A Study of Inner Representations in Deep Neural Networks for Comprehending Network Behaviors Using Persistent Homology

ディープニューラルネットワークの挙動解析を目的としたパーシステントホモロジーを用いたネットワーク内部構造の研究

July, 2022

Satoru WATANABE
渡辺　聡

A Study of Inner Representations in Deep Neural Networks for Comprehending Network Behaviors Using Persistent Homology

ディープニューラルネットワークの挙動解析を目的としたパーシステントホモロジーを用いたネットワーク内部構造の研究

July, 2022

Waseda University Graduate School of Fundamental Science and Engineering

Department of Computer Science and Communications Engineering, Research on Parallel and Distributed Architecture

Satoru WATANABE
渡辺　聡

Abstract

Deep neural networks (DNNs) have demonstrated remarkable performance in various fields, including image analysis, speech recognition, and text classification. However, inner representations in DNNs are indecipherable. This makes it difficult to tune DNN models, control their training processes, and interpret their outputs. The inner representation retains the knowledge learned through the training and determines the network behaviors. Thus, the investigation of the inner representation is essential to improve the comprehension of network behaviors.

This thesis investigates the inner representation of DNNs using persistent homology (PH). The inner representation retains the knowledge learned through the training, which is represented by the combination of the neurons. Furthermore, PH reveals the combinational effects of multiple neurons in DNNs from the viewpoints of the number and stability of holes, which are difficult to capture without using PH. Thus, the comprehension of network behaviors can be improved through the investigation of the inner representation of DNNs using PH.

PH is designed to be applied to simplicial complexes in topological spaces. Due to the difference between network parameters in DNNs and simplicial complexes in topological spaces, PH cannot be straightforwardly applied to the investigation of DNNs. Consequently, this thesis aims to fulfill the following goals: developing a method to investigate the inner representation of DNNs using PH, and improving the comprehension of network behaviors using PH. To achieve these goals, this thesis proposes a construction method for clique complexes on DNNs (Chapter 2). Moreover, the changes in PH involved in different network parameters of DNNs are analyzed (Chapter 3). Additionally, this work proposes two investigation methods using PH to improve the comprehension of their inner representation of DNNs (Chapters 3 and 4).

In Chapter 2, this thesis proposes a construction method for clique complexes on DNNs through the introducion of two techniques: normalization and propagation. Furthermore, this thesis mathematically proves the correctness of the construction method. Additionally, the PH calculation method is formalized for the clique complexes con-

structed on DNNs. The construction of clique complexes and formalization in PH calculation provide a foundation for studying on the inner representation of DNNs using PH.

In Chapter 3, this thesis analyzes changes in PH involved in different network parameters of DNNs using PH diagrams. PH diagrams enable us to review the number and the stability of holes that appeared in the clique complexes obtained from trained DNNs. The analysis reveals that PH changes with the difficulty of the problem for which the DNNs are trained. This indicates that PH reflects the inner representation of trained DNNs.

In Chapters 4 and 5, two investigation methods of DNNs are proposed for overfitting detection and network pruning, respectively. The overfitting-detection method enables us to filter overfitted DNNs without relying on the training data. The network-pruning method prunes 95% of the edges from DNNs with 12% higher accuracy than a common baseline of pruning methods in our evaluation settings including dataset, network structure, and training process. These methods are refined by the investigation of DNNs using PH. Thus, they demonstrate improvements in the comprehension of network behaviors using PH.

Acknowledgements

I was working full-time while pursuing my Ph. D. and received immense support from the people around me during these three years.

First, I would like to express my gratitude to Prof. Yamana. He accepted my application to conduct my Ph. D. study in his laboratory. We first discussed the engagement in March 2019. I began my research in September 2019 after passing the entrance examination in June. Throughout the research, he advised me on how to improve the study, particularly in the theoretical and descriptive parts. I am also grateful for his numerous support in the application for my doctorate.

I would also like to express my gratitude to the members of the examination committee: Profs. Sugawara, Uchida, and Kasai. They provided me with useful suggestions for enhancing this thesis. Though the discussion with the committee, I found that my publications include inadequate descriptions in several essential parts. Then, the thesis's description of the proposed technology was significantly improved through the discussion.

I would also like to thank my colleagues at Hitachi Ltd. and Hitachi America Ltd. They allowed me to attend the weekly seminar in the laboratory and adjusted the schedule to avoid conflict with the seminar. They were tolerant of the delay in my response due to the study, which extended especially when I was inspired by new ideas about the study.

The members of the Yamana laboratory provided me with new insights into emerging technologies in the weekly seminar. The members of the big data group in the laboratory attended the bi-weekly discussion with Prof. Yamana. Owing to these periodic meetings, I could control my progress of the study. I would like to express my gratitude to them.

Finally, my sincerest gratitude goes to my family. They encouraged my challenge, which convinced me to engage in the study. I appreciate their continued encouragement guiding me in difficult moments. Last but not least, my wife, Kaoru, regularly supported my study after working hours and on weekends. Moreover, her specialty as a nutritionist improved my health, which was essential throughout the study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Challenges

Artificial intelligence is utilized in various fields, including image analysis, speech recognition, and text classification, where deep neural networks (DNNs) are essential in learning from large amounts of data [33, 91]. However, inner representations in DNNs are indecipherable. This makes it difficult to tune DNN models, control their training processes, and interpret their outputs.

This thesis investigates the inner representation of DNNs using topological data analysis (TDA). TDA employs results from geometry and topology [61, 81], which has provided new insights in various fields such as neuroscience [12, 17, 62, 75, 87], proteomics [11, 29, 86], and material science [35, 41]. Persistent homology (PH) [10, 26, 70] is a prominent method in TDA because of its theoretical foundation, practical computability, and robustness with small perturbations [61]. These advantages are beneficial for investigating DNNs. Theoretical foundation and practical computability are fundamental for extracting knowledge from empirical observations, while robustness is indispensable for investigating DNNs involving parameter perturbations. Thus, PH is a prominent method to investigate DNNs for improving the comprehension of network behaviors.

The inner representation of DNNs constitutes network parameters calibrated with network training. The inner representation retains the knowledge learned through the training and determines the network behaviors. Therefore, the investigation of the inner representation is essential in improving the comprehension of network behaviors. Meanwhile, PH is designed to be applied to simplicial complexes in topological spaces. Due to the

difference between network parameters in DNNs and simplicial complexes in topological spaces, PH cannot be straightforwardly applied to the investigation of DNNs. Consequently, this thesis aims to fulfill the following goals.

(G1) Developing a method to investigate the inner representation of DNNs using PH

(G2) Improving the comprehension of network behaviors using PH

1.2 Intuition behind Topological Analysis on DNNs

1.2.1 Inner Representation in DNNs and Topological Analysis

DNNs work as knowledge-distilling pipelines [46]. Alternatively, the degree of feature abstraction increases with the depth of DNN layers, where higher-level features are obtained by combining lower-level features. For example, images of cats are incrementally abstracted from pixels to diagonal lines and from diagonal lines to ear shapes. Then, DNNs can detect cats using a combination of the high-level features, such as ear and body shapes [14]. In this process, the neurons in the DNNs represent the features. Thus, the combination of the neurons, described by the network parameters, represents the implementation of knowledge retained in DNNs.

PH is a method for computing the topological features of a simplicial complex. One-dimensional PH counts the number of holes in a simplicial complex. The combination of the neurons develops holes in the simplicial complex, where the neurons and the connections between the neurons are considered as vertexes and edges in the simplicial complex, respectively. Further, the one-dimensional PH measures the stability of the holes, which varies depending on the number of vertexes and the distance between the vertexes. Then, it can reveal the cooperation among neurons for activating their subsequent neurons, which is hereafter called combinational effect of neurons. Accordingly, PH reveals the combinational effects of multiple neurons in DNNs from the viewpoints of the number and stability of holes, which are difficult to capture without using PH. Thus, the comprehension of network behaviors can be improved through the investigation of the inner representation of DNNs using PH.

1.2.2 Topological Analysis on Human Brain

Previous studies have demonstrated that PH can be used for comparing and characterizing human brains. Cassidy et al. employed PH as a tool for comparing human brains using functional magnetic resonance imaging (fMRI) [12]. Petri et al. demonstrated that psilocybin affects the homological structure of the brain's functional patterns [62]. Furthermore, Sizemore et al. employed PH to highlight the crucial features of human brains from diffusion spectrum imaging (DSI) [75]. However, it is often difficult to quantify the activation of neurons by fMRIs and DSIs. Hence, PH is more useful in the computer science, particularly for DNNs because their network structures and the activation of neurons can be described mathematically. In this work, we employed PH to investigate the process of training a DNN and evaluate its knowledge representation complexities.

1.2.3 Investigation of DNNs using PH

Functional connectivity is an essential concept in neuroscience [20, 44, 71]. It is defined as the synchronization of spatially remote neurophysiological events [28]. It enables us to know the collaboration among neurons to achieve brain tasks. Considering the importance of functional connectivity in neuroscience, this thesis applied the concept of functional connectivity to DNNs.

To apply the concept to DNNs, this thesis defined relevance values among neurons based on the contribution to the activation of the following neuron. Furthermore, the clique complex induces subgraphs comprising synchronized neurons, called simplexes. This thesis proved that the simplexes forms a simplicial complex on DNNs, which enables us to study DNNs using PH.

PH investigates simplicial complexes at different resolutions. Edges and faces (1- and 2-simplexes) appear and disappear depending on the resolution, which develop and vanish holes on the simplicial complex. The one-dimensional PH investigates the number and stability of the holes. Thus, PH investigates the sets of synchronized neurons at different resolutions for revealing the inner representation of DNNs from the standpoint of functional connectivity.

1.3 Overview of Contributions

To achieve the goals, this thesis proposes a construction method for clique complexes on DNNs and analyzes the changes in PH involved in different network parameters of DNNs. Additionally, this thesis proposes two investigation methods for DNNs using PH to improve the comprehension of the inner representation of DNNs. Sections 1.3.1–1.3.4 summarize the four contributions ((C1)–(C4)).

1.3.1 Proposal of a Construction Method for Clique Complex to DNNs (C1)

This thesis develops a construction method for clique complexes on DNNs through the introduction of two techniques: normalization and propagation. These techniques are inspired by the deep Taylor decomposition method, which has been designed to reveal the influential inputs to the outputs of DNNs. Furthermore, this thesis enhances the method for investigating the inner representation of DNNs and mathematically proves the correctness of the construction method.

Additionally, the PH calculation method is formalized for the clique complexes constructed on DNNs, considering three types of layers: dense, convolutional, and pooling layers. These layers are prevalent in many DNN applications, such as image analysis, speech recognition, and text classification. The construction of clique complexes and formalization in PH calculations provide a foundation for studying on the inner representation of DNNs using PH.

1.3.2 Analysis on the Changes in PH Involved in Different Network Parameters of DNNs (C2)

This thesis analyzes the changes in PH involved in different network parameters of DNNs using PH diagrams. PH diagrams enable us to review the number and the stability of holes that appeared in the clique complexes obtained from trained DNNs. DNNs are trained varying the difficulty of the problem. Then, the inner representations are analyzed using PH. This analysis reveals that PH changes with the problem for which the DNNs are trained. This indicates that PH reflects the inner representation of trained DNNs.

To confirm the robustness of DNNs' investigation using PH, this thesis conducts each experiment 10 times with 30 different settings. The process is carried out using random initial values of network weights, resulting in a total of 300 experiments. These experiments reveal that the results obtained from the investigation using PH are robust with the network's settings and initial weights.

1.3.3 Proposal of an Overfitting Measurement of DNNs Using PH (C3)

Overfitting reduces the generalizability of DNNs. Overfitting is generally detected by comparing the accuracies and losses of the training and validation data, where the validation data are formed from a portion of the training data. However, detection methods are ineffective for pretrained networks distributed without the training data. Thus, this work proposes a method to detect the overfitting of DNNs using the trained network weights inspired by the dropout technique, which enables us to compare the degrees of overfitting among trained DNNs. The dropout technique has been employed to prevent DNNs from overfitting, where the neurons in the DNNs are invalidated randomly during their training. It has been hypothesized that this technique prevents DNNs from overfitting by restraining co-adaptations among neurons. This hypothesis implies that the overfitting of DNNs results from co-adaptations among neurons and can be detected by investigating the inner representation of DNNs. The proposed persistent homology-based overfitting measure (PHOM) method constructs clique complexes in DNNs using trained network weights. Moreover, one-dimensional PH investigates co-adaptations among neurons.

In addition, this thesis enhances PHOM to normalized PHOM (NPHOM) to mitigate the fluctuation in PHOM caused by the difference in network structures. The proposed methods are applied to convolutional neural networks trained for the classification problems on the CIFAR-10, street view house number, Tiny ImageNet, and CIFAR-100 datasets. The experimental results demonstrate that PHOM and NPHOM can indicate the degree of the overfitting of DNNs. Therefore, these methods enable us to filter overfitted DNNs without requiring the training data.

Table 1.1: Layout of this thesis

Chapter	Description	Contribution	Challenge
1	Introduction	Not applicable	
2	Construction of clique complex on DNNs	C1	G1
3	Analysis on the changes in PH involved in different network parameters of DNNs	C2	G1
4	Proposal of overfitting measure of DNNs using PH	C3	G2
5	Proposal of network pruning method of DNNs using PH	C4	G2
6	Conclusion	Not applicable	

1.3.4 Proposal of a Network Pruning Method Using PH (C4)

The consumption of enormous computation resources prevents DNNs from operating on small computers such as edge sensors and handheld devices. Network pruning (NP), which removes parameters from trained DNNs, is a prominent method of reducing the resource consumption of DNNs. This thesis proposes a PH-based NP method (PHNP). PH investigates the inner representation of knowledge in DNNs, and PHNP utilizes the investigation in NP to improve the efficiency of pruning. PHNP prunes DNNs in ascending order of magnitudes of the combinational effects among neurons, which are calculated using one-dimensional PH, to prevent the deterioration of the accuracy. PHOM is compared to the global magnitude pruning method (GMP), which is a common baseline for evaluating pruning methods. The evaluation results show that the classification accuracy of DNNs pruned by PHNP outperforms that pruned by GMP.

1.4 Layout of This Thesis

Table 1.1 lists the correspondences among chapters, contributions, and challenges. Chapters 2 and 3 contain the contributions of (C1) and (C2), respectively. The contributions of (C1) and (C2) achieve the challenge of (G1), enabling us to investigate the inner representation of DNNs using PH. Furthermore, this thesis provides the contributions of (C3) and (C4) by proposing two investigation methods for DNNs in Chapters 4 and 5. These methods are reinforced by the investigation using PH and demonstrate that the contribu-

tions of (C3) and (C4) achieve the challenge of (G2), meaning that the comprehension of networks' behaviors is improved using PH.

Chapter 2

Construction Method for Clique Complexes on DNNs

2.1 Introduction

2.1.1 Application Strategy of PH to DNNs

Brain connectivity has been studied using functional connectivity [20, 44, 71], which is defined as the synchronization of spatially remote neurophysiological events [28]. Synchronization denotes the collaboration among neurons to achieve brain tasks. Therefore, functional connectivity is essential in studying the mechanism in brains.

This thesis applied the concept of functional connectivity to DNNs. To extract the synchronized neurons in DNNs, the following three steps were used;

(i) define a relevance between adjacent neurons using deep Taylor decomposition (DTD) [55];

(ii) introduce normalization and propagation techniques to extend the relevance among neurons in remote layers;

(iii) induce subgraphs comprising the synchronized neurons using the clique complex.

DTD defines the relevance among adjacent neurons which indicates the degree of the preceding neuron's contribution to the activation of the following neuron. Two neurons with a high relevance value tend to activate simultaneously, indicating that they are synchronized. In the step (ii), the two techniques were introduced to extend the relevance among neurons in remote layers. Finally, the clique complex induces subgraphs comprising synchronized neuron. Various methods have been proposed to construct simplical

complex on point cloud data, such as Čech, Vietoris-Rips, alpha, and clique complexes. In these methods, the clique complex is suitable for graph and network analysis [64]. Moreover, it is prominently used in brain-network analysis [18, 44]. Thus, we employed the clique complex to extract cliques of synchronized neurons from DNNs.

2.1.2 Investigation of DNNs Using PH

PH investigates simplicial complexes at different resolutions. All vertexes in the simplicial complex are isolated when the resolution is set to the maximum. Meanwhile, edges and faces (1- and 2-simplexes) appear and disappear when the resolution decreases, developing and vanishing holes in the simplicial complex. The one-dimensional PH investigates the holes' number and stability.

The stability of holes varies depending on the number and topological structure of neurons forming the hole. Since simplexes on DNNs were defined by the sets of synchronized neurons, the holes comprise synchronized neurons. Thus, the neurons in stable holes can be activated even when some neurons in the holes are inactive. Contrary, the activation in unstable holes is vulnerable to the inactivation of other neurons. That is, the stability of holes correlates with the number of neurons supporting the activation. Thus, PH investigates the sets of synchronized neurons at different resolutions for revealing the inner representation of DNNs from the standpoint of functional connectivity.

This chapter formalizes the PH calculation method for the clique complexes constructed on DNNs, considering three types of layers: dense, convolutional, and pooling layers. These layers are prevalent in many DNN applications, such as image analysis, speech recognition, and text classification. The construction of clique complexes and formalization in PH calculation provides a foundation for studying on the inner representation of DNNs using PH.

Additionally, this chapter also provides an example of a clique complex on a DNN to illustrate PH's calculation. Finally, an algorithm is proposed to identify all simplexes from the clique complex, which is required for the calculation of PH using public libraries. The algorithm enables us to utilize the high-performance implementation of public libraries in calculating the PH of DNNs.

2.2 Preliminary

The terms of TDA and PH can be understood based on previous studies [23,37,61], while introductory videos explaining TDA and PH can be found on on-demand video services[1].

2.2.1 Persistent Homology

The homology groups of orders zero and one represent the number of connected components and holes, respectively. PH is a method for computing the homology groups at different resolutions. The formal definition of PH is provided in Definitions 1–4.

Definition 1. *An abstract simplicial complex is a finite collection of sets $\mathscr{K}$ such that $X \in \mathscr{K}$ and $Y \subseteq X$ implies $Y \in \mathscr{K}$.*

The sets X in $\mathscr{K}$ denote its simplices. The dimension of a simplex is $\dim X = \operatorname{card} X - 1$, where $\operatorname{card} X$ denotes the cardinality of X. The dimension of an abstract simplicial complex is the maximum dimension of any of its simplices. The vertex set is the set consisting of all the simplices of dimension 0, while the face of a simplex X is a non-empty subset $Y \subseteq X$.

A p-chain c of a simplicial complex $\mathscr{K}$ is a formal sum of p-simplices in $\mathscr{K}$, that is, $c = \sum a_i X_i$, where X_i are p-simplices and a_i are coefficients. We employ module-2 coefficients, that is, a_i are either 0 or 1 and $1+1=0$. The binary arithmetic of two p-chains $c = \sum a_i X_i$ and $c' = \sum b_i X_i$ is defined as $c + c' = \sum (a_i + b_i) X_i$, where the coefficients are of modulo-2. The p-chain forms a group denoted as C_p.

A boundary operator ∂_p is a map from a p-simplex to the sum of its $(p-1)$-simplices. Formally, $\partial_p X = \sum_{j=0}^{p} [v_0, \ldots, \hat{v_j}, \ldots, v_p]$, where $[v_0, \ldots, v_p]$ is the simplex with vertices, while the hat indicates that v_j is removed. A chain complex is the sequence of chain groups connected by boundary operators, $\cdots \xrightarrow{\partial_{p+2}} C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \cdots$. A p-cycle is a p-chain with an empty boundary forming a group denoted as $Z_p = \ker \partial_p$. A p-boundary is a p-chain, that is, the image of a $(p+1)$-chain forming a group denoted as $B_p = \operatorname{im} \partial_{p+1}$.

[1] `https://www.youtube.com/watch?v=akgU8nRNIp0`, `https://www.youtube.com/watch?v=2PSqWBIrn90`

Definition 2. *The p-th homology group denoted as $H_p(= Z_p/B_p)$ is the p-th cycle group modulo the p-th boundary group. The p-th Betti number β_p is the rank of H_p.*

Definition 3. *A filtration of the simplicial complex $\mathscr{K}$ is a sequence of simplicial complex such that $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = \mathscr{K}$.*

For every $i \leq j$, there is an induced homomorphism in each dimension p, $f_p^{i,j}$ from $H_p(K_i)$ to $H_p(K_j)$. $f_p^{i,j}$ satisfies the condition of $f_p^{k,j} \circ f_p^{i,k} = f_p^{i,j}$ for all $0 \leq i \leq k \leq j \leq n$.

Definition 4. *Let $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = \mathscr{K}$ be a filtration. The p-th PH of $\mathscr{K}$ is the pair $(\{H_p(K_i)\}_{0\leq i\leq n}, \{f_p^{i,j}\}_{0\leq i\leq j\leq n})$, where the homomorphism $f_p^{i,j} : H_p(K_i) \to H_p(K_j)$ represents the maps induced by including maps $K_i \to K_j$.*

A homology $\gamma \in H_p(K_i)$ can be said to be born at K_i if $\gamma \notin \mathrm{im} f_p^{i-1,i}$. Furthermore, if γ is born at K_i, then it dies entering K_j if $f_p^{i,j-1}(\gamma) \notin \mathrm{im} f_p^{i-1,j-1}$ but $f_p^{i,j}(\gamma) \in \mathrm{im} f_p^{i-1,j}$. The lifetime of γ is represented by the half-open interval $[i, j)$. If $f_p^{i,j}(\gamma) \neq 0$ $(i \leq \forall j \leq n)$, γ can be said to live forever, and its lifetime is the interval $[i, \infty)$.

2.2.2 Diagrams

PH is a method to compute the topological features of a space. The zero- and one-dimensional PHs count the number of connected components and holes, respectively, whose formal definition can be found in the literature [23]. Fig. 2.1(a) shows points with oblique lined circles in $\mathbb{R}^2$. When the radius of the circles is small, the points are isolated. We gradually increase the radius of the circles, and then the circles produce the encircled regions in $\mathbb{R}^2$. The appearance of these encircled regions corresponds to the birth of homologies. We then increase the radius of the circles further, which causes the encircled regions to disappear. The disappearance of the encircled region corresponds to the death of homologies.

Fig. 2.1(b) shows the PH diagram of Fig. 2.1(a). Here, the X and Y axes present the birth and death of the homologies, respectively. The coordinate values on the axes are determined by the radius of the oblique lined circles. The two points in Fig. 2.1(b) correspond to the two enlarged regions in Fig. 2.1(a). Note that the small region in Fig. 2.1(a) is less stable than the large region. The stability of the regions is indicated by the

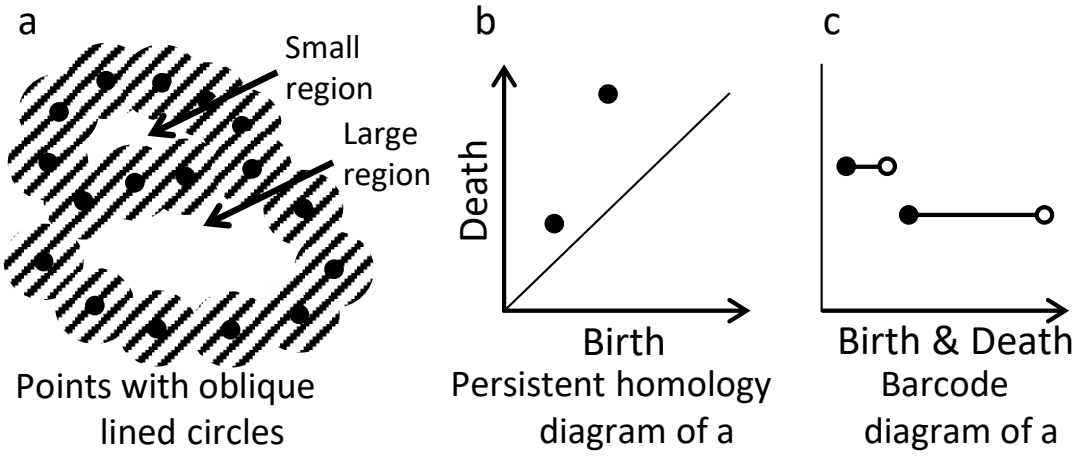


Figure 2.1: (a) Examples of persistent homology; (b) persistent homology diagram of (a); (c) barcode diagram of (a).

distance from the diagonal line in Fig. 2.1(b), i.e., the distance from the diagonal line of the point corresponding to the small region is shorter than that to the large region.

A barcode diagram provides the same information as the PH diagram. The barcode diagram of Fig. 2.1(a) is shown in Fig. 2.1(c), where the start and end points of the lines parallel to the X-axis show the birth and death of homologies, respectively. The short and long lines correspond to the small and large regions, respectively. The stability of the regions is indicated by the line length. The Y-axis in Fig. 2.1(c) indicates no meaningful information, which is sometimes used to group the results by the dimension of PH.

The computation of PH involves the explosion of the complexity due to the increase in the number of vertices. Different methods have been implemented to realize this computation [61], including GUDHI [53], JavaPlex [78], and Dionysus [25, 57].

2.3 Construction of Clique Complexes on DNNs

2.3.1 Dense Layers

We consider a set of neurons as vertices $V = \{v_0, \ldots, v_n\}$, where $n+1$ is the number of neurons. DNNs are considered as directed graphs with weights w_{ij}, where w_{ij} denotes the weight between v_i and v_j; here, w_{ij} is zero if v_i and v_j are not connected. We set the value of the relevance of identical neurons to one and the relevance R_{ij} between the connected neurons v_i and v_j as the normalized weight. Formally we set

$$R_{ij} = \begin{cases} 1 & (i = j) \\ w^+_{ij} / \sum_{i, i \neq j} w^+_{ij} & (i \neq j), \end{cases} \tag{2.1}$$

where w_{ij}^{+} denotes the positive part of the weight, i.e. $w_{ij}^{+} = \max\{0, w_{ij}\}$. R_{ij} indicates the relevance between v_i and v_j because the input to the j-th neuron is calculated by $\sum_i a_i w_{ij} + b_j$ in DNNs, where a_i is the activation of the i-th neuron and b_j is the bias [14]. We employed the positive part of the weight and ignored the bias, in a manner similar to the z^{+}-rule defined in deep Taylor decomposition [55].

To construct clique complexes on DNNs, the relevance was extended to indirectly connected neurons. For example, when v_0 and v_2 are connected to a path $v_0 \to v_1 \to v_2$, the relevance between v_0 and v_2 corresponding to the path is defined as $R_{01}R_{12}$. The intuition behind the definition is as follows: R_{01} and R_{12} indicate the contributions of v_0 and v_1 to the increase in the inputs of v_1 and v_2, respectively; $R_{01}R_{12}$ indicates the contribution of v_0 to the increase in the input of v_2. Formally we set

$$\widetilde{R_{ij}} = \max_{(v_i, v_{m_1} \ldots, v_{m_k}, v_j) \in L_{ij}} R_{v_i v_{m_1}} \cdots R_{v_{m_k} v_j}, \tag{2.2}$$

where L_{ij} denotes the set of all possible paths from v_i to v_j. It is possible to define $\widetilde{R_{ij}}$ using multiple paths in L_{ij}. However, the maximum was employed in Eq. (2.2) to improve computational efficiency.

Masulli et al. constructed a clique complex $K(G)$ on a finite directed weighted graph $G = (V, E)$ with vertex set V and edge set E with no self-loops and no double edges [54]. They defined the clique complex $K(G)$ as $K(G)_0 = V$ and $K(G)_p = \{(v_{K_0}, \ldots, v_{K_p})\ ;\ v_{K_i} \in V, (v_{K_i}, v_{K_j}) \in E \text{ for all } K_i < K_j\}$ (for $p \geq 1$), where $K(G)_p$ denotes the set of p-simplices on G.

Correspondingly, $\widetilde{R_{ij}}$ enables the construction of a clique complex and filtration on V. The neurons were numbered in ascending order from the output to input layers. Hence, the numbers of neurons in the closer layer to the output layer are smaller than those in the farther layer, where the distance is indicated by the number of edges from the output layer. Using this numbering, we set p-simpleces on V as

$$K_p^t = \begin{cases} V & (p = 0) \\ \{(v_{k_0}, \ldots, v_{k_p})\ ;\ v_{k_i} \in V, \widetilde{R_{k_i k_j}} \geq t \ \text{ for all } k_i > k_j\} & (p \geq 1), \end{cases} \tag{2.3}$$

where t is a threshold value ($0 \leq t \leq 1$).

Proposition 1. *Let $V = (v_0, \ldots, v_n)$ be a finite set, and $\{w_{ij}\}$ $(0 \leq i, j \leq n)$ be a set of real numbers. Let $\widetilde{R_{ij}}$ $(0 \leq i, j \leq n)$ be the relevance defined by Eqs. (2.1) and (2.2)*

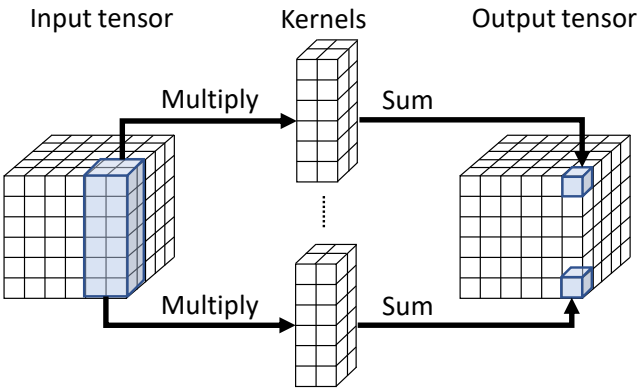


Figure 2.2: Overview of the convolution operation between input tensor and kernels to obtain output tensor.

using $\{w_{ij}\}$. *Let* K_p^t *be the p-simplices defined by Eq. (2.3), where t is a threshold value* ($0 \leq t \leq 1$). *Then, a finite collection of sets* $K^t = K_0^t \cup K_1^t \cup \cdots \cup K_n^t$ *is an abstract simplicial complex.*

Proof. Let $X = \{v_{X_0}, \ldots, v_{X_p}\}$ be an element of K^t. Then, $\widetilde{R_{X_i X_j}}$ is greater than or equal to t for all $X_i > X_j$. Let $Y = \{v_{Y_0}, \ldots, v_{Y_q}\}$ be a subset of X. Then, $\widetilde{R_{Y_i Y_j}}$ are greater than or equal to t for all $Y_i > Y_j$. Therefore, $X \in K^t$ and $Y \subseteq X$ imply $Y \in K^t$. □

Proposition 2. *Let* $(t_i)_{i=1}^n$ *be a monotonically decreasing sequence ranging from* 1 *to* 0. *Then,* $K_0 = \emptyset$ *and* $K_i = K^{t_i}$ $(1 \leq i \leq n)$ *form a filtration of* K^{t_n}.

Proof. $K_p^{t_k}$ is included in $K_p^{t_l}$ $(1 \geq t_k > t_l \geq 0)$ from Eq. (2.3). It implies $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n = K^{t_n}$. □

2.3.2 Convolution Layers

Eq. (2.1) defines the normalized relevance among neurons in the dense layer using network weights. Notably, for convolution layers, the output tensor is obtained using the convolution operation between the input tensor and the kernels. Fig. 2.2 provides an overview of the convolution operation. The input tensor values are multiplied by those in the kernels' values, and the multiplication results are added together for the output tensor values. The kernels associate the input tensor with the output tensor, similar to the network weights in the dense layers. Thus, we define the relevance among neurons using the kernel weights similar to Eq. (2.1).

Formally, the l^{th} convolution layer correlates the input neurons $a^{[l-1]}$ of the size of $(n_H^{[l-1]}, n_W^{[l-1]}, n_C^{[l-1]})$ to the output neurons $a^{[l]}$ of the size of $(n_H^{[l]}, n_W^{[l]}, n_C^{[l]})$ according to Eqs. (2.4) and (2.5).

$$a^{[l]} = [conv(a^{[l-1]}, K^{(1)}), conv(a^{[l-1]}, K^{(2)}), \ldots, conv(a^{[l-1]}, K^{(n_C^{[l]})})] \tag{2.4}$$

$$conv(a^{[l-1]}, K^{(n)})_{x,y} = \psi^{[l]}(\sum_{i=1}^{f^{[l]}} \sum_{j=1}^{f^{[l]}} \sum_{k=1}^{n_C^{[l-1]}} K_{i,j,k}^{(n)} a^{[l-1]}{}_{(x-1)s^{[l]}-p^{[l]}+i,(y-1)s^{[l]}-p^{[l]}+j,k} + b_n^{[l]}) \\ (1 \leq x \leq n_H^{[l]}, 1 \leq y \leq n_W^{[l]}), \tag{2.5}$$

where $n_C^{[l]}$ is the number of kernels; $K^{(n)}(1 \leq n \leq n_C^{[l]})$ are the kernels with the dimension $(f^{[l]}, f^{[l]}, n_C^{[l-1]})$; $p^{[l]}$ is the padding; $s^{[l]}$ is the stride; $b_n^{[l]}(1 \leq n \leq n_C^{[l]})$ are the bias; and $\psi^{[l]}$ is the activation function[2]. From Eqs. (2.4) and (2.5), the output neuron $a_{x,y,z}^{[l]}$ is influenced by the input neurons $a_{(x-1)s^{[l]}-p^{[l]}+i,(y-1)s^{[l]}-p^{[l]}+j,k}^{[l-1]}(1 \leq i, j \leq f^{[l]}, 1 \leq k \leq n_C^{[l-1]})$ with the coefficient of $K_{i,j,k}^{(z)}$.

We consider the correlation between two neurons $a_{X,Y,Z}^{[L]}$ and $a_{x,y,z}^{[l]}$, where $X = (x-1)s^{[l]} - p^{[l]} + i$ and $Y = (y-1)s^{[l]} - p^{[l]} + j$. Then, we obtain two equations $i = X - (x-1)s^{[l]} - p^{[l]}$ and $j = Y - (y-1)s^{[l]} - p^{[l]}$. Under these considerations, we define the normalized relevance between two neurons $a_{X,Y,Z}^{[L]}$ and $a_{x,y,z}^{[l]}$ in convolution layers using the normalization factor $N^{(z)+} = \sum_{i=1}^{f^{[l]}} \sum_{j=1}^{f^{[l]}} \sum_{k=1}^{n_C^{[l-1]}} K_{i,j,k}^{(z)+}$,

$$R_{(L,X,Y,Z),(l,x,y,z)} = \begin{cases} 1 & ((L,X,Y,Z) = (l,x,y,z)) \\ K_{X-(x-1)s^{[l]}+p^{[l]},Y-(y-1)s^{[l]}+p^{[l]},Z}^{(z)+} / N^{(z)+} & \\ \quad (L = l-1, 1 \leq Z \leq n_C^{[l-1]}, & \\ \quad 1 \leq X-(x-1)s^{[l]}+p^{[l]} \leq f^{[l]}, & \\ \quad 1 \leq Y-(y-1)s^{[l]}+p^{[l]} \leq f^{[l]}) & \\ 0 & (otherwise), \end{cases} \tag{2.6}$$

where $K^+ = \max\{0, K\}$ is the positive part of K.

2.3.3 Pooling Layers

Pooling layers are important in CNNs because they attempt to downsample the features of the input. Many functions, e.g., maximum and average functions, are used for the

[2]We used the notation from `https://towardsdatascience.com/convolutional-neural-networks-mathematics-1beb3e6447c0` with modifications based on our understanding.

downsampling, but the input neurons correlate equally with the output neurons in these functions. Thus, we define the relevance in pooling layers based on the size of the pooling filter.

Here, using Eq. (2.7), the l^{th} convolution layer correlates the input neurons $a^{[l-1]}$ with the size of $(n_H^{[l-1]}, n_W^{[l-1]}, n_C^{[l-1]})$ to the output neurons $a^{[l]}$ of the size of $(n_H^{[l]}, n_W^{[l]}, n_C^{[l]})$.

$$a^{[l]}_{x,y,z} = pool(a^{[l-1]})_{x,y,z} = \phi((a^{[l-1]}_{(x-1)s^{[l]}-p^{[l]}+i,(y-1)s^{[l]}+p^{[l]}+j,z})_{(1\leq i,j\leq f^{[l]})}), \tag{2.7}$$

where ϕ is the pooling function; $f^{[l]}$ is the size of the pooling filter; $s^{[l]}$ is the stride; and $p^{[l]}$ is the padding[1]. From Eq. (2.7), we define the normalized relevance between two neurons $a^{[L]}_{X,Y,Z}$ and $a^{[l]}_{x,y,z}$ in pooling layers,

$$R_{(L,X,Y,Z),(l,x,y,z)} = \begin{cases} 1 & ((L,X,Y,Z) = (l,x,y,z)) \\ 1/(f^{[l]})^2 & \\ & (L = l-1, Z = z, \\ & 1 \leq X-(x-1)s^{[l]}+p^{[l]} \leq f^{[l]}, \\ & 1 \leq Y-(y-1)s^{[l]}+p^{[l]} \leq f^{[l]}) \\ 0 & (otherwise). \end{cases} \tag{2.8}$$

2.3.4 Clique Complexes on DNNs

We consider sequential DNNs in which all neurons only connect with the neurons in the consecutive layers. Sequential DNNs can be considered a finite directed weighted graph with no self-loops and double edges, where the neurons and the connections between them are represented by vertexes and edges, respectively.

Proposition 1 assumes Eqs. (2.1) and (2.2) designed for dense layers. However, the proof of Proposition 1 does not employ Eqs. (2.1) and (2.2). Then, Proposition 1 is generalized to Proposition 3, which requires no modification in the proof.

Proposition 3. *Let $V = (v_0, \ldots, v_n)$ be a finite set, and $\widetilde{R_{ij}}$ $(0 \leq i, j \leq n)$ be a set of real numbers. Let K_p^t be the p-simplices defined by Eq. (2.3), where t is a threshold value ($0 \leq t \leq 1$). Then, a finite collection of sets $K^t = K_0^t \cup K_1^t \cup \cdots \cup K_n^t$ is an abstract simplicial complex.*

We used Eqs. (2.1), (2.6), and (2.8) to define the normalized relevance R in dense, convolution, and pooling layers, respectively. Additionally, we used Eq. (2.2) to define

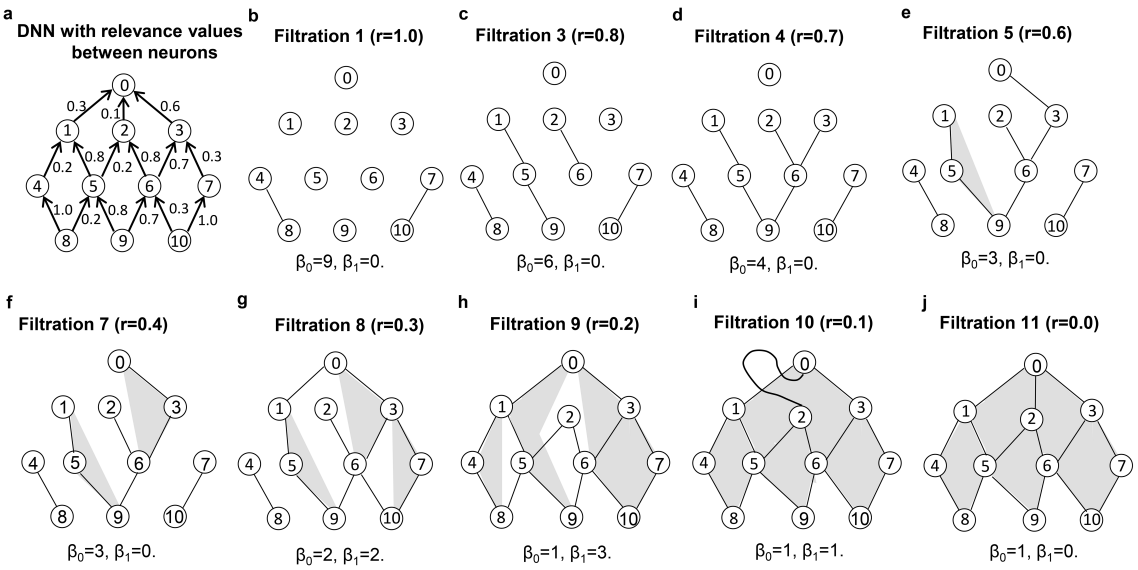


Figure 2.3: (a) Example of DNN with weights; (b–h) simplicial complexes and betti numbers corresponding to the filtration.

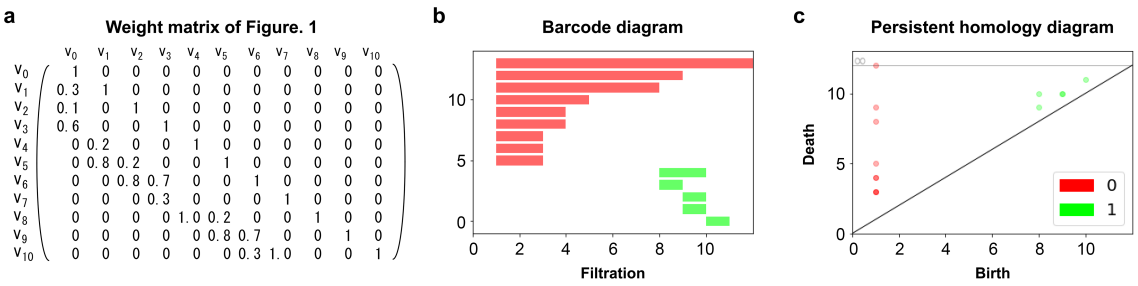


Figure 2.4: (a) Weight matrix of Fig. 2(a); (b,c) barcode and PH diagrams illustrated using GUDHI library.

$\widetilde{R}$ for all layers, which defines a set of real numbers between neurons in CNNs. Thus, an abstract simplicial complex K^t is defined on CNNs as shown in Proposition 3.

2.3.5 Example of Clique Complex on DNN

Fig. 2.3(a) illustrates a four-layered DNN with an output neuron v_0. The values adjacent to the arrows denote the weight between two neurons, and the weight matrix is presented in Fig. 2.4(a) where the (i,j) element denotes the weight between the i-th and j-th neurons. Fig. 2.3(b) illustrates the simplicial complex of $K_{r=1.0}$ with Betti number $\beta_0 = 9$. The decrease of the Betti number β_0 according to the filtration can be observed in Fig. 2.3(c) to (h). Fig. 2.3(e) illustrates a 2-simplex represented with the gray triangle.

Fig. 2.3(g) and 2.3(h) illustrate the increase of the Betti number β_1 corresponding to the occurrences of the cycle. If the vertices representing the features of input images are

connected straightforwardly to the output neurons, the knowledge in the DNN is considered to be simple because it is equivalent to feature detection. In contrast, the increase of the Betti number β_1 indicates that the DNN classifies the input based on the combination of features. From these viewpoints, we can assume the increase in the Betti number β_1 reflects the complexity of knowledge in the DNN. Filtration 10 (Fig. 2.3(i)) has Betti number $\beta_1 = 1$. While [0, 2] is a simplex in Filtration 10, it is not included in another simplex [0,...,10] and produces $\beta_1 = 1$.

The computation of PH involves the explosion of the complexity caused by the increase of vertices, several implementations of which are publicly available [61]. We employed the GUDHI [9, 53], JavaPlex [78], and Dionysus 2 [25, 57] libraries for the computation and visualization. These libraries require registering simplexes in each filtration to calculate PH.

Algorithm 1 identifies all simplexes from a vertex s up to the limit of the threshold of relevance t using the recursive procedure call. All simplexes in each filtration are identified using this procedure and registered to the libraries. Fig. 2.4(b) and (c) are barcode and PH diagrams illustrated by the GUDHI library, respectively. The library employed red and green for indicating zero- and one-dimensional homologies, respectively. The Betti numbers in Fig. 2.4(b) correspond to the number of the intersections between the bars and the perpendicular lines to the X-axis (remembering that the lifetime of homologies is defined by the half-open interval $[birth, death)$). The GUDHI library illustrates Betti numbers using color shades in PH diagrams shown in Fig. 2.4(c). PH was calculated using the Dionysus 2 and JavaPlex libraries, resulting in the same diagrams.

In each layer of DNNs, Algorithm 1 executes the recursive procedure call. When the DNN comprises L layers with N_i neurons in the i^{th} layer, the maximum number of the recursive procedure call is $\prod_{i=1}^{L} N_i$. The recursive procedure call is discontinued when the relevance value between two neurons reaches the threshold value. However, the computational complexity of Algorithm 1 can reach $\mathcal{O}(\prod_{i=1}^{L} N_i)$.

The recursive procedure call imposes a high computational overhead. In Chapter 5, a stack-based algorithm is developed to reduce the computational overhead. Furthermore, in Chapter 4, an algorithm is proposed for enhancing the computation efficiency

Algorithm 1 Algorithm for obtaining simplexes from a vertex s using a threshold t

procedure GETSIMPLEX(M, s, t) ▷ where M: $n \times n$-matrix, s: array, t: threshold
  $relevance \leftarrow 1.0$, $result \leftarrow \emptyset$, $origin \leftarrow s[0]$
  for $dest = s[0]$ to $s[|s|-1]$ **do** ▷ calculate the relevance from $s[0]$ to $s[|s|-1]$.
    $relevance \leftarrow relevance \times M[origin][dest]$ ▷ $s[|s|-1]$ is the last element of s.
    $origin \leftarrow dest$
  end for
  if $relevance \geq t$ **then**
    $result.append(combination(s))$ ▷ append all the combinations of the elements in s.
    $lastPoint \leftarrow s[|s|-1]$
    for $i = 0$ to $n-1$ **do** ▷ check if the last point has connections.
      if $M[lastPoint][i] > 0$ and $i \neq lastPoint$ **then**
        $ss \leftarrow$deep copy of s
        $recResult \leftarrow getSimplex(M, ss.append(i), t)$ ▷ recursive call with extended array.
        for e in $recResult$ **do**
          $result.append(combination(e))$ ▷ append all the combinations of the elements in e.
        end for
      end if
    end for
  end ifreturn $unique(result)$ ▷ return deduplicated array
end procedure

by limiting the number of DNN layers to three. The results obtained using the proposed algorithms are identical, except for the duplications and the order of outputs.

2.4 Discussion

In this section, the assumptions used in this study are explained and the application of the topological measurement of DNNs is discussed.

2.4.1 Assumptions

The assumptions of this study include the follows: (1) the knowledge in DNNs can be investigated from their network weights among neurons and (2) PH reveals the knowledge complexity of DNNs. The first assumption is acceptable because the weights are the outcome of the training process. The second assumption is based on the observations from previous works described in Sec.1.2 [14, 46]. PH reveals the births and deaths of feature combinations, which are difficult to be captured without using PH. The effectiveness of

the second assumption can be evaluated from the usability, which changes depending on the application.

2.4.2 Applications

One of the most important applications of the proposed method is recognizing the quality of DNN training. In particular, the performance of DNNs can deteriorate for many reasons, including a shortage of data, overfitting, and improper hyper-parameter setting [5, 77].

In Chapter 3, we provide evaluation results of the proposed method, in which the excess of the output neurons produces homologies near the dialog line. These results imply that the shortage of data can be indicated by the PH. Furthermore, we propose an overfitting measure in Chapter 4 for selecting appropriate DNN architectures, which is one of the major challenges when utilizing DNNs [30, 59].

2.5 Related Work

Bianchini et al. investigated the upper and lower bounds of network complexity from the viewpoint of topological concepts [6]. We addressed the inner representations of DNNs with small perturbations. Our evaluation results revealed that small perturbations such as the number of output neurons and a variety of input data have significant impact on PH. Thus, the sensitivity of PH requires a careful investigation for securing comparability.

Rieck et al. investigated the complexity of the inner representation of DNNs using zero-dimensional PH [69]. Zero-dimensional PH counts the number of connected components in DNNs. Fig. 2.3(f) and (g) have $\beta_0 = 3$ and $\beta_0 = 2$ corresponding to the connected components, respectively. In contrast, the Betti number β_1 reveals the combinations among neurons illustrated in Fig. 2.3(g), where the neurons one and three collaborate to increase the Betti number β_1. Thus, we believe that one-dimensional PH can reveal the combination of neurons and access essential aspects of DNNs that are difficult to be accessed using other methods.

2.6 Conclusion

This chapter provided a foundation for studying on the inner representation of DNNs using PH by proposing a construction method of clique complex and formalizing PH calculation on DNNs. The normalization and propagation techniques were introduced for constructing clique complex on DNNs, where mathematical proof was presented to show the construction method's correctness. The formalization was achieved in dense, convolutional, and pooling layers, which are prevalent in many DNN applications. An example of a clique complex on a DNN was presented to illustrate the PH calculation. An algorithm was also provided for obtaining simplexes from complexes, which enables us to utilize the high-performance implementation of public libraries in calculating the PH of DNNs.

Chapter 3

Changes in PH Involved in Different Network Parameters of DNNs

3.1 Introduction

This study assumes that the inner representation of DNNs affects the PHs obtained from the DNNs. For confirming the assumption, DNNs are developed varying the difficulty of problems for which the DNNs trained. The difficulty of problems is adjusted by reducing the number of classes that the DNNs learn. DNNs are also developed varying the number of neurons in the DNNs. The changes of PH are illustrated with PH diagrams for overviewing the number and stability of homologies in DNNs.

Evaluations are conducted using fully connected networks (FCNs) and networks combining FCNs and convolutional neural networks (CNNs) trained on the MNIST and CIFAR-10 datasets. Evaluation results demonstrate that the PH of DNNs reflects both the excess of neurons and problem difficulty, making PH one of the prominent methods for investigating the inner representation of DNNs.

Additionally, the robustness of PH is investigated in this chapter. We conduct each experiment 10 times with 30 different settings using random initial values of network weights, resulting in a total of 300 experiments. These experiments reveal that the results obtained from the investigation using PH are robust with the network's settings and initial weights.

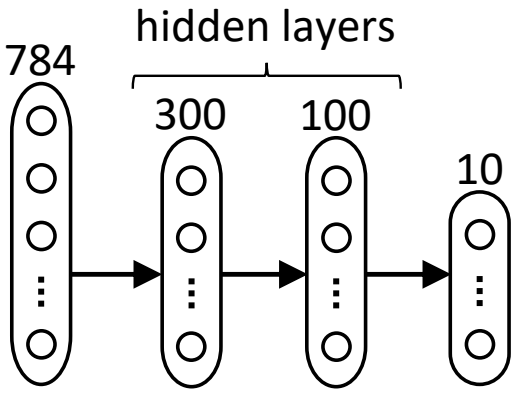


Figure 3.1: DNN for handwritten number recognition.

Table 3.1: Overview of the data sets and network types employed in this study

Data set	Content	Data size	Network type
MNIST	handwritten digits	784 (28×28 grayscale)	FCN
CIFER-10	photographs	3072 (32×32 color)	CNN, FCN

3.2 Changes in PH Involved in Different Network Parameters

3.2.1 Datasets and Setting of DNNs

The MNIST and CIFAR-10 data sets were employed in the evaluation [42,47]. As shown in Table 3.1, the contents of the MNIST and CIFAR-10 data sets are 28×28 grayscale handwritten digits and 32×32 color photographs, respectively. The CIFAR-10 data set comprises the photographs of 10 types of objects such as airplanes, automobiles, birds, etc. All experiments were conducted using Keras and Tensorflow [1,14], and DNNs were developed based on the examples in Keras 2.3.0 [1].

For the classification of the MNIST data set, we employed an FCN with two hidden layers of sizes 300 and 100, the ReLU activation function in the hidden layers and 10 output neurons with the sigmoid activation function (Fig. 3.1). The models were traind for 10 epochs with a batch size of 64, and all models achieved an accuracy of over 97% on the test data.

[1]The source code used in the evaluation can be accessed at `https://github.com/satoru-watanabe-aw/DNNtopology`.

For the classification of the CIFAR-10 data set, we employed DNNs consisting of a CNN and an FCN. The CNN was used to extract features from the photographs, while the FCN was used to classify the photographs based on the combination of the features. The proposed method was applied to the FCN since the purpose of this study was to examine the complexity of the knowledge in DNNs represented in the combination of features.

We employed the CNN from an example network included in Keras 2.3.0 without modifications. This CNN comprises multiple layers, including two-dimensional convolution, max pooling, and dropout layers. Two FCNs with sizes of (300, 100, 10) and (512, 512, 10) were used for examining the sensitivity of the proposed method to the network structures[2]. The DNNs were trained for 30 epochs with a batch size of 32.

3.2.2 Resolution of Persistent Homology

PH investigates simplicial complexes at different resolutions, which are defined by filtration. Filtration can be set automatically by libraries [61] that adjust the resolution according to the simplicial complexes. However, this adjustment makes it difficult to compare the PH diagrams among the evaluation results. Thus, we used a fixed resolution in this study.

From Proposition 2, a filtration of K^{t_n} is defined by a monotonically decreasing sequence $(t_i)_{i=1}^n$ ranging from 1 to 0. Here, we employ the sequence $(t_i)_{i=1}^n (1 \leq n \leq 64)$ defined as $t_i = (1 - 0.1 \times (l-1)) \times 10^{-m}$, where m and l are the quotient and remainder, respectively, when i is divided by 9. The sequence $(t_i)_{i=1}^n$ defines a filtration $\emptyset = K_0 \subseteq K^{t_1=1.0} \subseteq K^{t_2=0.9} \subseteq \cdots \subseteq K^{t_{10}=1.0^{-1}} \subseteq K^{t_{11}=0.09} \subseteq \cdots \subseteq K^{t_{64}=1.0^{-7}}$. Note that this setting is consistent throughout this thesis.

[2]The following network structures are employed: input(3072)–Conv2D(32 filters, 3×3 kernel, ReLu activation)–Conv2D(32 filters, 3×3 kernel, ReLu activation)–MaxPooling2D(2×2 pool)–Dropout(dropout ratio 0.25)–Conv2D(64 filters, 3×3 kernel, ReLu activation)–Conv2D(64 filters, 3×3 kernel, ReLu activation)–MaxPooling2D(2×2 pool)–Dropout(dropout ratio 0.25)–Flatten–Dense(300 or 512, ReLu activation)–Dropout(dropout ratio 0.5)–Dense(100 or 512, ReLu activation)–Dense(10, softmax activation).

3.3 Evaluation Results

3.3.1 MNIST Dataset

Figs. 3.2(a–j) illustrate PH diagrams of the FCNs produced using the Dionysus 2 library, where the number of input digits used to train the FCN models was varied. In particular, we extracted the images of the target digits from the MNIST data set and trained FCN models using the images of digits 0–9 (Fig. 3.2(a)), digits 0–8 (Fig. 3.2(b)), and so on. The Dionysus 2 library allows to visualize the overlapping quantity of homologies using different colors as indicated by the legends in Fig. 3.2. The values of birth and death in the axes on PH diagrams indicate the order of the 64 threshold values defined in Section 2.3. Let m and l are the quotient and remainder when the values of birth and death are divided by 9, respectively, the threshold values corresponding to the values in the axes on PH diagrams are $(1-0.1\times(l-1))\times 10^{-m}$. This correspondence is consistent through the paper.

The following three observations can be made from Figs. 3.2(a–j): (1) points are plotted in the belt-like area ($birth+5<death<birth+20$) parallel to the dialog line; (2) some figures have points below the belt-like area; and (3) some figures have points over the belt-like area.

With respect to observation (2), the number of points below the belt-like area increases from Fig. 3.2(a) to Fig. 3.2(g) and decreases from Fig. 3.2(h) to Fig. 3.2(j). This pattern reflects both the excess of the output neurons and problem difficulty. It can be further observed that the diagrams seem to reflect the degree of confidence of the FCN models, i.e., the excess of the output neurons reduced the confidence, whereas the simplicity of the problem increases it. For further investigation, we classified five digits using five output neurons (Fig. 3.2(k)) and 10 digits using 20 output neurons (Fig. 3.2(l)). In contrast to Fig. 3.2(f), the points below the belt-like area disappeared in Fig. 3.2(k). The opposite can be observed in Figs. 3.2(a) and 3.2(l).

Table 3.2 lists the number of points plotted in Fig. 3.2(a–e), 3.2(i), and 3.2(j). We categorized the points using the representative cycles calculated by the JavaPlex based on the following two conditions: (c1) the homology includes unused output neurons and (c2)

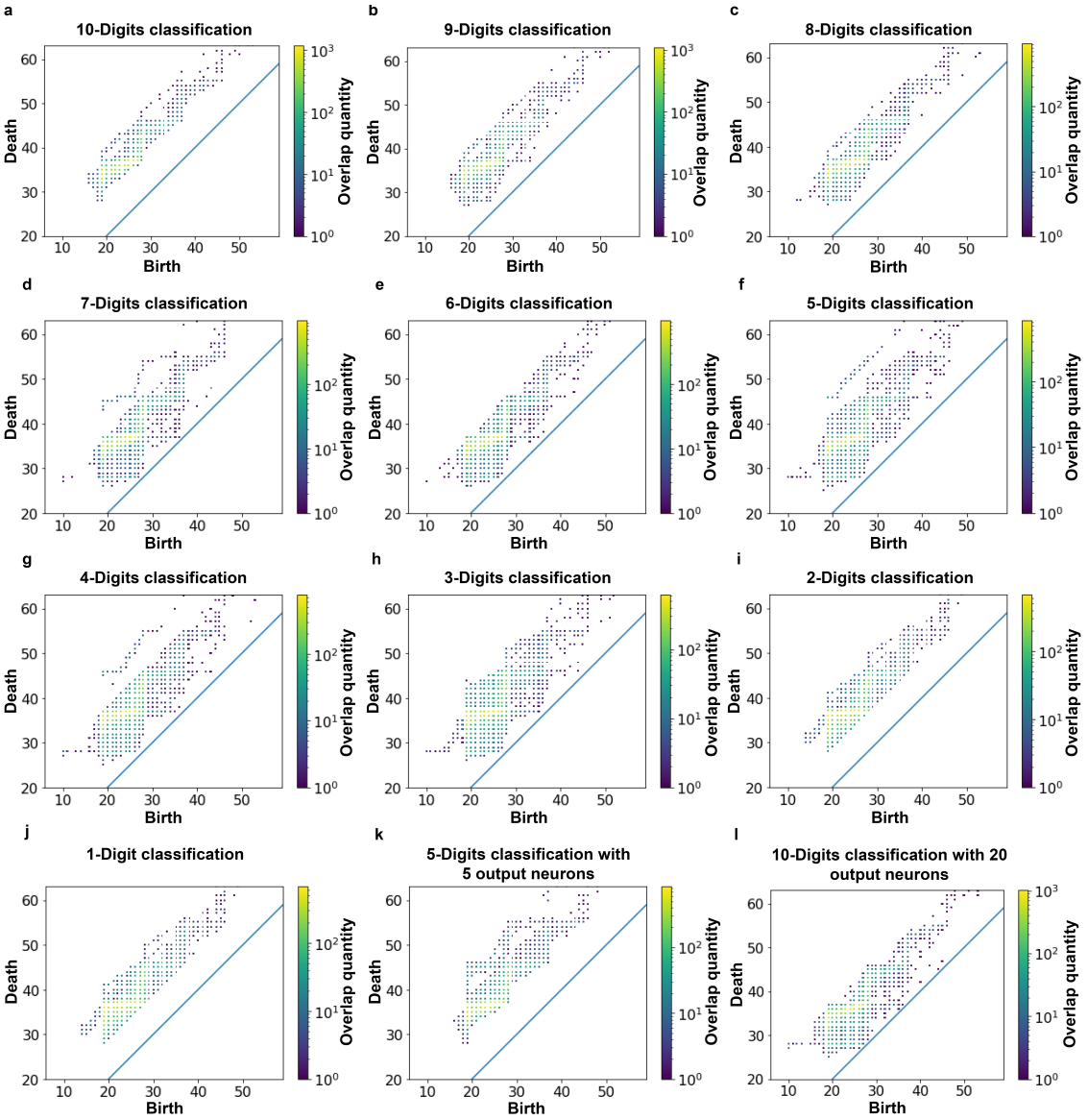


Figure 3.2: (a–j) PH diagrams of the FNC models trained to classify handwritten digits based on a varying number of input digits from 10 to 1; (k) persistent diagram of the FCN model trained to classify five digits using five output neurons; (l) persistent diagram of the FCN model trained to classify 10 digits using 20 output neurons

the points are under the belt-like area ($death \leq birth + 5$). While the number of points that include unused output neurons in Figs. 3.2(i) and 3.2(j) is more than twice of that in Fig. 3.2(e), these points are not plotted below the belt-like area. The simplicity of the problem led to no points being plotted under the belt-like area.

3.3.2 CIFAR-10 Dataset

Figs.3.3(a–j) illustrate PH diagrams of the DNN models combining a CNN and an FCN (300, 100, 10), where the number of classes used to train the models was varied. In

Table 3.2: Number of points in Figs. 3.2(a–e), 3.2(i), and 3.2(j)

	(a)	(b)	(c)	(d)	(e)	(i)	(j)
Total number	16,420	16,399	16,150	16,222	16,133	15,857	15,531
(c1)	N/A	1,317	2,034	1,700	2,972	8,226	13,123
(c2)	0	45	26	254	273	0	0
(c1) and (c2)	N/A	45	26	254	40	0	0

particular, we extracted photographs of the target classes from the CIFAR-10 data set and trained the DNN models using the photographs of 10 classes (Fig. 3.3(a)), nine classes (Fig.3.3(b)), and so on.

As described in Section 3.2, the contents of the CIFAR-10 data set differs from that of the MNIST data set in terms of the image size, tone, and represented object. Unlike FCN-based models traind on the MNIST data set, CNNs were employed in addition to FCNs to classify the CIFAR-10 data set.

Despite these differences, Figs. 3.3 demonstrate similar patterns to those in Figs. 3.2. In particular, the points under the belt-like area appear only in Figs. 3.3(d–h); Fig. 3.3(k), where the photographs of five classes are classified using five output neurons, has no points under the belt-like area, whereas Fig. 3.3(l), where the photographs of 10 classes are classified using 20 output neurons, has points under the belt-like area.

A further experiment was conducted using the DNN models combining a CNN and an FCN (512, 512, 10). The results of this experiment are illustrated in Figs. 3.4. A similar patterns regarding the appearance and disappearance of points under the belt-like area can be observed from Fig. 3.4; that is, only Figs. 3.4(d–h) and 3.4(l) have the points under the belt-like area. This result suggests that the observation is robust to not only the network type and content of data sets but also number of neurons in FCNs.

Two additional observations can be made from Fig. 3.3 and 3.4: (i) the numbers of points in Figs. 3.4 are larger than those in Figs. 3.3; (ii) the sizes of the areas that points are plotted in Figs. 3.4 are larger than those in Figs. 3.3. Tables 3.3 and 3.4 list the numbers of points and sizes of the convex hull of the points plotted in Fig. 3.3(a)–(j) and 3.4(a)–(j), respectively. The numbers of points in Fig. 3.4 are 8.81 to 9.31 times larger than those in Fig. 3.3. The sizes of the convex hulls in Fig. 3.4 are 1.05 to 2.57 times larger than those in Fig. 3.3.

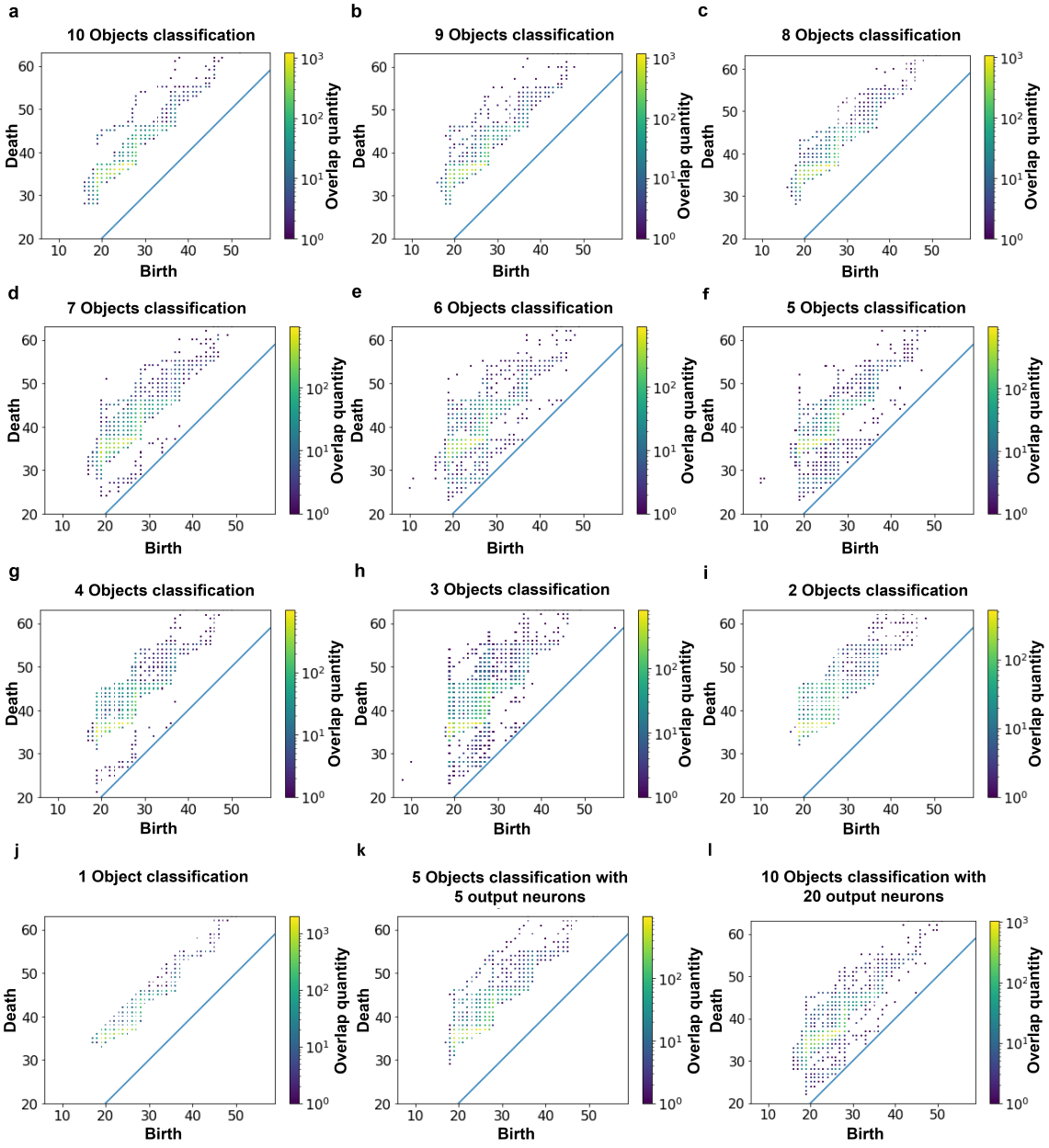


Figure 3.3: (a–j) PH diagrams of the DNNs using the FCN (300, 100, 10) trained to classify photographs based on a varying number of input classes from 10 to 1; (k) PH diagram of the DNN using the FCN (300, 100, 10) trained to classify five classes using five output neurons; (l) PH diagram of the DNN using the FCN (300, 100, 10) trained to classify 10 classes using 20 output neurons

Table 3.3: Number of points in Figs.3.3(a–j) and 3.4(a–j)

	(A) Fig. 3.3: FCN (300, 100, 10)	(B) Fig. 3.4: FCN (512, 512, 10)	(B) / (A)
(a)	16,214	142,768	8.81
(b)	16,278	139,783	8.59
(c)	15,702	142,016	9.04
(d)	15,421	141,027	9.15
(e)	15,274	138,732	9.08
(f)	15,759	136,508	8.66
(g)	14,878	133,503	8.97
(h)	14,348	124,919	8.71
(i)	11,496	106,983	9.31
(j)	15,073	132,775	8.81

The number of points reflects the difference of expressiveness of the FCN (512, 512, 10) and FCN (300, 100, 10). The FCN (512, 512, 10) has more parameters compared to the FCN (300, 100, 10), which results in the ability of the FCN (512, 512, 10) to learn knowledge is higher than that of the the FCN (300, 100, 10) and produces many homologies. As a rough approximation, the FCN (512,512,10) has $512 \times 512 + 512 \times 10$ of weight parameters, whereas the FCN (300, 100, 10) has $300 \times 100 + 100 \times 10$ of them. The ratio 8.62 ($= (512 \times 512 + 512 \times 10)/(300 \times 100 + 100 \times 10)$) provides the explanation for the increase in the values listed in Table 3.3.

The increase in the size of convex hull is smaller than that of the number of points, which indicates that the FCNs (512, 512, 10) have duplicated homologies approximately 4 to 8 times more often compared to the FCNs (300, 100, 10). It implies that the FCNs (512, 512, 10) have duplicated homologies with different neurons, which can be achieved with expressive training to the data set. The interpretation of the PH diagrams requires further investigation, which we left as a task for future work because the purpose of this study was only to examine the prominence of the topological measurement of DNNs.

3.4 Robustness on Weight Initialization

We conducted additional experiments by varying the initial values of network weights to investigate the robustness of the PH diagrams' transitions described in Subsections 3.3.1 and 4.3. Keras framework starts the training with random initial values of network weights

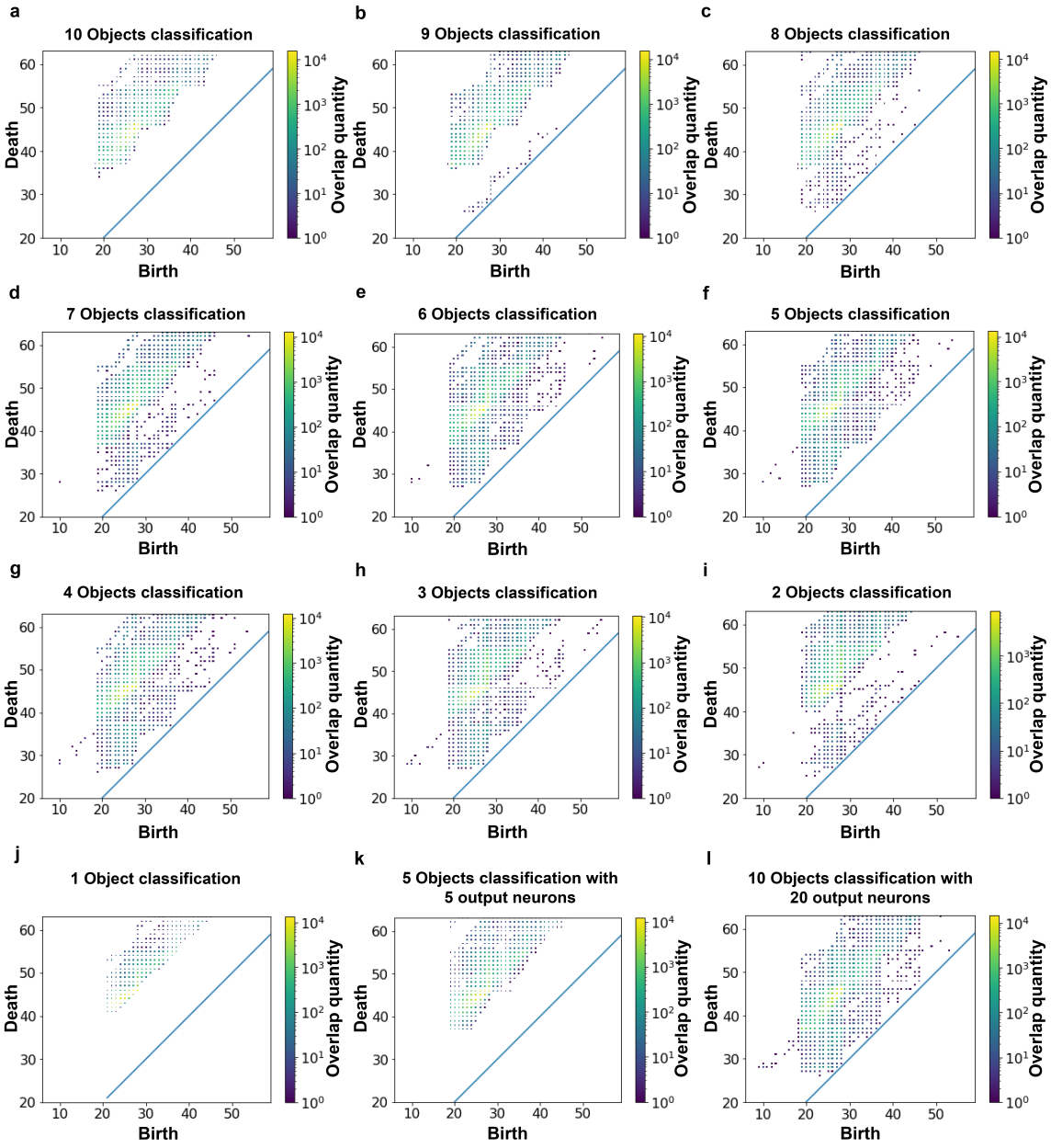


Figure 3.4: (a–j) PH diagrams of the DNN using the FCN (512, 512, 10) trained to classify photographs based on a varying number of input classes from 10 to 1; (k) PH diagram of DNN using the FCN (512, 512, 10) trained to classify five classes using five output neurons; (l) PH diagram of DNN using the FCN (512, 512, 10) trained to classify 10 classes using 20 output neurons

Table 3.4: Size of the convex hull in Figs. 3.3(a–j) and 3.4(a–j)

	(A) Fig. 3.3: FCN (300, 100, 10)	(B) Fig. 3.4: FCN (512, 512, 10)	(B) / (A)
(a)	445.5	492.5	1.11
(b)	477.0	737.5	1.55
(c)	406.0	881.0	2.17
(d)	710.5	1029.5	1.45
(e)	823.0	959.5	1.17
(f)	836.0	904.8	1.08
(g)	634.5	964.5	1.52
(h)	992.0	1041.5	1.05
(i)	413.5	1061.0	2.57
(j)	232.5	254.0	1.09

[14]. We repeated each experiment 10 times by varying the number of input classes from 10 to 1 with the three network types, MNIST $(300, 100, 100)$, CIFAR-10 $(300, 100, 100)$, and CIFAR-10 $(512, 512, 10)$, resulting in a total of 300 additional experiments.

Fig. 3.5 shows the minimum, average, and maximum size of convex hulls of the points in the PH diagrams. The differences between the maximum and minimum values indicate the degree of vibration of the experiment results. All the three graphs are approximately convex upward, indicating that the PH diagrams transit the shape in a similar manner to those described in Subsections 3.3.1 and 4.3, and the transitions are robust on the initial values of network weights.

In Subsections 3.3.1 and 4.3, we observed the transition of the PH diagrams that the number of points near the dialog line ($death \leq birth + 5$) changes by varying the number of input classes. No point near the dialog line appeared when the number of input classes was set to 10 and 1. Additionally, the number of points near the dialog line increased and decreased with the decrease in the number of input classes from 10 to 8 and 3 to 1, respectively.

Table 3.5 lists the minimum, average, and maximum numbers of points near the dialog line regarding the additional experiments. We observed that no point appeared near the dialog line when the number of input classes was set to 10 and 1 in all the additional experiments. Additionally, the increase and decrease followed the same trend in the additional experiments, shown in Table 3.5, meaning that the observations obtained in Subsections

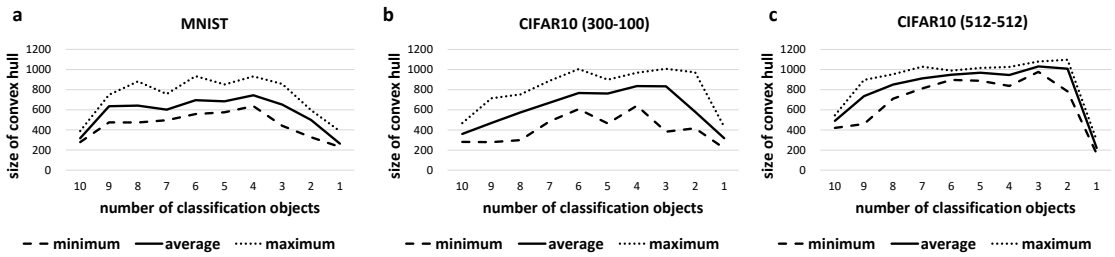


Figure 3.5: (a)–(c) Size of the convex hull of points in the PH diagrams with MNIST using the FCN $(300, 100, 10)$, CIFAR-10 using the FCN $(300, 100, 10)$,and CIFAR-10 using FCN $(512, 512, 10)$ by varying the number of input classes, respectively

Table 3.5: Number of points near the dialog line ($death \leq birth + 5$)

Number of	MNIST			CIFAR-10 (300-100)			CIFAR-10 (512-512)		
input classes	min.	avg.	max.	min.	avg.	max.	min.	avg.	max.
10	0	0	0	0	0	0	0	0	0
9	57	96	132	0	11	59	0	115	234
8	110	150	199	0	33	102	79	273	497
7	141	209	297	0	78	143	278	375	451
6	141	269	348	0	136	284	209	376	571
5	137	332	528	0	142	334	52	380	620
4	111	308	524	48	196	321	13	423	823
3	46	131	207	0	158	365	591	764	909
2	0	0	1	0	36	252	145	581	936
1	0	0	0	0	0	0	0	0	0

3.3.1 and 4.3 are robust on the initial values of network weights.

3.5 Conclusion

We evaluated the changes of PH involved in different network parameters of DNNs. The evaluation results demonstrated that the one-dimensional PH of DNNs can reflect both the excess of neurons and problem difficulty, which implies that PH can become one of the prominent methods for investigating the inner representation of DNNs.

The investigation about the robustness of PH was also conducted in this chapter. Additional experiments revealed that the results obtained from the investigation using PH are robust with the network's settings and initial weights.

Chapter 4

Overfitting Measurement of Deep Neural Networks Using Training Network Weights

Overfitting reduces the generalizability of deep neural networks (DNNs). Overfitting is generally detected by comparing the accuracies and losses of the training and validation data, where the validation data are formed from a portion of the training data; however, detection methods are ineffective for pretrained networks distributed without the training data. Thus, in this paper, we propose a method to detect overfitting of DNNs using the trained network weights inspired by the dropout technique. The dropout technique has been employed to prevent DNNs from overfitting, where the neurons in the DNNs are invalidated randomly during their training. It has been hypothesized that this technique prevents DNNs from overfitting by restraining the co-adaptations among neurons, and this hypothesis implies that the overfitting of DNNs results from co-adaptations among neurons and can be detected by investigating the inner representation of DNNs. The proposed persistent homology-based overfitting measure (PHOM) method constructs clique complexes in DNNs using the trained network weights, and the one-dimensional persistent homology investigates co-adaptations among neurons. In addition, we enhance PHOM to normalized PHOM (NPHOM) to mitigate fluctuation in PHOM caused by the difference in network structures. The methods are applied to convolutional neural networks trained for the classification problems on the CIFAR-10, street view house number, Tiny ImageNet, and CIFAR-100 datasets. Experimental results demonstrate that PHOM and NPHOM can indicate the degree of overfitting of DNNs, which suggests that these

methods enable us to filter overfitted DNNs without requiring the training data.

4.1 Introduction

Overfitting is defined as "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably [22]." Overfitting is a major contributor to reduced generalizability of data analytics methods, including deep neural networks (DNNs).

Overfitting of DNNs is generally detected by comparing the accuracies and losses of the training and validation data, where a portion of the training data is used to form the validation data [7]. It is detected by the higher accuracy and lower loss to the training data compared with those to validation data. However, the detection method is ineffective for pretrained networks distributed without the associated training data. Due to the large amount of training data, trained networks are typically distributed without the training data to reduce the data handling costs.

Srivastava et al. previously proposed a dropout method that prevents DNNs from overfitting [77]. In this method, the neurons in the DNNs are invalidated randomly during their training. They hypothesized that "for each hidden unit, dropout prevents co-adaptation by making the presence of other hidden units unreliable." This hypothesis implies that the overfitting of DNNs is a result of co-adaptations among neurons and can be detected by investigating the inner representation of DNNs.

In this paper, as an enhancement of our previous work [83], we propose a persistent homology-based overfitting measure (PHOM) that uses trained network weights to measures the degree of overfitting of DNNs. We investigate the possibility of overfitting detection without relying on the training data. PHOM constructs clique complexes on trained DNNs using the network weights, kernel weights, and pooling sizes in the dense, convolution, and pooling layers, respectively. The one-dimensional PH investigates the co-adaptations among neurons from the viewpoint of the number and stability of synchronized neurons as described in Section 2.1.1 and 2.1.2; thus, PHOM does not require the training data to detect overfitting. In addition, we extend PHOM to normalized PHOM (NPHOM) to mitigate fluctuation in PHOM results caused by the difference in

network structures. The proposed methods are applied to convolutional neural networks (CNNs) trained for the classification problems in the CIFAR-10, street view house number (SVHN), Tiny ImageNet, and CIFAR-100 datasets, and we experimentally investigate the ability of PHOM and NPHOM to separate overfitted DNNs using no training data.

The investigation of this paper shows that the overfitting of DNNs can be detected via the inspection of the inner representation of trained DNNs. We demonstrate that the detection can be achieved with the assistance of topological data analysis.

The remainder of this paper is organized as follows. PHOM is proposed in Section 4.2. The experimental setup and results are described in Sections 4.3 and 4.4, respectively. Section 4.5 describes the fundamental analysis of PHOM, and Section 4.6 enhances PHOM to NPHOM. Section 4.7 discusses PHOM. Related work is described in Section 4.8. Finally, conclusions are presented in Section 4.9.

4.2 Persistent Homology-Based Overfitting Measurement

4.2.1 Definition of Overfitting Measure

The overfitting of DNNs is hypothesized as a result of the co-adaptation of neurons, which induces the synchronization among neurons. The synchronized neurons produce stable homologies, and the stable homologies are plotted at a distance from the dialog line in the PH diagram. Thus, it is assumed that the overfitting of DNNs results in the decrease of points near the dialog line.

To measure the degree of overfitting, we set a belt area in the PH diagram with the condition of $death \leq birth + 10$. Then, we defined the overfitting measure of DNNs by the number of points in the belt area. We employed a provisional threshold value of 10, considering other counting techniques as described in Section 4.7.4.

4.2.2 Computation Algorithm

DNNs can comprise millions of neurons, which makes it difficult to compute the PH of DNNs. To reduce this difficulty, we consider vertex-induced subgraphs of DNNs. A vertex-induced subgraph is a subset of the vertices of the graph together with any edges whose endpoints are present in the subset. Here, we select vertexes from DNNs as fol-

lows:

(i) select two successive layers from DNNs;

(ii) select input neurons from the first layer;

(iii) select neurons that are connecte to the input neurons.

The sequential nature of DNNs reduces the difficulty of the identification of simplexes because no vertex in simplexes has branches, as shown in Proposition 1. Based on Proposition 1, we identify all the simplexes in the subgraph using Algorithm 2. Here, there are two "loop in 1st layer"s to leverage the identification using the sparse matrix library provided in the SciPy [79].

Proposition 1. *Let K^t be an abstract simplicial complex in sequential DNNs as defined by Eqs. (2.2) and (2.3). Then, all simplexes will have no vertex that is a shared start point of multiple edges.*

Proof. Let $v = (v_0, \ldots, v_p) \in K^t$ is a simplex, and let v_0 be a shared start point of multiple edges. Then, the end points v_i and v_j belong to the next layer of v_0 due to the sequential nature of the DNN, i.e., $\widetilde{R_{ij}}$ is undefined because there is no path from v_i to v_j. □

4.3 Evaluation Setup

The CIFAR-10 and SVHN datasets were employed in our evaluations [42, 60]. The CIFAR-10 and SVHN datasets contain color photographs (32×32 pixels). The CIFAR-10 dataset contains images of 10 types of objects, e.g., airplanes, automobiles, and birds, with 50,000 and 10,000 images for training and testing, respectively. The SVHN dataset contains images of real-world house numbers obtained from Google Street View images. Here, we used "format two" in SVHN dataset, which was generated by cropping digits and resizing the images to a fixed resolution of 32×32 pixels.

Evaluations were conducted using a network with a combination of dense and convolutional layers (Fig. 4.1), which was developed based on the examples in Keras 2.3.0. Here, we varied the dropout rate in the DNNs to consider networks with and without

Algorithm 2 Algorithm for identifing all simplexes

1: **procedure** IDFYSIMPLEX($\{R_{ij}\}$, $(t_i)_{i=1}^{L}$, (N_1,N_2,N_3)) ▷ $\{R_{ij}\}$: $N \times N$ matrix defined in Eqs. (2.2), (2.5), and (2.8); $(t_i)_{i=1}^{L}$: threshold values; N_1,N_2,N_3: number of neurons in 1st, 2nd, and 3rd layers, respectively $(N = N_1+N_2+N_3)$.
2: **for** $i = 0$ to $N-1$ **do**
3: $simplex[1] \leftarrow \{(v_i)\}$ ▷ register 0-simplexes
4: **end for**
5: **for** $i = N_1+N_2$ to $N-1$ **do** ▷ loop in 3rd layer
6: **for** $j = N_1$ to N_1+N_2-1 **do** ▷ loop in 2nd layer
7: **if** $R_{ij} \neq 0$ and $i \neq j$ **then**
8: call setSimplex(R_{ij}, $\{(v_i,v_j)\}$)
9: **for** $k = 0$ to N_1-1 **do** ▷ loop in 1st layer
10: **if** $R_{jk} \neq 0$ and $j \neq k$ **then**
11: call setSimplex(R_{jk}, $\{(v_j,v_k)\}$)
12: **end if**
13: **end for**
14: **end if**
15: **end for**
16: **for** $j = 0$ to N_1-1 **do** ▷ loop in 1st layer
17: $\{M_k\}_{k=0}^{N-1} \leftarrow \{R_{ix}\}_{x=0}^{N-1} \odot \{R_{yj}\}_{y=0}^{N-1\intercal}$ ▷ $\odot$ and $\intercal$ denote Admiral product and transpose, respectively.
18: **for** $k = 0$ to $N-1$ **do**
19: **if** $M_k \neq 0$ **then**
20: call setSimplex(M_k, $\{(v_i,v_j)\}$)
21: call setSimplex(M_k, $\{(v_i,v_k,v_j)\}$)
22: **end if**
23: **end for**
24: **end for**
25: **end for**
26: **end procedure**
27: **procedure** SETSIMPLEX(R, e) ▷ R and e are the relevance value and simplex, respectively.
28: $p \leftarrow \min\{l; R \geq t_l \ (1 \leq l \leq L)\}$
29: $simplex[p] \leftarrow e$
30: **end procedure**

overfitting to the training data. The DNNs were trained with a batch size of 128 and 30 epochs, where 20% of the training data were used as the validation data.

We selected two target layers to apply PHOM, i.e., the 1st and 2nd groups in Fig. 4.1. For the 1st group, we selected all the neurons in the first dense layer as the input neurons (Section 4.2.2). For the 2nd group, we selected input neurons with a size of $3 \times 3 \times 32$ to reduce computation complexity. Here, the size corresponds to the kernel size in the convolution layer.

We also used the Tiny ImageNet dataset [19] and VGG16 network [74] in our evaluations. Tiny ImageNet dataset contains 64×64 color photographs of 200 types of objects with 100,000 training and 10,000 validation images. Keras provides a VGG16 network pretrained on ImageNet, which includes 13 convolutional and three dense layers [32]. We transferred the convolutional layers and appended two dense layers linking with 1,000 and 200 neurons, respectively (Fig. 4.2). The DNN was fine-tuned for the Tiny ImageNet dataset with a batch size of 128 and 30 epochs. Here, the network weights are frozen in the convolutional layers. Fine-tuning with frozen network weights is an efficient practice in transfer learning [88]. We applied PHOM to the dense layers, which are denoted as the 3rd group in Fig. 4.2. From this group, we selected 512 neurons in the first dense layer as the input neurons (Section 4.2.2). Note that all the results presented in Section 4.4 can be reproducible using the publicly available source code[1].

4.4 Evaluation Results

4.4.1 CIFAR-10 Dataset

The DNNs were trained varying dropout rate from 0.0 to 0.6 in 0.2 increments. PHOM was applied to the trained DNNs, and the results are shown in Figs. 4.3, where Figs. 4.3 (a)–(d) show the results of the dropout rates from 0.0, 0.2, 0, 4, and 0.6, respectively. The graphs of Fig. 4.3 (a-1)–(d-1) and (a-2)–(d-2) show the accuracy and loss of the training and validation data, respectively.

The graphs shown in Figs. 4.3 (a-1) and (a-2) suggest that the DNN overfits of the

[1]The source code and models used in the evaluation can be accessed at `https://github.com/satoru-watanabe-aw/phom/`.

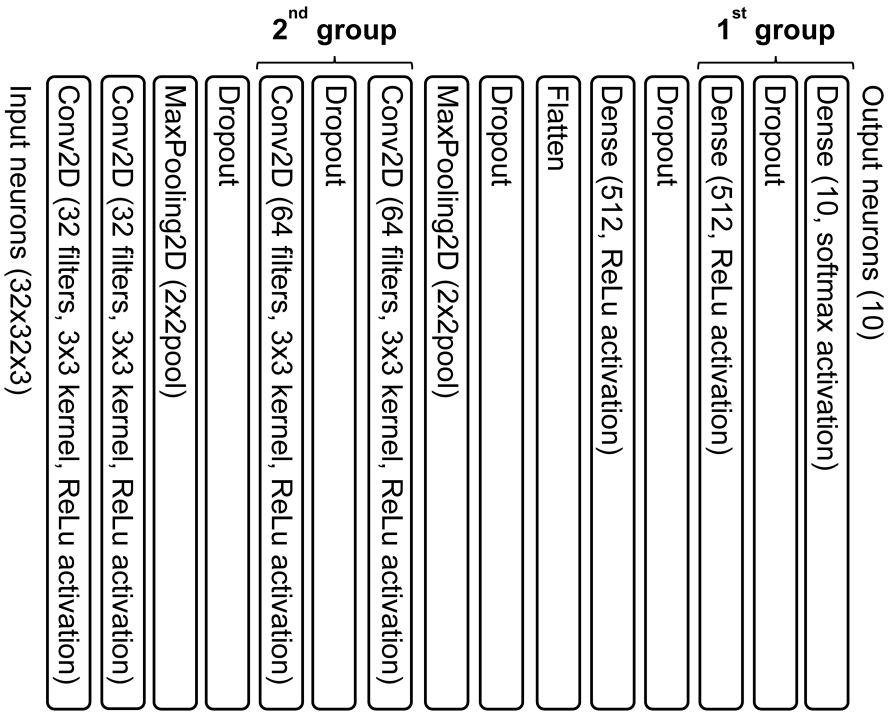


Figure 4.1: Network structure and layer groups used in the classification of the CIFAR-10 and SVHN datasets.

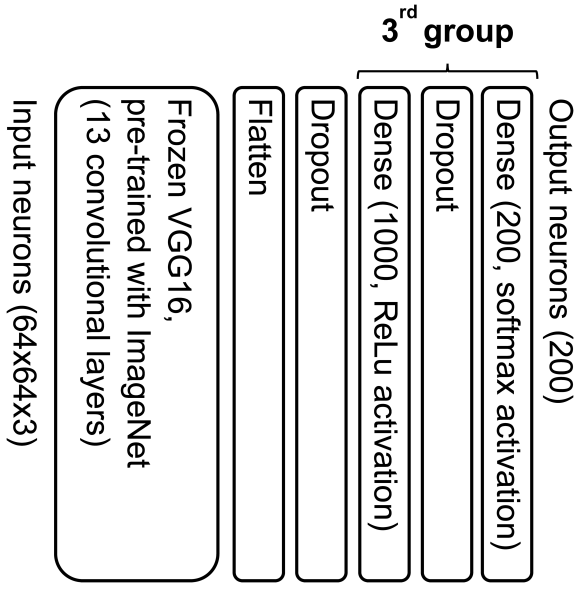


Figure 4.2: Network structure and layer group used in the classification of the Tiny ImageNet dataset.

training data, which is indicated by the observations that the accuracy values to the training data were higher than those of the validation data, whereas the loss values of the training data were less than those of the validation data. Overfitting of the DNNs was mitigated by the increase in dropout rate (Figs. 4.3 (a-1)–(d-1) and (a-2)–(d-2)). The graphs of Figs. (b-1) and (b-2) show that the DNN still overfitted to the training data, which is indicated by the differences between the measured values of the training and validation data. The DNN with a dropout rate of 0.6 underfitted to the training data, which is indicated by the graphs in Fig. 4.3 (d-1), where the accuracy values drop by over 10% from those in Fig. 4.3 (c-1). From the graphs in Fig. 4.3 (a-1)–(d-1) and (a-2)–(d-2), the DNN with a dropout rate of 0.4 was the best.

The PH diagrams of the 1st and 2nd groups are shown in Figs. 4.3 (a-3)–(d-3) and (a-4)–(d-4), respectively. The PH diagrams in Section 4.4 were drawn using the Dionysus library. For the PH diagrams of the 1st group, Fig. 4.3 (a-3) shows no point near the diagonal line, where the DNN overfitted to the training data. The number of points near the diagonal line increased according to the increase of the dropout rate. These observations imply that the degree of overfitting can be measured by the points near the diagonal line in the PH diagrams. For counting the number of points near the diagonal line, we defined a belt area in the PH diagram with the condition of $death \leq birth + 10$. Table 4.1 lists the number of points in the PH diagrams and that in the belt area, wherein the increase in the number of points in the belt area is presented explicitly as the rate of dropout increases.

For the PH diagrams of the 2nd group, the changes in Figs. 4.3 (a-4)–(d-4) were unclear compared to that of the 1st group, which means that Figs. 4.3 (a-4)–(d-4) had no point adjacent to the diagonal line. However, Table 4.1 shows the changes explicitly, where the number of points in the belt area increased with the increase of the dropout rate. Overfitting of the DNNs to the training data is indicated repeatedly by the PH diagrams for the 2nd group.

4.4.2 SVHN Dataset

We applied PHOM to the trained DNNs and varied the dropout rate from 0.0 to 0.6 in 0.2 increments. The results are shown in Fig. 4.4 similarly to Fig. 4.3. The DNN with

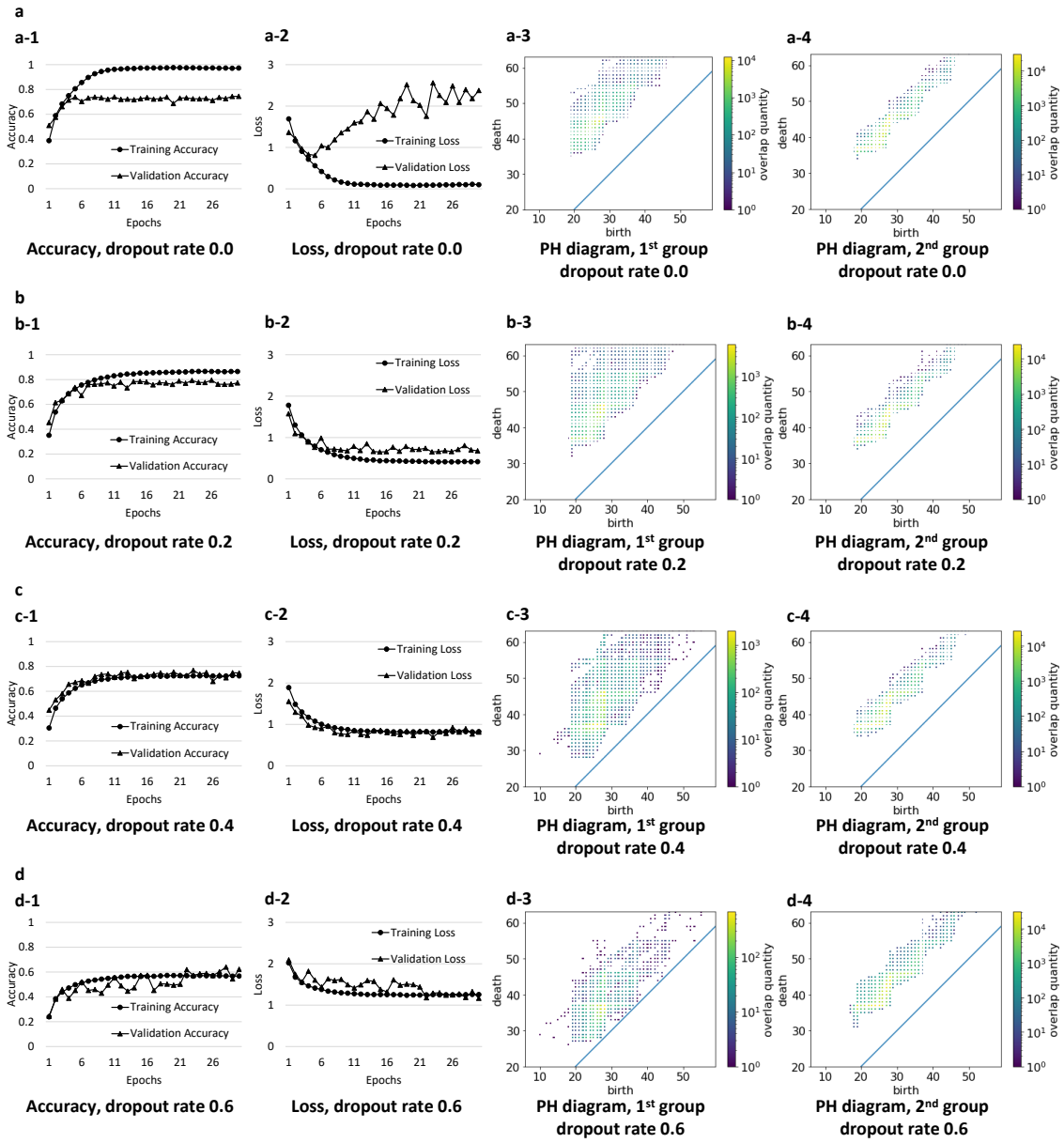


Figure 4.3: PH diagrams of trained DNNs with CIFAR-10 dataset; Figs. (a-1)–(d-1) show the accuracy values on the training and validation data; Figs. (a-2)–(d-2) show the loss values on the training and validation data; Figs. (a-3)–(d-3) show the PH diagrams of the 1st group; Figs. (a-4)–(d-4) show the PH diagrams of the 2nd group.

a dropout rate of 0.0 overfitted to the training data, and the DNN with a dropout rate of 0.6 underfitted to the training data, as shown in Fig. 4.4 (a-1), (a-2), (d-1), and (d-2). The accuracy in Fig. 4.4 (b-1) is less than that in Fig. 4.4 (c-1), and the loss in Fig. 4.4 (b-2) is greater than that in Fig. 4.4 (c-2). These results suggest that the DNN with a dropout rate of 0.4 underfitted to the training data, and the DNN with a dropout rate of 0.2 was the best.

The PH diagrams of the 1st and 2nd groups are shown in Figs. 4.4 (a-3)–(d-3) and (a-

Table 4.1: Total number of points in the PH diagrams and number of points in belt area ($death \leq birth + 10$) for CIFAR-10 dataset.

Dropout rate	1st group total	1st group belt area	2nd group total	2nd group belt area
0.0	135,279	0	492,725	63
0.2	104,478	104	475,969	153
0.4	49,319	7,462	484,871	1,584
0.6	12,840	7,590	559,605	4,951

Table 4.2: Total number of points in the PH diagrams and the number of points in belt area ($death \leq birth + 10$) for SVHN dataset.

Dropout rate	1st group total	1st group belt area	2nd group total	2nd group belt area
0.0	120,124	972	510,394	0
0.2	49,683	9,023	504,812	1,601
0.4	26,514	10,898	591,696	3,569
0.6	2,375	1,745	594,779	6,807

4)–(d-4), respectively. For the PH diagrams of the 1st group, the number of points near the diagonal line followed the same trend in Fig. 4.3. That is, Fig. 4.4 (a-3) has no point near the diagonal line, where the DNN overfitted to the training data. The number of points in the belt area increased with the increase of the dropout rate (Figs. 4.4 (a-3)–(d-3) and Table 4.2).

For the PH diagrams of the 2nd group, the number of points in the belt area increased with the increase of the dropout rate (Table 4.2). Among the datasets and groups, the overfitted DNNs produced PH diagrams that had zero or smaller numbers of points in the belt area, whereas the underfitted DNNs produced PH diagrams that had larger rate of the number of points in the belt area.

4.4.3 Comparison of CIFAR-10 and SVHN Datasets

As described in Sections 4.4.1 and 4.4.2, the DNN with a dropout rate of 0.2 on the CIFAR-10 dataset overfitted to the training data, whereas that in the SVHN dataset did not overfit to the training data. The difference appeared in the number of points in the belt area with a dropout rate of 0.2. Namely, for the CIFAR-10 dataset, the numbers of points in the belt area in the 1st and 2nd groups were 104 and 153, respectively. Contrariwise, for the SVHN dataset, the numbers of points in the belt area in the 1st and 2nd groups were

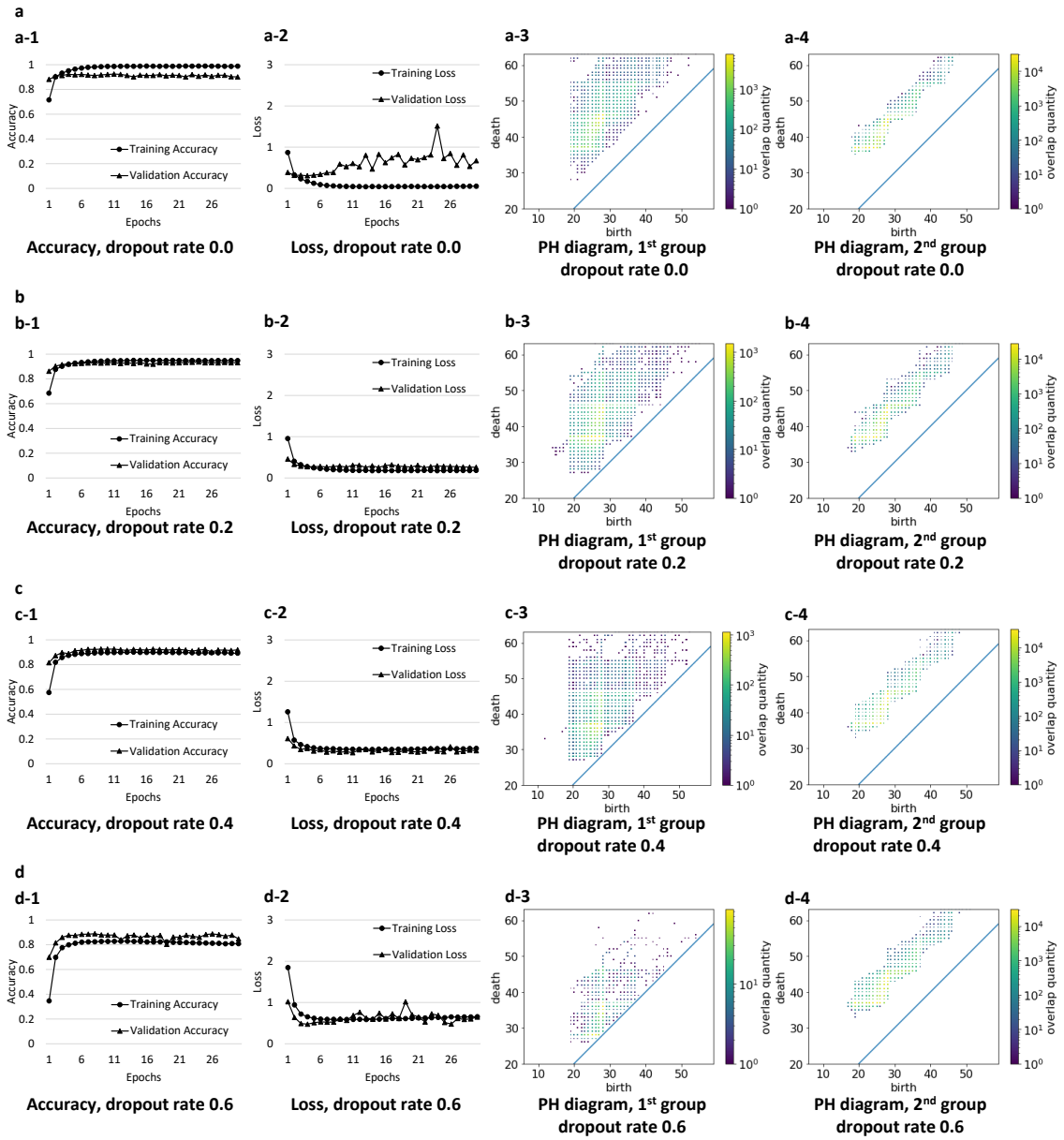


Figure 4.4: PH diagrams of trained DNNs with SVHN dataset; Figs. (a-1)–(d-1) show the accuracy values on the training and validation data; Figs. (a-2)–(d-2) show the loss values on the training and validation data; Figs. (a-3)–(d-3) show the PH diagrams of the 1st group; Figs. (a-4)–(d-4) show the PH diagrams of the 2nd group.

9,023 and 1,601, respectively. In addition, the points shown in Fig. 4.4 (c-3) are adjacent to the diagonal line, whereas those in Fig. 4.3 (c-3) are not.

These results imply that the degree of overfitting in DNNs can be measured by the number of points in the belt area. The correlation between the degree of overfitting and the number of points should be investigated further, but we set that aside for future work, because this study focused on investigating the possibilities of overfitting detection using no training data.

4.4.4 Comparison of 1st and 2nd Groups

The PH diagrams of the 1st group transited explicitly with the increase of the dropout rate, which means that both the number and the location of points transited with the increase of the dropout rate. Contrariwise, those of the 2nd group transited in the location of points, where the number of points did not decreases with the increased dropout rate.

The DNNs extract features in the convolutional layers (2nd group) and classify the objects in the fully connected layers (1st group) [14]. The difference between the transitions of the 1st and 2nd groups could be related to the difference in their functionalities. In other words, the amount of knowledge learned by the classifiers decreased with the increased dropout rate in the 1st group. In addition, the number of features learned in the 2nd group did not change because the number of features was defined intrinsically by the number of kernels in the DNNs.

4.4.5 Tiny ImageNet Dataset

We trained the DNN by varying the dropout rate from 0.0 to 0.6 in 0.2 increments. The results are shown in Fig. 4.5, similar to Figs. 4.3 and 4.4. Due to the increased number of the object classes and the frozen layers in the model training, the accuracy value in Fig. 4.5 is less than those in Figs. 4.3 and 4.4. We found that the DNN with a dropout rate of 0.0 overfitted to the training data, and the DNN with a dropout rate of 0.6 underfitted to the training data, as shown in Figs. 4.5 (a-1), (a-2), (d-1), and (d-2). The graphs of Figs. 4.5 (b-1) and (b-2) indicated that the DNN with a dropout rate of 0.2 overfitted to the training data. For the DNN with a dropout rate of 0.4, the accuracies and losses on the training and validation data were almost the same ((c-1) and (c-2)), which suggests that the DNN was the best.

The PH diagrams for the 3rd group are shown in Figs. 4.5 (a-3)–(d-3). We repeatedly observed that the number of points near the diagonal line followed the same trend in Figs. 4.3 and 4.4, i.e., Fig. 4.5 (a-3) shows no point near the diagonal line, where the DNN overfitted to the training data. The total number of points decreased with the increase of the dropout rate, and the number of points in the belt area increased with it (Figs. 4.5 (a-3)–(d-3) and Table 4.3). We found that a dropout rate of 0.2 mitigated (but did not

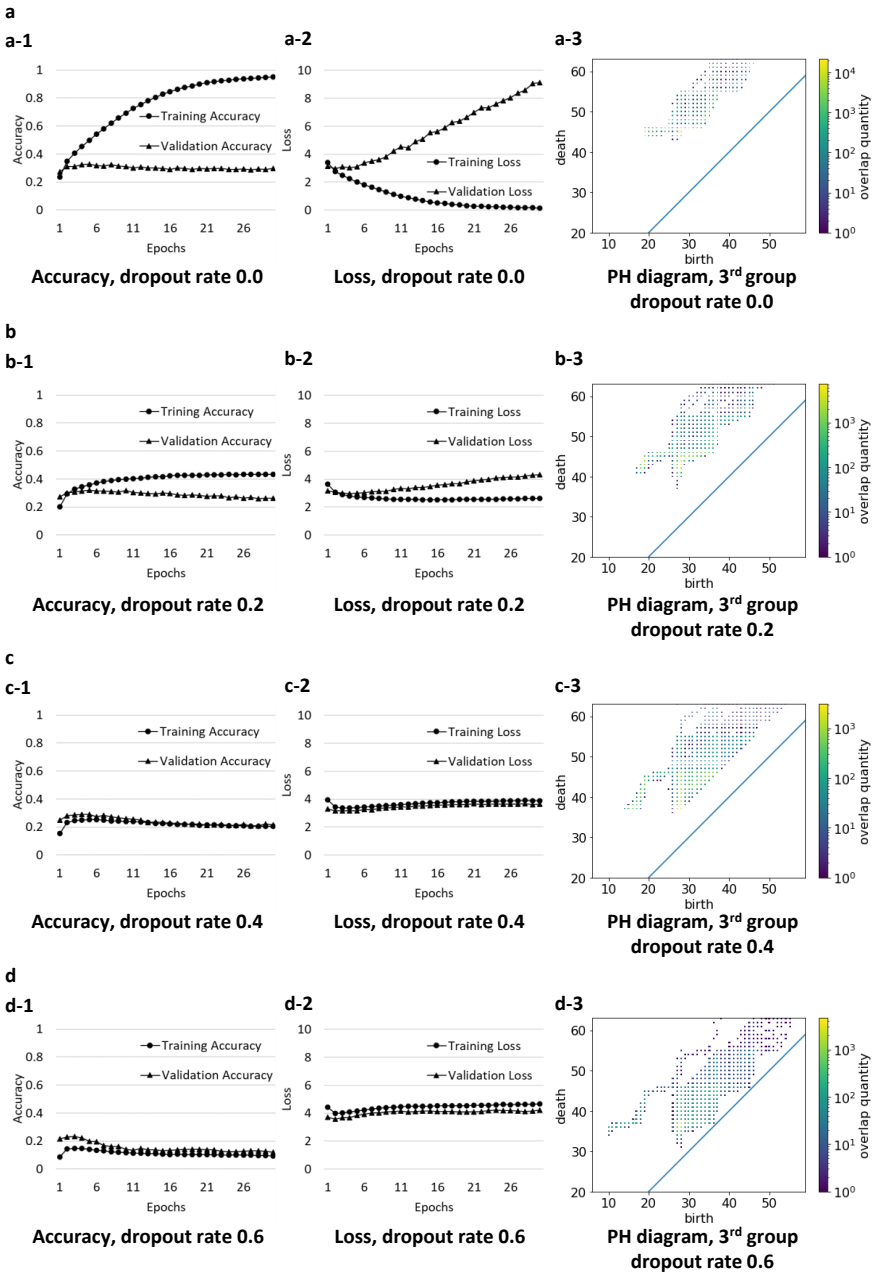


Figure 4.5: PH diagrams of trained DNNs with Tiny ImageNet dataset; Figs. (a-1)–(d-1) show the accuracy values on the training and validation data; Figs. (a-2)–(d-2) show the loss values on the training and validation data; Figs. (a-3)–(d-3) show the PH diagrams of the 3rd group.

eliminate) the degree of overfitting. This was suggested by the small number of points in the belt area, which was commonly observed with the CIFAR-10 dataset (Table 4.1). Among the three datasets, the image size, number of classes, and network structure of the DNNs were unique in the Tiny ImageNet dataset. However, PHOM detected overfitting of the DNNs to the Tiny ImageNet dataset in a manner similar to the other datasets.

Table 4.3: Total number of points in the PH diagrams and number of points in belt area ($death \leq birth + 10$) for Tiny ImageNet dataset.

Dropout rate	3rd group	
	total	belt area
0.0	103,495	0
0.2	75,738	115
0.4	50,311	11,811
0.6	31,377	21,796

4.5 Fundamental Analysis of PHOM

PHOM calculates the PH of a trained DNN using the relevance values among neurons defined in Eqs. (2.2), (2.5), and (2.8). These values include two types of information, i.e., (i) the links among neurons, and (ii) the normalized weights of the links. PHOM reveals the structure of trained DNNs utilizing information (i) and (ii), though only information (ii) can be used to uncover the structure to a certain extent. This section investigates the advantages of utilizing both information (i) and (ii) in PHOM.

4.5.1 Investigation on the 1st and 3rd Groups

Using only the normalized weights of the links, Figs. 4.6 and 4.7 show histograms of the normalized weights for the 1st and 3rd groups trained on the CIFAR-10 and Tiny ImageNet datasets, respectively. The 1st and 3rd groups comprise the two dense layers that link among (512, 512, 10) and (2048, 1000, 200) neurons, respectively. Therefore, the 1st and 3rd groups have 267,264 ($= 512 \times 512 + 512 \times 10$) and 2,248,000 ($= 2,048 \times 1,000 + 1,000 \times 200$) weights, respectively. Figs. 4.6 and 4.7 highlight the relevance values (excluding zeros).

According to the increase in dropout rate, as shown in Figs. 4.6 (a)–(d) and Figs. 4.7 (a)–(d), we made the following observations: (A) the number of positive weights in the 2nd layer decreased, and (B) the variance of the weights in the 2nd layer increased. The numbers of weights in Figs. 4.6 (a) and 4.7 (a) are greater than those in Figs. 4.6 (b)–(d) and Figs. 4.7 (b)–(d), respectively, due to the logarithmic scale of the Y-axis in Figs. 4.6 and 4.7. The same observations are evident in the 1st group trained on the SVHN dataset, as summarized in Table 4.4. The neurons connected with output neurons contribute to the activation of the output neuron. Thus, these observations implied that

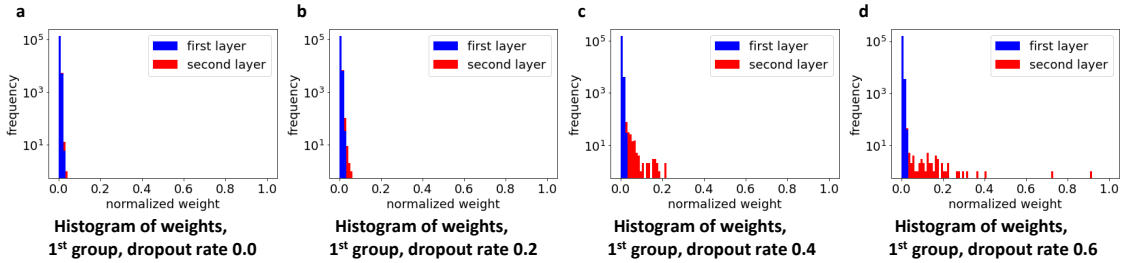


Figure 4.6: Histograms of normalized weights in 1st group trained on CIFAR-10 dataset with different dropout rates (0.0 to 0.6).

the number of neurons contributing to the activation of the output neurons decreased as the dropout rate increased. Homologies comprising only a few neurons result in unstable structures, which are plotted near the diagonal line in the PH diagram. Thus, observations (A) and (B) partially explain the increase in the number of points in the belt area in the PH diagrams (Table 4.4).

However, these observations do not provide integrated measures which can be applied to different DNN structures. In other words, the number of positive weights decreased to 2,892 with a dropout rate of 0.6 in the 3rd group; however, this number was even larger than those obtained with a dropout rate of 0.0 in the 1st group, i.e., 1,871 and 1,064 for the CIFAR-10 and SVHN datasets, respectively (Table 4.4). Further, the variance of weights increased from 1.97×10^{-5} to 2.92×10^{-3} as the dropout rate increased from 0.0 to 0.6 in the 3rd group. However, the variance of weights in the 1st group considerably increased from 1.35×10^{-5} to 2.36×10^{-2} and 8.02×10^{-5} to 9.77×10^{-2} in CIFAR-10 and SVHN datasets, respectively (Table 4.4).

Thus, we could not induce the integrated measure of overfitting in DNNs for the 1st and 3rd groups using only observations of (A) and (B). In contrast, PHOM indicates the overfitting of DNNs, regardless of the groups, using the number of points in the belt area. The advantage of utilizing observations (i) and (ii) in PHOM is a integrated perspective that is independent of network structure.

4.5.2 Investigation on the 2nd Group

Figs. 4.8 (a)–(d) show the histograms of the normalized weights of the 2nd group trained with CIFAR-10 dataset with varying the dropout rate from 0.0 to 0.6 in 0.2 increments.

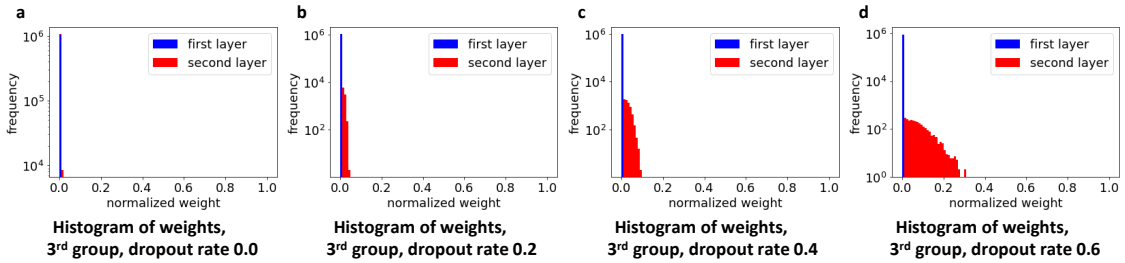


Figure 4.7: Histograms of normalized weights in 3rd group trained on Tiny ImageNet dataset with different dropout rates (0.0 to 0.6).

Table 4.4: Number and variance of weights excluding zeros in DNNs trained on CIFAR-10, SVHN, and Tiny ImageNet datasets for 1st and 3rd groups.

Dataset	Network group	Dropout rate	Number of weights		Variance of weights	
			1st layer	2nd layer	1st layer	2nd layer
CIFAR-10	1st group	0.0	137,992	1,871	8.49×10^{-6}	1.35×10^{-5}
		0.2	136,150	1,163	9.46×10^{-6}	4.60×10^{-5}
		0.4	152,619	321	7.51×10^{-6}	1.33×10^{-3}
		0.6	159,478	64	6.89×10^{-6}	2.36×10^{-2}
SVHN	1st group	0.0	138,155	1,064	8.40×10^{-6}	8.02×10^{-5}
		0.2	136,875	255	8.82×10^{-6}	2.27×10^{-3}
		0.4	145,786	121	7.60×10^{-6}	7.10×10^{-3}
		0.6	155,115	21	6.49×10^{-6}	9.77×10^{-2}
Tiny ImageNet	3rd group	0.0	1,047,176	27,074	5.09×10^{-7}	1.97×10^{-5}
		0.2	1,025,407	16,204	5.70×10^{-7}	6.32×10^{-5}
		0.4	970,559	8,431	6.54×10^{-7}	2.71×10^{-4}
		0.6	886,864	2,892	8.01×10^{-7}	2.92×10^{-3}

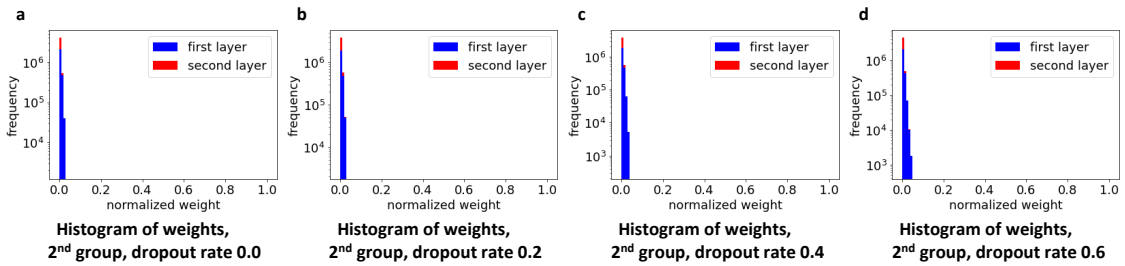


Figure 4.8: Histograms of normalized weights in the 2nd group trained with CIFAR-10 dataset with varying the dropout rate from 0.0 to 0.6.

The 2nd group is comprises the two convolution layers that link among (10816, 14400, 7200) neurons. Therefore, the 2nd group has 259,430,400 ($= 10,816 \times 14,400 + 14,400 \times 7,200$) weights. Figs. 4.8 (a)–(d) highlight the relevance values, excluding zeros.

In the 2nd group, we could not obtain the observations of (A) and (B), but we made an observation of (C): the variance of weights in the 1st layer increased as the dropout rate increased. Observation (C) was also obtained in the 2nd group trained with SVHN dataset, as summarized in Table 4.5. Due to the small change shown in Figs. 4.8 (a)–(d) and Table 4.5, it was difficult to distinguish the overfitted DNNs only relying on (C). However, PHOM explicitly indicates the overfitting of DNNs: the number of points in the belt area increased from 63 to 4,951, and 0 to 6,807 in CIFAR-10 and SVHN datasets, respectively (Tables 4.1 and 4.2)).

In addition, the inconsistency of the observations in the 1st, 2nd, and 3rd groups prevented them from inducing the integrated overfitting measures of DNNs. The inconsistency resulted from the differences in their network structures and layer types. The three groups had different network structures, and only the 2nd group had a unique layer type among the three groups. It would be an essential difficulty to induce the integrated overfitting measures of DNNs by only relying on (ii). However, PHOM could overcome the difficulty by utilizing both information (i) and (ii), meaning that PHOM revealed the increase of the number of points in the belt area commonly in 1st–3rd groups as described Section 4.4.

Table 4.5: Number and variance of weights excluding zeros in DNNs trained on CIFAR-10 and SVHN datasets for the 2nd group.

Dataset	Network group	Dropout rate	Number of weights		Variance of weights	
			1st layer	2nd layer	1st layer	2nd layer
CIFAR-10	2nd group	0.0	2,582,274	2,085,096	2.42×10^{-5}	8.12×10^{-6}
		0.2	2,453,820	2,039,842	2.65×10^{-5}	9.25×10^{-6}
		0.4	2,380,529	2,100,432	3.03×10^{-5}	9.85×10^{-6}
		0.6	2,524,244	2,497,610	3.26×10^{-5}	7.74×10^{-6}
SVHN	2nd group	0.0	2,546,256	2,227,047	2.48×10^{-5}	7.73×10^{-6}
		0.2	2,498,033	2,187,973	2.78×10^{-5}	8.58×10^{-6}
		0.4	2,541,696	2,685,200	3.14×10^{-5}	6.43×10^{-6}
		0.6	2,505,344	2,671,352	3.58×10^{-5}	6.19×10^{-6}

4.6 Normalization of PHOM

The results of PHOM were compared among the same network structure in Section 4.4. Here, we investigate PHOM's ability to compare DNNs with different network structures.

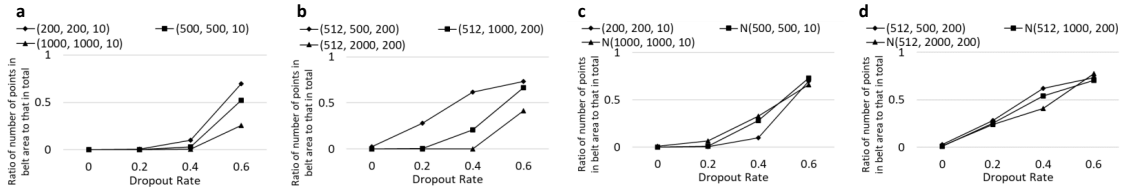
4.6.1 PHOM on Different Network Structure

First, we straightforwardly applied PHOM to neural networks with different network structures. We varied the number of neurons in the 1st group to {(200, 200, 10), (500, 500, 10), (1000, 1000, 10)} for the classification of CIFAR-10 dataset, where the sets of three numbers indicate the number of neurons in 1st–3rd layers in the group. And we also varied the number of neurons in the 3rd group to {(512, 500, 200), (512, 1000, 200), (512, 2000, 200)} for the classification of the Tiny ImageNet dataset. The networks were trained using the parameters described in Section 4.3 with a dropout rate from 0.0 to 0.6 in 0.2 increments.

Figs. 4.9 (a) and (b) show the ratio of the number of points in the belt area ($death \leq birth + 10$) to the total number of points. We found that the difference in network structure caused fluctuations in the PHOM results, which suggests that PHOM has difficulties when comparing networks with different structures.

4.6.2 Normalized PHOM

The normalized PHOM (NPHOM) was inspired by network pruning (NP) technology to mitigate the above mentioned difficulties. NP reduces the number of parameters in networks without reducing their performance [49], which shrinks networks and restrains the



Ratio of number of points in belt area, 1st group varying dropout rate and network structure calculated using PHOM

Ratio of number of points in belt area, 3rd group varying dropout rate and network structure calculated using PHOM

Ratio of number of points in belt area, 1st group varying dropout rate and network structure calculated using NPHOM

Ratio of number of points in belt area, 3rd group varying dropout rate and network structure calculated using NPHOM

Figure 4.9: Ratio of number of points in the belt area ($death \leq birth + 10$) to that in total, in 1st and 3rd groups trained for the classification of CIFAR-10 and Tiny ImageNet datasets, respectively, obtained by varying the network structures and dropout rate from 0.0 to 0.6 calculated with PHOM and NPHOM. The sets of three numbers in the legends indicate the number of neurons in 1st–3rd layers in the groups.

loss of the knowledge learned by the original network. Magnitude-based NP (MNP) is a common technique, where neurons are selected from the original networks in descending order of network weights to preserve influential neurons. Although the strategy of MNP is uncomplicated, MNP has been shown to be competitive with more complex methods [8].

NPHOM shrinks the network size while remaining influential neurons using a similar approach to MNP. Algorithm 3 shows the selection procedure, where the inputs comprise the network weights, target output neurons, and the number of input neurons to be selected. NPHOM calculates the influences of the input neurons with respect to the output neurons and selects input neurons in descending order of the influences.

Figs. 4.9 (c) and (d) show the ratios of the number of points in the belt area to the total number of points obtained using NPHOM. Here, the networks of (500, 500, 10) and (1000, 1000, 10) were reduced to (200, 200, 10) in Fig. 4.9 (c), and the networks of (512, 1000, 200) and (512, 2000, 200) were reduced to (512, 500, 200) in Fig. 4.9 (d), which are marked as "N" in the legends. These figures show that NPHOM allows us compare networks of different structures by reducing fluctuation caused by differences in the network structures.

The difference between the training and validation accuracies is an indicator of the degree of overfitting, which decreases according to the mitigation of overfitting (Figs. 4.3–4.5). Table 4.6 presents the correlation coefficient between A and B, where A is the absolute value of the difference between the training and validation data, and B is the ratio of the number of points in the belt area to the total number of points calculated by PHOM

and NPHOM. The negative correlation between A and B was strengthened by NPHOM. Consequently, the correlation coefficients obtained on the CIFAR-10 and Tiny ImageNet datasets reached -0.64 and -0.89, respectively. This strong negative correlation suggests that we could select networks that are less overfitted to the training data using NPHOM.

Algorithm 3 Algorithm for selecting influential neurons

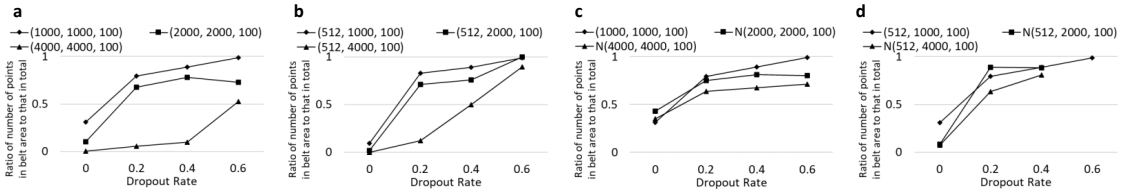
1: **procedure** SELECTNUERON($\{w_{ij}\}$, $(I_k)_{k=1}^{L}$, N) ▷ $\{w_{ij}\}$: weight matrix, w_{ij} is the weight between input neuron i and output neuron j; $(I_k)_{k=1}^{L}$: list of output neurons; N: number of input neurons to be selected.
2: $v_{ij} \leftarrow \max(0, w_{ij})$ for $\forall i,\ \forall j \in (I_k)_{k=1}^{L}$ ▷ PHOM uses positive part of the weight
3: $v_{ij} \leftarrow 0$ for $\forall i,\ \forall j \notin (I_k)_{k=1}^{L}$ ▷ set 0 when w_{ij} is not related to the output neurons
4: $S_i \leftarrow \sum_j v_{ij}$ ▷ calculate the influence of input neurons
5: $(O_k)_{k=1}^{N} \leftarrow \{j; S_j$ is in larger N elements of $\{S_i\}\}$ ▷ select input neurons in descending order of S_i
6: **return** $(O_k)_{k=1}^{N}$
7: **end procedure**

4.6.3 NPHOM on Additional Dataset

We also used the CIFAR-100 dataset [42] to investigate the effectiveness of PHOM and NPHOM on different datasets. CIFAR-100 consists of 50,000 training images with the size of 32×32 color that are classified into 100 types of objects. We trained the 1st and 3rd groups varying the network size as {(1000, 1000, 100), (2000, 2000, 100), (4000, 4000, 100)} and {(512, 1000, 100), (512, 2000, 100), (512, 4000, 100)} for the classification, respectively. The networks were trained with a batch size of 128 and 30 epochs, where 20% of the training data were used as validation data.

Figs. 4.10 (a) and (b) show the ratios of the number of points in the belt area to that in total calculated using PHOM. We repeatedly observed that the difference in network structures fluctuates the PHOM results. This means that the result of network size of (4000, 4000, 100) in Fig. 4.10 (a) and that of (512, 4000, 100) in Fig. 4.10 (b) increase gradually compared with other results. This difference prevents PHOM from comparing networks with different structures.

Figs. 4.10 (c) and (d) show the ratios of the number of points in the belt area to that in total calculated using NPHOM. Here, NPHOM aligned the network size to the smallest size i.e., the networks of size (2000, 2000, 100) and (4000, 4000, 100) in Fig. 4.10 (c)



Ratio of number of points in belt area, 1st group varying dropout rate and network structure calculated using PHOM

Ratio of number of points in belt area, 3rd group varying dropout rate and network structure calculated using PHOM

Ratio of number of points in belt area, 1st group varying dropout rate and network structure calculated using NPHOM

Ratio of number of points in belt area, 3rd group varying dropout rate and network structure calculated using NPHOM

Figure 4.10: Ratio of number of points in the belt area ($death \leq birth + 10$) to that in total, in 1st and 3rd groups trained for the classification of CIFAR-100 datasets obtained by varying the network structures and dropout rate from 0.0 to 0.6 calculated with PHOM and NPHOM. The sets of three numbers in the legends indicate the number of neurons in 1st–3rd layers in the groups.

and those of (512, 2000, 100) and (512, 4000, 100) in Fig. 4.10(d) were reduced to (1000, 1000, 100) and (512, 1000, 100), respectively (marked with "N" in the legends). The difference in PHOM results were mitigated due to the normalization proposed in NPHOM. The homologies of the networks N(512, 2000, 100) and N(512, 4000, 100) with a dropout rate of 0.6 did not appear in the PH diagrams. This phenomenon occurred occasionally in our investigations when the target network was excessively underfitted to the training data.

Table 4.6 presents the correlation coefficient between A and B, where A is the absolute value of the difference between the training and validation data, and B is the ratio of the number of points in the belt area to the total number of points calculated by PHOM and NPHOM. Due to the fluctuation caused by the difference in network structures, the correlation efficient of "all sizes" can be less than that of each network. We found that NPHOM strengthened the negative correlation between A and B, with an exception caused by the disappearance of homologies in the network of size (512, 2000, 100). The strong negative correlation and mitigation of fluctuations by NPHOM suggest that NPHOM is essential in selecting networks that are less overfitted to the training data, even when the networks have different structures.

Table 4.6: Correlation coefficient between A and B of the networks trained for the classification of CIFAR-10, Tiny ImageNet, and CIFAR-100 datasets, where A is the absolute value of the difference between the training and validation data, and B is the ratio of the number of points in the belt area to that in total calculated by PHOM and NPHOM.

Dataset	Network size	Sample points	PHOM	NPHOM
CIFAR-10	(200, 200, 10)	4	−0.48	−0.48
	(500, 500, 10)	4	−0.54	−0.77
	(1000, 1000, 10)	4	−0.37	−0.70
	all sizes	12	−0.44	−0.64
Tiny ImageNet	(512, 500, 200)	4	−0.88	−0.88
	(512, 1000, 200)	4	−0.78	−0.98
	(512, 2000, 200)	4	−0.45	−0.83
	all sizes	12	−0.66	−0.89
CIFAR-100	(1000, 1000, 100)	4	−0.90	−0.90
	(2000, 2000, 100)	4	−0.80	−0.82
	(4000, 4000, 100)	4	−0.79	−0.79
	all sizes	12	−0.64	−0.80
CIFAR-100	(512, 1000, 100)	4	−0.98	−0.98
	(512, 2000, 100)	4	−0.96	−0.95*
	(512, 4000, 100)	4	−0.84	−0.99*
	all sizes	12	−0.89	−0.97**

* calculated with three samples due to the missing value
** calculated with ten samples due to the missing values

4.7 Discussion

4.7.1 Intension of Overfitting Measurement Using PH

It has been hypothesized that the dropout technique prevents DNNs from overfitting by reducing co-adaptation [77]. The hypothesis suggests that overfitting is a result of neuron co-adaptations. We have investigated the number of points in the belt area, which counts the points near the diagonal line. The results imply that the degree of neuron co-adaptation, assumed a cause of overfitting in [77], is indicated by the number of points near the diagonal line.

Fig. 2.1 shows that the points adjacent to each other in $\mathbb{R}^2$ produced the homology mapped near the diagonal line in the PH diagram; whereas, the points that are distant from each other produced the homology mapped at a distance from the diagonal line. In Section 4.2, the relevance between neurons was defined in proportion as the weight between them, meaning that the neurons connected with large weights are located "close." As a result, the degree of overfitting is indicated by the number of points in the belt area.

The deep Taylor decomposition (DTD) method reveals the influential inputs to DNN output. This study investigates the stability of the knowledge retained in DNNs based on DTD, which is measured by the number of neurons supporting the output. The overfitted DNNs had a large number of stable homologies. The result is consistent with the hypothesis that DNN overfitting is due to the co-adaptations among neurons.

4.7.2 Underfitting Measurement of DNNs

Regarding the 1st and 3rd groups, the total number of points in the PH diagram decreases as the dropout rate increases. This observation was common throughout this study as shown in Tables 4.1, 4.2, and 4.3. This implies that the total number of points in the PH diagram measures the degree of DNNs' underfittings.

Therefore, when two models have the close number of points near the diagonal line, the total number of points helps to select the better model, meaning that the less underfitted model (better model) has a larger number of points. Otherwise, the better model is selected based on the balance between the overfitting and underfitting degrees. We set the investigation of the balance aside as a task for future work.

4.7.3 Plot Area of PH Diagram

The PH diagram varies the shape of the plot area depending on the knowledge retained in DNNs. Unstable homologies, plotted near the diagonal line, indicate the degree of overfittings of DNNs, as shown in Figs. 4.3, 4.4, and 4.5. PHOM detects the overfittings of DNNs using the number of unstable homologies.

Stable homologies, plotted at a distance from the diagonal line, also appear depending on the degree of overfitting of DNNs in the 1st and 3rd groups. This implies that both the unstable and stable homologies contribute to improving the general performance of DNNs. Thus, if the numbers of homologies are close in two models, the model whose PH diagram has a wider plot area will have better general performance. However, PHOM employs the number of unstable homologies for detecting overfitting since the change in the PH diagram for the 2nd group is unclear about the stable homologies as shown in Figs. 4.3 and 4.4.

4.7.4 Threshold Value Defining Belt Area

For counting the number of unstable homologies, PHOM employed a threshold value of 10 for defining the belt area ($birth + 10 \leq death$). Furthermore, we considered other counting techniques using statistical values, such as the center and variance of points. However, the plot area expands differently in Figs. 4.3, 4.4, and 4.5, meaning that Fig. 4.5 expands the area with the range of 10–25 birth values, which is almost unobserved in Figs. 4.3 and 4.4. Consequently, PHOM employed a provisional threshold value of 10 for the detection.

The proper threshold value can change depending on the network structures and training datasets. We have started the development of a method for searching the proper threshold value using the representative cycle [24]. The representative cycle enables us to know the neurons included in each homology, which helps to reveal the mechanism that generates homologies in DNNs. The revelation of the mechanism will derive the method. Thus, we set the development of the method for searching the proper threshold value as our future work.

4.8 Related Work

4.8.1 Persistent Homology for Investigating DNNs

PH is a prominent method for investigating DNNs. Rieck et al. proposed a neural persistence that employs the zero-dimensional PH to characterize and monitor the structural properties of DNNs [69]. The zero-dimensional PH represents the number of connected components in simplicial complexes. It is closely related to the training progress because the DNN parameters converge during training, and the convergence of the parameters results in the convergence of the number of connected components in DNNs.

The one-dimensional PH has also been employed to investigate DNNs [16, 82, 84]. Carneanu et al. estimated the performance gap between training and testing data using the one-dimensional PH [16]. The one-dimensional PH represents the number of holes on simplicial complexes, which can reveal the combinational effects among neurons. Thus, the one-dimensional PH is a prominent method for revealing the co-adaptations among

neurons in DNNs. Carneanu et al. employed the training data to calculate the PH of DNNs. Meanwhile, we defined simplicial complexes on DNNs from the trained parameters of DNNs, which enables PHOM to detect the overfitting of DNNs without requiring training data.

4.8.2 Detection of Overfitting

Beyond the conventional method to detect overfitting, which compares the accuracy and loss values of training and validation data, the detection of overfitting in DNNs has been improved in various ways. Werpachowski et al. proposed an error estimation method that is based on adversarial examples generated from the test data [85]. Groose et al. eliminated the need for large amounts of labeled data using a local measurement around a small number of unlabeled test points [31], which suggests that overfitting is a local characteristic of DNNs.

This concept was pushed further in this study, assuming that overfitting is an intrinsic characteristic of DNNs without relying on training data. To confirm this assumption, we investigated the possibility of overfitting detection using no training data. We employed PH of this investigation because it is a prominent method for investigating the inner representation of DNNs.

To the best of our knowledge, previous methods for detecting overfitting of DNNs rely on the test data. However, the inner representation of DNNs can be investigated using statistical methods without relying on the test data, such as canonical correlation analysis (CCA). Singular vector [66] and projection-weighted CCA [56] have been proposed to apply CCA to deep neural networks by mitigating its sensitivity to parameter perturbations. The authors of [40] introduced centered kernel alignment, which is closely related to CCA. They provided similarity indexes among trained models, which can be used to filter overfitted DNNs by distinguishing them from not-overfitted DNNs. However, non-overfitted DNNs are required for detecting overfitting using the similarity indexes. Contrary, PHOM detects overfitted DNNs by investigating the combination among neurons in a trained DNN. Thus, we believe that the approach in PHOM is efficient for detecting overfitting without relying on the test data.

4.9 Conclusion

In this paper, we have proposed the PHOM method to detect overfitted DNNs using no training data. PHOM constructs clique complexes in DNNs using the trained parameters of DNNs, and the one-dimensional PH investigates the co-adaptations among neurons. Thus, PHOM does not require training data to detect overfitting. In addition, we extended PHOM to normalized PHOM (NPHOM) to mitigate fluctuations in PHOM caused by differences in network structures. We applied the proposed methods to DNNs trained for the classification problems of the CIFAR-10, SVHN, Tiny ImageNet, and CIFAR-100 datasets. The experimental results demonstrated that PHOM and NPHOM can measure the degree of overfitting to the training data. PHOM demonstrated that the number of points in the belt area indicates the degree of overfitting in the comparison between two DNNs with same network structure, meaning that the overfitted DNN has smaller number of points in the belt area when the total numbers of points are approximately same. NPHOM enhanced PHOM for applying different network structures. These results implied that the methods could remove the necessity to use training data for filtering overfitted DNNs.

Chapter 5

Deep Neural Network Pruning Using Persistent Homology

Deep neural networks (DNNs) have improved the performance of artificial intelligence systems in various fields including image analysis, speech recognition, and text classification. However, the consumption of enormous computation resources prevents DNNs from operating on small computers such as edge sensors and handheld devices. Network pruning (NP), which removes parameters from trained DNNs, is one of the prominent methods of reducing the resource consumption of DNNs. Although NP consumes computation resources, NP can be executed on large computers for developing pruned DNNs for operating on small computers. In this paper, we propose a novel method of NP, hereafter referred to as PHNP, using persistent homology (PH). PH investigates the inner representation of knowledge in DNNs, and PHNP utilizes the investigation in NP to improve the efficiency of pruning. PHNP prunes DNNs in ascending order of magnitudes of the combinational effects among neurons, which are calculated using the one-dimensional PH, to prevent the deterioration of the accuracy. We compared PHNP with global magnitude pruning method (GMP), which is one of the common baselines to evaluate pruning methods. PHNP improves the accuracy of pruned networks in return for the larger computation resources compared with GMP. Evaluation results show that the classification accuracy of DNNs pruned by PHNP outperforms that pruned by GMP.

5.1 Introduction

Deep neural networks (DNNs) play an essential role in artificial intelligence (AI) systems for improving performance in various fields including image analysis, speech recognition, and text classification [33, 91]. They improve the performance of AI systems using knowledge learned from big data, which sometimes surpass human levels. However, they consume enormous computation resources, which prevents them from operating on small computers such as edge sensors and handheld devices [63].

Network pruning (NP), which removes parameters from trained DNNs, is one of the prominent methods for reducing the computation resources required by DNNs. The removal of parameters relieves the computation and memory capacity for operating DNNs. However, NP deteriorates the performance of DNNs because the performance of DNNs with a large number of parameters is generally higher than that with a small number of parameters. To achieve the best balance between performance and efficiency, many techniques have been proposed in previous research [27, 89].

The inner representations of knowledge in DNNs are indecipherable, which makes the achievement of the balance difficult. The limited comprehension of the inner representation causes over- and under- cutting of parameters. Although the number of parameters removed from DNNs is not proportional to the amount of knowledge removed from DNNs, we need to achieve a proportion that balances efficiency and performance despite the limited comprehension.

In this paper, we propose a novel method of pruning DNNs using persistent homology (PH). PH reveals the synchronizations among neurons as described in Section 2.1.1 and 2.1.2, which is one of the inner representation of the knowledge retained in DNNs. Then, the efficiency of NP can be improved by remaining the synchronized neurons selectively. For applying PH to NP, we constructed clique complexes on trained DNNs and calculated the one-dimensional PH of DNNs. The one-dimensional PH reveals the combinational effects of multiple neurons in DNNs at different resolutions. The persistent-homology-based pruning method (PHNP) prunes DNNs in ascending order of the magnitudes of the effects for preventing accuracy deterioration. We applied PHNP in DNNs that comprise a

convolutional neural network (CNN) and a fully connected network (FCN). The accuracy of PHNP was compared with that of global magnitude pruning method (GMP), which is a common baseline for evaluating pruning methods and has demonstrated to be competitive with more complex pruning methods [8]. We explored the ability of PH for investigating the inner representation of knowledge in DNNs by applying PH to NP.

The reminder of the paper is organized as follows. In Section 5.2, we will explain PH and propose PHNP. In Section 5.3 and 5.4, we will present the evaluation result comparing PHNP and GPM and discuss the differences between them. In Section 5.5 and 5.6, we will explain related work and conclude this paper.

5.2 Neural Network Pruning Using Persistent Homology

PHNP comprise three steps. In the first step, PHNP constructs simplicial complexes on trained DNNs by considering DNNs as weighted directed graphs. Although PHNP was developed on the basis of constructing simplicial complexes on DNNs [84], the following two points differ from this basis: (1) PHNP employs the absolute values of weights of DNNs, where positive values are used in the previous method described in Chapter 2; (2) PHNP employs the stack-based algorithm, where the previous method employs the recursive-call-based algorithm. The algorithm change from recursive-call-based to stack-based improves the performance, but we confirmed that the results of both algorithms are essentially the same, namely, the results are identical except for the duplications and the order of outputs.

In the second step, PHNP calculates the PH of the simplicial complexes using the computational libraries [25, 61, 78]. To perform the above calculation, it is required to define the resolution of PH, which is called filtration. We employed the same filtration setting as in the previous method [84]. We employed Dionysus and JavaPlex libraries for the computation [25, 57, 78].

In the third step, PHNP prunes the edges of DNNs as following: (i) sort homology by the sum of birth and death; (ii) select homologies in ascending order of the sum, in which the number of edges included in the selected homologies achieves the target pruning ratio; and (iii) set the edge weights not included in the selected homologies to zero. The one-

dimensional PH investigates the synchronized neurons in DNNs as described in Section 2.1.1 and 2.1.2. And, the filtration of PH is designed in descending order as described in Section 3.2.2. Thus, PHNP selects the synchronized neurons in descending order of the relevance values among neurons. In other words, PHNP remains the synchronized neurons connected with large relevance values. For identifying the edges included in homologies in (ii), we employed the annotation functionality of JavaPlex, which computes the representative cycle for each homology [78].

Through (i)–(iii), the weight values of edges included in the selected homologies remain the same in the trained DNNs, and the others are set to zero. The sum of birth and death correlates with the denseness of point. We defined the "closeness" of neurons based on the weight values in DNNs, and it indicates the magnitudes of the combinational effects among neurons. Therefore, PHNP prunes DNNs in ascending order of magnitudes of the combinational effects among neurons.

5.3 Evaluation and Result

5.3.1 Evaluation Setup

The CIFAR-10 data set was employed in the evaluation [42], which is used in the evaluations in network pruning [13, 92]. The contents of the data set are 32 x 32 color images, which comprises the images of 10 types of objects such as airplanes, automobiles, birds. It contains 50,000 and 10,000 images for training and testing, respectively. All experiments were conducted using Keras and Tensorflow [1, 14], and DNNs were developed based on the examples in Keras 2.3.0.

For the classification of the CIFAR-10 data set, we employed DNNs consisting of a CNN and an FCN, which are developed based on the sample networks of Keras 2.3.0. The CNN comprises multiple layers, including two-dimensional convolution, max pooling, and dropout layers. The FCN comprises three layers with sizes of (300, 100, 10)[1]. The

[1]The following network structures are employed: input(3072)–Conv2D(32 filters, 3×3 kernel, ReLu activation)–Conv2D(32 filters, 3×3 kernel, ReLu activation)–MaxPooling2D(2×2 pool)–Dropout(dropout ratio 0.2)–Conv2D(64 filters, 3×3 kernel, ReLu activation)–Conv2D(64 filters, 3×3 kernel, ReLu activation)–MaxPooling2D(2×2 pool)–Dropout(dropout ratio 0.2)–Flatten–Dense(300, ReLu activation)–Dropout(dropout ratio 0.2)–Dense(100, ReLu activation)–Dropout(dropout ratio 0.2)–Dense(10, softmax activation).

DNNs were trained for 100 epochs with a batch size of 32.

The CNN was used to extract features from the images, while the FCN was used to classify the images based on the combination of the features. PHNP was applied to the FCN since the inner representation of knowledge in DNNs is represented in the combination of features. For pruning both FCN and CNN, PHNP needs to be integrated with pruning techniques for CNNs. However, we have set this aside as a task for future work because this paper explores the ability of PH to investigate of the inner representation of knowledge in DNNs.

5.3.2 Result

We compared PHNP with GMP, which is a common baseline and has demonstrated to be competitive with more complex pruning methods [8]. GMP prunes the edges with the lowest absolute value anywhere in the DNNs.

Table 5.1 shows the object classification accuracies of pruned networks. The pruning ratio is defined as $(m–n)/m$, where m and n are the number of edges in original and pruned FCNs, respectively. The value of m was set to $31,000 (= 300 \times 100 + 100 \times 10)$, because we employed the FCN with 300, 100, and 10 neurons in each layer.

The accuracies of both methods deteriorated as edges were removed from DNNs. However, the accuracy of PHNP was higher than that of GMP in the pruning ratios of 0.8 and 0.9, which indicates that PHNP can remove edges from DNNs efficiently. PHNP could prune 95% of edges from DNNs with 12% higher accuracy than the DNN pruned by GMP in our evaluation setting including dataset, network structure, and training process. It means that PHNP can remove the edges from DNNs with less loss of the knowledge in the networks.

Fig. 5.1 and 5.2 show the object classification accuracies of DNNs pruned by GMP and PHNP, respectively. The accuracies of DNNs pruned by GMP deteriorate mainly in the object classification of dog, mobile, flog, horse, bird, and ship. Contrary, the accuracies of DNNs pruned by PHNP deteriorate mainly in the object classification of mobile and bird. Consequently, the overall accuracies of DNNs pruned by PHNP outperform those pruned by GMP, as shown in Table 5.1.

Table 5.1: Accuracies of object classification of pruned networks

Pruning Ratio	0	0.6	0.7	0.8	0.9	0.95
Acc. of GMP	0.77	0.77	0.76	0.74	0.66	0.43
Acc. of PHNP	0.77	0.77	0.74	0.74	0.69	0.56

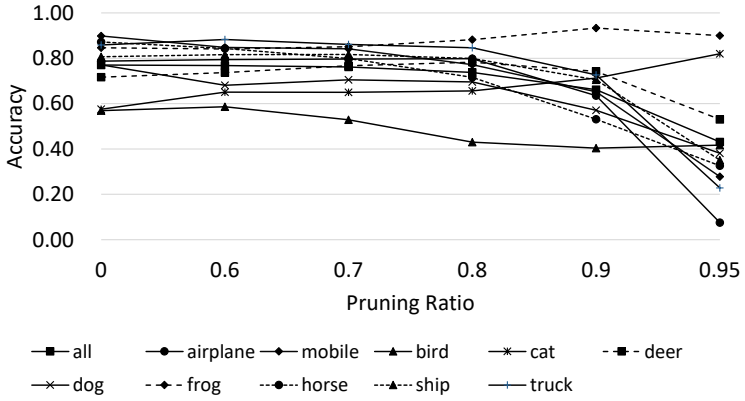


Figure 5.1: Accuracies of object classification of DNN pruned by GMP

5.4 Discussion

Fig. 5.1 reveals a shortcoming in GMP; that is, it failed to keep the balance of the knowledge pruned from DNNs. Contrarily, PHNP kept the balance by pruning DNNs based on the investigation of the inner representation of DNNs. The result thus suggests the following two points. First, PH is useful to investigate the inner representation of DNNs. PH reveals the births and deaths of the combination of neurons, which are difficult to be captured without using PH. Second, we will be able to improve pruning methods by leveling the knowledge pruned from DNNs. The accuracies of mobile and cat declined mainly in Fig. 5.2. The overall accuracies will be improved by preventing the deterioration of the two objects classification. The development of the leveling technique in network pruning using PH remains in future works.

5.5 Related Work

Network pruning has been investigated over three decades [38, 58], and its popularity has increased in this decade owing to the rise of DNNs [8]. To confirm the superiority of the pruning method, it is required to examine the method in various conditions such as network structure, size, and data sets. In addition, the broad varieties of proposals of the

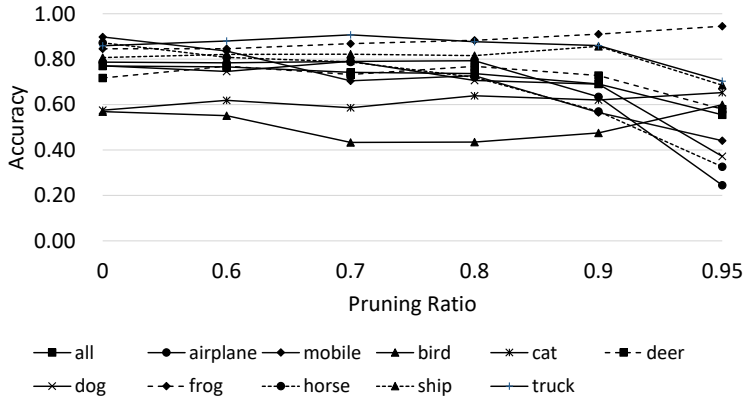


Figure 5.2: Accuracies of object classification of DNN pruned by PHNP

pruning method prevent us from comparing them under equitable conditions. Blalock et al. proposed ShrinkBench for facilitating the standardized evaluation of pruning methods. PHNP needs to be evaluated using various conditions for demonstrating superiority to conventional pruning methods. However, we have set it as future work because the result demonstrates the ability of PH for investigating the inner representation of DNNs, which could provide novel insights into the research of PH and DNNs.

5.6 Conclusion

We proposed a novel method (PHNP) to prune DNNs using PH. The inner representation of knowledge in DNNs is investigated by PH, and PHNP utilizes the investigation in NP. The one-dimensional PH reveals the combinational effect of multiple neurons in DNNs, and PHNP prunes DNNs in ascending order of magnitudes of the effect for preventing the deterioration of accuracy. We compared PHNP with GMP, which is a common baseline for evaluating pruning methods. The evaluation result showed that PHNP outperforms GMP by utilizing the analysis of PH. It implies that PH is one of the prominent methods for investigating the inner representation of knowledge in DNNs.

Chapter 6

Conclusion

6.1 Achievements

Deep neural networks (DNNs) play an important role in improving the performance of artificial intelligent systems [33, 91]. DNNs extract knowledge from big data, thereby eliminating the need to develop custom representations for target applications. The combination of DNNs and big data is a powerful technique to obtain knowledge that can be used for several applications.

Although the use of trained DNNs leads to obvious performance improvement, interpreting the inner representation of trained DNNs is challenging. The uninterpretability hinders the tuning of DNN models, the control of their training processes, and the interpretation of their outputs. The difficulty is caused from two problems: DNNs contain tens of millions parameters [2], and the parameters perturb in each training [27]. The large amount and perturbation of parameters make it difficult to examine and verify the inner representation of DNNs.

Investigating the inner representation of DNNs requires a data analysis method that can overcome the aforementioned problems. Given these considerations, we used persistent homology (PH) for the investigation. PH is a data analysis method that has three advantages: theoretical foundation, practical computability, and robustness with small perturbations [61]. Furthermore, we adopted a one-dimensional PH for the investigation, which can reveal the combinational effects among neurons in DNNs.

Functional connectivity is defined as the synchronization of spatially remote neurophysiological events [28], which plays an important role in investigating brains. Consid-

ering the importance of functional connectivity in neuroscience, this thesis applied the concept of functional connectivity to DNNs. Thus, this thesis contributed to reveal the fundamental aspects of the inner representation of DNNs by investigating DNNs from the standpoint of functional connectivity.

PH is designed to investigate simplicial complexes in a topological space. The discordance between simplicial complexes and network parameters in DNNs prevents us from applying PH to DNNs. Thus, this study aims to achieve the following goals.

(G1) Developing a method to investigate the inner representation of DNNs using PH;

(G2) Improving the comprehension of network behaviors using PH.

The goals and achievements are summarized in Table 6.1. In Chapter 2, a construction method for clique complexes on DNNs was presented, which defined sets of synchronized neurons as simplexes for investigating the inner representation of DNNs from the stand point of functional connectivity. Two techniques were introduced in the construction: normalization and propagation. The normalization technique enables us to compare the influences among neurons. Meanwhile, the propagation technique defines the influences among the neurons connected with multiple edges. In this chapter, the mathematical proof of the correctness of the construction method was provided. Additionally, the calculation methods of PH were formalized in dense, convolution, and pooling layers. Finally, an algorithm was proposed to identify all simplexes in the clique complex constructed on DNNs. The proposed algorithm enables us to utilize the high-performance implementation of the computation libraries in PH calculation.

In Chapter 3, the changes in PH were examined while varying the difficulty of the problem for which DNNs were trained. The difficulty of the problem was adjusted by varying the number of classes of training data. Further, the number of output neurons of DNNs was varied. Then, four observations were obtained by counting the number of unstable homologies, which are plotted near the diagonal line in the PH diagram:

(i) The number of unstable homologies increased with the decrease in the number of classes, from 10 to 5, under the network setting of 10 output neurons;

Table 6.1: Summary of goals and achievements of this thesis

Goal	Achievement	Chapter
G1	- Development of a construction method for clique complexes on DNNs with mathematical proof of the correctness of the method - Formalization of the calculation method of PH in dense, convolutional, and pooling layers - Proposal of an algorithm to identify simplexes in the clique complex constructed on DNNs	2
	- Confirmation of the reflection in PH corresponding to the difficulty of the problem for which DNNs are trained - Confirmation of the robustness of PH regarding the network's settings and initial weights of DNNs	3
G2	- Proposal of a PH-based overfitting measure (PHOM) method, which detects overfitted DNNs using trained network weights - Verification of the effectiveness of PHOM, which enables us to filter overfitted DNNs without relying on the training data	4
	- Proposal of a PH-based network pruning (PHNP) method, which reduces network parameters from DNNs using PH - Verification of the effectiveness of PHNP, which showed that PHNP could prune 95% of edges from DNNs with 12% higher accuracy than the DNN pruned by a common baseline method in our evaluation setting including dataset, network structure, and training process	5

(ii) The number of unstable homologies decreased with the decrease of the number of classes, from 5 to 1, under the network setting of 10 output neurons;

(iii) The unstable homologies were disappeared when the number of classes was equal to 5 under the network setting of 5 output neurons;

(iv) The unstable homologies were appeared when the number of classes was equal to 10 under the network setting of 20 output neurons.

The observations (i)–(iv) indicated that PH reflected the changes in the inner representation of DNNs. Additionally, we evaluated the robustness of PH investigation by repeating the experiments using random initial weights with three network structures and two datasets. The evaluation showed that the observations (i)–(iv) were common in the experiments, thus confirming the robustness of the PH regarding the network settings and weights of DNNs.

In Chapter 4, a PH-based overfitting measure (PHOM) method was proposed. This method detects overfitted DNNs using trained network weights. PHOM was inspired by the dropout technique, which prevents the overfitting of DNNs under the hypothesis that overfitting is a result of the co-adaptations among neurons. PHOM investigates the co-adaptations by calculating the ratio of A to B, where A is the number of unstable homologies, and B is the number of total homologies. PHOM was evaluated by varying the dropout rate in the DNNs to control the ir overfitting. The following are the observations of the PHOM method:

(v) The ratio increases according to the mitigation of the overfitting of DNNs;

(vi) Observation (v) is common in dense and convolutional layers.

Observations (v) and (vi) were confirmed with two network structures and four datasets. Additionally, PHOM was enhanced to normalized PHOM (NPHOM) for reducing the fluctuation caused by the difference in the network structures. NPHOM enables us to compare networks with different structures. The effectiveness of NPHOM was verified using two networks with varying the network structures under three settings. These investigations revealed that PHOM and NPHOM are effective in filtering overfitted DNNs without relying on the training dataset; the same was difficult to achieve without using

PH.

In Chapter 5, a PH-based network pruning (PHNP) method was introduced. This method reduces network parameters from trained DNNs using PH. Consuming enormous computation resources prevents DNNs from operating on small computers, such as hand-held devices. PHNP prunes the edges in DNNs in ascending order of magnitude of the combinational effects among neurons using PH. Thus, PHNP can reduce the parameters from DNNs while restraining the loss of knowledge in DNNs. The effectiveness of PHNP was evaluated by comparing PHNP with the global magnitude pruning (GMP) method, which is one of the common baselines to evaluate pruning methods. The evaluation revealed that PHNP could prune 95% of the edges from DNNs with 12% higher accuracy than the DNN pruned by GMP method in our evaluation setting including dataset, network structure, and training process.

6.2 Related Work

In this thesis, the inner representation in DNNs was investigated by examining the interactions among neurons using PH. Previous works to investigate DNNs using PH were described in Section 4.8. Furthermore, the inner representation can be examined using two strategies: model-agnostic and model-specific strategies [50]. This section describes the related works, categorizing previous works into strategic types.

6.2.1 Model-agnostic Strategy

Model-agnostic strategy investigates the inner representation in DNNs by observing the sensitivity of the outputs to the inputs. A visualization technique has been introduced in [90], which reveals the inputs that contribute to activating outputs at any layer in the DNNs. The authors of [4] proposed a gradient-based method that provides local explanation vectors applicable to any classification method. The vectors help to understand prediction results for single data instances. The local interpretable model-agnostic explanations (LIME) method was introduced in [67]. LIME interprets the behavior of DNNs with the superimposition of simple models assuming the locally linearity of DNNs. Another model-agnostic interpretation method was introduced in [68], which provides the

high-precision if-then rules to represent sufficient conditions for explaining an individual prediction.

The model-agnostic strategy has been improved in many studies [21, 39, 51, 80]. It provides the local explainability, which reveals the reason for each prediction obtained by the prediction methods. The model-agnostic strategy has the advantage of few limitations to the applicable prediction methods due to the independence on the mechanism in the models. However, the independence property makes it difficult to delineate the inner representation in the prediction models. Thus, this thesis employed the model-specific strategy for trained DNNs. As the result, we achieved the detection of overfitted DNNs, which is difficult to achieve through the model-agnostic strategy.

6.2.2 Model-specific Strategy

Model-specific strategy is applied to a specific family of algorithms [50]. To investigate the similarity among the representations of DNNs, canonical correlation analysis (CCA) is used. CCA is a statistical method that measures the associations between two sets of variables. Singular vector [66] and projection-weighted CCA [56] have been proposed to apply CCA to DNNs by mitigating its sensitivity to parameter perturbations. The authors of [40] introduced centered kernel alignment (CKA) as a similarity index of DNNs, which is closely related to the CCA. The CKA can reliably identify the correspondence among representations in DNNs trained from different initializations. Previous studies enhanced statistical methods to mitigate the parameter perturbation in DNNs. However, PH is designed intrinsically to be robust with the parameter perturbation, which makes PH prominent for investigating the inner representation in DNNs.

Furthermore, network weights in DNNs are calibrated using the error gradient, which indicates the direction and magnitude for updating the weights to minimize the loss function [32]. The inner representation of DNNs can be investigated using the calibration technique. Gradient-based saliency detection methods have been studied [15], which aim to highlight salient regions in images, texts, and videos. A gradient-based attribution method was proposed in [73] computing saliency map corresponding to the gradient of the output neurons. Further, the authors of [45] developed the saliency model that outperforms state-

of-the-art models by a large margin using a the well-known network developed in [43]. In [3], layer-wise relevance propagation is proposed for understanding classification decisions by the pixel-wise decomposition of nonlinear classifiers. To backpropagate the activations in the output layer to the input layer of DNNs, the deep Taylor decomposition (DTC) was proposed [55]. The DTC calculates the relevance among neurons based on the gradient technique, which was used in this thesis for constructing clique complexes in DNNs. Additionally, this thesis provides a mathematical proof of the correctness of the construction and formulas for calculating PH in DNNs. The construction of clique complexes and the formalization in PH calculation provide a foundation for studying on the inner representation of DNNs using PH.

6.3 Conclusion and Future Work

This thesis used PH to examine the inner representation in DNNs to improve network behavior comprehension. Four contributions of (C1)–(C4) helped to realize the two goals of (G1) and (G2), as described in Section 6.1.

This thesis established an investigation approach for fully connected and convolutional layers in DNNs for the (G1: developing an approach to investigate the inner representation of DNNs using PH). However, several deep learning models have been proposed, which can be categorized into generative, discriminative, and hybrid architectures [76]. On the basis of this categorization, this study focused on the discriminative architecture implemented with fully-connected and convolutional layers. However, the discriminative architecture can also use recurrent neural networks and long short-time memory [36]. Furthermore, autoencoders [34] and deep Boltzmann machines [72] can be used to implement the generative and hybrid architectures. Thus, one of the future works is to enhance the proposed investigation approach to expand the applicable architectures.

Regarding the (G2: improving the comprehension of network behaviors using PH), this study applied the proposed investigation approach to the overfitting detection and DNN network pruning. Furthermore, the investigation of DNNs using PH can be utilized in other applications, for example, the authors of [93] applied PH to trojan network detection, and PH was used in the performance prediction of DNNs [16]. These studies

improved the comprehension of DNNs using the novel insights on DNNs provided by PH. Therefore, one of the future works is the development of PH's applications in DNNs.

In the investigation in G2, we observed that the PH diagram occasionally produced a line on the upper side, as described in Section 3. Although the outlier will represent some DNN characteristics, the meaning of the outlier remained unrevealed in this study. Moreover, we investigated the change in the PH diagram during the training. We confirmed that there was no major change in the PH diagram after the conversion of the accuracy and loss. This change can be examined by adjusting the learning rate. Furthermore, we attempted to calculate the upper and lower bounds of the number of homologies appeared in DNNs. However, because of the combination forming the homology, the number of homologies can be substantial. Hence, proper limitations will be required for estimating the upper and lower bounds. Therefore, we set the outlier, convergence, and bound investigations in the PH diagram for future works.

The integration and segregation of the brain network, including small-worldness, hubs, and modularity, have been investigated using topological data analysis [71]. Previous studies revealed that Alzheimer's disease causes the loss of small-worldness in brain networks [48, 65], which can be detected using PH [18, 44]. Considering the observation in human brains, the integration and segregation also play an important role in DNNs. For example, it is possible that the loss of small-worldness harms the functionality of DNNs. Thus, the relationship between DNNs' functionality and network structure is an interesting future work, in which PH will provide a powerful tool for the investigation.

Moreover, in this study, we had difficulties in computing the PH of DNNs. The computation approaches of PH have been enhanced both theoretically and practically [24, 52, 61]. However, the expensive computation restricted the target layers to compute PH in this study. Improvements in the computation approaches will enable a more comprehensive understanding of DNNs. Therefore, the improvement in the computation approach of PH was set as one of the future works.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *Advances in Neural Information Processing Systems*, 27, 2014.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

[5] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554. 2011.

[6] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565, 2014.

[7] Christopher M Bishop. *Pattern recognition and machine learning*. springer, Switzerland, 2006.

[8] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems*, volume 2, pages 129–146, 2020.

[9] Jean-Daniel Boissonnat and Clément Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70(3):406–427, 2014.

[10] Francesca Cagliari, Massimo Ferri, and Paola Pozzi. Size functions from a categorical viewpoint. *Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, 67(3):225–235, 2001.

[11] Zixuan Cang and Guo-Wei Wei. Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. *International Journal for Numerical Methods in Biomedical Engineering*, 34(2):e2914, 2018.

[12] B. Cassidy, F. D. Bowman, C. Rae, and V. Solo. On the reliability of individual brain activity networks. *IEEE Transactions on Medical Imaging*, 37(2):649–662, Feb 2018.

[13] Shih-Kang Chao, Zhanyu Wang, Yue Xing, and Guang Cheng. Directional pruning of deep neural networks. 33:13986–13998, 2020.

[14] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2017.

[15] Runmin Cong, Jianjun Lei, Huazhu Fu, Ming-Ming Cheng, Weisi Lin, and Qingming Huang. Review of visual saliency detection with comprehensive information. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(10):2941–2959, 2018.

[16] Ciprian A Corneanu, Sergio Escalera, and Aleix M Martinez. Computing the testing error without a testing set. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2677–2685, 2020.

[17] Carina Curto. What can topology tell us about the neural code? *Bulletin of the American Mathematical Society*, 54(1):63–78, 2017.

[18] Madelaine Daianu, Neda Jahanshad, Talia M Nir, Clifford R Jack Jr, Michael W Weiner, Matt A Bernstein, Paul M Thompson, and Alzheimer's Disease Neuroimaging Initiative. Rich club analysis in the alzheimer's disease connectome reveals a relatively undisturbed structural core network. *Human Brain Mapping*, 36(8):3087–3103, 2015.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[20] Rahul S Desikan, Florent Ségonne, Bruce Fischl, Brian T Quinn, Bradford C Dickerson, Deborah Blacker, Randy L Buckner, Anders M Dale, R Paul Maguire, Bradley T Hyman, et al. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *Neuroimage*, 31(3):968–980, 2006.

[21] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in Neural Information Processing Systems*, 31, 2018.

[22] Oxford Dictionary. Oxford dictionaries. *Language Matters*, 2014.

[23] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., Providence, Rhode Island, 2010.

[24] Herbert Edelsbrunner, John Harer, et al. Persistent homology-a survey. *Contemporary Mathematics*, 453:257–282, 2008.

[25] Herbert Edelsbrunner and Dmitriy Morozov. Persistent homology: theory and practice. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2012.

[26] Herbert Edelsbrunner and Afra Zomorodian. Computing linking numbers of a filtration. *Homology, Homotopy and Applications*, 5(2):19–37, 2003.

[27] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[28] Karl J Friston, Chris D Frith, Peter F Liddle, and Richard SJ Frackowiak. Functional connectivity: the principal-component analysis of large (pet) data sets. *Journal of Cerebral Blood Flow & Metabolism*, 13(1):5–14, 1993.

[29] Marcio Gameiro, Yasuaki Hiraoka, Shunsuke Izumi, Miroslav Kramar, Konstantin Mischaikow, and Vidit Nanda. A topological measurement of protein compressibility. *Japan Journal of Industrial and Applied Mathematics*, 32(1):1–17, 2015.

[30] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, 2017.

[31] Kathrin Grosse, Taesung Lee, Youngja Park, Michael Backes, and Ian M. Molloy. A new measure for overfitting and its implications for backdooring of deep learning. *CoRR*, abs/2006.06721, 2020.

[32] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, Birmingham, UK, 2017.

[33] William Grant Hatcher and Wei Yu. A survey of deep learning: platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018.

[34] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[35] Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[37] Danijela Horak, Slobodan Maletić, and Milan Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034, 2009.

[38] Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[39] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in Neural Information Processing Systems*, 29, 2016.

[40] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[41] M Kramar, A Goullet, L Kondic, and K Mischaikow. Persistence of force networks in compressed granular media. *Physical Review E-Statistical, Nonlinear, and Soft Matter Physics*, 87(4):042207, 2013.

[42] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's Thesis, University of Tront*, 2009.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[44] Liqun Kuang, Yan Gao, Zhongyu Chen, Jiacheng Xing, Fengguang Xiong, and Xie Han. White matter brain network research in alzheimer's disease using persistent features. *Molecules*, 25(11):2472, 2020.

[45] M Kümmerer, L Theis, and M Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. In *International Conference on Learning Representations*, 2014.

[46] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[47] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[48] Hyekyoung Lee, Moo K Chung, Hyejin Kang, Bung-Nyun Kim, and Dong Soo Lee. Discriminative persistent homology of brain networks. In *2011 IEEE International Symposium on Biomedical Imaging: from nano to macro*, pages 841–844, 2011.

[49] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[50] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2021.

[51] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[52] Nicholas O Malott, Rishi R Verma, Rohit P Singh, and Philip A Wilsey. Distributed computation of persistent homology from partitioned big data. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 344–354. IEEE, 2021.

[53] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

[54] Paolo Masulli and Alessandro EP Villa. The topology of the directed clique complex as a network invariant. *SpringerPlus*, 5(1):388, 2016.

[55] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[56] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems*, pages 5727–5736, 2018.

[57] Dmitriy Morozov. Dionysus library for computing persistent homology, 2012. *URL http://www. mrzv. org/software/dionysus*.

[58] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, pages 107–115, 1989.

[59] Supun Nakandala, Yuhao Zhang, and Arun Kumar. Cerebro: A data system for optimized deep learning model selection. *Proceedings of the VLDB Endowment*, 13(12):2159–2173, 2020.

[60] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[61] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, 2017.

[62] Giovanni Petri, Paul Expert, Federico Turkheimer, Robin Carhart-Harris, David Nutt, Peter J Hellyer, and Francesco Vaccarino. Homological scaffolds of brain functional networks. *Journal of The Royal Society Interface*, 11(101):20140873, 2014.

[63] Thorbjörn Posewsky and Daniel Ziener. A flexible fpga-based inference architecture for pruned deep neural networks. In *International Conference on Architecture of Computing Systems*, pages 311–323. Springer, 2018.

[64] Chi Seng Pun, Si Xian Lee, and Kelin Xia. Persistent-homology-based machine learning: a survey and a comparative study. *Artificial Intelligence Review*, pages 1–45, 2022.

[65] Tiantian Qiu, Xiao Luo, Zhujing Shen, Peiyu Huang, Xiaojun Xu, Jiong Zhou, Minming Zhang, Alzheimer's Disease Neuroimaging Initiative, et al. Disrupted brain network in progressive mild cognitive impairment measured by eigenvector centrality mapping is linked to cognition and cerebrospinal fluid biomarkers. *Journal of Alzheimer's Disease*, 54(4):1483–1493, 2016.

[66] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, pages 6076–6085, 2017.

[67] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[68] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[69] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *International Conference on Learning Representations*, 2018.

[70] Vanessa Robins. Towards computing homology from finite approximations. In *Topology Proceedings*, volume 24, pages 503–532, 1999.

[71] Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.

[72] Ruslan Salakhutdinov and Hugo Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700, 2010.

[73] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

[74] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[75] Ann E Sizemore, Chad Giusti, Ari Kahn, Jean M Vettel, Richard F Betzel, and Danielle S Bassett. Cliques and cavities in the human connectome. *Journal of Computational Neuroscience*, 44(1):115–145, 2018.

[76] S Smys, Joy Iong Zong Chen, and Subarna Shakya. Survey on neural network architectures with deep learning. *Journal of Soft Computing Paradigm (JSCP)*, 2(03):186–194, 2020.

[77] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[78] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at `http://appliedtopology.github.io/javaplex/`.

[79] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

[80] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[81] Larry Wasserman. Topological data analysis. *Annual Review of Statistics and Its Application*, 5:501–532, 2018.

[82] Satoru Watanabe and Hayato Yamana. Deep neural network pruning using persistent homology. In *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 153–156. IEEE, 2020.

[83] Satoru Watanabe and Hayato Yamana. Overfitting measurement of deep neural networks using no data. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2021.

[84] Satoru Watanabe and Hayato Yamana. Topological measurement of deep neural networks using persistent homology. *Annals of Mathematics and Artificial Intelligence*, 90(1):75–92, 2022.

[85] Roman Werpachowski, András György, and Csaba Szepesvari. Detecting overfitting via adversarial examples. In *Advances in Neural Information Processing Systems*, pages 7858–7868, 2019.

[86] Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International Journal for Numerical Methods in Biomedical Engineering*, 30(8):814–844, 2014.

[87] Jaejun Yoo, Eun Young Kim, Yong Min Ahn, and Jong Chul Ye. Topological persistence vineyard for dynamic functional brain connectivity during resting and gaming stages. *Journal of Neuroscience Methods*, 267:1–13, 2016.

[88] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[89] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[90] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[91] Qingchen Zhang, Laurence T Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[92] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2019.

[93] Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Appendix A: Publications

A.1 International Journals

Satoru Watanabe and Hayato Yamana, Overfitting Measurement of Convolutional Neural Networks Using Trained Network Weights, International Journal of Data Science and Analytics, pp. 1–18, Springer, 2022, doi: 10.1007/s41060-022-00332-1.

Satoru Watanabe and Hayato Yamana, Topological Measurement of Deep Neural Networks Using Persistent Homology. Annals of Mathematics and Artificial Intelligence 90(1) pp. 75–92, Springer, January 2022, doi: 10.1007/s10472-021-09761-3.

A.2 International Conferences

Satoru Watanabe and Hayato Yamana, Overfitting Measurement of Deep Neural Networks Using No Data, IEEE 8th International Conference on Data Science and Advanced Analytics, pp. 1–10, October 2021, doi: 10.1109/DSAA53316.2021.9564119.

Satoru Watanabe and Hayato Yamana, Overfitting Measurement of Deep Neural Networks Homology, IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering, pp. 153–156, December 2020, doi: 10.1109/AIKE48582.2020.00030.

Satoru Watanabe and Hayato Yamana, Topological Measurement of Deep Neural Networks Using Persistent Homology, International Symposium on Artificial Intelligence and Mathematics, January 2020.

Appendix B: Row Data of Figures

B.1 Row Data of Figures in Chapter 3

Table B.1 shows the row data of Fig. 3.5.

Table B.1: row data of Figs. 3.5 (a)–(c).

	number of classification objects	size of convex hull mininum	average	maximum
(a) MNIST	10	279	317	389
	9	476	635	749
	8	474	641	882
	7	496	601	757
	6	558	695	935
	5	575	684	852
	4	637	744	932
	3	441	652	857
	2	327	499	597
	1	232	264	388
(b) CIFAR10 (300–100)	10	282	361	469
	9	280	469	714
	8	300	575	754
	7	486	670	889
	6	608	767	1006
	5	465	763	900
	4	641	836	969
	3	383	833	1008
	2	417	581	973
	1	218	318	430
(c) CIAFAR10 (512–512)	10	421	491	546
	9	460	736	898
	8	711	851	954
	7	813	913	1029
	6	897	949	990
	5	888	968	1017
	4	837	947	1027
	3	978	1031	1081
	2	783	1009	1098
	1	164	222	298

B.2 Row Data of Figures in Chapter 4

Tables B.2–B.5 show the row data of Fig. 4.3.

Table B.2: row data of Figs. 4.3 (a-1) and (a-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.387	0.510	1.691	1.363
	2	0.589	0.576	1.160	1.212
	3	0.683	0.661	0.906	0.970
	4	0.749	0.714	0.716	0.833
	5	0.806	0.738	0.559	0.809
	6	0.856	0.703	0.417	1.041
	7	0.896	0.731	0.295	1.003
	8	0.926	0.738	0.214	1.188
	9	0.944	0.734	0.164	1.354
	10	0.955	0.722	0.135	1.448
	11	0.963	0.739	0.112	1.595
	12	0.965	0.720	0.109	1.622
	13	0.968	0.723	0.103	1.863
	14	0.970	0.717	0.099	1.684
CIFAR10	15	0.973	0.726	0.085	2.062
a-1, a-2	16	0.972	0.734	0.092	1.942
	17	0.973	0.722	0.088	1.781
	18	0.973	0.725	0.091	2.183
	19	0.974	0.739	0.086	2.518
	20	0.976	0.686	0.081	2.131
	21	0.975	0.730	0.088	2.022
	22	0.974	0.733	0.090	1.753
	23	0.974	0.725	0.090	2.559
	24	0.974	0.725	0.093	2.260
	25	0.973	0.728	0.096	2.088
	26	0.973	0.711	0.094	2.486
	27	0.972	0.736	0.105	2.091
	28	0.972	0.727	0.099	2.397
	29	0.971	0.743	0.109	2.182
	30	0.973	0.745	0.099	2.376

Table B.3: row data of Figs. 4.3 (b-1) and (b-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.351	0.454	1.778	1.573
	2	0.538	0.614	1.300	1.093
	3	0.628	0.635	1.061	1.050
	4	0.686	0.686	0.901	0.893
	5	0.724	0.737	0.791	0.761
	6	0.756	0.672	0.703	0.983
	7	0.776	0.759	0.644	0.719
	8	0.797	0.762	0.586	0.721
	9	0.811	0.766	0.544	0.700
	10	0.820	0.774	0.519	0.677
	11	0.830	0.749	0.496	0.787
	12	0.837	0.779	0.477	0.685
	13	0.845	0.733	0.453	0.850
	14	0.844	0.783	0.456	0.663
CIFAR10	15	0.853	0.785	0.435	0.650
b-1, b-2	16	0.852	0.781	0.437	0.657
	17	0.855	0.757	0.432	0.767
	18	0.857	0.776	0.434	0.667
	19	0.858	0.772	0.425	0.786
	20	0.859	0.761	0.424	0.712
	21	0.861	0.788	0.422	0.715
	22	0.862	0.771	0.416	0.739
	23	0.865	0.793	0.414	0.653
	24	0.866	0.777	0.411	0.662
	25	0.866	0.776	0.412	0.684
	26	0.865	0.795	0.411	0.659
	27	0.865	0.763	0.411	0.712
	28	0.863	0.762	0.418	0.804
	29	0.865	0.763	0.411	0.701
	30	0.865	0.773	0.415	0.676

Table B.4: row data of Figs. 4.3 (c-1) and (c-2).

	epoch	accuracy		loss	
		training	validation	training	validation
CIFAR10 c-1, c-2	1	0.304	0.448	1.888	1.555
	2	0.464	0.531	1.486	1.294
	3	0.539	0.581	1.300	1.201
	4	0.588	0.656	1.172	0.984
	5	0.624	0.672	1.077	0.927
	6	0.651	0.685	1.003	0.902
	7	0.668	0.665	0.952	0.963
	8	0.681	0.720	0.919	0.798
	9	0.696	0.735	0.890	0.767
	10	0.699	0.739	0.875	0.759
	11	0.711	0.714	0.848	0.856
	12	0.709	0.747	0.846	0.755
	13	0.715	0.752	0.838	0.738
	14	0.714	0.702	0.839	0.851
	15	0.720	0.718	0.830	0.854
	16	0.719	0.730	0.834	0.803
	17	0.724	0.738	0.819	0.781
	18	0.721	0.748	0.825	0.755
	19	0.721	0.730	0.815	0.839
	20	0.724	0.755	0.820	0.733
	21	0.725	0.729	0.823	0.835
	22	0.723	0.727	0.827	0.833
	23	0.724	0.769	0.816	0.690
	24	0.724	0.738	0.823	0.818
	25	0.726	0.752	0.826	0.785
	26	0.723	0.679	0.822	0.925
	27	0.722	0.729	0.822	0.815
	28	0.723	0.709	0.824	0.897
	29	0.727	0.749	0.820	0.767
	30	0.724	0.748	0.824	0.814

Table B.5: row data of Figs. 4.3 (d-1) and (d-2).

	epoch	accuracy		loss	
		training	validation	training	validation
CIFAR10 d-1, d-2	1	0.239	0.245	2.019	2.093
	2	0.385	0.382	1.674	1.747
	3	0.438	0.460	1.543	1.563
	4	0.471	0.389	1.466	1.820
	5	0.498	0.452	1.413	1.605
	6	0.513	0.521	1.378	1.437
	7	0.527	0.452	1.339	1.636
	8	0.535	0.460	1.318	1.596
	9	0.543	0.430	1.298	1.624
	10	0.549	0.495	1.290	1.492
	11	0.553	0.555	1.275	1.415
	12	0.559	0.489	1.266	1.491
	13	0.565	0.446	1.256	1.596
	14	0.566	0.473	1.257	1.575
	15	0.564	0.573	1.256	1.382
	16	0.565	0.574	1.253	1.321
	17	0.566	0.451	1.254	1.610
	18	0.569	0.506	1.251	1.482
	19	0.573	0.503	1.239	1.509
	20	0.572	0.494	1.245	1.497
	21	0.571	0.502	1.244	1.440
	22	0.571	0.620	1.249	1.178
	23	0.567	0.572	1.252	1.289
	24	0.572	0.586	1.249	1.299
	25	0.570	0.589	1.250	1.241
	26	0.570	0.572	1.248	1.247
	27	0.570	0.603	1.251	1.275
	28	0.568	0.640	1.253	1.188
	29	0.567	0.545	1.258	1.328
	30	0.567	0.622	1.257	1.166

Tables B.6–B.9 show the row data of Fig. 4.4.

Table B.6: row data of Figs. 4.4 (a-1) and (a-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.716	0.882	0.871	0.388
	2	0.902	0.907	0.328	0.317
	3	0.932	0.912	0.233	0.319
	4	0.951	0.922	0.171	0.305
	5	0.964	0.918	0.126	0.322
	6	0.973	0.922	0.092	0.342
	7	0.979	0.916	0.072	0.381
	8	0.982	0.912	0.061	0.387
	9	0.984	0.918	0.054	0.591
	10	0.986	0.921	0.049	0.528
	11	0.986	0.924	0.047	0.605
	12	0.987	0.922	0.046	0.527
	13	0.988	0.913	0.041	0.804
	14	0.987	0.901	0.047	0.470
SVHN	15	0.988	0.915	0.043	0.828
a-1, a-2	16	0.989	0.913	0.043	0.624
	17	0.988	0.913	0.044	0.744
	18	0.987	0.920	0.047	0.823
	19	0.988	0.910	0.047	0.571
	20	0.987	0.916	0.049	0.730
	21	0.988	0.913	0.045	0.694
	22	0.989	0.901	0.046	0.747
	23	0.988	0.920	0.044	0.811
	24	0.989	0.907	0.045	1.514
	25	0.989	0.917	0.045	0.726
	26	0.988	0.906	0.047	0.849
	27	0.988	0.915	0.051	0.561
	28	0.987	0.915	0.052	0.811
	29	0.986	0.902	0.054	0.533
	30	0.988	0.901	0.054	0.672

Table B.7: row data of Figs. 4.4 (b-1) and (b-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.685	0.862	0.955	0.458
	2	0.876	0.903	0.405	0.328
	3	0.902	0.916	0.323	0.286
	4	0.918	0.921	0.275	0.272
	5	0.927	0.922	0.245	0.272
	6	0.933	0.926	0.228	0.280
	7	0.938	0.928	0.213	0.265
	8	0.942	0.930	0.201	0.277
	9	0.944	0.928	0.193	0.293
	10	0.946	0.929	0.188	0.261
	11	0.946	0.932	0.185	0.310
	12	0.947	0.924	0.181	0.308
	13	0.947	0.931	0.183	0.255
	14	0.950	0.924	0.177	0.295
SVHN	15	0.948	0.932	0.178	0.256
b-1, b-2	16	0.949	0.920	0.179	0.300
	17	0.949	0.919	0.179	0.319
	18	0.949	0.931	0.178	0.287
	19	0.949	0.930	0.179	0.278
	20	0.949	0.928	0.178	0.263
	21	0.949	0.930	0.178	0.307
	22	0.948	0.934	0.183	0.250
	23	0.949	0.934	0.182	0.275
	24	0.949	0.937	0.176	0.293
	25	0.948	0.930	0.178	0.284
	26	0.947	0.931	0.183	0.281
	27	0.949	0.929	0.178	0.274
	28	0.948	0.933	0.180	0.267
	29	0.949	0.930	0.177	0.263
	30	0.948	0.931	0.185	0.260

Table B.8: row data of Figs. 4.4 (c-1) and (c-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.575	0.816	1.259	0.610
	2	0.819	0.873	0.575	0.433
	3	0.858	0.897	0.464	0.346
	4	0.874	0.889	0.416	0.372
	5	0.883	0.912	0.387	0.310
	6	0.889	0.915	0.371	0.340
	7	0.890	0.924	0.366	0.283
	8	0.894	0.923	0.364	0.314
	9	0.896	0.925	0.355	0.279
	10	0.896	0.925	0.351	0.295
	11	0.898	0.925	0.353	0.268
	12	0.898	0.920	0.350	0.339
	13	0.900	0.914	0.348	0.347
	14	0.900	0.923	0.347	0.289
SVHN	15	0.898	0.920	0.354	0.321
c-1, c-2	16	0.898	0.918	0.351	0.351
	17	0.899	0.925	0.348	0.273
	18	0.897	0.922	0.353	0.278
	19	0.900	0.918	0.351	0.330
	20	0.897	0.919	0.353	0.296
	21	0.896	0.923	0.358	0.282
	22	0.898	0.917	0.356	0.307
	23	0.896	0.909	0.362	0.370
	24	0.893	0.917	0.367	0.306
	25	0.894	0.923	0.361	0.290
	26	0.895	0.902	0.365	0.413
	27	0.895	0.917	0.364	0.288
	28	0.896	0.918	0.362	0.302
	29	0.892	0.914	0.369	0.331
	30	0.892	0.919	0.373	0.316

Table B.9: row data of Figs. 4.4 (d-1) and (d-2).

	epoch	accuracy		loss	
		training	validation	training	validation
	1	0.346	0.699	1.847	1.020
	2	0.698	0.815	0.943	0.638
	3	0.776	0.859	0.720	0.490
	4	0.799	0.876	0.653	0.474
	5	0.812	0.875	0.617	0.510
	6	0.820	0.884	0.597	0.530
	7	0.823	0.885	0.592	0.524
	8	0.824	0.888	0.592	0.526
	9	0.826	0.879	0.593	0.603
	10	0.826	0.878	0.589	0.567
	11	0.827	0.878	0.590	0.685
	12	0.828	0.840	0.591	0.763
	13	0.828	0.872	0.587	0.624
	14	0.826	0.878	0.593	0.595
SVHN	15	0.825	0.855	0.595	0.739
d-1, d-2	16	0.822	0.878	0.610	0.599
	17	0.822	0.857	0.611	0.725
	18	0.823	0.875	0.611	0.612
	19	0.823	0.803	0.609	1.016
	20	0.819	0.862	0.616	0.705
	21	0.817	0.860	0.625	0.646
	22	0.818	0.876	0.625	0.528
	23	0.815	0.868	0.632	0.726
	24	0.813	0.860	0.639	0.688
	25	0.814	0.883	0.634	0.516
	26	0.812	0.887	0.651	0.477
	27	0.811	0.882	0.646	0.612
	28	0.808	0.868	0.651	0.591
	29	0.809	0.877	0.651	0.602
	30	0.808	0.846	0.656	0.648

Tables B.10–B.13 show the row data of Fig. 4.5.

Table B.10: row data of Figs. 4.5 (a-1) and (a-2).

	epoch	accuracy		loss	
		training	validation	training	validation
TinyImageNet a-1, a-2	1	0.238	0.274	3.421	3.145
	2	0.350	0.312	2.759	2.966
	3	0.406	0.310	2.479	3.103
	4	0.453	0.326	2.246	3.022
	5	0.499	0.327	2.030	3.091
	6	0.543	0.319	1.825	3.375
	7	0.581	0.315	1.639	3.515
	8	0.620	0.324	1.461	3.602
	9	0.659	0.314	1.297	3.814
	10	0.693	0.311	1.146	4.243
	11	0.726	0.302	1.008	4.518
	12	0.755	0.308	0.891	4.475
	13	0.781	0.301	0.785	4.896
	14	0.805	0.300	0.690	5.090
	15	0.828	0.299	0.603	5.543
	16	0.846	0.292	0.532	5.614
	17	0.863	0.303	0.470	5.894
	18	0.877	0.297	0.417	6.281
	19	0.888	0.291	0.373	6.379
	20	0.900	0.297	0.331	6.660
	21	0.910	0.293	0.297	6.993
	22	0.917	0.297	0.270	7.333
	23	0.923	0.299	0.249	7.327
	24	0.928	0.292	0.230	7.607
	25	0.934	0.290	0.210	7.839
	26	0.937	0.295	0.199	8.049
	27	0.942	0.288	0.186	8.381
	28	0.944	0.293	0.178	8.589
	29	0.948	0.290	0.166	9.054
	30	0.950	0.297	0.156	9.136

Table B.11: row data of Figs. 4.5 (b-1) and (b-2).

	epoch	accuracy		loss	
		training	validation	training	validation
TinyImageNet b-1, b-2	1	0.203	0.275	3.638	3.159
	2	0.294	0.301	3.070	3.061
	3	0.327	0.309	2.908	3.007
	4	0.345	0.316	2.814	2.977
	5	0.361	0.323	2.744	3.000
	6	0.372	0.315	2.692	3.042
	7	0.382	0.314	2.655	3.094
	8	0.389	0.312	2.625	3.141
	9	0.395	0.309	2.601	3.153
	10	0.401	0.319	2.577	3.273
	11	0.405	0.306	2.572	3.346
	12	0.408	0.302	2.561	3.318
	13	0.412	0.299	2.552	3.398
	14	0.417	0.295	2.536	3.417
	15	0.418	0.301	2.536	3.471
	16	0.424	0.296	2.532	3.580
	17	0.426	0.288	2.534	3.627
	18	0.428	0.282	2.531	3.699
	19	0.426	0.287	2.551	3.701
	20	0.428	0.284	2.552	3.815
	21	0.429	0.276	2.555	3.903
	22	0.430	0.281	2.560	3.971
	23	0.431	0.280	2.573	3.989
	24	0.433	0.267	2.574	4.091
	25	0.432	0.273	2.585	4.129
	26	0.432	0.264	2.598	4.169
	27	0.433	0.270	2.609	4.163
	28	0.433	0.260	2.619	4.225
	29	0.432	0.263	2.629	4.304
	30	0.433	0.265	2.631	4.336

Table B.12: row data of Figs. 4.5 (c-1) and (c-2).

	epoch	accuracy		loss	
		training	validation	training	validation
TinyImageNet c-1, c-2	1	0.154	0.251	3.947	3.301
	2	0.232	0.280	3.446	3.179
	3	0.247	0.289	3.383	3.158
	4	0.251	0.290	3.388	3.171
	5	0.252	0.290	3.406	3.185
	6	0.252	0.275	3.429	3.275
	7	0.250	0.284	3.468	3.233
	8	0.245	0.273	3.510	3.334
	9	0.245	0.266	3.543	3.378
	10	0.240	0.261	3.590	3.400
	11	0.238	0.257	3.625	3.444
	12	0.237	0.249	3.663	3.435
	13	0.233	0.231	3.691	3.517
	14	0.227	0.238	3.724	3.525
	15	0.226	0.234	3.740	3.532
	16	0.222	0.229	3.766	3.576
	17	0.220	0.228	3.797	3.599
	18	0.220	0.224	3.811	3.610
	19	0.219	0.220	3.824	3.606
	20	0.213	0.221	3.840	3.607
	21	0.216	0.214	3.857	3.677
	22	0.214	0.215	3.859	3.612
	23	0.213	0.219	3.861	3.647
	24	0.211	0.212	3.871	3.658
	25	0.210	0.212	3.892	3.661
	26	0.209	0.219	3.886	3.661
	27	0.206	0.208	3.896	3.641
	28	0.205	0.208	3.907	3.690
	29	0.207	0.221	3.901	3.633
	30	0.205	0.215	3.903	3.656

Table B.13: row data of Figs. 4.5 (d-1) and (d-2).

	epoch	accuracy		loss	
		training	validation	training	validation
TinyImageNet d-1, d-2	1	0.088	0.214	4.436	3.715
	2	0.144	0.231	4.007	3.583
	3	0.149	0.234	4.025	3.677
	4	0.147	0.223	4.090	3.687
	5	0.142	0.198	4.167	3.863
	6	0.135	0.194	4.242	3.908
	7	0.130	0.170	4.312	4.028
	8	0.126	0.162	4.367	4.042
	9	0.122	0.161	4.409	4.088
	10	0.118	0.141	4.439	4.137
	11	0.114	0.136	4.476	4.153
	12	0.114	0.147	4.487	4.101
	13	0.111	0.139	4.510	4.122
	14	0.110	0.139	4.501	4.146
	15	0.109	0.130	4.510	4.164
	16	0.105	0.135	4.534	4.118
	17	0.107	0.138	4.535	4.120
	18	0.105	0.140	4.545	4.110
	19	0.105	0.142	4.544	4.096
	20	0.105	0.143	4.547	4.102
	21	0.103	0.138	4.557	4.085
	22	0.103	0.138	4.575	4.116
	23	0.102	0.127	4.578	4.180
	24	0.100	0.124	4.594	4.233
	25	0.100	0.125	4.622	4.181
	26	0.101	0.127	4.620	4.193
	27	0.101	0.130	4.628	4.155
	28	0.098	0.131	4.644	4.130
	29	0.098	0.126	4.651	4.175
	30	0.095	0.120	4.678	4.233

Tables B.14–B.17 show the row data of Fig. 4.6.

Table B.14: row data of Fig. 4.6 (a).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	132775	134463	0.35–	0	0	0.70–	0	0
0.01–	5211	5386	0.36–	0	0	0.71–	0	0
0.02–	6	13	0.37–	0	0	0.72–	0	0
0.03–	0	1	0.38–	0	0	0.73–	0	0
0.04–	0	0	0.39–	0	0	0.74–	0	0
0.05–	0	0	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.15: row data of Fig. 4.6 (b).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	129849	130626	0.35–	0	0	0.70–	0	0
0.01–	6267	6574	0.36–	0	0	0.71–	0	0
0.02–	34	101	0.37–	0	0	0.72–	0	0
0.03–	0	9	0.38–	0	0	0.73–	0	0
0.04–	0	2	0.39–	0	0	0.74–	0	0
0.05–	0	1	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.16: row data of Fig. 4.6 (c).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	148489	148586	0.35–	0	0	0.70–	0	0
0.01–	4101	4164	0.36–	0	0	0.71–	0	0
0.02–	29	77	0.37–	0	0	0.72–	0	0
0.03–	0	31	0.38–	0	0	0.73–	0	0
0.04–	0	26	0.39–	0	0	0.74–	0	0
0.05–	0	14	0.40–	0	0	0.75–	0	0
0.06–	0	15	0.41–	0	0	0.76–	0	0
0.07–	0	5	0.42–	0	0	0.77–	0	0
0.08–	0	4	0.43–	0	0	0.78–	0	0
0.09–	0	1	0.44–	0	0	0.79–	0	0
0.10–	0	2	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	2	0.47–	0	0	0.82–	0	0
0.13–	0	2	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	3	0.50–	0	0	0.85–	0	0
0.16–	0	3	0.51–	0	0	0.86–	0	0
0.17–	0	2	0.52–	0	0	0.87–	0	0
0.18–	0	1	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	2	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.17: row data of Fig. 4.6 (d).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	155913	155915	0.35–	0	0	0.70–	0	0
0.01–	3526	3531	0.36–	0	1	0.71–	0	0
0.02–	39	44	0.37–	0	0	0.72–	0	1
0.03–	0	5	0.38–	0	0	0.73–	0	0
0.04–	0	2	0.39–	0	0	0.74–	0	0
0.05–	0	4	0.40–	0	1	0.75–	0	0
0.06–	0	1	0.41–	0	0	0.76–	0	0
0.07–	0	1	0.42–	0	0	0.77–	0	0
0.08–	0	2	0.43–	0	0	0.78–	0	0
0.09–	0	3	0.44–	0	0	0.79–	0	0
0.10–	0	2	0.45–	0	0	0.80–	0	0
0.11–	0	1	0.46–	0	0	0.81–	0	0
0.12–	0	5	0.47–	0	0	0.82–	0	0
0.13–	0	2	0.48–	0	0	0.83–	0	0
0.14–	0	2	0.49–	0	0	0.84–	0	0
0.15–	0	1	0.50–	0	0	0.85–	0	0
0.16–	0	4	0.51–	0	0	0.86–	0	0
0.17–	0	3	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	2	0.54–	0	0	0.89–	0	0
0.20–	0	1	0.55–	0	0	0.90–	0	0
0.21–	0	1	0.56–	0	0	0.91–	0	1
0.22–	0	2	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	1	0.61–	0	0	0.96–	0	0
0.27–	0	1	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	1	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	1	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Tables B.18–B.21 show the row data of Fig. 4.7.

Table B.18: row data of Fig. 4.7 (a).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	1047176	1065754	0.35–	0	0	0.70–	0	0
0.01–	0	8496	0.36–	0	0	0.71–	0	0
0.02–	0	0	0.37–	0	0	0.72–	0	0
0.03–	0	0	0.38–	0	0	0.73–	0	0
0.04–	0	0	0.39–	0	0	0.74–	0	0
0.05–	0	0	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.19: row data of Fig. 4.7 (b).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	1025407	1032444	0.35–	0	0	0.70–	0	0
0.01–	0	6007	0.36–	0	0	0.71–	0	0
0.02–	0	2930	0.37–	0	0	0.72–	0	0
0.03–	0	228	0.38–	0	0	0.73–	0	0
0.04–	0	2	0.39–	0	0	0.74–	0	0
0.05–	0	0	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.20: row data of Fig. 4.7 (c).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	970559	972653	0.35–	0	0	0.70–	0	0
0.01–	0	1871	0.36–	0	0	0.71–	0	0
0.02–	0	1669	0.37–	0	0	0.72–	0	0
0.03–	0	1313	0.38–	0	0	0.73–	0	0
0.04–	0	859	0.39–	0	0	0.74–	0	0
0.05–	0	413	0.40–	0	0	0.75–	0	0
0.06–	0	149	0.41–	0	0	0.76–	0	0
0.07–	0	46	0.42–	0	0	0.77–	0	0
0.08–	0	15	0.43–	0	0	0.78–	0	0
0.09–	0	2	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.21: row data of Fig. 4.7 (d).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	886863	887118	0.35–	0	1	0.70–	0	0
0.01–	1	291	0.36–	0	0	0.71–	0	0
0.02–	0	255	0.37–	0	0	0.72–	0	0
0.03–	0	218	0.38–	0	0	0.73–	0	0
0.04–	0	241	0.39–	0	1	0.74–	0	0
0.05–	0	217	0.40–	0	0	0.75–	0	0
0.06–	0	206	0.41–	0	1	0.76–	0	0
0.07–	0	192	0.42–	0	0	0.77–	0	0
0.08–	0	172	0.43–	0	0	0.78–	0	0
0.09–	0	153	0.44–	0	0	0.79–	0	0
0.10–	0	124	0.45–	0	0	0.80–	0	0
0.11–	0	109	0.46–	0	0	0.81–	0	0
0.12–	0	88	0.47–	0	0	0.82–	0	0
0.13–	0	76	0.48–	0	0	0.83–	0	0
0.14–	0	52	0.49–	0	0	0.84–	0	0
0.15–	0	55	0.50–	0	0	0.85–	0	0
0.16–	0	45	0.51–	0	0	0.86–	0	0
0.17–	0	24	0.52–	0	0	0.87–	0	0
0.18–	0	28	0.53–	0	0	0.88–	0	0
0.19–	0	25	0.54–	0	0	0.89–	0	0
0.20–	0	13	0.55–	0	0	0.90–	0	0
0.21–	0	9	0.56–	0	0	0.91–	0	0
0.22–	0	8	0.57–	0	0	0.92–	0	0
0.23–	0	6	0.58–	0	0	0.93–	0	0
0.24–	0	6	0.59–	0	0	0.94–	0	0
0.25–	0	7	0.60–	0	0	0.95–	0	0
0.26–	0	5	0.61–	0	0	0.96–	0	0
0.27–	0	2	0.62–	0	0	0.97–	0	0
0.28–	0	1	0.63–	0	0	0.98–	0	0
0.29–	0	1	0.64–	0	0	0.99–	0	0
0.30–	0	2	0.65–	0	0			
0.31–	0	1	0.66–	0	0			
0.32–	0	1	0.67–	0	0			
0.33–	0	1	0.68–	0	0			
0.34–	0	1	0.69–	0	0			

Tables B.22–B.25 show the row data of Fig. 4.8.

Table B.22: row data of Fig. 4.8 (a).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	2079249	4090364	0.35–	0	0	0.70–	0	0
0.01–	462368	536349	0.36–	0	0	0.71–	0	0
0.02–	39424	39424	0.37–	0	0	0.72–	0	0
0.03–	1233	1233	0.38–	0	0	0.73–	0	0
0.04–	0	0	0.39–	0	0	0.74–	0	0
0.05–	0	0	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.23: row data of Fig. 4.8 (b).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	1923489	3870941	0.35–	0	0	0.70–	0	0
0.01–	478423	570137	0.36–	0	0	0.71–	0	0
0.02–	50144	50820	0.37–	0	0	0.72–	0	0
0.03–	1764	1764	0.38–	0	0	0.73–	0	0
0.04–	0	0	0.39–	0	0	0.74–	0	0
0.05–	0	0	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.24: row data of Fig. 4.8 (c).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	1843412	3841334	0.35–	0	0	0.70–	0	0
0.01–	469760	570411	0.36–	0	0	0.71–	0	0
0.02–	61771	63630	0.37–	0	0	0.72–	0	0
0.03–	5194	5194	0.38–	0	0	0.73–	0	0
0.04–	196	196	0.39–	0	0	0.74–	0	0
0.05–	196	196	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.25: row data of Fig. 4.8 (d).

range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer	range	frequency first layer	frequency second layer
0.00–	2018898	4447654	0.35–	0	0	0.70–	0	0
0.01–	425182	490994	0.36–	0	0	0.71–	0	0
0.02–	67854	70558	0.37–	0	0	0.72–	0	0
0.03–	10112	10450	0.38–	0	0	0.73–	0	0
0.04–	1806	1806	0.39–	0	0	0.74–	0	0
0.05–	392	392	0.40–	0	0	0.75–	0	0
0.06–	0	0	0.41–	0	0	0.76–	0	0
0.07–	0	0	0.42–	0	0	0.77–	0	0
0.08–	0	0	0.43–	0	0	0.78–	0	0
0.09–	0	0	0.44–	0	0	0.79–	0	0
0.10–	0	0	0.45–	0	0	0.80–	0	0
0.11–	0	0	0.46–	0	0	0.81–	0	0
0.12–	0	0	0.47–	0	0	0.82–	0	0
0.13–	0	0	0.48–	0	0	0.83–	0	0
0.14–	0	0	0.49–	0	0	0.84–	0	0
0.15–	0	0	0.50–	0	0	0.85–	0	0
0.16–	0	0	0.51–	0	0	0.86–	0	0
0.17–	0	0	0.52–	0	0	0.87–	0	0
0.18–	0	0	0.53–	0	0	0.88–	0	0
0.19–	0	0	0.54–	0	0	0.89–	0	0
0.20–	0	0	0.55–	0	0	0.90–	0	0
0.21–	0	0	0.56–	0	0	0.91–	0	0
0.22–	0	0	0.57–	0	0	0.92–	0	0
0.23–	0	0	0.58–	0	0	0.93–	0	0
0.24–	0	0	0.59–	0	0	0.94–	0	0
0.25–	0	0	0.60–	0	0	0.95–	0	0
0.26–	0	0	0.61–	0	0	0.96–	0	0
0.27–	0	0	0.62–	0	0	0.97–	0	0
0.28–	0	0	0.63–	0	0	0.98–	0	0
0.29–	0	0	0.64–	0	0	0.99–	0	0
0.30–	0	0	0.65–	0	0			
0.31–	0	0	0.66–	0	0			
0.32–	0	0	0.67–	0	0			
0.33–	0	0	0.68–	0	0			
0.34–	0	0	0.69–	0	0			

Table B.26 shows the row data of Fig. 4.9.

Table B.26: row data of Fig. 4.9.

		dropout ratio			
		0.0	0.2	0.4	0.6
a	(200,200,10)	0.0006	0.0056	0.0995	0.6988
	(500,500,10)	0.0000	0.0002	0.0310	0.5226
	(1000,1000,10)	0.0000	0.0000	0.0080	0.578
b	(512,500,200)	0.0263	0.2811	0.6199	0.7317
	(512,1000,200)	0.0000	0.0039	0.2080	0.6635
	(512,2000,200)	0.0000	0.0000	0.0005	0.4154
c	(200,200,10)	0.0006	0.0056	0.0995	0.6988
	N(500,500,10)	0.0032	0.0125	0.2820	0.7320
	N(1000,1000,10)	0.0095	0.0690	0.3310	0.6620
d	(512,500,200)	0.0263	0.2811	0.6199	0.7317
	N(512,1000,200)	0.0090	0.2558	0.5393	0.7034
	N(512,2000,200)	0.0144	0.2396	0.4079	0.7753

Table B.27 shows the row data of Fig. 4.10.

Table B.27: row data of Fig. 4.10.

		dropout ratio			
		0.0	0.2	0.4	0.6
a	(1000,1000,100)	0.3098	0.7946	0.8912	0.9903
	(2000,2000,100)	0.1022	0.6751	0.7808	0.7277
	(4000,4000,100)	0.0036	0.0560	0.0990	0.5252
b	(512,1000,100)	0.0927	0.8299	0.8897	0.9903
	(512,2000,100)	0.0161	0.7109	0.7560	1.0000
	(512,4000,100)	0.0006	0.1215	0.4964	0.8951
c	(1000,1000,100)	0.3098	0.7946	0.8912	0.9903
	N(2000,2000,100)	0.4284	0.7509	0.8094	0.8028
	N(4000,4000,100)	0.3482	0.6380	0.6726	0.7119
d	(512,1000,100)	0.0927	0.8299	0.8897	0.9903
	N(512,2000,100)	0.0833	0.8902	0.8824	NA
	N(512,4000,100)	0.0737	0.6337	0.8091	NA

B.3 Row Data of Figures in Chapter 5

Table B.28 shows the row data of Fig. 5.1.

Table B.28: row data of Fig. 5.1.

	pruning ratio					
	0.0	0.6	0.7	0.8	0.9	0.95
all	0.77	0.77	0.76	0.74	0.66	0.43
airplane	0.79	0.79	0.80	0.80	0.63	0.08
mobile	0.90	0.85	0.84	0.77	0.65	0.28
bird	0.57	0.59	0.53	0.43	0.40	0.42
cat	0.57	0.65	0.65	0.66	0.71	0.82
deer	0.72	0.74	0.77	0.78	0.74	0.53
dog	0.77	0.68	0.70	0.70	0.57	0.38
frog	0.85	0.84	0.85	0.88	0.93	0.90
horse	0.87	0.84	0.80	0.72	0.53	0.33
ship	0.81	0.82	0.82	0.80	0.70	0.35
truck	0.86	0.88	0.86	0.85	0.73	0.23

Table B.29 shows the row data of Fig. 5.2.

Table B.29: row data of Fig. 5.2.

	pruning ratio					
	0.0	0.6	0.7	0.8	0.9	0.95
all	0.77	0.77	0.74	0.74	0.69	0.55
airplane	0.79	0.78	0.79	0.79	0.63	0.25
mobile	0.90	0.84	0.70	0.73	0.56	0.44
bird	0.57	0.55	0.43	0.44	0.47	0.60
cat	0.57	0.62	0.59	0.64	0.62	0.65
deer	0.72	0.77	0.73	0.77	0.73	0.58
dog	0.77	0.75	0.79	0.71	0.69	0.37
frog	0.85	0.85	0.87	0.88	0.91	0.94
horse	0.87	0.81	0.79	0.72	0.57	0.33
ship	0.81	0.82	0.82	0.82	0.86	0.69
truck	0.86	0.88	0.91	0.88	0.86	0.70