's generalization performance for specific tasks. Ecker et al. [20] explored patterns in the primary visual cortex (V1) by making convolution kernels robust to rotation through kernel adjustments and rotations for different angles. However, this approach is primarily focused on local texture information, making it less suitable for general recognition tasks.

Marcos et al.[43] proposed a method to learn explicitly rotation-invariant rotatable filters using standard CNNs. Their approach ties the weights of filter groups in the first layer of a

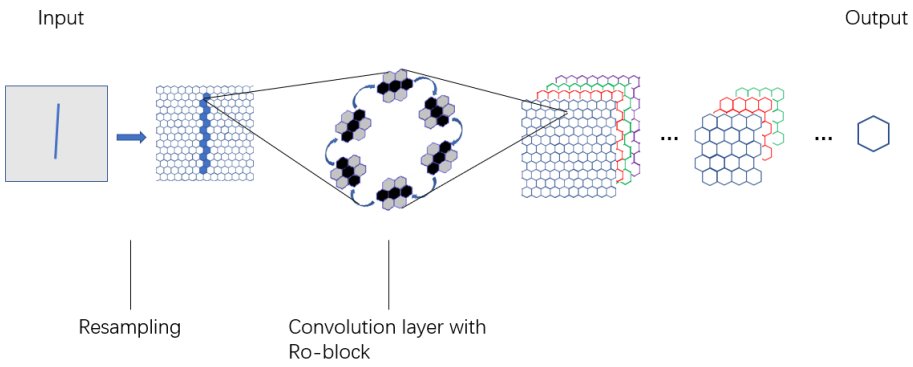


Figure 5.1: Illustration of the Ro-block structure used in convolution network. In this paper, a traditional rectangular input image is first hexagonized by way of resampling. It is then put into the network layer for passing and computation. In order to achieve full rotational invariance, we replace all the convolution layers in the FCN with our Ro-block structure and the size of the data in the last layer is 1x1.

shallow CNN to enhance rotation invariance, but it is limited to texture classification tasks. Cohen et al.[13] introduced Group Equivariant CNN (G-CNN) by partially masking the feature map in the network layer, rotating it, and obtaining multi-angle feature information. However, their method is learning-based and does not achieve complete rotational invariance. In contrast, our Ro-block structure includes a specific pooling layer that actively selects the most suitable kernel for the current feature map from a set containing rotation. Our method is adaptive and can effectively handle 60 degree-wise rotations.

Chidester et al.[8] improved the recognition rate of microscope images using a rotation-robust kernel set, but this is an approximate rotation, and the method has limitations when dealing with small-size matrices like convolution kernels. Marcos et al.[44] proposed a method using larger convolution kernels (up to 9× 9) and a vector field to record rotation angle and position information. However, the method relies on bilinear interpolation for kernel rotation, leading to approximated values. Delchevalerie et al. [19] proposed an involving the utilization of the convolution kernel with an even bigger size (15×15), which draws inspiration from the Bessel function They aim to attain comprehensive rotational invariance under the infinite input space. However, they have no theoretical guarantee about finite input space. It's also worth noting that the efficacy of this method notably depends on parameter selection. In addition, the efficacy of this method depends on parameter selection and faces limitations when the convolution kernel size diminishes due to discretization errors. Bekkers et al.[4] and Weiler et al.[59] also contributed to the rotation invariance task. However, these methods,

including [13, 8, 4, 59, 44], enhance the network's generalization ability to rotate targets without achieving complete rotational invariance. The commonality among these related methods is their use of the standard square lattice. However, the hexagonal grid, which has more rotating symmetrical axes, fits 60 degree-wise rotations. Thus, we anticipate that the hexagonal structure is more robust to input rotations compared to the square one.

The use of hexagonal structured kernels is not a novel concept and has Several applications in computer graphics tasks. The hexagonal grid, with its 6-fold symmetry, offers distinct advantages over the regular square grid, especially in fields like map coordinate positioning [33]. In the context of convolutional neural networks (CNNs), the hexagonal representation can store more data on the same sampling points, reducing calculation time and quantization errors [26, 65]. Zhao et al. demonstrated that CNNs with hexagonal kernels follow the same back-propagation computation process as standard CNNs [65]. Luo et al. and Hoogeboom et al. presented construction methods for hexagonal structures that enable lossless rotation of hexagonal images, proposing the use of hexagonal kernels for Group Equivariant CNNs (G-CNNs)[13][39, 30]. However, these methods did not address the challenge of achieving complete rotation invariance in G-CNN networks. In contrast, our method employs a specific hexagonal kernel structure and, critically, achieves provable complete rotation invariance (e.g., $f(x) = f(\bar{x})$). Schlosser et al. also explored the performance of hexagonal kernels in standard CNNs for classical recognition tasks [51].

Perraudin et al. introduced a CNN variant defined on the sphere with HEALPix sampling, where filters are constrained to be radial to ensure rotational invariance [47]. While this method provides an intriguing solution for achieving rotational invariance in 3-dimensional (3D) networks, it relies on complex 3D sampling techniques for input data, and sampling introduces some dependence on location for the action of a graph filter. Their work addresses rotational invariance in 3D input networks [14, 21], whereas our method focuses on 2D input networks. Steppa et al. proposed a method for processing hexagonally sampled data in a standard CNN, including convolution computation and pooling operations [54].

5.3 Preliminaries

We assume that the input of a neural network (e.g., images) is represented as points in the hexagonal grid. More precisely, each point and its neighbors in the hexagonal grid are represented as a two-dimensional vector in the following hexagonal coordinate system illustrated in Figure 5.2.

Definition 1. *The set of neighbors (including itself) of the origin* $(0,0)$ *is defined as*

$$\mathbf{N} = \{(-1,1),(0,1),(1,0),(1,-1),(0,-1),(-1,0),(0,0)\}.$$

For notational convenience, each neighbor in $\mathbf{N}$ *is specified as* $\mathbf{N}[\mathbf{k}]$ *for* $\mathbf{k} \in \{1,\ldots,7\}$. *Each neighbor of a point* (x,y) *is defined, by using translation, as* $\mathbf{N}[\mathbf{k}] + (x,y)$ *for* $\mathbf{k} \in \{1,\ldots,7\}$.

The definition of neighbors naturally induces the L-1 distance of each point between the origin $(0,0)$. Let $X_R = \{(x,y) \mid (x,y)$ is a point within distance R from the origin $\}$. For example, Fig. 5.2 shows X_3.

To convert a square-based rectangular image to a hexagon, the simplest way is to translate the odd rows of the original rectangular image by half a unit to construct a pseudo-hexagonal structure [3]. In addition, there are some methods of transforming the image through the affine matrix [16], methods of enlarging a single pixel to obtain an approximate hexagonal lattice [15, 29, 24]. In this paper, we use the resampling method [22] to hexagonize the input images in the experiments.

Further, we assume that the CNN is a fully convolutional network, which does not have any extra layer. Each convolution layer has input $I \in \mathbb{R}^{X_k}$ for some k in the hexagonal coordinate system. We denote $I[(x,y)] \in \mathbb{R}$ as the value corresponding to position (x,y), and $I_T \in \mathbb{R}$. For each layer $2 \le t \le T$, and each channel $1 \le c \le C_{t+1}$, we define a kernel as $K_t^c \in \mathbb{R}^{C_t \times |\mathbf{N}|}$. To define hexagonal convolution calculation, we let $K_t^c[(l,\mathbf{k})]$ give the connection between a pixel in channel $l = 1,\ldots,C_t$ of the t-th convolution layer input I_t (written as $I_{t,l}$) and channel $c = 1,\ldots,C_{t+1}$ of the $t+1$-th convolution layer input I_{t+1} (written as $I_{t+1,c}$, also is the output of t-th convolution layer), that is

$$I_{t+1,c}[(i,j)] = \sigma\Big(\sum_{\mathbf{k},1\leqslant l\leqslant C_t} I_{t,l}[(i,j)+\mathbf{N}[\mathbf{k}]] \times K_t^c[(l,\mathbf{k})]\Big). \tag{5.1}$$

where $\mathbf{k}$ is the index defined in Definition 1 and $\sigma()$ is a activation function, e.g., Relu function.

5.4 Proposed Method

During training, a well-trained CNN can obtain the convolution kernels that reflect the features of the input data. However, these kernels only record and learn the features they have learned from input data. Therefore, it leads to the problem that it is hard for these kernels to recognize the same object after rotation.

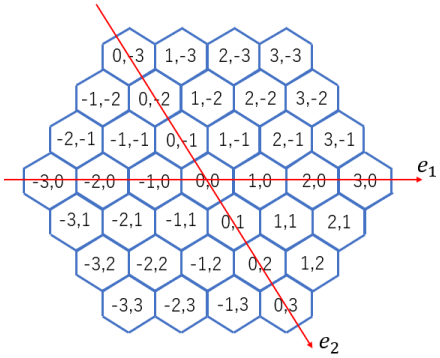


Figure 5.2: Illustration of the Hexagonal coordinate representation. The e_1 and e_2-axis of the hexagonal coordinate system can similarly correspond to the X and Y-axis of the rectangular coordinate system. It should be noted that the e_2 axis is oriented obliquely downward.

To enable the CNN to handle the rotated input, the common is to use data augmentation technologies, such as adding rotating images at the desired angle into the input data as a supplement. This technology allows the network to cope easily with the rotated images with the same object, but the rotation angle must be known in advance. It is undoubtedly undesirable to perform data augmentation by increasing the rotation angle indefinitely.

By adjusting the convolution kernel and expanding their channels, we propose a special structure, named Ro-block, that can learn, choose and merge multi-angle information to enhance the network's generalization ability to handle rotated targets without any data augmentation by simply replacing the standard convolution layer with it.

Our structure is similar to that used in ResNext [61], which extends the original convolution layer into multiple independent convolution parts to widen the network and reduce the parameters involved in the matrix multiplications. Since the hexagonal convolution kernel is good at handling 60 degree-wise rotation, our Ro-block structure can enrich the rotation information of the kernels as much as possible to enhance the network's generalization ability and achieve complete rotation invariance in the fully convolution network with our Ro-block.

5.4.1 Ro-block: multi-angle arrangement method

We consider a multi-channel structure named Ro-block that can simultaneously calculate the kernel for 60 degree-wise angles. As shown in Fig. 5.3, the Ro-block structure contains a set of kernels which is a core part corresponding to the 60 degree-wise rotated hexagonal input. Each kernel within it will perform the convolution calculation with the same input, respectively. Then, the Ro-block performs the Max/Mean operation to compress the calculation result of the most

suitable kernel and to keep the output size of the Ro-block consistent with that of the standard convolution. More precisely, the Ro-block structure is defined as a series of compositions of the convolutions given in equation (5.2)~ (5.6).

To obtain multi-angle rotation information, we generate the rotation kernels by expanding a new channel by rotating the initial convolution kernels (the left part in Fig. 5.3) on $60 * r$ ($r = 0, \ldots, 5$) degrees, respectively, and use them to perform the convolution calculation to obtain feature maps as temporary output. Set $(0, 0)$ be the origin of kernel K_t^c, the rotation of the kernel can be regarded as a movement of the kernel neighbor coordinates.

By Definition 1, for each $r \in \{0, \ldots, 5\}$, we define the r-th rotated kernel of K_t^c as

$$K_t^{c,r}[(l, \mathbf{k})] = \begin{cases} K_t^c[(l, \mathbf{k})], & \text{if } \mathbf{k} = 7 \\ K_t^c[(l, (\mathbf{k} + r) \bmod 6)], & \text{if } \mathbf{k} \neq 7 \end{cases}. \tag{5.2}$$

Let $\hat{I}_{t+1,r,c}$ be a output from convolution calculation of I_t with r times rotated kernel $K_t^{c,r}$. For $r = 0, \ldots, 5$, the convolution calculation for pixel (i, j) on $\hat{I}_{t+1,r,c}$ is written as:

$$\hat{I}_{t+1,r,c}[(i, j)] = \sum_{\mathbf{k},l} I_{t,l}[(i, j) + \mathbf{N}[\mathbf{k}]] \times K_t^{c,r}[(l, \mathbf{k})]. \tag{5.3}$$

Finally, as the output of Ro-block, for $c = 1, \ldots, C_{t+1}$, we use the following 2 ways to merge the feature information of different channels.
Max-type:

$$I_{t+1,c}[(i, j)] = \sigma(\max_{r \in \{0,\ldots,5\}} (\hat{I}_{t+1,r,c}[(i, j)])), \tag{5.4}$$

Mean-type:

$$I_{t+1,c}[(i, j)] = \sigma(\frac{\sum_{r=0,\ldots,5} \hat{I}_{t+1,r,c}[(i, j)]}{6}), \tag{5.5}$$

where σ is a Relu activation function and has $\sigma(x) = \max(0, x)$.

The combination of equation (5.3)~ (5.5) naturally defines the function F_t^c which maps $I_{t,l}$ ($l = 1, \ldots C_t$) to $I_{t+1,c}$ for any channel c. That is,

$$I_{t+1,c} = F_t^c((I_{t,l})_{1 \leq l \leq C_t}), \tag{5.6}$$

As we mentioned before, our Ro-block can be used directly as a replacement for the standard convolutional layer.

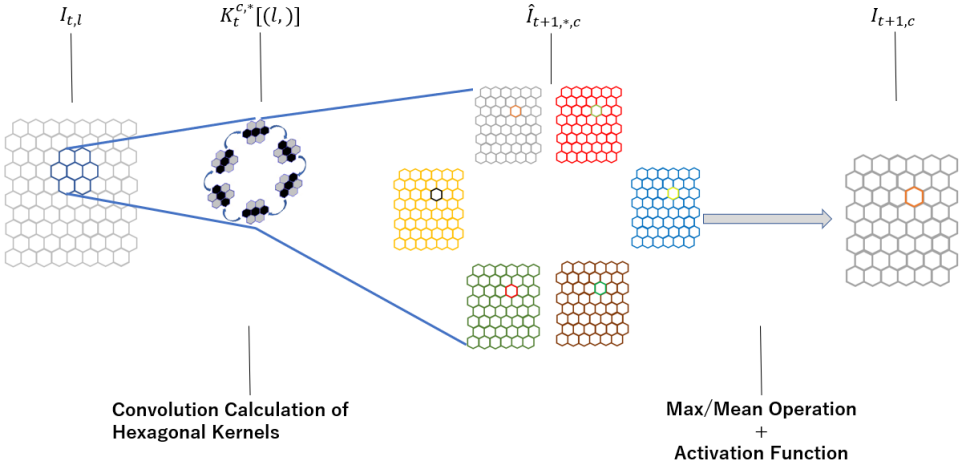


Figure 5.3: Illustration of Hexagonal convolution layer with Ro-block structure. When $K_t^{c,*}[(l,)]$ and $\hat{I}_{t+1,*,c}$ with symbol $*$ means that this is the set of rotations about themselves. On each channel l of the input I_t of the layer, for each set of hexagonal lattices of size $|\mathbf{N}|$, a set of rotated hexagonal convolution kernels are computed with it and the result of output channel c is $\hat{I}_{t+1,*,c}$. Finally, the Max/Mean operation is used to select the most suitable result for the input to pass to the next layer.

Definition 2. *A Hexagonal Convolution Neural Network (HCNN) is a fully convolutional network that has $I_T \in \mathbb{R}$ and each convolution layer is replaced with our Ro-blcok defined in equation* (5.6).

As shown in Figure 5.4, no matter how many times the network accepts a 60 degree-wise rotated input, after the calculation process of the network, these outputs will become I_T, that is, a real number. A real number does not essentially have the concept of rotation, so we have achieved the rotation invariance.

5.4.2 Rotational Invariance of HCNN

We prove the rotational invariance of HCNN by observing the convolution calculation of the hexagonal input and the hexagonal convolution kernel set. All hexagonal operations here are based on the hexagonal coordinate system, as shown in Fig. 5.2. The hexagonal lattice is rotated about origin $(0,0)$ losslessly in 60 degree-wise rotations, which means that not any single pixel value needs to be missing or changed.

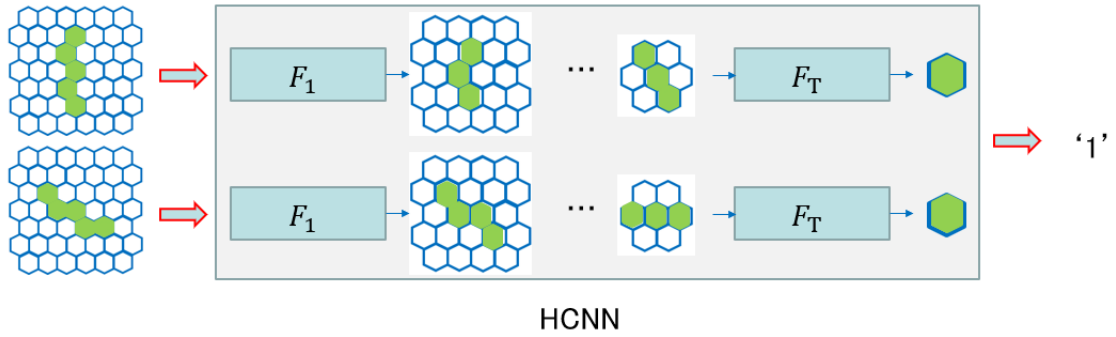


Figure 5.4: A Sample of HCNN

Let $ro((x,y))$ be the function that represents the rotation of point (x,y) by 60 degrees counterclockwise about the origin $(0,0)$ in the hexagonal coordinate system. This function is a bijection. We will abuse the notation for a hexagonal layer I. That is, $ro(I)$ represents a layer where each point is a rotation of some point in I (only the coordinate of the pixel is changed under the following rule.).

Proposition 1. *For every point (x,y), we have $I[(x,y)] = ro(I)[(x+y,-x)]$.*

Proof:
Now, we transform this problem into whether function $ro()$ applies to the rotation affine matrix based on Two-dimensional space. Let $(0,0)$ be the origin, $e_1 = (1,0)$ be the unrotated vector, and $e_2 = (\frac{1}{2}, \frac{\sqrt{3}}{2})$ be the vector after a 60-degree rotation. When (x,y) is the coordinate before rotation and $ro((x,y)) = (x',y')$ is the coordinate after a 60-degree rotation, the rotation matrix is satisfied:

$$x'\begin{bmatrix}1\\0\end{bmatrix} + y'\begin{bmatrix}\frac{1}{2}\\\frac{\sqrt{3}}{2}\end{bmatrix} = \begin{bmatrix}\cos -60 & -\sin -60\\ \sin -60 & \cos -60\end{bmatrix}\left(x\begin{bmatrix}1\\0\end{bmatrix} + y\begin{bmatrix}\frac{1}{2}\\\frac{\sqrt{3}}{2}\end{bmatrix}\right)$$
$$= (x+y)\begin{bmatrix}1\\0\end{bmatrix} - x\begin{bmatrix}\frac{1}{2}\\\frac{\sqrt{3}}{2}\end{bmatrix}$$

■

Then, we can combine Proposition 1 with Definition 1 to obtain Proposition 2.

Proposition 2. $\mathbf{N}[\mathbf{k}-] = (-\mathbf{N}[\mathbf{k}]_2, \mathbf{N}[\mathbf{k}]_1 + \mathbf{N}[\mathbf{k}]_2)$.

To prove the rotational invariance of HCNN, we have Lemma 1.

Lemma 1. *Let I_t be the input of the t-th hexagonal convolution layer of HCNN as defined in equation* (5.6)*. For any $c \leq C_{t+1}$, we have $ro(I_{t+1,c}) = F_t^c(ro(I_{t,l})_{1\leq l\leq C_t})$.*

Proof:
Let $ro(I_{t,c})[(x,y)]$ be the pixel value of the input $ro(I_{t,c})$ at the coordinate (x,y) for channel c at convolution layer t.

By using the proposition 1 of $ro()$, we have

$$ro(I_{t,c})[(x,y)] = I_{t,c}[(-y, x+y)]. \tag{5.7}$$

For the neighbor with coordinate (x,y) in $ro(I)$ and the neighbor with coordinate $(-y, x+y)$ in I, have the following relationship. Using Proposition 2, for each k and $\mathbf{N}[\mathbf{k}]$, the first and second coordinates are denoted as $\mathbf{N}[\mathbf{k}]_1$ and $\mathbf{N}[\mathbf{k}]_2$, respectively.

$$\begin{aligned} ro(I_{t,c})[\mathbf{N}[\mathbf{k}] + (x,y)] &= I_{t,c}[(-y - \mathbf{N}[\mathbf{k}]_2, \\ x + y + \mathbf{N}[\mathbf{k}]_1 + \mathbf{N}[\mathbf{k}]_2)] & \\ &= I_{t,c}[\mathbf{N}[\mathbf{k}-] + (-y, x+y)]. \end{aligned} \tag{5.8}$$

where $\mathbf{k} \in \{1, \ldots, 7\}$.

The results $\hat{I}_{t+1,r,c}$ and $ro(\hat{I}_{t+1,r,c})$ can be obtained by the convolution calculation of I_t and $ro(I_t)$ with the weights $K_t^{c,r}$ in equation (5.3), where $r = 0, \ldots, 5$.

For $r = 0, \ldots, 5$, $c = 1, \ldots, C_{t+1}$, we can rewrite equation (5.3) and have

$$\hat{I}_{t+1,r,c}[(x,y)] = \sum_{l,\mathbf{k}} I_{t,l}[(x,y) + \mathbf{N}[\mathbf{k}]] \times K_t^{c,r}[(l,\mathbf{k})], \tag{5.9}$$

and

$$\begin{aligned} ro(\hat{I}_{t+1,r,c})[(x,y)] = \sum_{l,\mathbf{k}} & ro(I_{t,l})[\mathbf{N}[\mathbf{k}] + (x,y)] \\ & \times K_t^{c,r}[(l,\mathbf{k})] \end{aligned} \tag{5.10}$$

By equation (5.2) and definition of hexagonal kernel set K, for $r = \{0, \ldots, 5\}$ and $K^{c,-1} =$

$K^{c,5}$, we have

$$\begin{aligned}
&\sum_{l,\mathbf{k}} ro(I_{t,l})[\mathbf{N}[\mathbf{k}] + (x,y)] \times K_t^{c,r}[(l,\mathbf{k})] \\
&= \sum_{l,\mathbf{k}} I_{t,l}[\mathbf{N}[\mathbf{k}-] + (-y, x+y)] \times K_t^{c,r-1}[(l,\mathbf{k})] \\
&= \hat{I}_{t+1,r-1,c}[(-y,x+y)].
\end{aligned} \tag{5.11}$$

By using equation (5.4), equation (5.9), equation (5.10) and equation (5.11), we have

$$\begin{aligned}
ro(I_{t+1,c})[(x,y)] &= I_{t+1,c}[(-y,x+y)] \quad \text{By Proprosition 1} \\
&= \sigma(\max_{r\in\{1,\dots,6\}}(\hat{I}_{t+1,r-1,c}[(-y,x+y)])) \quad \text{By equation (5.4)} \\
&= \sigma(\max_{r\in\{0,\dots,5\}}(ro(\hat{I}_{t+1,r,c})[(x,y)])) \quad \text{By equation (5.11)} \\
&= F_t^c(ro(I_{t,l})_{1\le l\le C_t})[(x,y)]
\end{aligned} \tag{5.12}$$

where $c = 1, \dots, C_{t+1}$. ■

Since HCNN is a fully convolutional network with $I_T \in \mathbb{R}$ and $ro(I_T) = I_T$. By combining the observation and Lemma 1, we obtain the following result.

Theorem 2. *HCNN is invariant with 60-degree rotations about the origin. That is, for the function f represented by the network, $f(I) = f(ro(I))$.*

5.5 Experiments

In our experiments, we evaluate the performance of HCNN on various datasets, including a homemade simple pattern image dataset, the rotated MNIST dataset [44], a synthetic biomarker image dataset [8], and a microscope cell image dataset [9]. The goals of these experiments vary, ranging from verifying full rotational invariance in HCNN on the homemade dataset to assessing the performance of our proposed method on common neural networks for biological/cell image recognition tasks.

For the homemade simple pattern image dataset, we aim to confirm the complete rotational invariance of HCNN, ensuring the network's recognition of rotated hexagonal images. On the synthetic biomarker image dataset and the microscope cell image dataset, the focus is on observing the performance of our proposed method used in standard neural networks for biological/cell image recognition tasks. In particular, in the microscope cell image dataset

experiments, we rotate a portion of the test images by 360 degrees and evaluate the network's performance on these rotated images.

Considering the differences in image sizes and the number of classes across datasets, we employ AlexNet [35], ResNet50, and ResNet18 [28] as baseline architectures, respectively. Our proposed method, HCNN, is compared against standard CNNs, Hexagdly-CNNs [54], G-CNN [13], RoteqNet [44], and BesselCNN [19] in recognizing rotated test images. While we emphasize training HCNN without data augmentation, we acknowledge that general models often benefit from such techniques during training. In datasets where objects lack a consistent orientation, such as cells, variations in angles can be considered inherent augmentations. In addition, all input images for HCNN are hexagonalized.

The experiments are conducted using an Intel(R) Xeon(R) Gold 2.30GHz CPU with 9 cores and an NVIDIA Tesla P100 GPU with 1 node. The implementation is based on the Pytorch library [1]. Each dataset's classification task is formulated as a standard loss minimization problem with cross-entropy. We use the standard Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, weight decay of 0.0001, batch size of 32, and a learning rate starting at 0.01 and decaying to 10% at half and three-quarters of the total learning epochs.

5.5.1 Experiment on the Simple Pattern Image Dataset

In our experiment with the homemade simple pattern image dataset, we extend the investigation to include translation invariance in addition to rotational invariance using HCNN.

We set up a 2-classification problem to distinguish positive and negative images. Positive images consist of arrow patterns (Fig.5.5-Left), and negative images include other patterns like diamonds and wavy lines (Fig.5.5-Middle and Right). The positive images are generated by translating the arrow pattern, and the negative images are generated similarly using the other patterns.

The training set comprises 30 positive and 50 negative images. Subsequently, we generate 366 test images by rotating the training images by 60, 120, 180, 240, and 300 degrees around the origin. HCNN, utilizing five hexagonal convolution layers, processes each input image with a size of 11×11.

In Table 5.1, we define two types of test error rates: $terr$ for data without rotated images (akin to training error) and $terr_{ro}$ for the rotated test data. As summarized in Table 5.1, HCNN

[1] https://pytorch.org/

Figure 5.5: Illustration of the simple pattern images.

outperforms standard CNNs and previous methods by a substantial margin in error rate.

We observe that when the network comprises only hexagonal convolution layers with a nonlinear function (e.g., ReLU activation layer), the output for the image before and after rotation remains the same in each layer, indicating rotational invariance. And when a batch-normalization layer [31] is used in the network, the difference in the sum of output values corresponding to each layer is less than $\pm 1e-4 \sim 1e-6$ (for rotated and non-rotated inputs), which is caused by the normalization calculation. Since the difference is minimal and does not cause a change in the classification results, the batch-normalization layer seems to keep the rotational invariance highly approximately. However, when other layers (e.g., pooling layer) are used in the network, the error rate increases (e.g., replacing two convolution layers with a max-pooling layer results in a recognition error rate of approximately 5.48%).

Method	$terr(\%)$	$terr_{ro}(\%)$
AlexNet	0	48.4
G-CNN-p4 [13]	37.5	37.5
Hexagdly-AlexNet [54]	0	33.5
RoteqNet [44]	0	24
Mean-type HCNN	2.5	2.5
Max-type HCNN	1.37	1.37

Table 5.1: The recognition results of Simple Pattern Images.

5.5.2 Experiment on the Rotated MNIST Dataset

In the experiment with the Rotated MNIST dataset (MNIST-rot), we employed a dataset that applies random rotations to each 28×28 pixel image from the original MNIST dataset. The dataset consists of 10,000 training images and 50,000 testing images. The rotations are applied within the range of $[0, 2\pi)$.

We utilized the approach introduced by [44] to create the MNIST-rot dataset. For the training set, consistent data augmentation techniques are applied. In contrast, the test set underwent

rotations in increments of $[0, 60, 120, 180, 240, 300]$ degrees to evaluate the model's rotational invariance.

Method	A(%)	B(%)
AlexNet [35]	3.02	2.88
BesselCNN [19]	5.58	3.31
G-CNN-p4 [13]	2.52	1.83
Hexagdly-AlexNet [54]	2.42	1.45
RoteqNet [44]	1.53	1.49
Mean-type HCNN	2.03	1.46
Max-type HCNN	1.99	1.38

Table 5.2: A is the error rate of the test set in MNIST-rot dataset; B is the error rate of the test set in MNIST-rot dataset after removing "2", "5", "6" and "9" and retraining.

Upon observing Table 5.2-A, our method demonstrated suboptimal results on the test set. We attribute this to the challenges posed by similar features in handwritten digits, especially in a subset comprising "2," "5," "6," and "9," which are difficult to distinguish after rotation. To address this, we excluded these digits from consideration, resulting in a reduced training dataset (6,154 images) and test dataset (30,609 images).

After retraining and testing, as shown in Table 5.2-B, our method exhibited improved performance compared to previous works. Recognizing differently labeled but similar digits remains a challenging task, but, in this sense, can be seen as a unique expression of the rotational invariance of our method.

5.5.3 Experiment on the Synthetic Biomarker Image Dataset [8]

The Synthetic Biomarker Image Dataset aims to emulate real-world microscope images, providing a precise evaluation of the performance of encoding rotation equivariance. This dataset was created by [8], who used Gaussian Mixture Models (GMM) to simulate real-world fluorescence microscopy images of biological signals [64]. The GMM method allows for explicit control over the representation of rotations, as well as intra- and inter-class variations.

In this dataset, 50 classes are defined, with 100 examples per class for training and 200 samples per class for testing. It's important to note that the test dataset contains rotated images, providing a realistic evaluation scenario. We follow the same experiment setting as in [8], and the HCNN in Table 5.3 is obtained by replacing each standard convolution layer in the ResNet50 network with our Ro-block.

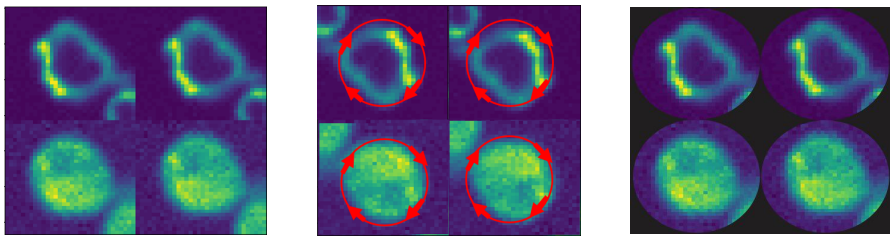
Method	Test error(%)
G-CNN-p4 [13]	93.34
BesselCNN [19]	39.17
Hexagdly-ResNet50 [54]	28.9
ResNet50	16.6
RoteqNet [44]	9.65
CFNET [8]	≈5 (reported in [8])
Max-type HCNN	3.06
Mean-type HCNN	2.44

Table 5.3: The recognition results of Synthetic Biomarker Images.

Given that the synthetic biomarker dataset already incorporates rotations, we do not introduce additional rotational elements or operations. In terms of experimental results, our method demonstrates superior performance compared to CFNET [8] and even outperformed other CNNs in recognizing biomarker images with inherent rotations.

5.5.4 Experiment on the Microscope Cell Images

The microscope cell image dataset is a standard dataset in the biological recognition field, created by [9]. This dataset comprises 21,882 training images, 4,491 verification images, and 4,516 test images. Each image has a size of 64×64 and consists of four sub-images with two channels, captured from eight different directions, as illustrated in Figure 5.6-A. These microscopic cell images, capturing protein cells, often lead to high similarity among images, presenting challenges in recognition.



A B C

Figure 5.6: Illustration of the image with the multi-angle target.

To evaluate global rotational invariance, we generate test images by rotating the four sub-images within each image simultaneously by 0 to 90 degrees, as depicted in Fig.5.6-B. In Fig.5.6-C, we apply a data pre-processing method called Inscribed-Circle-Crop (ICC) to maintain the image's contour consistently, regardless of rotation. Interestingly, this pre-processing method

does not significantly improve other networks and even results in an increased error rate due to the loss of detailed information. However, in the network using Ro-block, we found it to be effective in reducing the network's recognition error rate for rotated images. The HCNN in Table 5.7 is obtained by replacing all standard convolution layers in the ResNet18 network with our Ro-block.

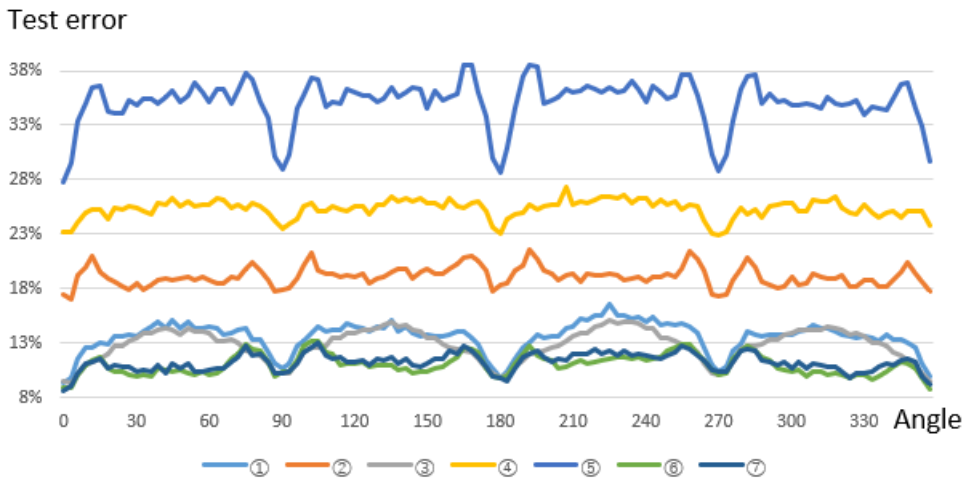


Figure 5.7: ①: ResNet18; ②: G-CNN-p4 [13]; ③: Hexagdly-ResNet18 [54]; ④: Rote-qNet [44]; ⑤: BesselCNN [19]; ⑥: Mean-type HCNN; ⑦: Max-type HCNN

	ASD(%)	SD(%)
①	14.96	1.4
②	19.98	0.87
③	14.28	1.47
④	26.16	0.83
⑤	37.24	2.15
⑥	11.94	0.94
⑦	12.02	0.81

Table 5.4: The measurement of results in the Experiment. We calculate the standard deviation (SD) and average + standard deviation (ASD) for each set of results in the experiment for analysis.

The experiment results are summarized in Figure 5.7, where the X coordinates correspond to the degrees of rotation. Our method (⑥ and ⑦ in Figure 5.7) shows excellent rotation robustness in the recognition of the microscopic cell recognition task. Method ① serves as the baseline, representing a common neural network for cell image recognition. We observe that the recognition rate curve of our network for rotated images changes more smoothly than in the

original ResNet18 network and other past works. Additionally, we achieve a lower test error rate than CFNet [8] and others for test data without rotation (at angle 0).

From Figure 5.7, we observe the following: Method ② has a similar recognition error as the unrotated image when the input image is rotated by 30 and 60 degrees, but it is difficult to compare because its error rate is too high. Method ③ is slightly better than the baseline, but the improvement is not significant, especially when the rotation angle is 30 and 60 degrees. Our methods ⑥ and ⑦ show strong rotation adaptation and achieve a lower error rate than others. In particular, after learning with ICC images, the result of ⑥ in Figure 5.7 indicates that the test error rate of the 60 degree-wise rotated test image is close to the non-rotated test image, suggesting an improvement in robustness to rotation. The training of ⑤ heavily relies on choosing two crucial parameters about the Bessel function within the network. Despite our rigorous efforts to experiment with various combinations of parameters, it is still challenging to achieve better results.

To more clearly measure changes in error rates at different angles, we utilize the standard deviation (SD), as well as the combination of average and standard deviation (ASD) of the test results, as indicators of both network invariance and network performance. Upon reviewing Table 5.4, the SD of ②, ④, ⑥, and ⑦ are all relatively low. Nevertheless, it is worth noting that ② and ④'s performance in terms of ASD is comparatively subpar. This implies that our approach effectively attains rotation robustness while concurrently minimizing the error rate.

5.6 Conclusion

We introduced HCNN, a modified version of classical fully convolutional neural networks (CNNs), designed to significantly enhance the network's expressive capacity without relying on data augmentation. Our Ro-block, incorporated into a fully convolutional network, demonstrates the achievement of complete rotational invariance. This implies that when the input is rotated by 60 degrees, the network's output remains unchanged. For general CNNs, our method effectively enhances rotational robustness. Our experimental results highlight that standard CNNs exhibit certain adaptability to slight angle rotations, with convolution kernels proving robust to 90-degree rotations. In contrast, our method, robust to 60-degree rotations, demonstrates improved generalization to rotational changes in images.

Chapter 6

Boosting-based Construction of BDDs for Linear Threshold Functions and Its Application to Verification of Neural Networks

6.1 Introduction

In this chapter, we focus on understanding the decision process within neural networks. This topic is referred to as explanatory artificial intelligence (XAI), also known as the Interpretability of neural networks.

The interpretability of Neural Networks (NNs) has been a challenging topic due to the intricacy of their behaviors. To enhance interpretability, several promising methods apply verification techniques of Boolean functions to gain insights into NNs. In this method, NNs are represented as equivalent Boolean functions, and various verification techniques are used to check criteria such as robustness [37, 41, 60, 63, 66]. While this approach holds promise by leveraging the extensive results of Boolean function verification, a significant hardness lies in this process of transforming an NN into a representation of an equivalent Boolean function.

The work presented by [52] proposes a method for transforming NNs into Boolean function representations. Their method involves two key steps: (i) converting each neuron, essentially a linear threshold function, into a Binary Decision Diagram (BDD), and (ii) amalgamating these BDDs to form a final Boolean function representation, such as Boolean circuits. The critical point in this process lies in transforming a linear threshold function into a BDD.

To address this, they adopt the transformation method in [7]. This method, rooted in dynamic programming, has a time complexity of $O(n2^{\frac{n}{2}})$ and yields a BDD of size $O(2^{\frac{n}{2}})$, where n represents the number of variables. Notably, the method mandates a predetermined order of n variables as input and produces the minimal BDD consistent with this order. To obtain the minimum BDD entails examining $n!$ potential orderings, it takes $O(n!n2^{\frac{n}{2}})$ time. Even without exhaustively searching through all orderings, selecting a good enough order remains a non-trivial task.

In this thesis, we introduce an alternative method for generating a specific type of BDD representation, termed Aligned Binary Decision Diagram (ABDD), for the linear threshold function. Our approach draws inspiration from the field of *Boosting*, a machine learning framework that enhances the performance of base classifiers by combining them into a more performance classifier. Specifically, our method is a modification of the boosting algorithm proposed by Mansour and McAllester [42]. The algorithm operates by considering a set of labeled instances of a linear threshold function and constructs a BDD in a top-down greedy way, ensuring consistency with the instances. Conceptually, the algorithm amalgamates elements of greedy decision tree learning with a node-merging process, demonstrating its functionality as a combination of these two approaches.

Given a linear threshold function $f(x) = \sigma(w \cdot x + b)$ where σ is the step function, we can apply the algorithm of Mansour and McAllester by feeding all 2^n possible labeled instances of f and obtain a BDD representation of f of the size $O(\frac{n^2}{\rho^4} \ln \frac{1}{\rho})$ in time $O(2^n \mathrm{poly}(1/\rho))$, where ρ is the margin of f, defined as $\rho = \min_{x \in \{-1,1\}^n} |w \cdot x + b|/\|w\|_1$. An advantage of the method is that the resulting BDD is small if the linear threshold function has a large margin. Another notable advantage of our method is its independence from a predetermined variable ordering as input. During our preliminary investigations, we observed that the algorithm is not efficient enough in practice. To address this, we obtain our boosting algorithm by modifying Mansour and McAllester's method, restricting the use of only one variable (base classifier) in each layer. Remarkably, despite this seemingly simple alteration, our modified algorithm retains the same theoretical guarantees as Mansour and McAllester's. Moreover, our modification enhances the efficiency of the merging process, resulting in significantly smaller BDDs in practice. While our modification may appear straightforward, its non-trivial nature is underscored in a theoretical sense. To maintain the same theoretical guarantee, we introduce a novel information-theoretic criterion for selecting variables, distinct from previous methodologies. This novel criterion represents one of our key technical contributions.

Table 6.1: Time and size for several methods to convert to DDs from a given linear threshold function (LTF, for short) of margin ρ. The fourth result is only for LTFs with integer weights whose $L2$-norm is W.

Method	DD type	Size	Time
[7]	OBDD	$O(2^{\frac{n}{2}})$	$O(n!n2^{\frac{n}{2}})$
[42]	BDD	$O(\frac{n^2}{\rho^4}\ln\frac{1}{\rho})$	$O(2^n\text{poly}(1/\rho))$
Ours	ABDD	$O(\frac{n^2}{\rho^4}\ln\frac{1}{\rho})$	$O(2^n\text{poly}(1/\rho))$
(cf. [52])	OBDD	$O(nW)$	$O(nW)$

In our experiments focusing on the verification tasks of Convolutional Neural Networks (CNNs), by following the same procedures as [52], we generate smaller BDDs and consequent Boolean representations of CNNs are more efficient than in past works. This advancement contributes significantly to the efficiency of the verification process, showing the practical benefits of our proposed method in the CNN verification task.

Related work The method in [45] introduces a precise Boolean encoding for Binary Neural Networks (BNNs), used in network verification tasks. However, their method is limited to small-sized networks. In [53], an Angluin-style learning algorithm is employed to convert BNNs, where both weights and inputs are binarized as $-1, 1$, and Ordered Binary Decision Diagrams (OBDDs) into Conjunctive Normal Form (CNF). Then, the Boolean Satisfiability (SAT) solver is then utilized to verify the equivalence of the resulting CNFs. This approach, however, involves multiple modifications to the OBDD and is constrained by the use of limited binary network weights. [7] focused on converting a linear threshold classifier with real-valued weights into an OBDD directly. Despite its effectiveness, this method exhibits a time complexity of $O(n!n2^{\frac{n}{2}})$ and an OBDD size complexity of $O(2^{\frac{n}{2}})$ when exploring the entire ordering, leading to exponential growth as n increases. Notably, [45, 53, 7] are only suitable for handling small-dimensional neural network weights. For cases with large Boolean expressions, they represent them as Sentential Decision Diagrams (SDDs), incurring enormous time complexity. Furthermore, [10] proposed a rule extraction method inspired by the 'rule extraction as learning' approach to express NNs as Reduced Ordered Decision Diagrams (RODDs). However, this method has a time complexity of $O(n2^{2n})$.

6.2 Preliminaries

6.2.1 Binary Neural Network

A binary neural network (BNN) is a variant of the standard NN with binary inputs and outputs [6]. In this paper, each neural unit, with a step activation function σ, is formulated as follows:

$$\sigma(\sum_i x_i w_i + b) = \begin{cases} 1, & \sum_i x_i w_i + b \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{6.1}$$

where $x \in \{-1, 1\}^n$, $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are the input, the weight vector and the bias of this neural unit, respectively.

6.2.2 Definition of BDD, OBDD and ABDD

A binary decision diagram (BDD) T is defined as a tuple $T = (V, E, l)$ with the following properties: (1) (V, E) is a directed acyclic graph with a root and two leaves, where V is the set of nodes, E is the set of edges such that $E = E_- \cup E_+$, $E_- \cap E_+ = \varnothing$. Elements of E_+ and E_-'s are called +-edges, and $-$-edges, respectively. Let $L = \{0\text{-leaf}, 1\text{-leaf}\} \subset V$ be the set of leaves. For each $v \in V$, there are two child nodes $v^-, v^+ \in V$ such that $(v, v^-) \in E_-$ and $(v, v^+) \in E_+$. (2) l is a function from $V \setminus L$ to $[n]$.

Given an instance $x \in \{-1, 1\}^n$ and a BDD T, we define the corresponding path $P(x) = (v_0, v_1, \ldots, v_{k-1}, v_k) \in V^*$ over T from the root to a leaf as follows: (1) v_0 is the root. (2) for any $j = 0, \ldots k-1$, we have $(v_j, v_{j+1}) \in E_+ \Leftrightarrow x_{l(v_j)} = 1$ and $(v_j, v_{j+1}) \in E_- \Leftrightarrow x_{l(v_j)} = -1$. (3) v_k is a leaf node. We say that an instance $x \in \{-1, 1\}^n$ reaches node u in T, if $P(x)$ contains u. Then, a BDD T naturally defines the following function $h_T : \{-1, 1\}^n \to \{-1, 1\}$ such that

$$h_T(x) \triangleq \begin{cases} -1, & x \text{ reaches 0-leaf} \\ 1, & x \text{ reaches 1-leaf.} \end{cases} \tag{6.2}$$

Given a BDD $T = (V, E, l)$, we define the depth of a node $u \in V$ as the length of the longest path from the root to u. An ordered BDD (OBDD) $T = (V, E, l)$ is a BDD satisfying an additional property: There is a strict total order $<_{[n]}$ on $[n]$ such that for any path $P = (v_0, \ldots, v_k)$ on from the root to a leaf, and any nodes v_i and v_j $(i < j < k)$, $l(v_i) <_{[n]} l(v_j)$. An Aligned BDD (ABDD) $T = (V, E, l)$ is defined as a BDD satisfying that for any nodes $u, v \in V \setminus L$ with the same depth, $l(u) = l(v)$. We employ $v_{i,j}$ to appear the positional information of a node in the

BDD, where j represents the depth of the node, and i represents the position of the node at depth j.

We show the details of BDD, OBDD and ABDD in Figure 6.1.

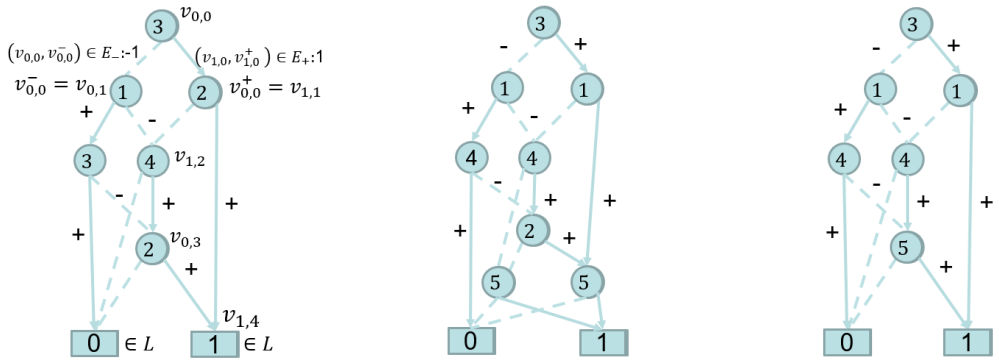


Figure 6.1: Examples of BDD (A), OBDD (B) and ABDD (C). To express the same linear threshold function in BDD form: nodes at the same depth can be labeled by different variables, which means that the number of variables does not limit the depth of BDD; in OBDD form: the nodes at the same depth are all labeled by the same variables, which results in the depth of OBDD is fixed as the number of variables; in ABDD form: nodes at the same depth are labeled by the same variables, and the depth of ABDD is solely determined by the entropy reduction to 0 in our algorithm.

6.2.3 Instance-based Robustness (IR), Model-based Robustness (MR) and Sample-based Robustness (SR)

Robustness is a fundamental property to measure the neural network, which represents the tolerance of the network to noise or white attacks. For binary input images, the robustness k represents that as long as at least k pixels are flipped from 0 to 1 or 1 to 0, and the neural network's output will be changed.

We define the IR and MR of a network as follows.

Definition 3. *(Instance-based Robustness) [52]*
Consider a classification function $f : \{-1,1\}^n \rightarrow \{-1,1\}$ *and a given instance* x. *The robustness of the classification of* x *by* f, *denoted by* $r_f(x)$. *If* f *is not a trivial function (always* $True$ *or* $False$*),*

$$r_f(x) = \min_{x': f(x) \neq f(x')} dis(x, x') \tag{6.3}$$

where $dis(x, x')$ denotes the Hamming distance between x and x'.

Definition 4. *(Model-based Robustness) [52]*
Consider a classification function $f : \{-1, 1\}^n \to \{-1, 1\}$. The Model-based Robustness of f is defined as:

$$MR(f) = \frac{1}{2^n} \sum_x r_f(x). \tag{6.4}$$

However, we contend that computing MR on full-size data lacks practical significance. In real-world applications, robustness validation using sample data is a common practice. Here, we treat the samples in the dataset as instances randomly drawn from the full-size data under the uniform distribution. With this perspective, we introduce the following definition.

Definition 5. *(Sample-based Robustness, SR)*
Consider a classification function $f : \{-1, 1\}^n \to \{-1, 1\}$. Given a sample S under uniform distribution from $\{-1, 1\}^n$. The Sample-based Robustness of f is defined as:

$$SR(f) = \frac{1}{|S|} \sum_{x \in S} r_f(x). \tag{6.5}$$

6.2.4 Overview of our method

In Section 6.3, we present an algorithm designed to construct an ABDD with a small training error concerning a specified sample of a target Boolean function. Our algorithm is grounded in boosting, a powerful machine-learning approach that creates a more accurate classifier by amalgamating "slightly accurate" classifiers.

In Section 6.4, we apply our boosting algorithm to identify an equivalent ABDD associated with a given linear threshold function. Making a natural assumption that the linear threshold function possesses a large "margin," we demonstrate that the size of the resulting ABDD is small.

In Section 6.5, we detail the process of converting a given BNN into an equivalent Boolean expression suitable for verification tasks. Specifically, (i) each neural unit undergoes conversion into an equivalent ABDD by running our boosting algorithm, (ii) each ABDD is further transformed into a Boolean circuit, and (iii) all circuits are integrated into the final circuit, equivalent to the provided BNN. Additionally, for a specific verification task, we convert the final circuit into an equivalent SDD.

6.3 Boosting

6.3.1 Outline of Boosting

Boosting is an approach to constructing a strongly accurate classifier by combining weakly accurate classifiers. The process of boosting like shown in Figure 6.2, the boosting algorithm here is seen as a Booster. When we run the algorithm at t interation ($t = 1, \ldots, T$), can get a distribution d_t over samples $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\{-1, 1\}^n \times \{-1, 1\})^m$ of m instances. This distribution is used to make a weak learner and return h_t, which is the weakly accurate classifier. Each weakly accurate classifier from hypotheses set $\mathcal{H}$ which is a search space. The final hypothesis $F_T : X \to \{\pm 1\}$ is constructed by combining $h_1, \ldots, h_T$, and has the error rate $\hat{err}(F_T) = \frac{1}{m} \sum_{i=1}^{m} I[F_T(x_i) \neq y_i] \to 0$.

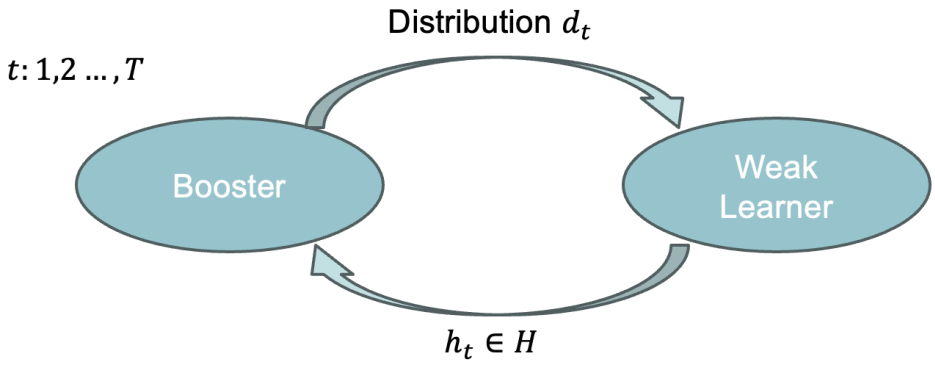


Figure 6.2: Outline of Boosting

Mansour and McAllester use the boosting-based method to construct a BDD [42]. Our proposed method can be seen as a revised version of theirs. Both two methods have the same theoretical guarantee for size and construction time, but our revised method is practically outstanding.

6.3.2 Problem Setting

We assume some unknown target function $f : \{-1, 1\}^n \to \{-1, 1\}$. When a sample $S = ((x_1, f(x_1)), \ldots, (x_m, f(x_m))) \in (\{-1, 1\}^n \times \{-1, 1\})^m$ of m instances labeled by f and a precision parameter ε is given, we want to find a classifier $g : \{-1, 1\}^n \to \{-1, 1\}$ such that its training error $\Pr_U\{g(x) \neq f(x)\} \leq \varepsilon$, where U is the uniform distribution over S. We are also given a set $\mathcal{H}$ of base classifiers from $\{-1, 1\}^n$ to $\{-1, 1\}$. We assume the following assumption

which is standard in the boosting literature [42].

Definition 6. *(Weak Hypotheses Assumption (WHA))*
A hypothesis set $\mathcal{H}$ satisfies γ-Weak Hypothesis Assumption (WHA) for the target function $f : \{-1,1\}^n \to \{-1,1\}$ if for any distribution d over $\{-1,1\}^n$, there exists $h \in \mathcal{H}$ such that $edge_{d,f}(h) \triangleq \sum_{x\in\{-1,1\}^n} d_x f(x)h(x) \geq \gamma$.

Intuitively, WHA ensures the set of $\mathcal{H}$ of hypotheses and f are "weakly" related to each other. The edge function $edge_{d,f}(h)$ takes values in $[-1,1]$ and equals to 1 if $f = h$. Under WHA, we combine hypotheses of $\mathcal{H}$ into a final hypothesis h_T represented by an ABDD T.

Our analysis is based on a generalized version of entropy [34]. A pseudo-entropy $G : [0,1] \to [0,1]$ is defined as $G(q) \triangleq 2\sqrt{q(1-q)}$. Like the Shannon entropy, G is concave and $G(1/2) = 1$, and $G(0) = G(1) = 0$. In particular, $\min(q, 1-q) \leq G(q)$. Then we will introduce the conditional entropy of f given a sample S and an ABDD T. For each node u in T, let $p_u \triangleq \Pr_U\{x \text{ reaches } u\}$ and $q_u \triangleq \Pr_U\{f(x) = 1 \mid x \text{ reaches } u\}$, respectively. Let $N(T)$ be the set of nodes in $V \setminus L$ whose depth is the maximum. We further assume that for each node u in $N(T)$, the x in u is assigned to 1-leaf if $q_u \geq 1/2$, and is assigned to 0-leaf, otherwise. Then, observe that $\Pr_U\{f(x) \neq h_T(x)\} = \sum_{u\in N(T)} p_u \min(q_u, 1-q_u)$. The conditional entropy of f given an ABDD T with respect to the distribution U is defined as

$$H_U(f|T) = \sum_{u\in N(T)} p_u G(q_u). \tag{6.6}$$

Then the conditional entropy gives an upper bound of the training error as follows.

Proposition 1. $\Pr_U\{h_T(x) \neq f(x)\} \leqslant H_U(f|T)$.

Proof. By using the property that $\min(q, 1-q) \leq G(q)$, $\Pr_U\{h_T(x) \neq f(x)\} = \sum_{u\in N(T)} p_u \min(q_u, 1-q_u) \leqslant \sum_{u\in N(T)} p_u G(q_u)$. □

Therefore, it is sufficient to find an ABDD T whose conditional entropy $H_U(f|T)$ is less than ε.

We will further use the following notations and definitions. When an ABDD T, S and $u \in N(T)$ is given, let $S_u = \{(x,y) \in S \mid x \text{ reaches } u\}$. The entropy $H_d(f)$ of $f : \{-1,1\}^n \to \{-1,1\}$ with respect to a distribution d over $\{-1,1\}^n$ is defined as $H_d(f) \triangleq G(q)$, where $q = \Pr_d\{f(x) = 1\}$. The conditional entropy $H_d(f|h)$ of f given $h : \{-1,1\}^n \to \{-1,1\}$ with respect to d is defined as $H_d(f|h) = \Pr_d\{h(x) = 1\}G(q^+) + \Pr_d\{h(x) = -1\}G(q^-)$, where $q^\pm = \Pr_d\{f(x) = 1 \mid h(x) = \pm 1\}$, repectively.

6.3.3 Our Boosting Algorithm

Our algorithm is a modification of the boosting algorithm proposed by Mansour and McAllester [42]. Both algorithms learn Boolean functions in the form of BDDs in a top-down manner. The difference between our algorithm and Mansour and McAllester's algorithm lies in the construction of the final Boolean function, where ours utilizes ABDDs, while Mansour and McAllester's algorithm does not. Although this change may appear subtle, it necessitates a new criterion for selecting hypotheses in $\mathcal{H}$ and demonstrates improved results in our experiments.

Our boosting algorithm iteratively grows an ABDD by adding a new layer at the bottom. More precisely, at each iteration k, when the current ABDD T_k is given, the algorithm performs the following two consecutive processes (as illustrated in Figure 6.3).

Split: It chooses a hypothesis $h_k \in \mathcal{H}$ using some criterion and adds two child nodes for each node in $N(T_k)$ in the next layer, where each child corresponds to ± 1 values of h_k. Let T'_k be the resulting DD.

Merge: It merges nodes in $N(T'_k)$ according to some rule and let T_{k+1} be the ABDD after the merge process.

The full description of the algorithm is given in Algorithm 1 and 2, respectively. For the split process, it chooses the hypothesis h_k maximizing the edge $edge_{\hat{d},f}(h)$ with respect to the distribution $\hat{d}$ specified in (6.17). For the merge process, we use the same way in the algorithm of Mansour and McAllester [42].

Definition 7. *[42] For δ and λ ($0 < \delta, \lambda < 1$), a (δ, λ)-net $\mathcal{I}$ is defined as a set of intervals $[v_0, v_1], [v_1, v_2], \ldots, [v_{w-1}, v_w]$ such that (i) $v_0 = 0$, $v_w = 1$, (ii) for any $I_k = [v_{k-1}, v_k]$ and $q \in I_k$, $\max_{q' \in I_k} G(q') \leq \max\{\delta, (1+\lambda)G(q)\}$.*

Mansour and McAllester showed a simple construction of (δ, λ)-net with length $w = O((1/\lambda)\ln(1/\delta))$ [42] and we omit the details. Our algorithm uses particular (δ, λ)-nets for merging nodes.

6.3.4 Analyses

The conditional entropy of a function $f : \{-1, 1\}^n \to \{-1, 1\}$ given an ABDD T and a hypothesis $h : \{-1, 1\}^n \to \{-1, 1\}$ with respect to distribution d over $\{-1, 1\}^n$ is defined as follows:

$$H_d(Y|T,h) = \sum_{u \in N(T)} (p_{u^+} G(q_{u^+}) + p_{u^-} G(q_{u^-})) \tag{6.7}$$

where $p_{u^+} = Pr_d\{x \text{ reaches } u | h(x) = 1\}$ and $q_{u^+} = Pr_d(f(x) = 1 | x \text{ reaches } u, h(x) = 1)$. p_{u^-} and q_{u^-} are defined similarly.

The following lemmas prove an effective weak learning algorithm under a variant balanced distribution $\bar{d}$, and it also can be achieved under any distribution. For $(x, y) \in S$:

$$\bar{d}_{(x,y)} = \frac{d_{(x,y)}}{2 \sum_{(x',y) \in S} d_{(x',y)}} \tag{6.8}$$

Lemma 3. *[56] Let d and $\bar{d}$ be any distribution over $\{-1,1\}^n$ and its balanced distribution with respect to some $f : \{-1,1\}^n \to \{-1,1\}$, respectively. Then, for any hypothesis with $edge_{\bar{d},f}(h) \geq \gamma$, $H_d(f|h) \leqslant (1 - \gamma^2/2) H_d(f)$.*

Based on Lemma 3, we establish the connection between γ and entropy function, and have a $\gamma^* \in (0,1)$ at each depth to reflect the entropy change under our algorithm.

Lemma 4. *Let $\hat{d}$ be the distribution over S specified in (6.17) when T_k, $\mathcal{H}$ and S is given by Algorithm 2 and let h_k be the output. If $\mathcal{H}$ satisfies γ-WHA, then the conditional entropy of f with respect to the distribution U over S given T_k and h_k is bounded as $H_U(f|h_k, T_k) \leqslant (1 - \gamma^2/2) H_U(f|T_k)$.*

Proof. For $(x, y) \in S_u$, we define the balanced distribution of node u as follows:

$$d^u_{(x,y)} = \begin{cases} \frac{1}{2 \times |\{(x',y') \in S_u | y=1\}|}, & \text{if } y = 1 \ \& \ (x,y) \in S_u \\ \frac{1}{2 \times |\{(x',y') \in S_u | y=-1\}|}, & \text{if } y = -1 \ \& \ (x,y) \in S_u \\ 0, & \text{if } (x,y) \notin S_u \end{cases} \tag{6.9}$$

Under the balanced distribution d^u, let

$$\gamma_{u,h_k} = edge_{d^u,f}(h_k) \tag{6.10}$$

be the edge of h_k with respect to d^u and f. By Lemma 3, we have

$$H_{d^u}(f|h) \leqslant (1 - \gamma^2_{u,h}/2) H_{d^u}(f). \tag{6.11}$$

$$H_{d^u}(Y|h) = H_d(Y|h, \ell_k(x) = u) \leqslant (1 - \frac{\gamma_{u,h}^2}{2}) H_{d^u}(Y) \tag{6.12}$$

Thus,

$$\begin{aligned} H_U(f|h_k, T_k) &= \sum_{u \in N(T_k')} p_u G(q_u) \\ &= \sum_{u \in N(T_k)} H_{d^u}(f|h_k) \\ &\leq \sum_{u \in N(T_k)} p_u (1 - \gamma_{u,h_k}^2/2) H_{d^u}(f) \\ &= \sum_{u \in N(T_k)} p_u H_{d^u}(f) \\ &\quad - \sum_{u \in N(T_k)} p_u \frac{\gamma_{u,h}^2}{2} H_{d^u}(f) \\ &= H_U(f|T_k) - \sum_{u \in N(T_k)} p_u' \frac{\gamma_{u,h_k}^2}{2} H_U(f|T_k), \end{aligned} \tag{6.13}$$

where $p_u' = p_u \frac{H_{d_u}(f)}{H_U(f|T_k)} = p_u \frac{G(q_u)}{\sum_{u \in N(T_k)} p_u G(q_u)}$ which is specified in line 3 of Algorithm 2. Here, we define a distribution $\hat{d}$ for samples in node u.

$$\hat{d}_{(x,y)} = \sum_{u \in N(T_k)} p_u' d^u_{(x,y)} \tag{6.14}$$

By using (6.13), (6.14), γ-WHA, and Jensen's inequality, we get

$$\begin{aligned} \sum_{u \in N(T_k)} p_u' \gamma_{u,h_k}^2 &= \sum_{u \in N(T_k)} p_u' \Big(\sum_{(x,y) \in S_u} d^u_{(x,y)} y h_k(x) \Big)^2 \\ &\geqslant \Big(\sum_{u \in N(T_k)} p_u' \sum_{(x,y) \in S_u} d^u_{(x,y)} y h_k(x) \Big)^2 \\ &= \Big(\sum_{u \in N(T_k)} \sum_{(x,y) \in S_u} p_u' d^u_{(x,y)} y h_k(x) \Big)^2 \\ &= \Big(\sum_{(x,y) \in S} \hat{d}_{(x,y)} y h_k(x) \Big)^2 \\ &= (edge_{\hat{d},f}(h_k))^2 \\ &\geqslant \gamma^2. \end{aligned} \tag{6.15}$$

By combining (6.13) and (6.14), (6.15) can be rewritten as

$$H_U(f|h_k, \ell_k) \leqslant (1-\gamma^2) H_U(f|T_k) \tag{6.16}$$

as desired. □

Lemma 5. *([42]) Assume that, before the merge process in Algorithm 1, $H_U(f|h_k, T_k) \leq (1-\lambda)H_U(f|T_k)$ for some λ ($0 < \lambda < 1$). Then, by merging based on the (δ, η)-net with $\delta = (\lambda/6)H_U(f|T_k)$ and $\eta = \lambda/3$, the conditional entropy of f with respect to the distribution U over S given T_{k+1} is bounded as $H_U(f|T_{k+1}) \leqslant (1-\lambda/2)H_U(f|T_k)$, where the width of T_{k+1} is $O((1/\lambda)(\ln(1/\lambda) + \ln(1/\varepsilon)))$, provided that $H_U(f|T_k) > \varepsilon$.*

Algorithm 1 ABDD Boosting

Input: a sample $S \in (\{-1,1\}^n \times (-1,1))^m$ of m instances by f, and a set $\mathcal{H}$ of hypotheses, and precision parameter ε ($0 < \varepsilon < 1$);
Output: ABDD T;

1: initialization: T_1 is the ABDD with a root and 0, 1-leaves, $k = 1$.
2: **repeat**
3: (**Split**) Let $h_k = Split(T, \mathcal{H}, S)$ and add child nodes with each node in $N(T_k)$. Let T'_k be the resulting ABDD.
4: **for** $u \in N(T'_k)$ **do**
5: merge u to the 0-leaf (the 1 leaf) if $q_u = 0$ ($q_u = 1$, resp.).
6: **end for**
7: (**Merge**) Construct a $(\hat{\delta}, \hat{\lambda}/3)$-net $\mathcal{I}_k$ with
8: $\hat{\lambda} = 1 - \frac{H_U(f|T_k, h_k)}{H_U(f|T_k)}$, and $\hat{\delta} = \frac{\hat{\lambda} H_U(f|T_k)}{6}$.
9: **for** $I \in \mathcal{I}_k$ **do**
10: merge all nodes $u \in N(T'_k)$ such that $q_u \in I$.
11: **end for**
12: Let T_{k+1} be the resulting ABDD and update $k \leftarrow k+1$.
13: **until** $H_U(f|T_k) < \varepsilon$
14: Output $T = T_k$.

Now we are ready to show our main theorem.

Theorem 6. *Given a sample S of m instances labeled by f, and a set $\mathcal{H}$ of hypotheses satisfying γ-WHA, Algorithm 1 outputs an ABDD T such that $\Pr_U\{h_T(x) \neq f(x)\} \leq \varepsilon$. The size of T is $O((\ln(1/\varepsilon)/\gamma^4)(\ln(1/\varepsilon) + \ln(1/\gamma)))$ and the running time of the algorithm is poly$(1/\gamma, n)m$.*

Proof. By definition of $\hat{\lambda}$ in Algorithm 1, we have $H_U(f|T_k, h_k) = (1-\hat{\lambda})H_U(f|T_k)$ at each iteration k. Therefore, by Lemma 5, $H_U(f|T_{k+1}) \leq (1-\hat{\lambda}/2)H_U(f|T_k)$. Furthermore, $\hat{\lambda}$ is

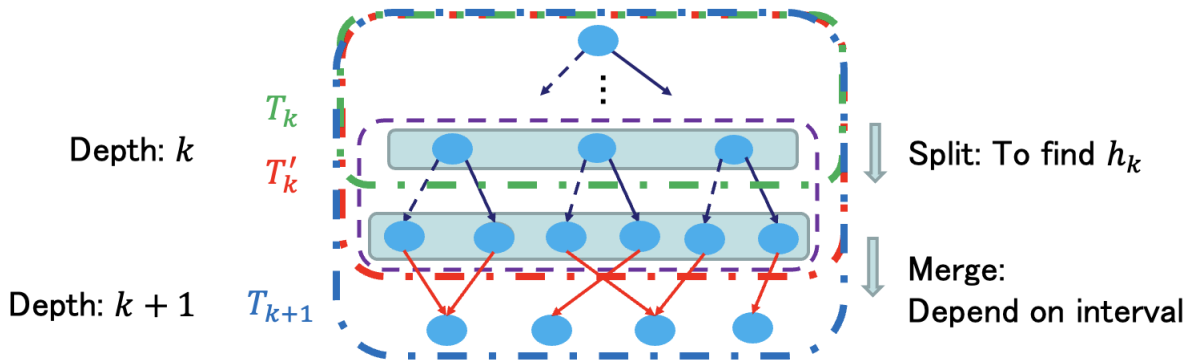


Figure 6.3: Illustration of our boosting algorithm. The purple dotted line part represents the process of merging the split temporary nodes into new nodes by searching the equivalence space that does not appear in the ABDD. The green dotted line part represents the DD with the depth k; the red dotted line part is the resulting DD after the Split; the blue dotted line part is the resulting DD after the Merge. Following algorithm 1, we find some hypothesis h to construct child nodes in the new layer and then merge nodes afterward.

always larger than $\gamma^2/2$, implying that $H_U(f|T_{k+1}) \leq (1-\gamma^2/4)H_U(f|T_k)$. Then, after K iterations, we have

$$H_U(f|T_K) \leq \left(1-\frac{\gamma^2}{4}\right)^K \leq e^{-\frac{K\gamma^2}{4}},$$

as $H_U(f) \leq 1$. Thus, $K = O((1/\gamma^2)\ln(1/\varepsilon))$ iterations suffices to ensure that $\Pr_U\{h_T(x) \neq f(x)\} \leq \varepsilon$. Since the width of the ABDD T is $O((1/\gamma^2)(\ln(1/\gamma)+\ln(1/\varepsilon)))$ by Lemma 5 and its depth is K, the size is bounded by $O((\ln(1/\varepsilon)/\gamma^4)(\ln(1/\varepsilon)+\ln(1/\gamma)))$, as claimed. □

6.4 ABDD Construction

We now apply the ABDD Boosting algorithm developed in the previous section to a given linear threshold function f to obtain an ABDD representation T for f. In particular, we show that the size of T is small when f has a large margin.

To be more specific, assume that we are given a linear threshold function $f : \{-1,1\}^n \to \{-1,1\}$ of the form

$$f(x) = \sigma(w \cdot x + b)$$

for some weight vector $w \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$, where σ is the step function, i.e., $\sigma(z)$ is 1 if $z \geq 0$ and -1 otherwise. Note that there are infinitely many (w,b) inducing the same

Algorithm 2 *Split*

Input: ABDD T, a set $\mathcal{H}$ of hypotheses, a set S of m instances labeled by f;
Output: $h \in \mathcal{H}$;

1: **for** $u \in N(T)$ **do**
2: Let

$$d^u_{(x,y)} = \begin{cases} \frac{1}{2\times|\{(x',y')\in S_u | y=1\}|}, & \text{if } y = 1 \ \&\ (x,y) \in S_u \\ \frac{1}{2\times|\{(x',y')\in S_u | y=-1\}|}, & \text{if } y = -1 \ \&\ (x,y) \in S_u \\ 0, & \text{if } (x,y) \notin S_u \end{cases}, \text{ for } (x,y) \in S.$$

3: $p'_u = \frac{p_u G(q_u)}{\sum_{u\in \mathbf{L}_k} p_u G(q_u)}$
4: Let

$$\hat{d}_{(x,y)} = \sum_{u\in N(T)} p'_u d^u_{(x,y)} \tag{6.17}$$

for $(x, y) \in S$.
5: **end for**
6: Output $h = \arg\max_{h'\in\mathcal{H}} edge_{\hat{d},f}(h)$.

function f. We define the margin ρ of f as the maximum margin $f(x)(w \cdot x + b)/\|w\|_1$ over all (w, b). We let our hypothesis set $\mathcal{H}$ consist of projection functions, namely, $\mathcal{H} = \{h_1, h_2, \ldots, h_n, h_{n+1}, h_{n+2}, \ldots, h_{2n}\}$, where $h_i : x \mapsto x_i$ if $i \le n$ and $h_i : x \mapsto -x_i$ otherwise, so that we can represent f as $f(x) = \sigma(\sum_{i=1}^{2n} w_i h_i(x) + b)$ for some non-negative $2n$-dimensional weight vector $w \ge 0$ and bias b. Then, we can represent the margin ρ of f as the solution of the optimal solution for the following LP problem:

$$\begin{aligned} &\max_{w,b,\rho} \rho \\ \text{s.t.} \quad &f(x)(\sum_{i=1}^{2n} w_i h_i(x) + b) \ge \rho \text{ for any } x \in \{-1,1\}^n, \\ &w \ge 0, \\ &\sum_i w_i = 1. \end{aligned} \tag{6.18}$$

Now we show that $\mathcal{H}$ actually satisfies ρ-WHA for f.

Lemma 7. *Let f be a linear threshold function with margin ρ. Then $\mathcal{H}$ satisfies ρ-WHA.*

Proof. Let ρ be the optimal solution of (6.18). By duality, ρ is identical to the optimal solution

γ of the dual problem of (6.18):

$$\begin{aligned} &\min_{d,\gamma} \gamma \\ \text{s.t.} \quad &\sum_{x\in\{-1,1\}^n} d_x f(x) h_i(x) \le \gamma \text{ for any } i \in [2n], \\ &d \ge 0, \\ &\sum_{x:f(x)=1} d_x = \sum_{x:f(x)=-1} d_x = 1/2. \end{aligned}$$

Apparently, the optimal γ should satisfy $\gamma = \max_{h\in\mathcal{H}} \sum_x d_x f(x)h(x)$, which implies the lemma. □

By the lemma and Theorem 6, we immediately have the following corollary.

Corollary 1. *Let f be a linear threshold function with margin ρ. Applying the ABDD Boosting algorithm with the sample $S = \{(x, f(x)) \mid x \in \{-1, 1\}^n\}$ of all (2^n) instances, our hypothesis set $\mathcal{H}$, and the precision parameter $\varepsilon = 1/2^n$, we obtain an ABDD T, which is equivalent to f of size $O\left(\frac{n}{\rho^4}(n + \ln(1/\rho))\right)$.*

6.5 Circuit and SDD Construction

Given our method for converting linear threshold functions into a Decision Diagram (DD) representation, the subsequent phase involves combining these representations following the NN structure to create an equivalent $(\vee, \wedge, \neg)$-circuit. This circuit serves as the foundation for verifying SR. Following this, we can transform the circuit into an SDD, a pivotal step in verifying the robustness.

The conversion process from a DD to a circuit follows a top-down manner. In Figure 6.4, the area enclosed by the red dashed box illustrates a conversion unit. This unit transforms a node x_1 and its two edges in the DD into four gates and a variable in the circuit. Subsequently, when the next node x_2 connects to a different edge of node x_1, a new circuit is generated starting with an 'or'-gate, connecting it to the 'and'-gate derived from the conversion of an edge associated with x_1. This process continues until all edges of the nodes lead to either the 0-leaf or the 1-leaf. The resulting circuit is equivalent to the DD and its corresponding neural unit. To build the equivalent circuit for the Neural Network (NN), we employ Algorithm 1 to generate

DDs for each neuron in the NN. Subsequently, these DDs are converted into circuit form. We then amalgamate these equivalent circuits based on the NN structure, establishing a one-to-one correspondence between the inputs and outputs of each neuron in the NN and the inputs and outputs of the circuit. This concludes the construction of the NN's equivalent circuit, a method also detailed in [52].

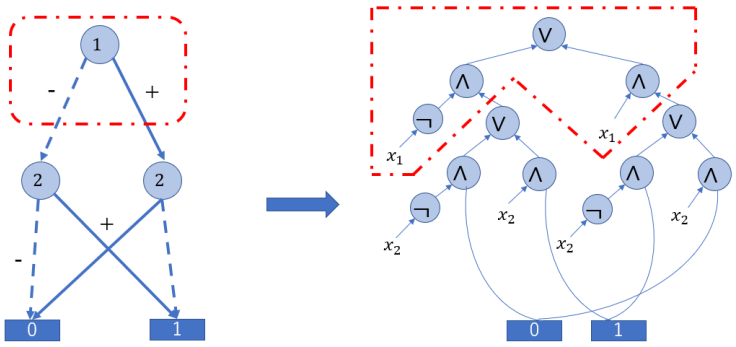


Figure 6.4: Example of converting BDD/OBDD/ABDD (Left) to the circuit (Right).

Once we obtain the equivalent circuit of an NN, the subsequent step involves converting it into an SDD. SDD belongs to a subclass of deterministic Decomposable Negation Normal Form (d-DNNF) circuits, characterized by stronger decomposability and a more robust form of determinism [17]. It's worth noting that the class of SDDs generalizes that of OBDDs, as every OBDD can be transformed into an SDD in linear time. Interestingly, certain Boolean functions possess polynomial-size SDD representations but only exponential-size OBDD representations [5]. In SDD, Boolean functions are represented through the introduction of "decision ($\vee$) nodes" and "conjunction ($\wedge$) nodes." This representation enhances the efficiency and expressiveness of the underlying Boolean functions.

Certainly, the size complexity of each $apply$ operation between two SDD nodes is directly proportional to the product of their internal nodes. Consequently, as the complexity of the circuit increases, constructing the corresponding SDD demands more space. Considering the conversion process according to ABDD, if the resulting circuit is smaller, it logically follows that the corresponding SDD constructed from it will also be reduced in size. The reduction in size during the circuit conversion directly impacts the resulting SDD, making it more compact. Therefore, by optimizing the BDD or circuit representation, we can achieve a more streamlined SDD.

6.6 Experiments

6.6.1 Experimental Setup

In our experiments, we employed the USPS digits dataset, which consists of hand-written digits represented by 16×16 binary pixel images. For SR verification, we specifically utilized image data labeled with 0 and 1. The NN architecture we used follows the design in [52], comprising two convolutional layers and a fully connected layer. During training, two convolutional layers with real-valued weights (kernel size 3, stride 2; kernel size 2, stride 2) are employed, each with a sigmoid activation function. The NN also includes a real-valued-weight fully connected layer. During testing, the sigmoid activation function is replaced with the previously mentioned step activation function. The experiments are conducted on a CPU of Intel(R) Xeon(R) Gold 2.60GHz. A batch size of 32 and a learning rate of 0.01, decaying to 10% at half and three-quarters of all learning epochs, are utilized. Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a weight decay of 0.0001 served as the optimizer for training the NN.

6.6.2 Sample-based Robustness (SR) validation

SR for a standard NN with real-valued outputs is typically achieved by identifying and modifying the pixels that have the most significant impact on recognition until the output changes. However, for BNNs with an input size of $h \times w$, the time complexity to compute robustness k using standard methods is $O((hw)^k)$, which is impractical for larger BNNs.

In contrast, when BDDs, OBDDs, and ABDD are easily represented as circuits, as illustrated in Figure 6.4, the circuits of various neural units are interconnected to form a circuit f representing the entire NN based on its network structure. Notably, we separately verify the SR of the OBDDs generated by methods based on Theorem 1 and Theorem 2 in [52] using a circuit representation. The integration of weights and bias in Theorem 2 is performed as follows.

For each w in weight W and bias b, We set $\alpha = \max\{|w_1|, \ldots, |w_n|, |b|\}$, then we turn them to integer weight $\hat{w}_i = \lfloor \frac{10^p}{\alpha} w_i \rfloor$ and bias $\hat{b} = \lfloor \frac{10^p}{\alpha} b \rfloor$ where p is the number of digits of precision.

As for $dis()$ in Definition 3, it's easy to express the $k \leq dis(x, x')$ between x and x' in a circuit form and likewise convert it to circuit $g_{k,x}$ denoted as follows:

$$g_{k,x}(x') = \begin{cases} 1, & if \mid x \oplus x' \mid \leq k \\ -1, & otherwise \end{cases} \tag{6.19}$$

We calculate the SR of f on (positive and negative) instance x, where have $f(x) = 1$ and $f(x) = -1$, by running Algorithm 3 of BDD, OBDD and ABDD on 10 CNNs, the results as shown in Table 6.2. Note that the SR of negative instances can be computed by invoking Algorithm 3 on function $\neg f$.

Algorithm 3 *SR*

Input: circuit f, (positive) instance $x \in \{-1, 1\}^n$;
Output: $r_f(x)$;

```
initial: r_f = 0;
for k = 1 to n do
    if g_{k,x} ∧ f̄ is satisfiable then
        break
    end if
end for
return k
```

Table 6.2

ID	SR	Time for SR (s)			Num of gates					
		Ours	(Shi.1)	(MM.)	Ours	(Shi.1)	(MM.)	(Shi.2 2)	(Shi.2 3)	(Shi.2 4)
1	1.83	3161	3994	4709	3737	5536	4433	48103	57079	57511
2	7.87	7142	9718	10626	1595	4039	2300	46972	57511	57505
3	3.79	3788	5050	5425	2384	4129	3146	51952	56611	54883
4	2.9	3970	4781	7655	3464	5128	5300	37474	57487	57493
5	4.94	5035	5902	7353	3235	3994	4249	43501	53971	57511
6	3.53	5229	7402	9731	3479	6049	4709	47083	55021	57511
7	2.84	4714	5839	7685	4963	5824	6613	47227	57511	57511
8	6.08	6559	8740	9334	3227	5233	3902	38983	57487	57511
9	4.01	5774	7323	8633	3659	5500	4577	47767	57487	57511
10	2.82	3711	5373	5393	2520	5497	3600	42439	57511	57511

In table 6.2, (Shi.1) circuits are generated using the Theorem 1 proposed in [52], while (MM.) circuits are generated using the method described in [42], and (Shi.2 p) circuits are generated using the Theorem 2 proposed in [52] under the number of digits of precision $p = 2, 3, 4$. As observed, the number of gates of (Shi.2 $2, 3, 4$) is significantly larger than the others, it also consumes more time during SR validation (over 10 hours).

6.6.3 Analysis

In our experiments, we trained a standard CNN that achieved a test set accuracy of 99.73%. Subsequently, by replacing the sigmoid activation function with the step activation function, the

accuracy of CNN slightly decreased to 99.22%. For the representation of BCNN, our algorithm demonstrates the capability to generate smaller ABDD compared to BDDs and OBDDs while maintaining the same recognition accuracy. This characteristic provides distinct advantages for subsequent work. It's noteworthy that the strategy employed in [52] involves randomly selecting multiple orders to generate several OBDDs and subsequently selecting the one with the smallest size. Despite setting the number of random orders to 100, the size of the OBDD remains larger than that of ABDD, showcasing the efficiency of our algorithm in comparison.

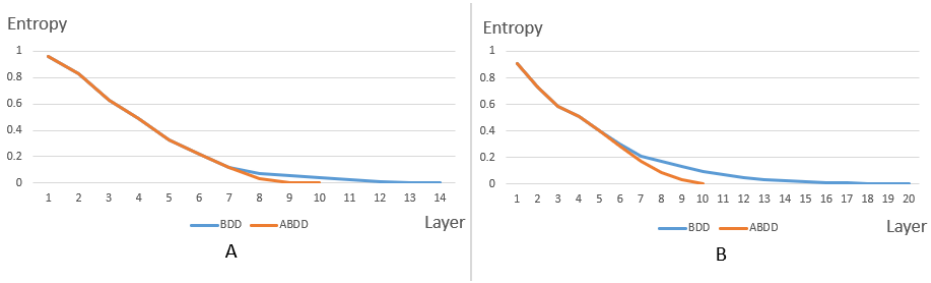


Figure 6.5: Illustration of the entropy change of ABDD and BDD at each depth. We show the change in the total entropy in each depth of the DD generated by the linear threshold functions of the 3×3 convolution layer (A) and the fully connected layer (B) when using the ABDD and BDD algorithms, respectively.

Comparing the methods in Table 6.1, the BDD algorithm proposed by Mansour and McAllester [42] is designed to identify the hypothesis h that leads to the most significant reduction in entropy at each node. Initially, this approach seems to produce smaller DDs due to the intuitive advantage of entropy reduction. However, our observations indicate that this advantage diminishes when the number of samples within a node changes during the merging process. Specifically, if the number of samples remains unchanged, i.e., the entropy of the child node does not decrease to zero, both BDD and our algorithm (ABDD) exhibit constant entropy at the same depth. However, once the phenomenon of entropy occurs, ABDD demonstrates a faster decrease in entropy compared to BDD, as illustrated in Figure 6.5. We attribute this phenomenon to the unique feature of our approach, which ensures that nodes at the same depth share the same variables. This characteristic provides us with a distinct advantage during the merging process, particularly when considering the entropy of each node. This leads to a faster reduction in entropy, as observed in our experimental results.

Comparing ABDD with OBDDs [52], we assert that ABDD provides a precise algorithm for finding the DD with the smallest size, eliminating the need for multiple randomizations.

Furthermore, when [52] applies OBDD based on neural unit weights, the time complexity increases exponentially with the size of the weights. In contrast, our algorithm exhibits minimal time consumption, unaffected by changes in weight size. Undoubtedly, this stands out as a significant advantage of our approach.

6.7 Conclusion

This chapter introduced a Boosting-based method that generates DD with smaller size and time complexities than conventional works. Our proposed method's resulting DD is named ABDD, a variant of standard DD, in which the same variable labels the nodes at the same depth, variables can be reused, and the depth is not limited to the number of dimensions of the variable. The experimental results highlight the versatility of ABDD, showcasing their capability to be connected in various configurations to effectively express NNs and facilitate efficient implementation of various verification tasks.

Chapter 7

Conclusion

In this paper, we address a series of topics related to neural networks, all of which revolve around image recognition tasks. The research initially begins with a Japanese handwritten character recognition task. While improving the recognition accuracy for 3-character ”Kuzushiji” images, we identified a limitation in traditional convolutional neural networks, which struggle to recognize rotated objects in the test set. We address this by introducing hexagonal convolutional kernels and a specialized network structure to achieve full rotation invariance in fully convolutional networks. Additionally, we explore methods to transform neural networks into more interpretable representations, allowing for a more efficient understanding of their various behaviors and decision-making processes.

In Chapter 3, we construct a more accurate dataset based on the CODH dataset and PRMU contest, making modifications to problematic labels and excluding questionable data. The 3-character ”Kuzushiji” images from this dataset are used in Chapter 4, where we introduce two segmentation methods to enhance the recognition accuracy of neural networks for these images. Both methods focus on the selection of bounding boxes used in character segmentation, one based on traditional object detection and clustering algorithms, and the other utilizing traditional image analysis tools in conjunction with the single character classifier for recognition. In Chapter 4.4, we extended the work in Chapter 4 to make it capable of running in the absence of bounding box information.

Chapter 5 concentrates on the issue of recognizing rotated objects with convolutional neural networks. We address how to make neural networks output the same results for both the unrotated images and the rotated images in the test set when trained on unrotated images. We propose HCNN, a modified version of traditional neural networks, which increases the network’s ability to recognize rotation features without data augmentation. We demonstrate that

in fully convolutional networks, it achieves full rotation invariance. For non-fully convolutional networks, our method is found to enhance the network's rotation invariance in experiments.

In Chapter 6, we delve into more efficient ways to explore various properties of neural networks. Our method is based on the basic of neural networks – neurons. We propose a method to transform each neuron in the network into a decision diagram that is more easily expressible. This approach relies on the Boosting algorithm to construct the decision diagram with new constraints called ABDD. Through experiments, we found that using ABDD to construct equivalent representations of neural networks could efficiently verify their properties while keeping the size minimal.

In conclusion, we not only improve neural networks for handwritten character recognition tasks but also demonstrate that using hexagonal kernels in fully convolutional networks can achieve full rotation invariance, contributing to the enhancement of the network's representation capabilities and generalizability. Furthermore, we propose a method to explore more properties of neural networks by converting each neuron into an equivalent Boolean function, allowing for the construction of more interpretable representations and efficient property verification.

Bibliography

[1] Adnan Amin. Off-line arabic character recognition: the state of the art. *Pattern recognition*, 31(5):517–530, 1998.

[2] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, 2007.

[3] A. Azeem, M. Sharif, J.H. Shah, and M. Raza. Hexagonal scale invariant feature transform (H-SIFT) for facial feature extraction. *Journal of applied research and technology*, 13:402 – 408, 2015.

[4] Erik J. Bekkers, Maxime W. Lafarge, Mitko Veta, Koen A. J. Eppenhof, Josien P. W. Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *Medical Image Computing and Computer Assisted Intervention*, volume 11070 of *Lecture Notes in Computer Science*, pages 440–448, 2018.

[5] Simone Bova. Sdds are exponentially more succinct than obdds. In *AAAI Conference on Artificial Intelligence*, pages 929–935, 2016.

[6] Nader H. Bshouty, Christino Tamon, and David K. Wilson. On learning width two branching programs. *Inf. Process. Lett.*, 65(4):217–222, 1998.

[7] Hei Chan and Adnan Darwiche. Reasoning about bayesian network classifiers. In *Conference in Uncertainty in Artificial Intelligence*, pages 107–115. Morgan Kaufmann, 2003.

[8] Benjamin Chidester, Tianming Zhou, Minh N. Do, and Jian Ma. Rotation equivariant and invariant neural networks for microscopy image analysis. *Bioinform.*, 35(14):i530–i537, 2019.

[9] Yolanda T. Chong, Judice L.Y. Koh, Helena Friesen, Supipi Kaluarachchi Duffy, Michael J. Cox, Alan Moses, Jason Moffat, Charles Boone, and Brenda J. Andrews. Yeast proteome dynamics from single cell imaging and automated analysis. *Cell*, 161(6):1413–1424, 2015.

[10] Jan Chorowski and Jacek M. Zurada. Top-down induction of reduced ordered decision diagrams from neural networks. In *International Conference on Artificial Neural Networks*, 2011.

[11] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep Learning for Classical Japanese Literature, 2018. URL `http://arxiv.org/abs/1812.01718`.

[12] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718, 2018. URL `http://arxiv.org/abs/1812.01718`.

[13] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2990–2999, 2016.

[14] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. In *International Conference on Learning Representations*, 2018.

[15] Sonya A. Coleman, Bryan W. Scotney, and Bryan Gardiner. Tri-directional gradient operators for hexagonal image processing. *J. Vis. Commun. Image Represent.*, 38:614–626, 2016.

[16] Laurent Condat, Dimitri Van De Ville, and Brigitte Forster-Heinlein. Reversible, fast, and high-quality grid conversions. *IEEE Trans. Image Process.*, 17(5):679–693, 2008.

[17] Adnan Darwiche. SDD: A new canonical representation of propositional knowledge bases. In Toby Walsh, editor, *International Joint Conference on Artificial Intelligence*, pages 819–826, 2011.

[18] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *International conference on Machine learning*, pages 233–240. ACM, 2006.

[19] Valentin Delchevalerie, Adrien Bibal, Benoît Frénay, and Alexandre Mayer. Achieving rotational invariance with bessel-convolutional neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 28772–28783, 2021.

[20] Alexander S. Ecker, Fabian H. Sinz, Emmanouil Froudarakis, Paul G. Fahey, Santiago A. Cadena, Edgar Y. Walker, Erick Cobos, Jacob Reimer, Andreas S. Tolias, and Matthias Bethge. A rotation-equivariant convolutional neural network model of primary visual cortex. In *International Conference on Learning Representations*, 2019.

[21] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical cnns. In *European Conference on Computer Vision*, volume 11217 of *Lecture Notes in Computer Science*, pages 54–70, 2018.

[22] Sadegh Fadaei and Abdolreza Rashno. A framework for hexagonal image processing using hexagonal pixel-perfect approximations in subpixel resolution. *IEEE Trans. Image Process.*, 30:4555–4570, 2021.

[23] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Trans. Syst. Sci. Cybern.*, 5(4):322–333, 1969.

[24] Bryan Gardiner, Sonya A. Coleman, and Bryan W. Scotney. Multiscale edge detection using a finite element framework for hexagonal pixel-based images. *IEEE Trans. Image Process.*, 25(4):1849–1861, 2016.

[25] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[26] Marcel J. E. Golay. Hexagonal parallel pattern transformations. *IEEE Trans. Computers*, 18(8):733–740, 1969.

[27] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutnk, Bas R. Steunebrink, and Jrgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28:2222–2232, 2015.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[29] Xiangjian He, Wenjing Jia, Qiang Wu, Namho Hur, Tom Hintz, Huaqing Wang, and Jinwoong Kim. Basic transformations on virtual hexagonal structure. In *International Conference on Computer Graphics, Imaging and Visualization*, pages 243–248, 2006.

[30] Emiel Hoogeboom, Jorn W. T. Peters, Taco S. Cohen, and Max Welling. Hexaconv. In *International Conference on Learning Representations*, 2018.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456, 2015.

[32] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Jintao Ke, Hai Yang, Hongyu Zheng, Xiqun Chen, Yitian Jia, Pinghua Gong, and Jieping Ye. Hexagon-based convolutional neural network for supply-demand forecasting of ride-sourcing services. *IEEE Trans. Intell. Transp. Syst.*, 20(11):4160–4173, 2019.

[34] Michael Kearns and Yishay Mansour. On the boosting ability of top–down decision tree learning algorithms. *Journal of Computer and System Sciences*, 58(1):109–128, 1999.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Annual Conference on Neural Information Processing Systems*, pages 1106–1114, 2012.

[36] Yann LeCun and Yoshua Bengio. *Convolutional networks for images, speech, and time series*, page 255–258. 1998.

[37] Bingyuan Liu, Christopher Malon, Lingzhou Xue, and Erik Kruus. Improving neural network robustness through neighborhood preserving layers. In *International Conference of Pattern Recognition, International Workshops and Challenges*, pages 179–195, 2020.

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[39] Junren Luo, Wanpeng Zhang, Jiongming Su, and Fengtao Xiang. Hexagonal convolutional neural networks for hexagonal grids. *IEEE Access*, 7:142738–142749, 2019.

[40] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, 2013.

[41] Ravi Mangal, Aditya V. Nori, and Alessandro Orso. Robustness of neural networks: a probabilistic and practical approach. In *International Conference on Software Engineering: New Ideas and Emerging Results*, pages 93–96. IEEE / ACM, 2019.

[42] Yishay Mansour and David McAllester. Boosting using branching programs. *Journal of Computer and System Sciences*, 64(1):103–112, 2002.

[43] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *International Conference on Pattern Recognition*, pages 2012–2017, 2016.

[44] Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *IEEE International Conference on Computer Vision*, pages 5058–5067, 2017.

[45] Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

[46] Hung Tuan Nguyen, Nam Tuan Ly, Kha Cong Nguyen, Cuong Tuan Nguyen, and Masaki Nakagawa. Attempts to recognize anomalously deformed Kana in Japanese historical documents. In *International Workshop on Historical Document Imaging and Processing*, pages 31–36, 2017.

[47] Nathanaël Perraudin, Michaël Defferrard, Tomasz Kacprzak, and Raphael Sgier. Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications. *Astron. Comput.*, 27:130–146, 2019.

[48] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[49] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[51] Tobias Schlosser, Michael Friedrich, and Danny Kowerko. Hexagonal image processing in the context of machine learning: Conception of a biologically inspired hexagonal deep learning framework. In *IEEE International Conference On Machine Learning And Applications*, pages 1866–1873, 2019.

[52] Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *International Conference on Principles of Knowledge Representation and Reasoning*, pages 882–892, 2020.

[53] Andy Shih, Adnan Darwiche, and Arthur Choi. Verifying binarized neural networks by angluin-style learning. In *Theory and Applications of Satisfiability Testing International Conference*, 2019.

[54] Constantin Steppa and Tim Lukas Holch. Hexagdly - processing hexagonally sampled data with cnns in pytorch. *SoftwareX*, 9:193–198, 2019.

[55] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[56] Eiji Takimoto and Akira Maruoka. Top-down decision tree learning as information based boosting. *Theoretical Computer Science*, 292(2):447–464, 2003.

[57] Yiping Tang, Kohei Hatano, Emi Ishita, Tetsuya Nakatoh, and Toshifumi Kawahira. Construction of japanese historical hand-written characters segmentation data from the codh data sets. In *Conference of Japanese Association for Digital Humanities*, pages 183–185. JADH2018, 2018.

[58] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2): 154–171, 2013.

[59] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[60] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018.

[61] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5987–5995, 2017.

[62] Tang Yiping, Kohei Hatano, Emi Ishita, Tetsuya Nakatoh, and Toshifumi Kawahira. Construction of Japanese Historical Hand-Written Characters Segmentation Data from the CODH Data Sets. In *Conference of Japanese Association for Digital Humanities*, pages 183–185, 2018.

[63] Fuxun Yu, Zhuwei Qin, Chenchen Liu, Liang Zhao, Yanzhi Wang, and Xiang Chen. Interpreting and evaluating neural network robustness. In *International Joint Conference on Artificial Intelligence*, pages 4199–4205, 2019.

[64] Ting Zhao and Robert F. Murphy. Automated learning of generative models for subcellular location: Building blocks for systems biology. *Cytometry Part A*, 71A, 2007.

[65] Yunxiang Zhao, Qiuhong Ke, Flip Korn, Jianzhong Qi, and Rui Zhang. Hexcnn: A framework for native hexagonal convolutional neural networks. In *IEEE International Conference on Data Mining*, pages 1424–1429, 2020.

[66] Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow. Improving the robustness of deep neural networks via stability training. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4480–4488, 2016.