



# Automated Hyperparameter Optimization and Novel Techniques for Reduced Computational Cost of Deep Neural Networks

坂井, 靖文

---

(Degree)

博士 (工学)

(Date of Degree)

2025-03-25

(Date of Publication)

2026-03-01

(Resource Type)

doctoral thesis

(Report Number)

甲第9219号

(URL)

<https://hdl.handle.net/20.500.14094/0100496500>

※ 当コンテンツは神戸大学の学術成果です。無断複製・不正使用等を禁じます。著作権法で認められている範囲内で、適切にご利用ください。



Doctoral Dissertation

Automated Hyperparameter Optimization and  
Novel Techniques for Reduced Computational  
Cost of Deep Neural Networks

(深層ニューラルネットワークの計算コスト削減に向けた自動ハイパーパラメータ最適化および新規手法)

January 2025

Graduate School of System Informatics  
Kobe University

Yasufumi Sakai



# Abstract

This dissertation presents computational cost reduction techniques of deep neural networks (DNNs), including automated hyperparameter search, not only for established fields such as image processing but also for emerging fields such as materials discovery.

Existing DNN computational cost reduction techniques often require complex manual hyperparameter tuning, limiting their efficiency and broad applicability. Furthermore, methods effective in established fields like image processing often prove inefficient when applied to emerging domains such as materials discovery. To address these limitations, we propose five novel techniques: (1) an 8-bit shifted dynamic fixed-point (S-DFP) quantization method minimizing accuracy loss; (2) a structured pruning method using zero-padding layers to improve compression efficiency, especially for networks with shortcut connections; (3) a gradient-aware automatic pruning rate search (GAPRS) algorithm automating optimal pruning rate selection; (4) a novel self-supervised learning method employing atom replacement for efficient learning with unlabeled materials data; and (5) a novel active learning method leveraging expanded features of material structural data to reduce the cost of generating labeled training data. We evaluated these techniques extensively. For image processing, experiments on ImageNet and CIFAR-10 using various ResNet architectures demonstrated significant reductions in model size and computational cost. For example, a 60.1% parameter reduction in ResNet-50 on ImageNet with 76.17% accuracy retained. For materials discovery, experiments on the Open Catalyst 2022 benchmark dataset using various graph neural networks (GNNs) such as PaiNN for material energy prediction showed that our method achieved the lowest prediction error with the fewest training data compared to existing methods.

Chapter 1 shows research background of the dissertation and basic characteristics for existing DNN computational cost reduction techniques such as quantization, pruning, self-supervised learning, and active learning.

In Chapter 2, challenges in DNN computational cost reduction are summarized: (1) trade-off between prediction accuracy, computation speed and model size of DNNs, (2) manual hyperparameter tuning for accelerating methods of DNN, and (3) DNNs computational cost for emerging fields such as materials discovery.

In Chapter 3, a quantization method using shifted dynamic fixed-point format for DNNs training is proposed. To prevent accuracy degradation due to quantized DNN training when using conventional DFP format, S-DFP shifts the data representable range of DFP from a large value area to a small value area by adding bias to the exponent of DFP. Using the proposed S-DFP format for quantized DNNs training, the quantized model can be trained using 8-bit fixed-point precision without significant accuracy degradation on the ImageNet task with ResNet-32, ResNet-50, ResNet-101, and ResNet-152. Since the validation accuracy of the quantized model with proposed S-DFP was achieved equivalent accuracy with FP32 regardless of the depth in ResNet, the proposed S-DFP is expected to significantly improve the accuracy of the quantized model regardless of the DNN model type, such as transformer other than ResNet. In addition, the proposed method achieved the equivalent accuracy with conventional 16-bit DFP instead of using 8-bit S-DFP.

In Chapter 4, to reduce the workload of inefficient manual pruning rate assignment, we describe our proposed automatic pruning rate search method named GAPRS for structured pruning. In the proposed method, by selecting a pruning rate in which the pruning error is smaller than a pre-defined threshold, the accuracy degradation of the pruned model is suppressed. Furthermore, we describe a structured pruning method without degrading the prediction accuracy of complex DNNs with shortcut connection. To improve the inference speed and the compression ratio for the DNN model with a shortcut connection such as ResNet, zero-padding layers are inserted into all input paths to the addition operator. The experimental results demonstrate that the pruned model accuracy can be effectively controlled by the proposed method. In addition, the pruning rate derived by the proposed method can be converged. We experimentally showed the superiority of GAPRS against various state-of-the-art methods on CIFAR-10 and ImageNet using various ResNet architectures. For ResNet-50, our method achieved a 60.1% reduction in parameters and a 59.8% reduction in FLOPs while maintaining 76.17% accuracy on ImageNet. These results demonstrate the effectiveness of GAPRS and our proposed method of inserting zero-padding layers. in achieving high compression ratios without model accuracy degradation.

Chapter 5 presents novel self-supervised learning (SSL) and active learning methods designed to reduce the computational cost of materials discovery using DNNs. Our

proposed mask-less SSL method employs atom replacement with unlabeled data to improve the accuracy of catalyst energy prediction using GNNs. Unlike existing SSL methods that utilize fictitious "mask" atoms, our approach leverages only real elements, thereby promoting more efficient learning. We evaluated this method on three benchmark datasets (OC22, Poisoned Catalyst, and Expanded Poisoned Catalyst) using CGCNN and PaiNN GNNs, achieving the lowest prediction error across all experiments. Furthermore, we introduce a novel active learning method to enhance the accuracy of DNN-based catalyst energy prediction with limited training data. This method leverages all available material properties (structure and energy) to minimize the required training data. Evaluations on three benchmark datasets (OC22, NRR, and ORR) using PaiNN and EquiformerV2 GNNs demonstrate that our method achieves the lowest mean absolute error (MAE) among all evaluated sampling methods even with the fewest data.

Chapter 6 summarizes the conclusions of this study. This dissertation presents methods for achieving low-cost DNN computation and automated hyperparameter optimization to accelerate DNNs, addressing both established applications such as image processing and emerging fields such as materials discovery. Our work contributes to bridging the DNN computational demand-supply gap in diverse fields, including image processing and materials discovery, by avoiding inefficient manual hyperparameter tuning.

*Keywords: Neural networks, Computational cost reduction, Model compression, Quantization, Pruning, Self-supervised learning, Active learning, Dynamic fixed-point, Structured pruning, Pruning rate search, mask less SSL, Feature sampling.*

# Table of Contents

|                                                                        |      |
|------------------------------------------------------------------------|------|
| Abstract .....                                                         | i    |
| Table of Contents.....                                                 | iv   |
| List of Figures .....                                                  | vii  |
| List of Tables .....                                                   | xii  |
| List of Algorithms.....                                                | xiii |
| Chapter 1 Introduction .....                                           | 14   |
| 1.1 Background of Research Area .....                                  | 14   |
| 1.2 Overview of This Study .....                                       | 21   |
| 1.3 Overview of This Dissertation.....                                 | 21   |
| Chapter 2 Challenges in DNN computational cost reduction .....         | 24   |
| 2.1 Performance Trade-off in lightweighted DNNs .....                  | 24   |
| 2.1.1 Challenges in the quantization of DNNs .....                     | 24   |
| 2.1.2 Challenges in the pruning of DNNs .....                          | 25   |
| 2.2 Manual hyperparameter tuning in DNN acceleration methods .....     | 28   |
| 2.3 DNNs computational cost for materials discovery .....              | 30   |
| 2.4 Summary .....                                                      | 31   |
| Chapter 3 Quantization for DNN acceleration .....                      | 32   |
| 3.1 S-DFP: 8-bit shifted dynamic fixed-point for quantized DNNs.....   | 33   |
| 3.1.1 Related works .....                                              | 33   |
| 3.1.1.1 Quantization for DNN training.....                             | 33   |
| 3.1.1.2 Dynamic fixed-point .....                                      | 33   |
| 3.1.2 Quantized DNNs training with shifted dynamic fixed-point .....   | 35   |
| 3.1.2.1 Analysis of accuracy degradation with dynamic fixed-point..... | 35   |
| 3.1.2.2 Shifted dynamic fixed precision format .....                   | 37   |
| 3.1.3 Experiment .....                                                 | 40   |
| 3.1.3.1 Parameters setting for S-DFP .....                             | 41   |

|                                                                        |                                                                                     |    |
|------------------------------------------------------------------------|-------------------------------------------------------------------------------------|----|
| 3.1.3.2                                                                | Experimental results.....                                                           | 43 |
| 3.2                                                                    | Summary .....                                                                       | 46 |
| Chapter 4 Automated hyperparameter search for DNN acceleration .. 47   |                                                                                     |    |
| 4.1                                                                    | GAPRS: Gradient-aware automatic pruning rate search for structured pruning48        |    |
| 4.1.1                                                                  | Related works .....                                                                 | 48 |
| 4.1.1.1                                                                | Importance for pruning .....                                                        | 48 |
| 4.1.1.2                                                                | Local pruning and global pruning.....                                               | 49 |
| 4.1.1.3                                                                | One-shot pruning and iterative pruning .....                                        | 49 |
| 4.1.2                                                                  | Automatic pruning rate search .....                                                 | 50 |
| 4.1.2.1                                                                | Pruning rate search by gradient-aware threshold .....                               | 51 |
| 4.1.2.2                                                                | Model pruning, fine-tuning, and accuracy evaluation.....                            | 56 |
| 4.1.2.3                                                                | Initializing and updating the upper bound of threshold .....                        | 56 |
| 4.1.3                                                                  | Structured pruning around shortcut connection with zero-padding layers              | 59 |
| 4.1.4                                                                  | Experiments.....                                                                    | 61 |
| 4.1.4.1                                                                | Performance comparisons .....                                                       | 61 |
| 4.1.4.2                                                                | Robustness evaluation for hyperparameter $Lm$ .....                                 | 63 |
| 4.1.4.3                                                                | Accuracy control of pruned model.....                                               | 65 |
| 4.1.4.4                                                                | Temporary pruning rate during pruning rate search.....                              | 65 |
| 4.1.4.5                                                                | Convergence of pruning rate search .....                                            | 67 |
| 4.1.4.6                                                                | Effectiveness of inserting zero-padding layers .....                                | 67 |
| 4.2                                                                    | Summary .....                                                                       | 69 |
| Chapter 5 Cost-effective DNN training for materials discovery ..... 70 |                                                                                     |    |
| 5.1                                                                    | Self-supervised learning with atom replacement for catalyst energy prediction ..... | 70 |
| 5.1.1                                                                  | Related works .....                                                                 | 71 |
| 5.1.2                                                                  | Self-supervised learning with atom replacement for energy prediction GNNs.....      | 72 |
| 5.1.3                                                                  | Experiments.....                                                                    | 73 |
| 5.1.3.1                                                                | Dataset.....                                                                        | 73 |
| 5.1.3.2                                                                | Training flow .....                                                                 | 76 |

|           |                                                                                       |     |
|-----------|---------------------------------------------------------------------------------------|-----|
| 5.1.3.3   | Experimental results.....                                                             | 78  |
| 5.2       | Active learning for graph neural network training in catalyst energy prediction ..... | 82  |
| 5.2.1     | Related works .....                                                                   | 83  |
| 5.2.2     | Active learning for material energy prediction GNNs.....                              | 85  |
| 5.2.2.1   | Existing active learning for energy prediction GNNs and the problems                  | 85  |
| 5.2.2.2   | Proposed active learning using expanded features .....                                | 87  |
| 5.2.3     | Experiment .....                                                                      | 88  |
| 5.2.3.1   | Dataset.....                                                                          | 88  |
| 5.2.3.2   | Models and hyperparameters for training.....                                          | 89  |
| 5.2.3.3   | Training flow with active learning.....                                               | 89  |
| 5.2.3.4   | Experimental results.....                                                             | 90  |
| 5.3       | Summary .....                                                                         | 94  |
| Chapter 6 | Conclusion .....                                                                      | 95  |
|           | References.....                                                                       | 99  |
|           | List of Publications and Presentations.....                                           | 111 |
|           | Publications in journals.....                                                         | 111 |
|           | Presentations at international conferences .....                                      | 111 |
|           | Awards .....                                                                          | 113 |
|           | Publications in Press Release .....                                                   | 113 |
|           | Publications in texts and books.....                                                  | 114 |
|           | Publications in Open-Source Software .....                                            | 114 |
|           | Acknowledgement .....                                                                 | 116 |

## List of Figures

|      |                                                                                                                                                                                                                                                                                                                                                                                     |    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.1  | Trends in DNN computation and single-processor computing power.....                                                                                                                                                                                                                                                                                                                 | 14 |
| 1.2  | Trends in DNN model size and single-processor memory capacity.....                                                                                                                                                                                                                                                                                                                  | 15 |
| 1.3  | The effects of quantization. ....                                                                                                                                                                                                                                                                                                                                                   | 16 |
| 1.4  | The effects of pruning.....                                                                                                                                                                                                                                                                                                                                                         | 16 |
| 1.5  | The degree of model size reduction in (a) quantization and (b) pruning. ....                                                                                                                                                                                                                                                                                                        | 17 |
| 1.6  | Examples of self-supervised learning.....                                                                                                                                                                                                                                                                                                                                           | 18 |
| 1.7  | Examples of active learning. ....                                                                                                                                                                                                                                                                                                                                                   | 18 |
| 1.8  | Quantized DNN model size vs. accuracy on ImageNet task [10]. ....                                                                                                                                                                                                                                                                                                                   | 19 |
| 1.9  | Pruned DNN model size vs. accuracy on ImageNet and Penn Tree Bank [11].<br>.....                                                                                                                                                                                                                                                                                                    | 19 |
| 1.10 | Examples of material structural data. ....                                                                                                                                                                                                                                                                                                                                          | 20 |
| 1.11 | Overview of this dissertation. ....                                                                                                                                                                                                                                                                                                                                                 | 23 |
| 2.1  | Overview of unstructured pruning and structured pruning. ....                                                                                                                                                                                                                                                                                                                       | 25 |
| 2.2  | Model implementation example on structured pruning in a channel-wise<br>manner. ....                                                                                                                                                                                                                                                                                                | 25 |
| 2.3  | Example of shortcut connection with an addition operator and how to work the<br>addition operator. (a) Feature maps from multiple layers are input to the<br>addition operator. (b) In the addition operator, input multiple feature maps are<br>added in an element-wise manner. (c) When the number of input feature maps<br>is different, the addition operator cannot work..... | 26 |
| 2.4  | Pruning procedure in the existing manual pruning.....                                                                                                                                                                                                                                                                                                                               | 29 |
| 2.5  | Pruning procedure in the proposed automatic pruning.....                                                                                                                                                                                                                                                                                                                            | 29 |
| 3.1  | IEEE-754 standard single precision floating-point format (FP32) and 8-bit<br>dynamic fixed-point format (DFP8). ....                                                                                                                                                                                                                                                                | 34 |
| 3.2  | Distribution of the weight gradient before and after quantization with DFP<br>format in the first convolution layer on ImageNet task with ResNet-50 trained<br>by FP32.....                                                                                                                                                                                                         | 34 |
| 3.3  | Top-1 validation accuracy on ImageNet task with ResNet-34: (a) accuracy<br>during 0 to 90 epochs and (b) accuracy during 60 to 90 epochs. ....                                                                                                                                                                                                                                      | 36 |
| 3.4  | Relation between the step size of the weight update and loss function: (a)                                                                                                                                                                                                                                                                                                          |    |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|      | because of the large step size when using DFP8, it is difficult to reach the minimum loss function; (b) because the step size is small when using FP32, the loss function can reach the minimum. ....                                                                                                                                                                                                                                                                                                                                                                                    | 36 |
| 3.5  | Overview of the proposed quantization method: (a) When using conventional DFP, the large value information remains, whereas the small value information is lost. (b) When using proposed S-DFP, the data representable range of DFP is shifted by adding bias to the exponent. Consequently, the large value information is lost and the small value information remains .....                                                                                                                                                                                                           | 38 |
| 3.6  | Validation accuracy on the ImageNet task with ResNet-34 when applying S-DFP8 using various <i>bias</i> values.....                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 40 |
| 3.7  | Minimum training loss function and maximum validation accuracy on various <i>bias</i> . ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 41 |
| 3.8  | Effects of S-DFP applying timing on the ImageNet classification task with ResNet-34. Suitable timing for applying S-DFP to quantized model training is after saturating validation accuracy. ....                                                                                                                                                                                                                                                                                                                                                                                        | 42 |
| 3.9  | Experimental results on the ImageNet classification task with ResNet-34: (left) Top-1 validation accuracy during whole training duration. (right) Top-1 validation accuracy after 60 epochs.....                                                                                                                                                                                                                                                                                                                                                                                         | 44 |
| 3.10 | Experimental results on the ImageNet classification task with ResNet-50. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 44 |
| 3.11 | Experimental results on the ImageNet classification task with ResNet-101. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 45 |
| 3.12 | Experimental results on the ImageNet classification task with ResNet-152. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 45 |
| 4.1  | Overview of our proposed pruning search. The pruning rate for each layer is chosen from the pruning rate candidates by comparing the pruning error with the threshold. The model is then pruned using the chosen pruning rates, and the pruned model is fine-tuned. If the accuracy of the fine-tuned model meets the target, the pruned fine-tuned model is moved to the next step (f); otherwise, the model is discarded and the pruning rates are re-searched. Our method iteratively searches for pruning rates until the pruning rate for all layers chosen in step (b) is 0%. .... | 50 |
| 4.2  | Proposed pruning method around shortcut connection. Zero-padding layers are                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |

|     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|     | inserted into all input paths to the addition operator. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 60 |
| 4.3 | Details of the proposed pruning method around shortcut connection. Inserted zero-padding layers append zero matrices to the output of the layers with fewer channels. The zero matrices are appended so that the number of channels of all input feature maps to the addition operator is aligned.....                                                                                                                                                                                                                                       | 60 |
| 4.4 | Pruned model performance against the hyperparameters on CIFAR-10 with ResNet-32. (a)Performance against $Lm$ ( $Acc_{ctrl} = 0\%$ ). The accuracy and compression ratios of the pruned model are robust for the $Lm$ choice when $Lm$ is greater than zero. (b) Performance against $Acc_{ctrl}$ ( $Lm = 0.1$ ). The pruned model accuracy can be effectively controlled by $Acc_{ctrl}$ .....                                                                                                                                               | 64 |
| 4.5 | Pruning rate of the entire model and the pruned model accuracy during pruning rate search on CIFAR-10 with ResNet-32. The temporary pruning rate is the pruning rate chosen by our method at the step in Fig. 4.1(b), and the temporary pruned model accuracy is the accuracy pruned by the temporary pruning rate. The accepted pruning rate is the pruning rate that is used for the fine-tuned model which satisfies the target accuracy, and the accepted pruned model accuracy is the accuracy pruned by the accepted pruning rate..... | 66 |
| 4.6 | Pruning rate of the entire model during pruning rate search. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 67 |
| 4.7 | Number of output channels before and after pruning.....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 68 |
| 5.1 | Overview of training framework with proposed SSL. (a) Our proposed method replace the atoms of input unlabeled data to realize mask-less SSL. (b) In our method, the pre-trained weights of the feature extractor are transferred to the GNN for fine-tuning. Then, the GNN is fine-tuned by labeled data for the downstream task. ....                                                                                                                                                                                                      | 73 |
| 5.2 | 3D view of the catalyst dataset for evaluation. OC22 dataset has various structures of catalysts. On the other hand, the poisoned catalyst dataset contains poisoning atoms causing catalyst poisoning.....                                                                                                                                                                                                                                                                                                                                  | 74 |
| 5.3 | GNN training flow in the experiments of this work. (a) Only fine-tune GNN (not using SSL). (b) Pre-train GNN by the existing mask-based SSL, then the pre-trained GNN is fine-tuned. (c) Pre-train GNN by the proposed mask-less SSL, then the pre-trained GNN is fine-tuned. ....                                                                                                                                                                                                                                                           | 76 |
| 5.4 | Learning curve of validation MAE of predicted energy during fine-tuning in                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |    |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |    |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
|      | PaiNN using (a) OC22 dataset, (b) Poisoned Catalyst dataset, and (c) Expanded Poisoned Catalyst dataset. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 78 |
| 5.5  | Two-dimensional visualization of extracted vector embeddings from PaiNN by t-SNE. The embeddings are obtained by OC22 IS2RE validation data. The color bar shows the total energy of each point in the t-SNE projection. (a) t-SNE projection from the PaiNN before fine-tuning without SSL, i.e. scratch PaiNN model. (b) t-SNE projection from fine-tuned PaiNN without SSL. (c) t-SNE projection from PaiNN pre-trained by SSL before fine-tuning. (d) t-SNE projection from fine-tuned PaiNN pre-trained by SSL. ....                                                                    | 79 |
| 5.6  | Scatter plots created by classifying all points obtained from the OC22 validation data with the specified range of total energy. “Toten” means total energy. Standard deviation $\sigma$ for each cluster is calculated by the residuals among the regression line by orthogonal distance regression and all points in each cluster. That is, $\sigma$ represents the variance of the points in the cluster separated by similar total energy. ....                                                                                                                                          | 80 |
| 5.7  | Training flow of neural network with active learning. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 83 |
| 5.8  | Example of sampled data by existing active learning [126]. Blue points are the feature map of the training data candidates. Red crosses are the sampled data by active learning. Since the data are sampled broadly from the population, independent of the distribution of the population, the sampled data capture the diversity of the population’s features. ....                                                                                                                                                                                                                        | 85 |
| 5.9  | (a) Feature map of neural networks training data for catalyst energy prediction [93, 110]. The x-axis and y-axis show the features of material structure. The z-axis shows the energy of catalysts. Even if the structural features are similar, i.e. the material structures are similar, the energies among the materials are different. (b) MAE of the predicted energy for the three validation datasets. The evaluated model is a published PaiNN model [104]. The energy prediction error increases when the validation data is composed of structures with large energy changes. .... | 86 |
| 5.10 | Overview of proposed active learning method. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 87 |
| 5.11 | Comparison of the MAE using different active learning on OC22 data. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 91 |
| 5.12 | Comparison of the MAE using different active learning on NRR data. ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 91 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |    |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 5.13 | Comparison of the MAE using different active learning on ORR data.....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 91 |
| 5.14 | The feature space of the proposed expanded features generated using PaiNN with OC22 training data. Red crosses show 500 sampled data points by various sampling methods. The top row 2-D feature maps represent the top view of 3-D feature space. The bottom row 2-D feature maps represent the side view. The distributions of the data sampled by our proposed method show a more even spread in terms of all dimensions, i.e., the structural features and the energy, compared to those sampled by random sampling and the existing active learning method, which are heavily influenced by the population distribution. .... | 92 |

# List of Tables

|     |                                                                                                                            |    |
|-----|----------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 | Comparison of Top-1 maximum validation accuracies for the respective quantization methods. In all cases, $bias=8$ .        | 41 |
| 3.2 | Comparison of Top-1 validation accuracies with related works on ImageNet task with ResNet-50.                              | 46 |
| 4.1 | Comparison of experimental results on CIFAR-10.                                                                            | 62 |
| 4.2 | Comparison of experimental results on ImageNet.                                                                            | 63 |
| 4.3 | Pruned model accuracy, compression ratio, and FLOPs with and without zero-padding layers.                                  | 68 |
| 5.1 | Overview of the evaluation dataset.                                                                                        | 74 |
| 5.2 | Hyperparameters in training. Other hyperparameters are the same as the published Open Catalyst setting in [105] and [106]. | 76 |
| 5.3 | Comparison of MAE of predicted energy on the evaluation dataset.                                                           | 78 |
| 5.4 | Euclidean distance of vector embeddings of PaiNN before and after fine-tuning calculated using the results of Fig. 6.5.    | 79 |
| 5.5 | Standard deviation of the population and the sampled data by various sampling method in Fig. 5.14.                         | 92 |

# List of Algorithms

|                                                                                         |    |
|-----------------------------------------------------------------------------------------|----|
| 3.1 DNN training procedure in the proposed quantization method.....                     | 39 |
| 4.1 Pruning rate search by comparing pruning error $\Delta W$ and threshold $T_h$ ..... | 54 |
| 4.2 Proposed automatic pruning rate search.....                                         | 58 |

# Chapter 1 Introduction

## 1.1 Background of Research Area

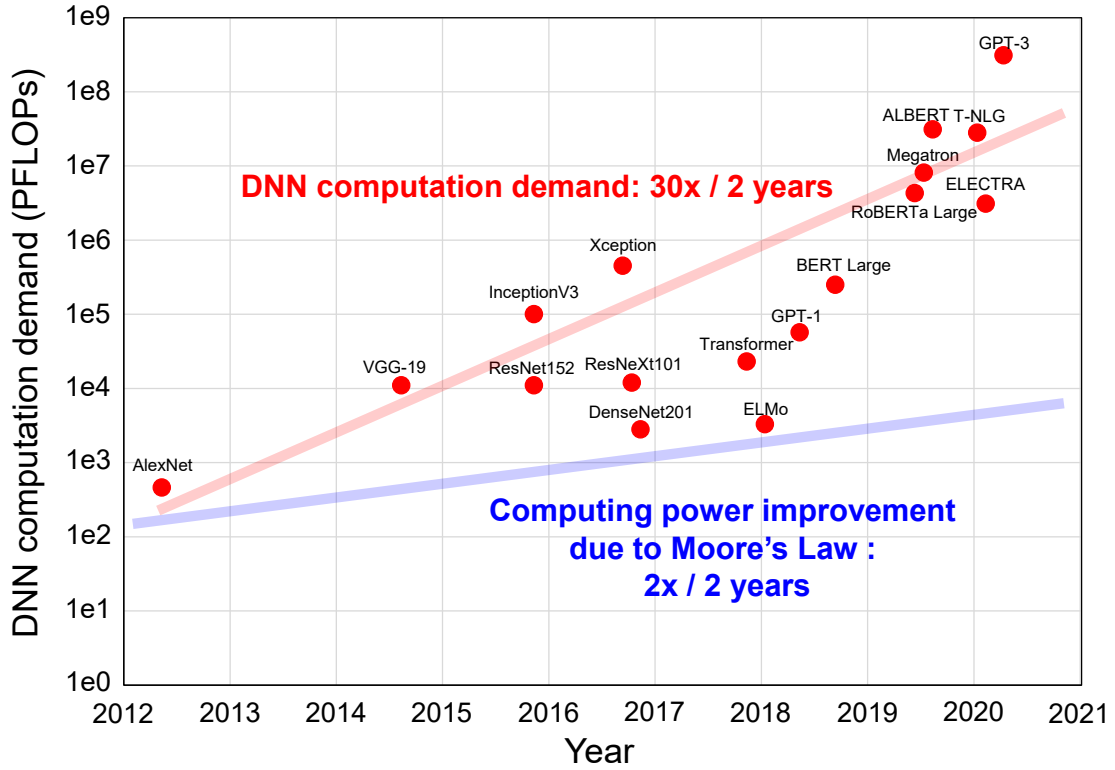


Fig. 1.1 Trends in DNN computation and single-processor computing power.

Deep neural networks (DNNs) have achieved great success in various areas such as image classification [1] and semantic segmentation [2]. However, as DNN models have become deeper and more complex alongside improvements to network performance, the deployment of DNNs on resource-constrained devices such as embedded systems and mobile phones is limited. [3] and [4] show the relationship between prediction accuracy and model size of DNNs. As the prediction accuracy of DNN improves, their computational complexity and the number of DNN parameters increase proportionally. Furthermore, the increasing trend in computational demands and model sizes for neural networks far exceeds the increasing trend in single-processor computing power and memory capacity [5, 6, 7, 8]. Figure 1.1 shows the year-by-year trends in DNN computation demand and single-processor computing power. While single-processor computing power roughly doubles every two years, DNN computational demands increase by a factor of 30 every two years. Figure 1.2 shows the year-by-year trends in

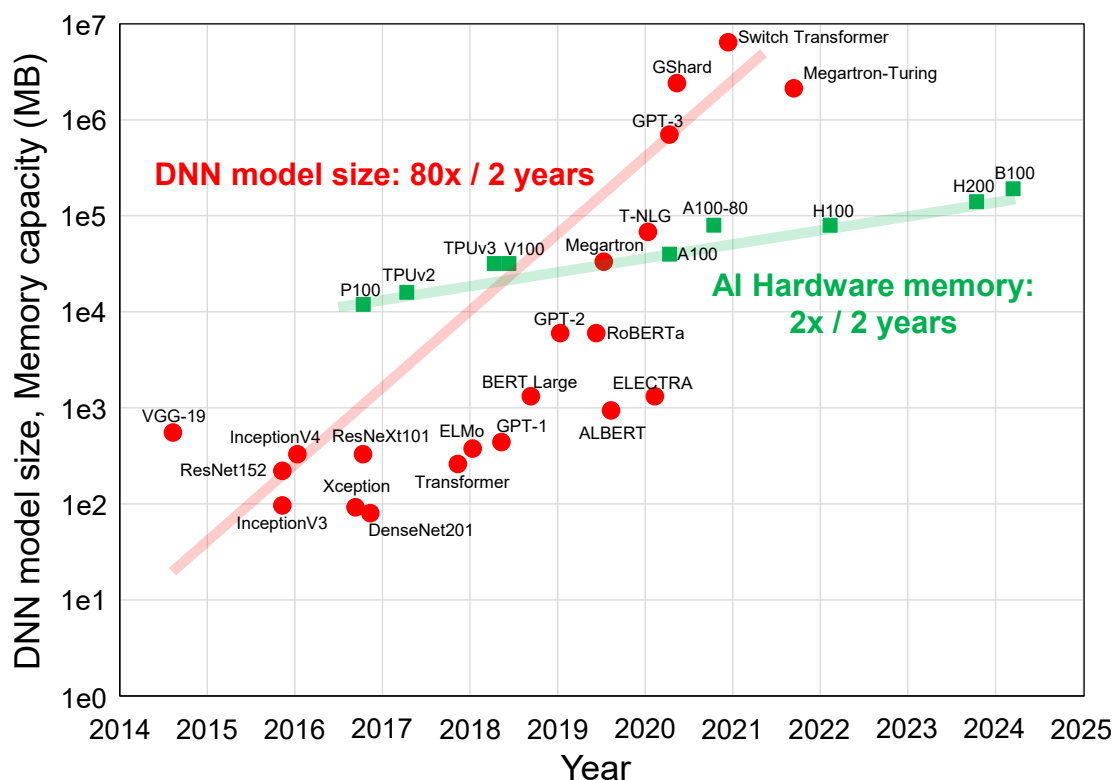


Fig. 1.2 Trends in DNN model size and single-processor memory capacity.

DNN model size and single-processor memory capacity. While single-processor memory capacity doubles every two years, DNN model size increases by a factor of 80 every two years. As these figures show, the recent increasing trends in DNN computation and model size far outpace the performance improvement trend of single processors. While DNNs have seen remarkable progress in domains such as image processing and natural language processing, their application to computationally intensive numerical simulations, particularly in materials discovery, is gaining traction as a means of accelerating calculations [91, 92, 93]. However, generating the labeled training data necessary for these DNNs relies on computationally expensive conventional numerical simulations, e.g., density functional theory (DFT) [88]. The substantial amount of labeled data required for high-accuracy DNNs results in a dramatically increased labeling cost in materials discovery compared to the established fields such as image processing and natural language processing. Consequently, the computational demand-supply gap in DNN-based materials discovery is further exacerbated. To bridge the demand-supply gap of DNN computation, it is necessary to introduce not only hardware technologies such as improvements in single-processor performance, but also various software technologies to reduce computational cost.

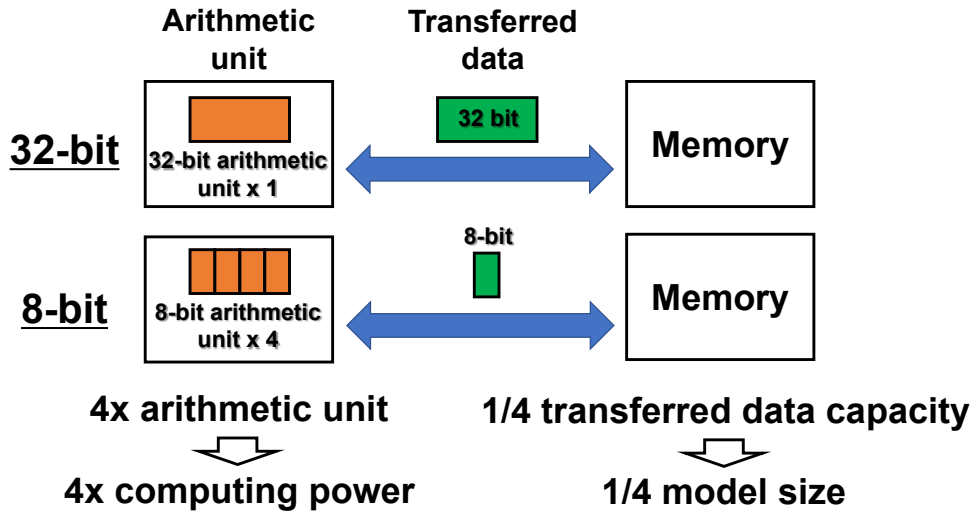


Fig. 1.3 The effects of quantization.

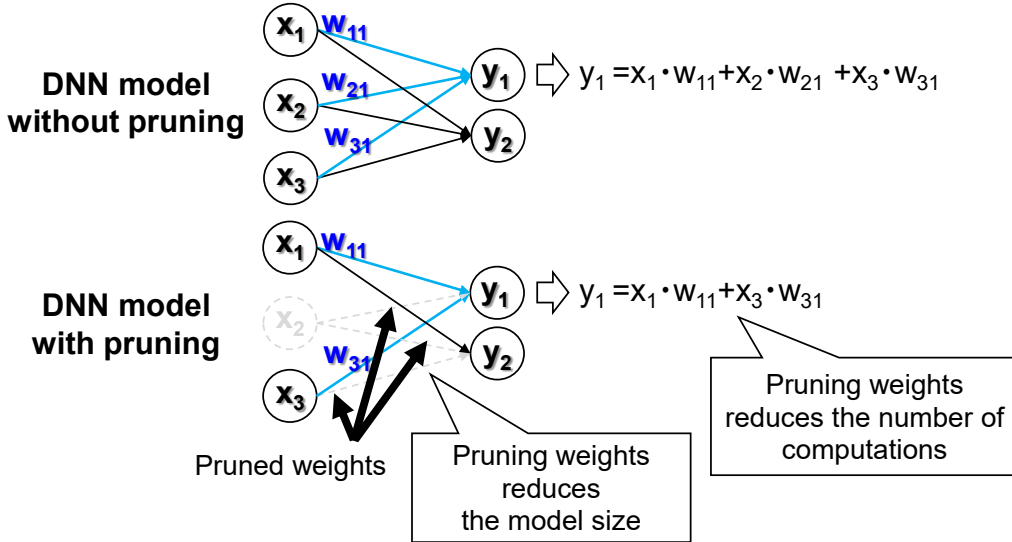


Fig. 1.4 The effects of pruning.

Quantization and pruning are proposed as representative software technologies that achieve both computational acceleration and model size reduction in DNNs [9]. In the field of deep learning, quantization refers to representing data expressed in 32-bit or more with a lower bit-width, such as 8-bit. As shown in Fig. 1.3, for example, representing data originally expressed in 32-bit with 8-bit allows four times as many arithmetic units to be placed on a chip of limited area, resulting in a fourfold increase in computing power. Furthermore, the data capacity is reduced to one-fourth, resulting in a one-fourth reduction in the DNN model size.

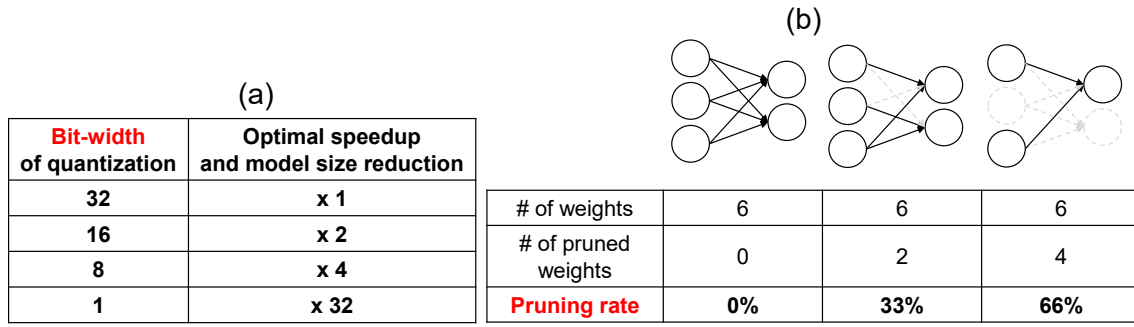
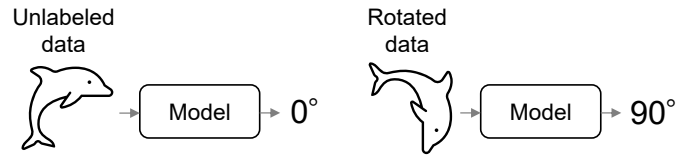


Fig. 1.5 The degree of model size reduction in (a) quantization and (b) pruning.

Pruning, on the other hand, is a technique that sets the weight values representing a DNN model to zero. A DNN model is generally represented by a directed graph, as shown in Fig. 1.4. The edges  $w_{11}$ ,  $w_{21}$ , and  $w_{31}$  represent the weights of the DNN. In the node part of the graph, the data output from each node and the weighted sum with the weights of the edges are calculated. In a pruned DNN model, the weights become zero. Therefore, since the edges are pruned, the number of weighted sum calculations performed in the node part also decreases. Furthermore, the model size of the DNN is reduced by the amount of the pruned weights. In quantization and pruning, the degree of model size reduction is determined by parameters such as “bit-width” and “pruning rate”. For example, in a quantized DNN model, as shown in Fig. 1.5(a), the degree of speedup and reduction in the quantized model varies depending on the bit-width used for quantization. The pruning rate, as shown in Fig. 1.5(b), represents the proportion of weights pruned by pruning. These bit-widths and pruning rate are manually set by the designer before applying quantization or pruning to the DNN. Such parameters that are manually set by the designer before program execution are called hyperparameters. Hyperparameters can also be defined as parameters whose values cannot be automatically determined by the algorithm executing the DNN.

Self-supervised learning (SSL) and active learning are proposed as software techniques to mitigate the high labeling costs associated with training data [139, 115]. SSL is a method for training DNNs using unlabeled data by creating labels from transformations applied to the unlabeled data itself, as illustrated in Fig. 1.6. This eliminates the need for manual labeling, thereby significantly reducing the labeling cost. In image processing, a common example is training a model to predict the rotation angle of unlabeled images that have been rotated by a known, self-determined amount.

TASK: The unlabeled data is rotated,  
and the model is trained to predict the direction of rotation.



TASK: A portion of the unlabeled data is masked,  
and the model is trained to predict the masked portion.

**that bird is small → that bird is [MASK]**

Unlabeled data

Masked data

Fig.1.6 Examples of self-supervised learning.

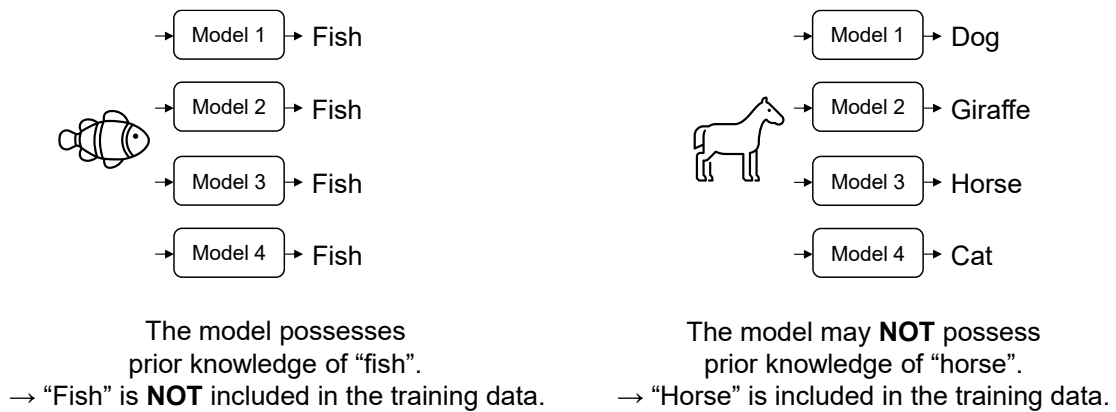


Fig.1.7 Examples of active learning.

The rotation angle, serving as the label, is inherently known, thus enabling supervised learning with only unlabeled data. Similarly, in natural language processing, a prevalent approach involves masking portions of text and training the model to predict the masked words. The location and identity of the masked words, which serve as labels, are known because they are self-determined. So, this method facilitates supervised learning using only unlabeled data. Active learning is a sampling method that selects informative data from a large pool of unlabeled data to train high-accuracy DNNs with limited labeled data. A representative active learning method, uncertainty sampling, is illustrated in Figure 1.7. The policy of uncertainty sampling is to reduce model uncertainty, i.e., to improve the model accuracy, by learning from highly uncertain data first. Thus, this method chooses

the data from the unlabeled data by calculating the uncertainty of all unlabeled data. For example, if multiple models accurately classify "fish," it indicates sufficient existing knowledge, so "fish" data are excluded from the training set. However, if multiple models produce inconsistent predictions for "horse", i.e., high uncertainty, it suggests a knowledge gap, leading to the inclusion of "horse" data in the training set.

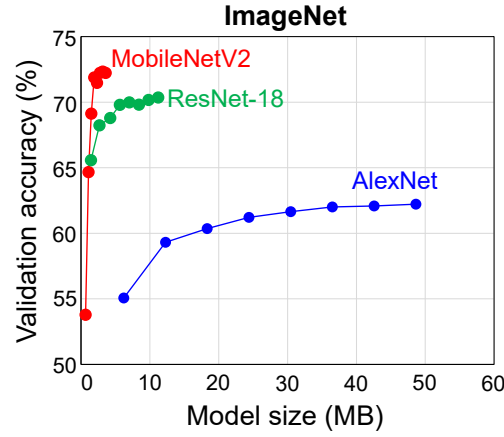


Fig. 1.8 Quantized DNN model size vs. accuracy on ImageNet task [10].

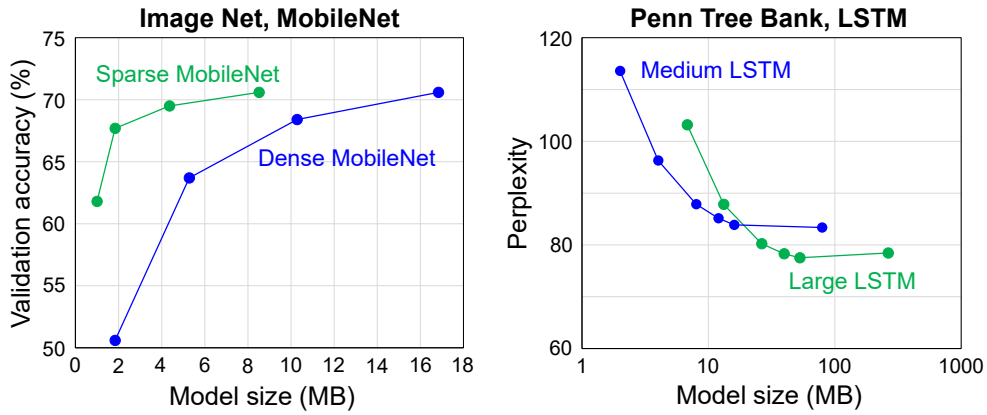


Fig. 1.9 Pruned DNN model size vs. accuracy on ImageNet and Penn Tree Bank [11].

Conventional computational cost reduction technologies have three main challenges. The first is the trade-off between DNN prediction accuracy and its computational speed and model size. Figure 1.8 and Figure 1.9 shows the relationship between model size and accuracy of compressed DNN models by quantization and pruning [10, 11]. As DNN models are compressed by quantization and pruning, they lose information including weights. Since weights determine the representational capacity of DNNs, the more weights are reduced to compress the DNN model, the more its accuracy tends to degrade.

The second challenge is the need for manual hyperparameter tuning to obtain a lightweight DNN model with the desired accuracy. As shown in Fig. 1.8 and Fig. 1.9, there is a trade-off between DNN accuracy and model size. Therefore, to obtain a lightweight model with the desired accuracy, it is necessary to set hyperparameters such as bit-width and pruning rate while considering this trade-off. However, many AI service developers lack expertise in DNN lightweighting techniques such as quantization and pruning. As mentioned in the first challenge, there is a trade-off between DNN prediction accuracy and computational speed and model size. Therefore, it is difficult for AI service developers to manually adjust hyperparameters to optimize this trade-off.

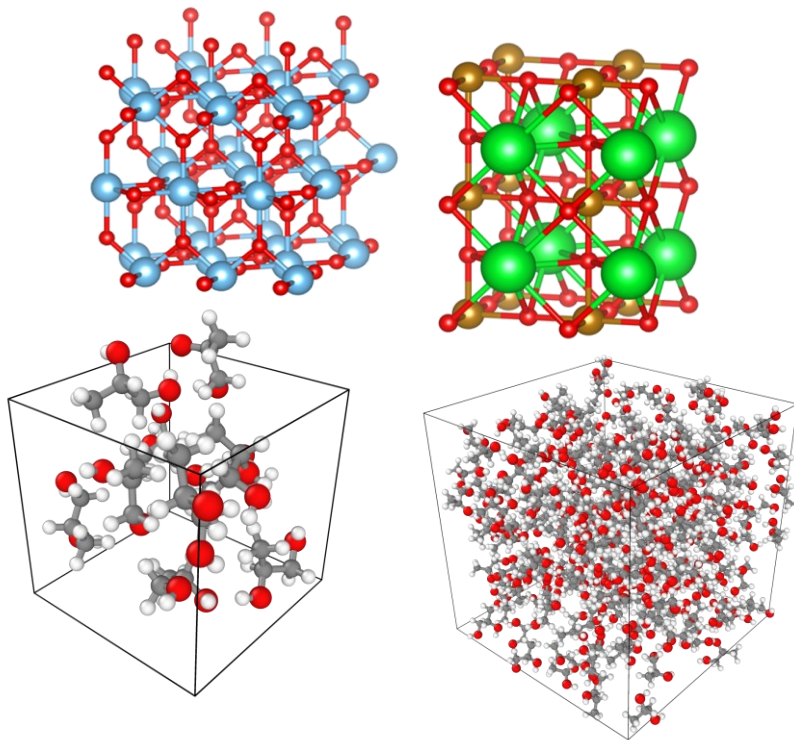


Fig. 1.10 Examples of material structural data.

The third challenge is DNNs computational cost for emerging fields such as materials discovery. As shown in [88], a major challenge in materials discovery is the high cost of generating labeled training data. Although SSL and active learning are promising approaches to reduce labeling costs, the challenge is the inefficiency of directly applying existing computational cost reduction technologies to emerging fields such as materials discovery. Materials discovery typically utilizes 3D structural data, where atoms are positioned in three-dimensional coordinates (Fig. 1.10), and the goal is to predict material

properties such as energy. Applying rotating SSL methods like those in Fig. 1.6 to this 3D structural data is problematic because rotating the structure does not change its properties; thus, rotation-based SSL is not applicable. In the case of applying masked SSL, since multiple atoms composing the material structure are masked, there is an inability to utilize the information of masked atoms to predict the other masked atoms. Furthermore, because these “masks” are artificially generated for SSL task, the data of the “mask” are not contained in the real data. Therefore, a mismatch occurs between the pre-training by the SSL task and the fine-tuning by the downstream task. The application of active learning using uncertainty sampling necessitates the evaluation of prediction variability across multiple models, resulting in increased computational cost.

## 1.2 Overview of This Study

This dissertation aims to research practical DNN computational cost reduction techniques not only for limited researchers and established fields like image processing and natural language processing, but also for AI service developers engaged in product development and emerging field such as materials discovery. Specifically, this research investigates methods to improve the trade-off between the accuracy of quantized and pruned DNNs and their computational speed and model size. In addition, using pruning as an example, this research investigates a method to automatically determine hyperparameters, such as the pruning rate, which determine the degree of computational acceleration and model compression, without sacrificing the prediction accuracy of the DNN. Furthermore, this research investigates methods to reduce labeling cost of training datasets used for DNNs utilized in materials discovery.

## 1.3 Overview of This Dissertation

Figure 1.11 illustrates an overview of this dissertation. Chapter 1 provides background, objectives, and an overview of this study. Chapter 2 discusses challenges in accelerating methods of DNNs computation: trade-off between computation speed, model size and prediction accuracy of DNN, and the difficulties of manual hyperparameter tuning for DNN accelerating methods. Chapters 3, 4, and 5 present novel techniques to address these challenges.

Chapter 3 presents a low bit-widths quantization method that maintains the prediction accuracy of complex DNNs. First, we propose an 8-bit shifted dynamic fixed-point format (S-DFP) for quantized DNNs. Then we demonstrate effectiveness of using our proposed S-DFP format in training quantized DNNs on the ImageNet task using ResNet-32, ResNet-50, ResNet-101, and ResNet-152. The model quantized by our proposed format can be trained using 8-bit fixed-point precision without significant accuracy degradation.

Chapter 4 introduces an automatic hyperparameter search for DNNs acceleration methods such as pruning to avoid inefficient manual hyperparameter assignment. This study shows an automatic pruning rate search method for structured pruning of DNNs to reduce the workload of inefficient manual pruning rate assignment for achieving the target accuracy of the pruned model. Because our method utilizes the pruning error (the difference in weight magnitude before and after pruning) to determine the pruning rate, it is applicable to quantization. Furthermore, we propose a structured pruning method without degrading the prediction accuracy of complex DNNs with shortcut connection. To improve the inference speed and the compression ratio for the DNN model with a shortcut connection such as ResNet, we propose structured pruning method that inserts zero-padding layers into all input paths to the addition operator. We demonstrate that our proposed method can reduce the model parameters and the number of floating-point operations (FLOPs) on CIFAR-10 and ImageNet with various ResNets while maintaining good accuracy.

Chapter 5 proposes novel SSL and active learning methods for materials discovery DNNs to reduce computational cost. In SSL study, we propose a mask-less SSL method with atom replacement using unlabeled data to improve the catalyst energy prediction accuracy of GNNs. The proposed SSL aims for efficient learning by replacing atoms by only using the actual existing elements, whereas the existing SSL methods use non-existing atoms of “mask”. We evaluate the proposed SSL on the three material benchmark datasets, OC22, Poisoned Catalyst, and Expanded Poisoned Catalyst, using energy prediction GNNs of CGCNN and PaiNN. The evaluation results show that the proposed SSL improves the prediction error of predicted energy in all experimental cases. In active learning study, we propose a novel active learning method to improve the catalyst energy prediction accuracy of GNNs with limited training data. Our method aims to reduce the number of training data by leveraging all available metrics, including material data such

as structure and energy. We demonstrate effectiveness of our method on three material benchmark datasets, OC22, NRR, and ORR, using energy prediction GNNs of PaiNN and EquiformerV2. The evaluation results show that our method improves the MAE of the predicted energy with limited training data among all evaluated sampling methods.

Finally, Chapter 6 summarizes the conclusions of this dissertation. This dissertation presents methods for achieving low-cost DNN computation and automatic hyperparameter optimization for DNN acceleration, addressing not only established applications such as image processing but also emerging fields such as materials discovery.

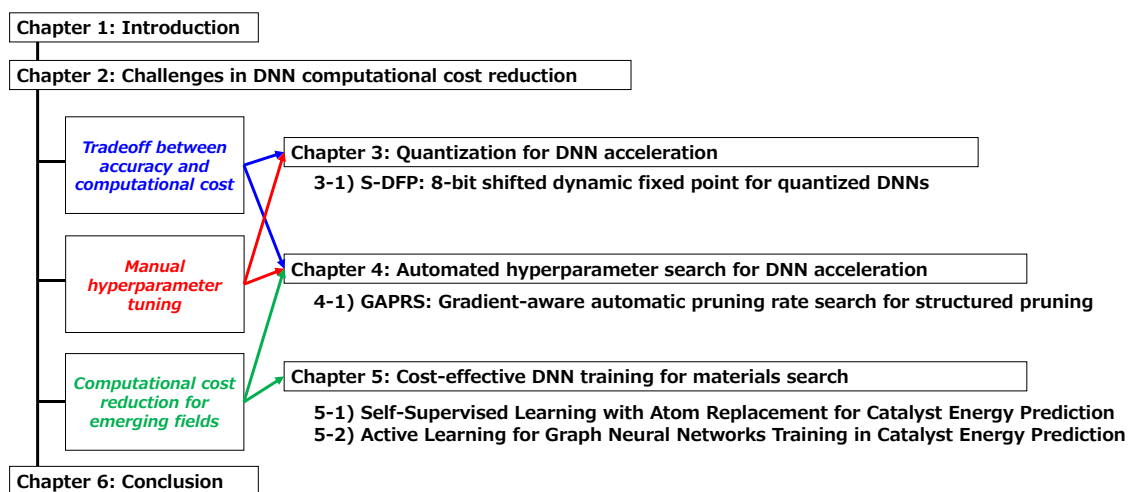


Fig. 1.11 Overview of this dissertation.

## Chapter 2 Challenges in DNN computational cost reduction

In this chapter, issues of DNN computational cost reduction techniques are summarized: (1) trade-off between prediction accuracy, computation speed and model size of DNNs, (2) manual hyperparameter tuning for accelerating methods of DNNs, and (3) DNNs computational cost for emerging fields such as materials discovery. First, issues of quantization and pruning are presented. Next, issues of manual hyperparameter tuning in acceleration methods for DNNs are introduced. Finally, we describe issues of labeling cost of DNNs utilized for materials discovery.

### 2.1 Performance Trade-off in lightweighted DNNs

#### 2.1.1 Challenges in the quantization of DNNs

To reduce the DNNs computation time such as training time, various quantization methods have been proposed. Two earlier reports [12, 13] have proposed DNN training methods using quantized weight and activation. One reported method [14] uses 16-bit floating-point (FP16) to quantize the gradient in addition to weight and activation for DNN training. The DNN training time is reduced according to the precision of datatype. Therefore, it is effective for using reduced precision for data representation and calculation [15]. Some earlier methods [16, 17] transform the data format representing single precision floating-point (FP32) into narrower bit-width floating-point format such as FP16 and 8-bit floating-point (FP8). Calculations using integer precision improve the energy efficiency of hardware for deep learning models in comparison with the use of floating-point format [18]. However, because the data representation range of integers is narrower than that of floating-point, the accuracy of the quantized model with integers is generally lower than when using floating-point for quantization [22]. Several methods [19, 20] have been proposed to prevent degradation of accuracy when using integers. One method [19] uses 16-bit dynamic fixed-point precision (DFP16) for DNN training on the ImageNet task with ResNet-50 [21] to prevent accuracy degradation. Although another method [20] proposes DNN training using 8-bit fixed-point, [20] does not evaluate the

accuracy using complex DNN models such as ResNet. Therefore, quantization methods with a low bit-width, such as 8-bit, without degrading the prediction accuracy of complex DNN models are highly expected.

### 2.1.2 Challenges in the pruning of DNNs

Pruning is one way to reduce the DNN model size and computational complexity. Pruning methods are classified as either “unstructured pruning” and “structured pruning”, as shown in Fig. 2.1.

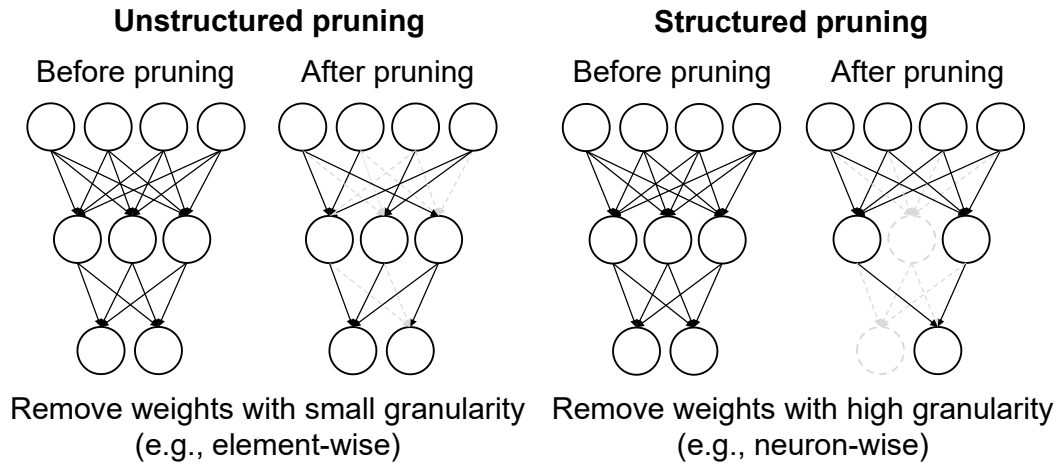


Fig.2.1 Overview of unstructured pruning and structured pruning.

#### Implementation example for convolutional layer in Pytorch

Before pruning `nn.Conv2d(in_channels = 32, out_channels = 32)`  
 After structured pruning `nn.Conv2d(in_channels = 10, out_channels = 16)`

Fig.2.2 Model implementation example on structured pruning in a channel-wise manner.

In unstructured pruning, unimportant connections are pruned based on defined criteria, and they are not necessarily small in value [24, 33, 53]. For example, when the value of weight is used for the pruning importance, small value elements in the weight tensor below a threshold determined by a pruning rate are removed. However, unstructured pruning leads to the generation of a sparse matrix. Since the inference speed of DNN models with a sparse matrix can only be accelerated by using special processors, the

practicality of unstructured pruning is limited. Structured pruning was therefore proposed to accelerate the inference speed on any processor [25, 26, 27, 28]. In structured pruning, pruning target weights are pruned with a higher granularity (e.g., neuron-wise manner) than in unstructured pruning, which means a model pruned by structured pruning can be easily represented on the code. The code in Fig. 2.2 shows an example when applying channel-wise structured pruning to a convolutional layer in Pytorch. Thanks to this feature, the inference speed of a model pruned by structured pruning can be accelerated on any processor.

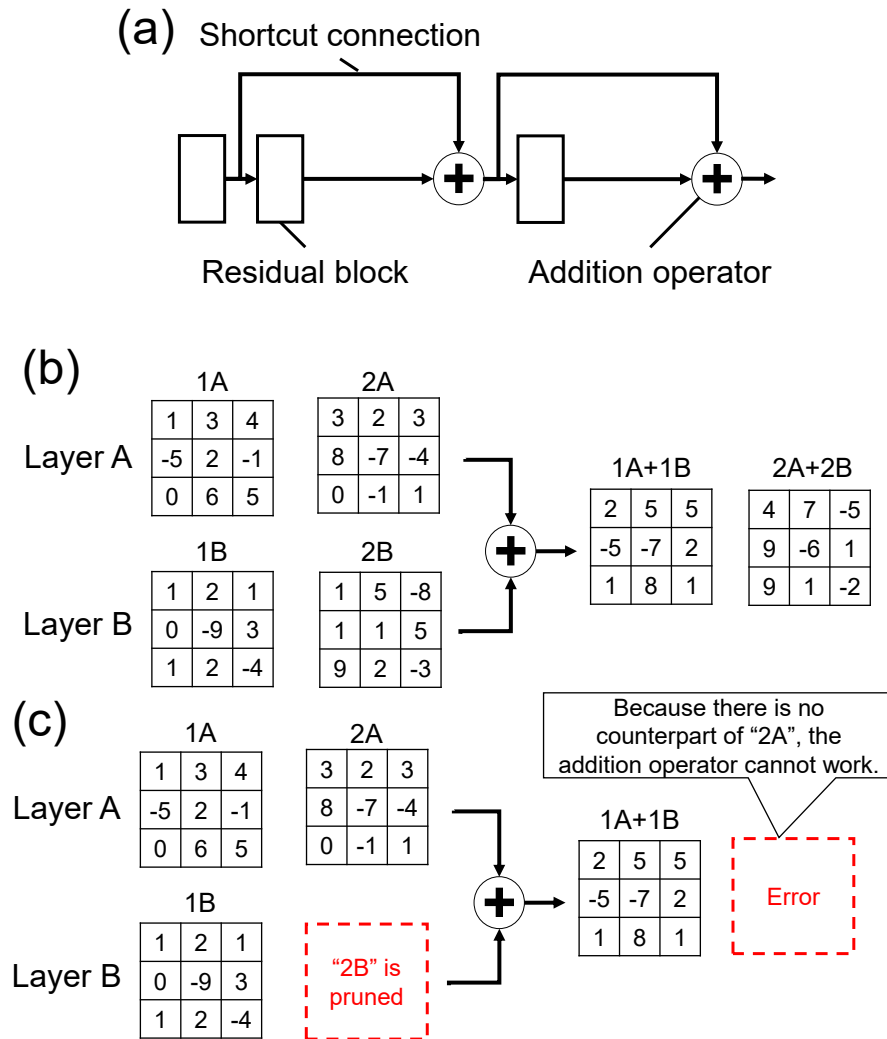


Fig.2.3 Example of shortcut connection with an addition operator and how to work the addition operator. (a) Feature maps from multiple layers are input to the addition operator. (b) In the addition operator, input multiple feature maps are added in an element-wise manner. (c) When the number of input feature maps is different, the addition operator cannot work.

Applying pruning results in a trade-off between prediction accuracy and model size [80], similar to quantization. Therefore, structured pruning methods that achieves low model size without degrading the prediction accuracy of complex DNN models are highly expected. A shortcut connection with an addition operator shown in Fig. 2.3(a) is often used for recent complex DNN models such as ResNet. Figure 2.3(b) shows how to work the addition operator. When multiple feature maps are input to the addition operator from multiple layers, the elements of the same index in the feature maps are added. Thus, the number of input feature maps must be the same. Here, when applying the pruning, the layers of the pruning target model are pruned independently, which means the number of input feature maps to the addition operator may not align, as shown in Fig. 2.3(c). In the case of Fig. 2.3(c), the addition operator cannot work. Therefore, to guarantee the calculation in the addition operator, the number of multiple input feature maps to the addition operator must be equal. In the original ResNet model [29], a zero-padding layer is inserted only into the shortcut connection to align the number of input feature maps, and no zero-padding layer is inserted into the residual block path because the number of feature maps from the shortcut connection is always less than the number of feature maps from the residual block. However, when all layers are candidates for pruning, the output channels of the residual block may be pruned, which leads to cases where there are fewer feature maps from the residual block path than from the shortcut connection, i.e., the numbers of each do not align.

## 2.2 Manual hyperparameter tuning in DNN acceleration methods

Existing DNN acceleration techniques such as quantization and pruning have many hyperparameters to optimize the lightweighted DNN model. For example, earlier pruning works [25, 26] have removed channels in accordance with the value ranking of the L1-norm of each pruning target filter. Another method [27] automatically derives the channels that should be pruned by using group LASSO and prunes filters to minimize the pruning error of feature maps connected to the output of the pruning target filters. There is also a method [28] that utilizes the gradient for pruning importance to consider the effect of pruning on the pruned model's accuracy. These existing methods prune weights in accordance with the pruning importance (e.g., the magnitude of the weight [24, 33]) and the pruning rate. Typically, the pruning rate is given manually, and the model is then pruned by the given pruning rate, as shown in Fig. 2.4. When an inappropriate pruning rate is given, the pruned model accuracy is degraded considerably as well as quantization [80]. Thus, the hyperparameters for acceleration methods of DNNs such as pruning rate and bit-width need to be iteratively searched by hand until the accuracy of the pruned model meets the target. However, there are many diverse applications in the industry such as image processing, natural language processing, materials discovery, and so on. Hence, various DNN models and datasets are utilized, which makes it impractical to implement manual hyperparameter searches for all practical cases. To reduce the workload of manually assigning hyperparameter in pruning and quantization to achieve the target accuracy of the pruned and the quantized model, we propose an automatic pruning rate search method for structured pruning shown in Fig.2.5. Because our method utilizes the pruning error (the difference in weight magnitude before and after pruning) to determine the pruning rate, it is applicable to quantization.

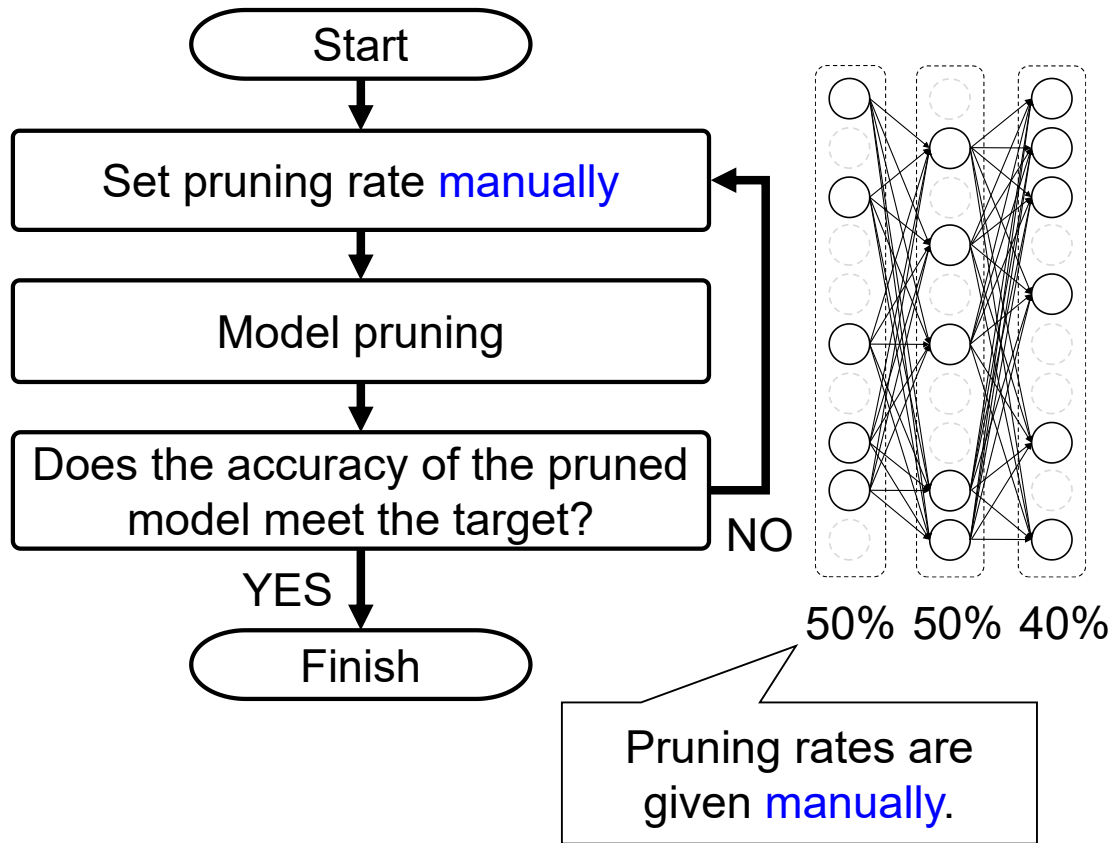


Fig.2.1 Pruning procedure in the existing manual pruning.

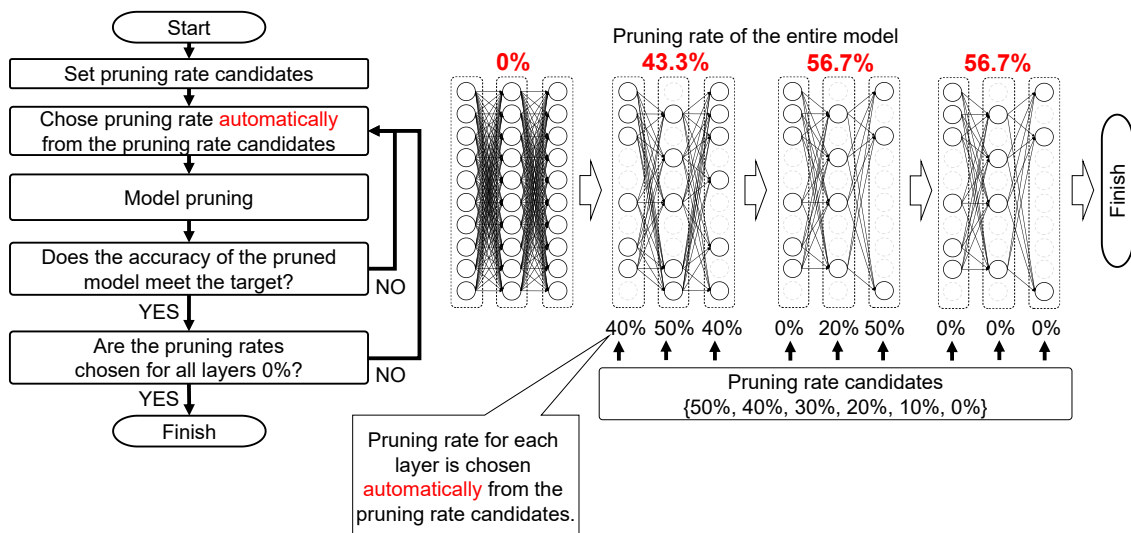


Fig.2.2 Pruning procedure in the proposed automatic pruning.

## 2.3 DNNs computational cost for materials discovery

To realize a carbon-free society, the sustainable production of various commodities must become economically viable. Ammonia has recently attracted attention as a next-generation energy carrier as its combustion does not result in carbon dioxide emissions [108]. It is currently produced via the Haber-Bosch process, which has a considerable carbon footprint [109]. Thus, much research effort has been put into the search for materials able to catalyze the electrochemical synthesis of ammonia at ambient temperature and pressure recently [81, 82, 83, 110]. Similarly, a fuel cell is a device that generates electricity by producing water from oxygen and hydrogen without emitting carbon dioxide. Identifying cheaper alternatives to the currently used Platinum-based fuel cell electrocatalysts is highly desirable [111, 112].

While the search for new catalysts was once led by experimentalists, first-principles calculations have in the last decade become the tool of choice [84, 85, 86, 110, 112,]. The use of computational methods makes it possible to scan through large groups of potential materials without having to produce them in the laboratory, saving both time and money. In the search for novel catalysts, quantities that characterize the chemical reactivity of a surface are of great importance, an example of which is the adsorption energy of adsorbates on the surface [87], which is calculated from the total electronic energy of the catalytic system. The electronic energy is obtained using density functional theory (DFT), which is a first principles-based methodology to find approximate solutions to the Schrodinger equation. With recent advances in algorithms and increasing computational resources, materials discovery using DFT is being widely employed. However, the computational cost of DFT calculations scales with the third power of the number  $N$  of electrons being modeled  $O(N^3)$ , so the computational cost becomes prohibitive for larger systems [88]. In the past few years, energy prediction methods using graph neural networks (GNNs) have been proposed to reduce the computational cost of energy calculations [90, 113, 114]. However, computationally heavy DFT calculations are still required to generate labeled training data to create highly accurate DNNs.

## 2.4 Summary

This section summarizes the issues of computational cost reduction methods of DNN :

- Trade-off between prediction accuracy and computational cost
- Manual hyperparameter tuning in acceleration method of DNNs
- DNNs computational cost for materials discovery

These issues must be considered to achieve both low computational cost and high accuracy of DNNs, even for users without specialized knowledge of DNNs. To solve these issues, acceleration method of DNNs computation for both established fields like computer vision and emerging field like materials discovery, and the automated hyperparameter tuning method must be integrated. In this dissertation, the novel techniques are presented in Chapters 3 to 6 to address the issues.

## Chapter 3 Quantization for DNN acceleration

This chapter describes the low bit-widths quantization method without degrading the prediction accuracy of complex DNN models. For this purpose, we studied following contents:

- 1) The 8-bit shifted dynamic fixed-point format for quantized DNNs is proposed.
- 2) Using our proposed format for quantized DNNs training, the quantized model can be trained using 8-bit fixed-point precision without significant accuracy degradation on the ImageNet task with ResNet-32, ResNet-50, ResNet-101, and ResNet-152.

Recent advances in deep neural networks have achieved higher accuracy with more complex models. Nevertheless, they require much longer training time. To reduce the training time, training methods using quantized weight, activation, and gradient have been proposed. Neural network computation by integer format improves the energy efficiency of hardware for deep learning models. Therefore, training methods for deep neural networks with fixed-point format have been proposed. However, the narrow data representation range of the fixed-point format degrades neural network accuracy. To solve these issues, we propose a new fixed-point format named shifted dynamic fixed-point (S-DFP) to prevent accuracy degradation in quantized neural networks training. S-DFP can change the data representation range of dynamic fixed-point format by adding bias to the exponent. We evaluated the effectiveness of S-DFP for quantized neural network training on the ImageNet task using ResNet-34, ResNet-50, ResNet-101 and ResNet-152. For example, the accuracy of quantized ResNet-152 is improved from 76.6% with conventional 8-bit DFP to 77.6% with 8-bit S-DFP.

## 3.1 S-DFP: 8-bit shifted dynamic fixed-point for quantized DNNs

### 3.1.1 Related works

#### 3.1.1.1 Quantization for DNN training

Floating-point format and integer format are used mainly for quantized neural network training. The floating-point format has a wide representable range. Therefore, the expected loss of information is small when applying floating-point format to quantization for DNN training. One reported method [14] proposed quantization method using FP16 with no marked accuracy degradation. Two earlier reports [16, 17] described the use of FP8 for DNN training. Neural network calculation by integer format improves the energy efficiency of hardware for deep learning models [18]. Therefore, training methods using fixed-point precision have also been proposed. Earlier studies [19, 20] have assessed training methods with fixed-point format, with high expected energy efficiency of hardware. One of those methods [19] uses DFP16 to prevent accuracy degradation. Although another method [20] uses a DNN training method with DFP8, its accuracy has not been evaluated the accuracy using complex DNN models such as ResNet.

#### 3.1.1.2 Dynamic fixed-point

An earlier report [19] described the DNN training method with DFP format. Figure 3.1 shows the IEEE-754 standard FP32 format and DFP8 format. Although floating-point has one exponent for each variable, DFP shares an exponent with multiple variables. The single shared exponent is applied generally to the respective tensors, such as the weight, activation, and gradient. When quantizing the FP32 vector of  $X=[x_1, \dots, x_n]^T$  using DFP, the value of shared exponent of DFP es is derived from the exponent of the absolute maximum value of  $X$  as presented below.

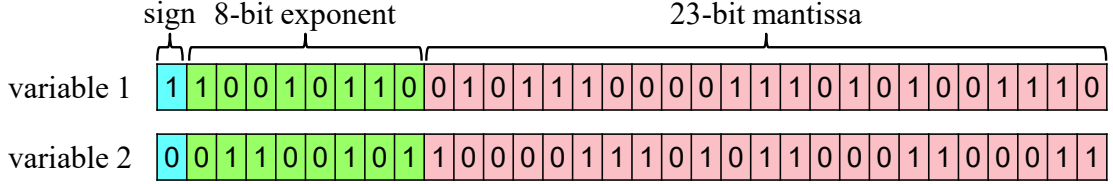
$$e_m = \text{exponent} \left( \max_{x \in X} |x| \right) \quad (3.1)$$

$$e_s = e_m - BW - 2 \quad (3.2)$$

where  $\text{exponent}()$  represents the exponent of input argument,  $e_m$  denotes the exponent of

absolute maximum value of  $X$ ,  $x$  is an element of  $X$ , and  $BW$  is the bit-width of the integer part of DFP, as shown in Fig. 3.1.

### Single precision floating-point (FP32)



### 8-bit Dynamic fixed-point (DFP8)

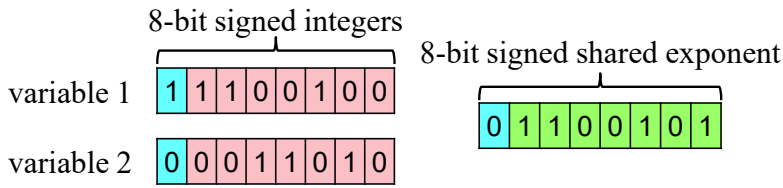


Fig. 3.1 IEEE-754 standard single precision floating-point format (FP32) and 8-bit dynamic fixed-point format (DFP8).

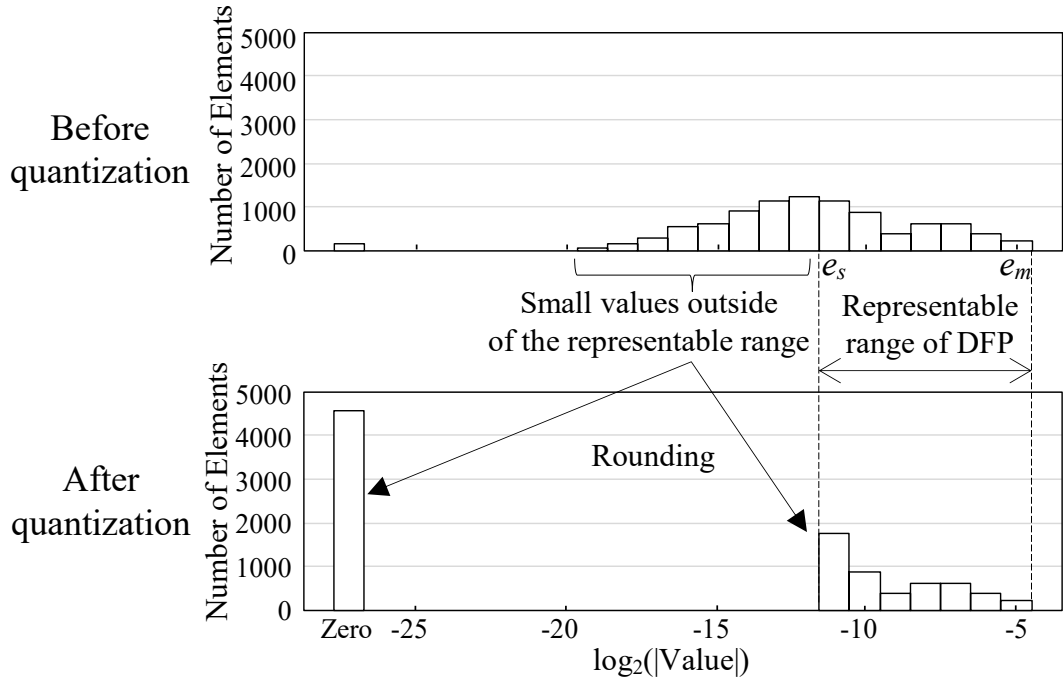


Fig. 3.2 Distribution of the weight gradient before and after quantization with DFP format in the first convolution layer on ImageNet task with ResNet-50 trained by FP32.

Figure 3.2 shows a distribution of weight gradient and data representation range before

and after quantization with DFP. The data representation range of DFP is represented the difference between  $e_s$  and  $e_m$  as shown in Fig. 3.2. Therefore, the bit-width of the integer part represents the data representation range, as given in Eq. 3.2. The value of quantized element with DFP is represented as shown below.

$$x_{i,q} = \text{round } x_i / 2^{e_s} \cdot 2^{e_s} \quad (3.3)$$

where  $x_i$  is the  $i$ -th element of  $X$  and  $x_{i,q}$  is the  $i$ -th element of  $X_q$ . Because the shared exponent of DFP  $e_s$  is derived from the absolute maximum value of quantization target vector  $X$ , the small value included in the vector before quantization is out of the data representation range of DFP, as shown in Fig. 3.2. Furthermore, the small value is rounded to zero or the minimum value that can be represented by DFP. Therefore, when the distribution is broad, most small values including the vector before quantization are rounded to zero when quantizing with DFP.

### 3.1.2 Quantized DNNs training with shifted dynamic fixed-point

To avoid information loss by rounding in quantization, we propose a training method using S-DFP format for quantized DNN training without any marked accuracy degradation. S-DFP can change the data representation range of DFP by adding bias to the exponent. This section first presents the cause of accuracy degradation when using DFP8 for quantized DNN training. Subsequently, we show the proposed quantized DNN training method with 8-bit S-DFP (S-DFP8).

#### 3.1.2.1 Analysis of accuracy degradation with dynamic fixed-point

Figure 3.3 shows the validation accuracy on the ImageNet task with ResNet-34 when using FP32 and DFP8. From Fig. 3.3(b), when using only DFP8 for training, the accuracy is degraded markedly. One reason for accuracy degradation when using DFP is that the step size in the weight update becomes large by quantizing the weight gradient with DFP in comparison with when using FP32. Figure 3.4 shows the relation between the step size in the weight and the loss function when using DFP8 and FP32. In DNN training using stochastic gradient descent (SGD), the loss function is reduced by updating the weight with the weight gradient. Additionally, the value of the loss function is optimized by the step size depending on the value of weight gradient. When using DFP for quantization,

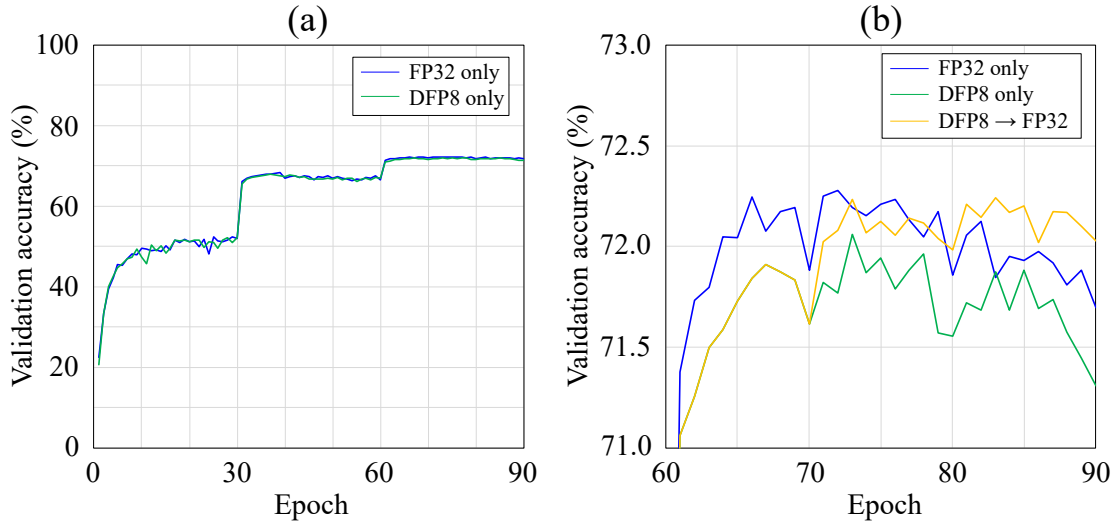


Fig. 3.3 Top-1 validation accuracy on ImageNet task with ResNet-34: (a) accuracy during 0 to 90 epochs and (b) accuracy during 60 to 90 epochs.

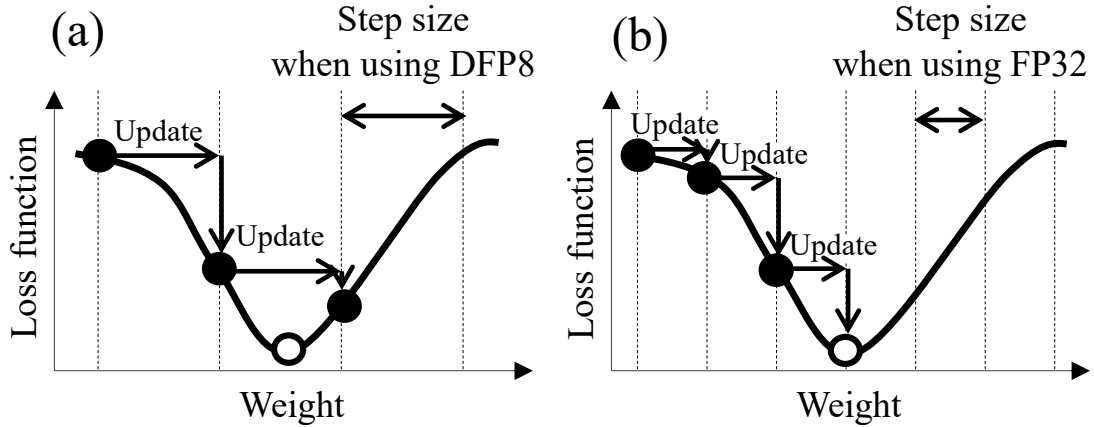


Fig. 3.4 Relation between the step size of the weight update and loss function: (a) because of the large step size when using DFP8, it is difficult to reach the minimum loss function; (b) because the step size is small when using FP32, the loss function can reach the minimum.

because the small values included in the vector before quantization are out of the data representation range of DFP, the small values are rounded to zero or the minimum value that can be represented by DFP, as shown in Fig. 3.2. Therefore, information of the small values which are containing the vector before quantization is lost. In contrast, since loss of information of large values contained in the vector before quantization is small, the weight gradient quantized with DFP includes more information of large values than

information of small values. Therefore, when using the weight gradient quantized with DFP, it is difficult for the weight to reach a position that minimizes the loss function because the weights are not updated by the small step size of the weight gradient as shown in Fig. 3.4(a). On the other hands, as shown in Fig. 3.4(b), when switching the data format of DNN model from DFP8 to FP32 in the middle of training, the step size in weight update is also switched from a large value to a small value as shown in Fig. 3.4(b). Consequently, because the weight can move the position at which the loss function is reduced, the accuracy of quantized model becomes equivalent to the accuracy achieved with FP32.

### 3.1.2.2 Shifted dynamic fixed precision format

Figure 3.5 shows an overview of the proposed S-DFP format to reduce the step size during weight updating. As shown in Fig. 3.5(a), the conventional DFP format is intended mainly to represent large values without overflow by application of large value shared exponents while the range of underflow becomes large. Therefore, because only the information of large values remains in the weight gradient quantized by the conventional DFP, the step size of the weight update becomes large. On the other hand, the proposed S-DFP format can reduce underflowed entries by changing the exponent value of DFP, as shown in Fig. 3.5(b). To shift the data representation range of DFP so that the data representation range of DFP includes small values in the vector before quantization, we modify the definition of shared exponents  $e_m$  and  $e_s$  to  $e_{m\_mod}$  and  $e_{s\_mod}$  respectively, as shown in Eq. 3.4 and Eq. 3.5.

$$e_{m\_mod} = e_m - bias \quad (3.4)$$

$$e_{s\_mod} = e_{m\_mod} - BW - 2 \quad (3.5)$$

In these equations,  $bias$  is the parameter to shift the data representation range of DFP. The value of quantized element with DFP using modified exponent is represented as presented below.

$$x_{i,q} = round \ x_i / 2^{e_{s\_mod}} \cdot 2^{e_{s\_mod}} \quad (3.6)$$

The proposed S-DFP format uses these modified exponents instead of conventional exponents.  $e_{s\_mod}$  becomes small by setting a positive value to  $bias$ , the data representation

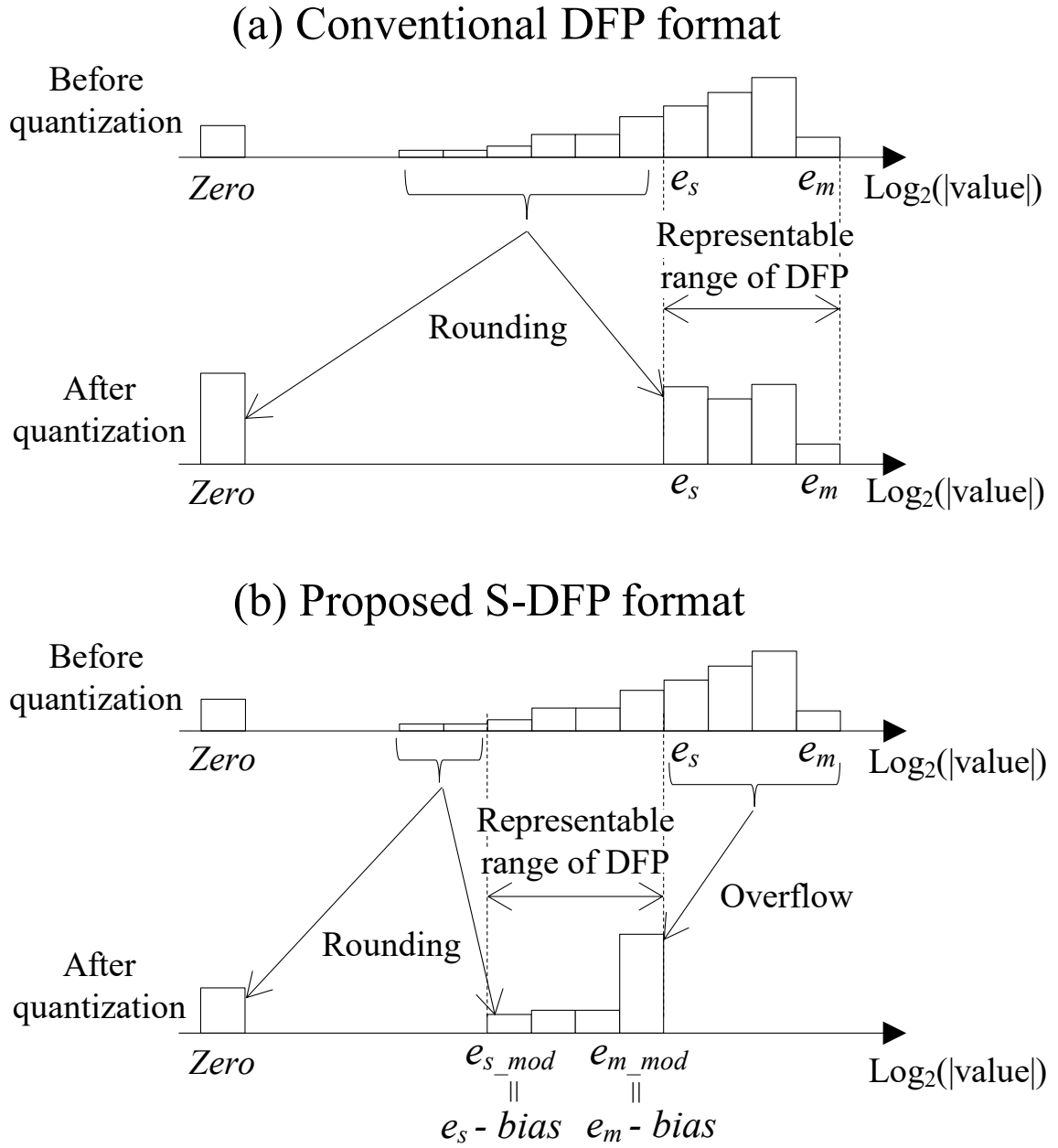


Fig.3.5 Overview of the proposed quantization method: (a) When using conventional DFP, the large value information remains, whereas the small value information is lost. (b) When using proposed S-DFP, the data representable range of DFP is shifted by adding bias to the exponent. Consequently, the large value information is lost and the small value information remains

range of DFP is shifted to the area of small values depending on the value of *bias*. Therefore, information of small values including the vector before quantization remains in the vector after quantization. In contrast, large values including the vector before

quantization are overflowed. Consequently, as shown in Fig.3.5, because the modified shared exponent quantizing the weight gradient by S-DFP, the step size during weight update becomes smaller than when quantizing the weight gradient using conventional DFP.

Algorithm 3.1 summarizes the training method for quantized DNN models using the proposed S-DFP.

Algorithm 3.1 DNN training procedure in the proposed quantization method.

```

1: Set the training duration to num.
2: for i=0 to num do
3:   Calculate the loss by the quantized weight and activation with DFP.
4:   Backpropagate by quantized activation gradient with DFP.
5:   if validation accuracy is saturated then
6:     Quantize the weight gradient with proposed S-DFP.
7:   else
8:     Quantize the weight gradient with DFP.
9:   end if
10:  Update weight with the quantized weight gradient.
11: end for

```

### 3.1.3 Experiment

We evaluated the proposed quantization method on the ImageNet classification task using ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The DNN models were trained by momentum SGD with L2 weight decay. The mini-batch size was 256. The momentum coefficient was 0.9. The coefficient of weight decay was  $10^{-4}$ . The initial learning rate was 0.1. The learning rate was multiplied by 0.1 every 30 epochs. We quantized the weight, activation, weight gradient and activation gradient using conventional DFP8. Furthermore, we quantized the weight gradient with proposed S-DFP8 from 70 epochs at when the validation accuracy is saturated. We used stochastic rounding for the quantized model training proposed in [22]. During the weight update, the FP32 weights were subtracted by the DFP8 weight gradient to prevent accuracy degradation, as demonstrated in [14].

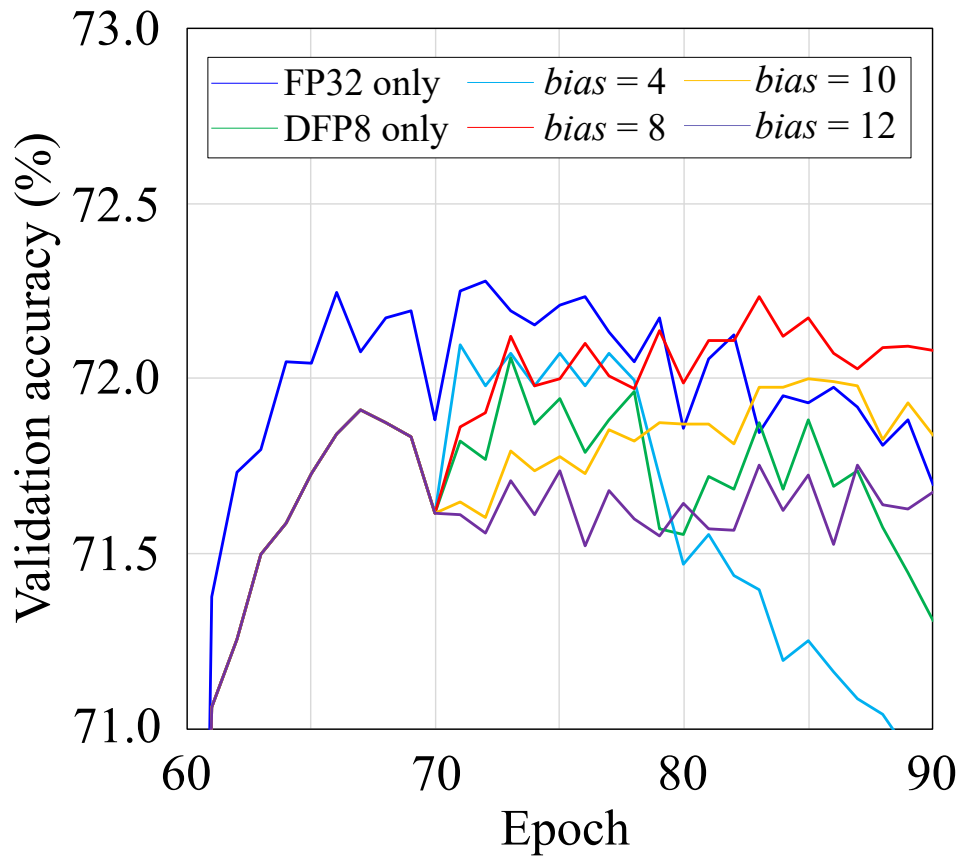


Fig. 3.6 Validation accuracy on the ImageNet task with ResNet-34 when applying S-DFP8 using various *bias* values.

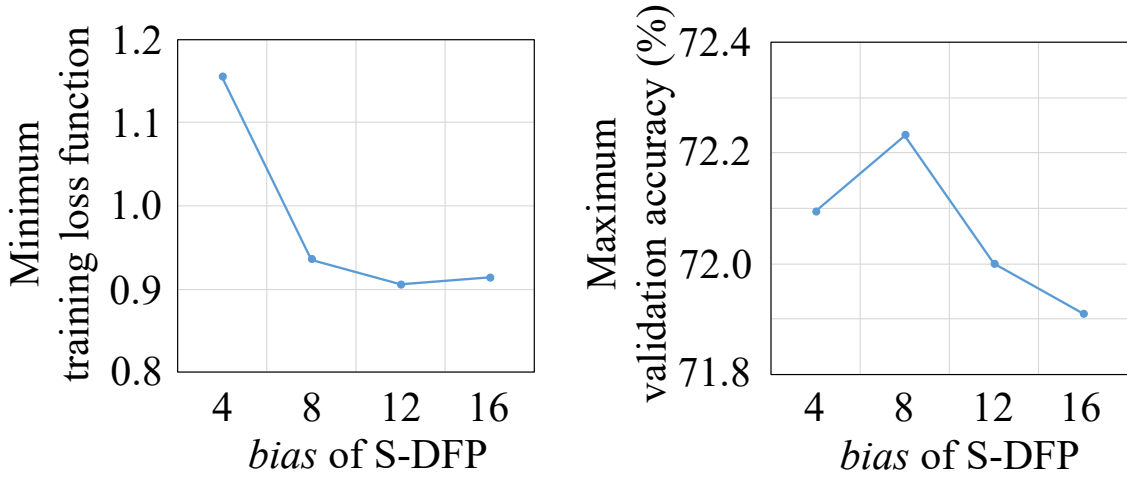


Fig. 3.7 Minimum training loss function and maximum validation accuracy on various *bias*.

Table 3.1 Comparison of Top-1 maximum validation accuracies for the respective quantization methods. In all cases, *bias*=8.

| Model      | Accuracy of FP32 (%) | Accuracy of DFP8 (%) | Accuracy of S-DFP8 (%) |
|------------|----------------------|----------------------|------------------------|
| ResNet-34  | 72.2                 | 72.0                 | 72.2                   |
| ResNet-50  | 75.5                 | 75.1                 | 75.4                   |
| ResNet-101 | 76.8                 | 75.9                 | 76.8                   |
| ResNet-152 | 77.6                 | 76.6                 | 77.6                   |

### 3.1.3.1 Parameters setting for S-DFP

Because the step size for the weight update is changed depending on the S-DFP value, the value of *bias* affects the validation accuracy and loss function during training. Figure 3.6 displays the relation among *bias*, validation accuracy and training loss function on the ImageNet task with ResNet-34 when switching the data format for quantization from conventional DFP to S-DFP in the middle of training for the quantized model. As shown in Fig. 3.7, when the *bias* is small, because the step size is large, the training loss function does not become small. As a result, the maximum validation accuracy is also small. When the *bias* is large, because the step size becomes small, the number of weights updating times becomes large to reduce the loss function sufficiently. Consequently,

unless the number of training times are increased, the maximum validation accuracy can be expected to be small. In this paper, *bias* for each evaluated model, we chose the *bias* which achieved the best maximum validation accuracy. The *bias* of each evaluated DNN model is shown in Table 3.1.

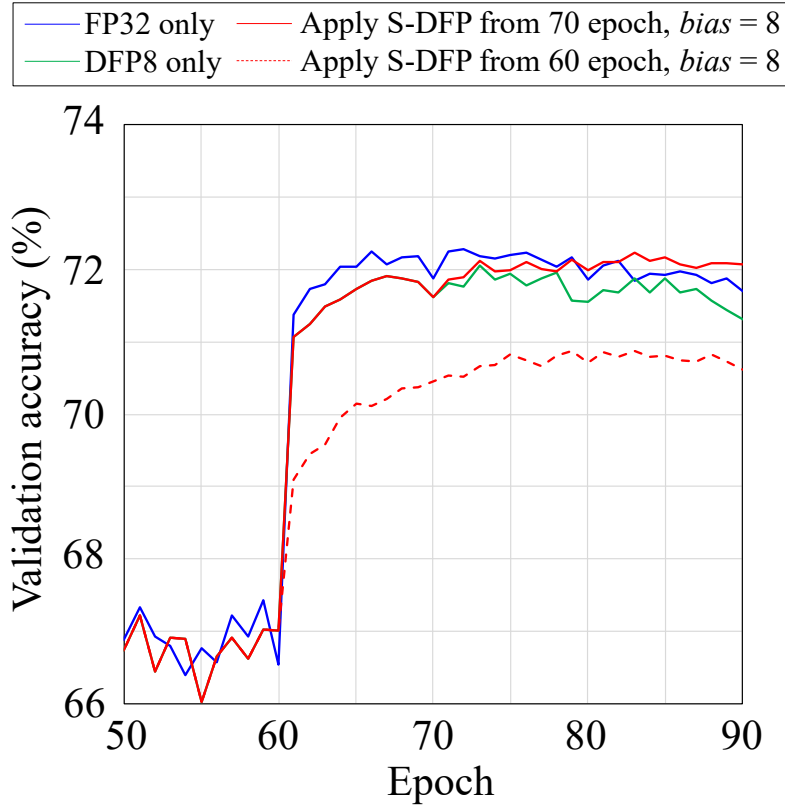


Fig. 3.8 Effects of S-DFP applying timing on the ImageNet classification task with ResNet-34. Suitable timing for applying S-DFP to quantized model training is after saturating validation accuracy.

Figure 3.8 shows the validation accuracy when applying S-DFP for weight gradient quantization at the timing of 60 epochs and 70 epochs. When applying S-DFP with a positive *bias* to weight gradient quantization, the step size for weight update becomes small. Therefore, when S-DFP is applied from 60 epochs with low accuracy, the maximum validation accuracy becomes small unless the number of training times are increased. In contrast, when S-DFP is applied from 70 epochs with high accuracy, because the maximum validation accuracy is improved to that of FP32 without increasing the number of training times, the timing of S-DFP applied for weight gradient quantization is suitable for after the validation accuracy saturation. In this paper, we

applied S-DFP for weight gradient quantization from 70 epochs when using ResNet-34, ResNet-101, and ResNet-152. Moreover, we applied S-DFP for weight gradient quantization from 80 epochs when using ResNet-50.

### 3.1.3.2 Experimental results

The experimental results of validation accuracy are shown in Fig. 3.9, Fig. 3.10, Fig. 3.11, Fig. 3.12, and Table 3.1. The validation accuracy of the quantized model with conventional DFP8 are degraded as the number of layers of ResNet gets deeper. On the other hand, the validation accuracy of the quantized model with proposed S-DFP was achieved equivalent accuracy with FP32 regardless of the depth in ResNet. Therefore, the proposed S-DFP is expected to significantly improve the accuracy of the quantized model regardless of the DNN model type, such as transformer other than ResNet.

Table 3.2 presents a comparison of the accuracies with those obtained from earlier studies [19] that used 16-bit DFP and [23] that used 8-bit DFP for quantized DNN training on ImageNet task with ResNet-50. Our proposed method achieved DNN training and inference with 8-bit quantization while maintaining 75.4% accuracy on ImageNet. These results demonstrate a smaller accuracy degradation than the that of conventional DFP8. In addition, the proposed method achieved the equivalent accuracy with conventional 16-bit DFP. Therefore, the proposed S-DFP format is more suitable for accelerating DNN training while maintaining accuracy in comparison with conventional DFP format.

## ResNet-34

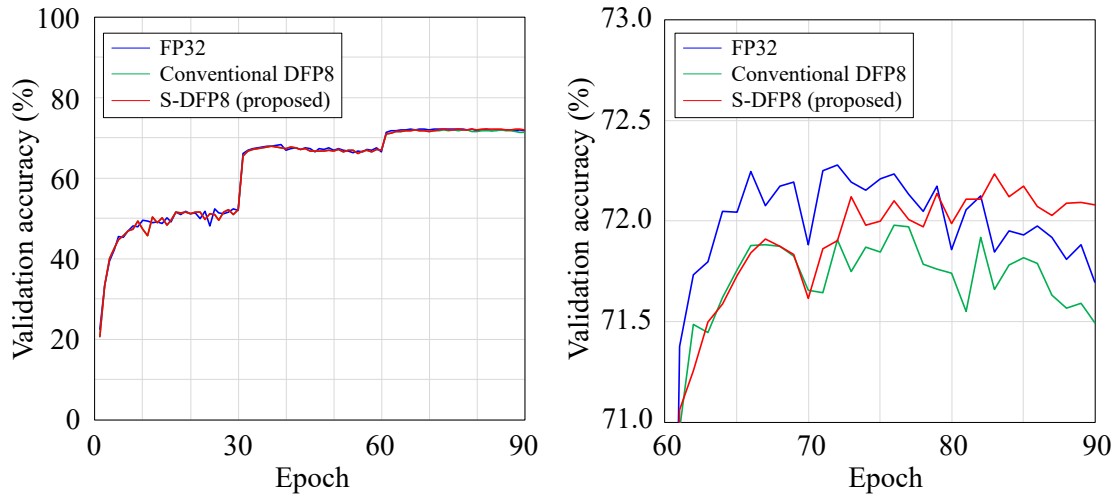


Fig. 3.9 Experimental results on the ImageNet classification task with ResNet-34: (left) Top-1 validation accuracy during whole training duration. (right) Top-1 validation accuracy after 60 epochs.

## ResNet-50

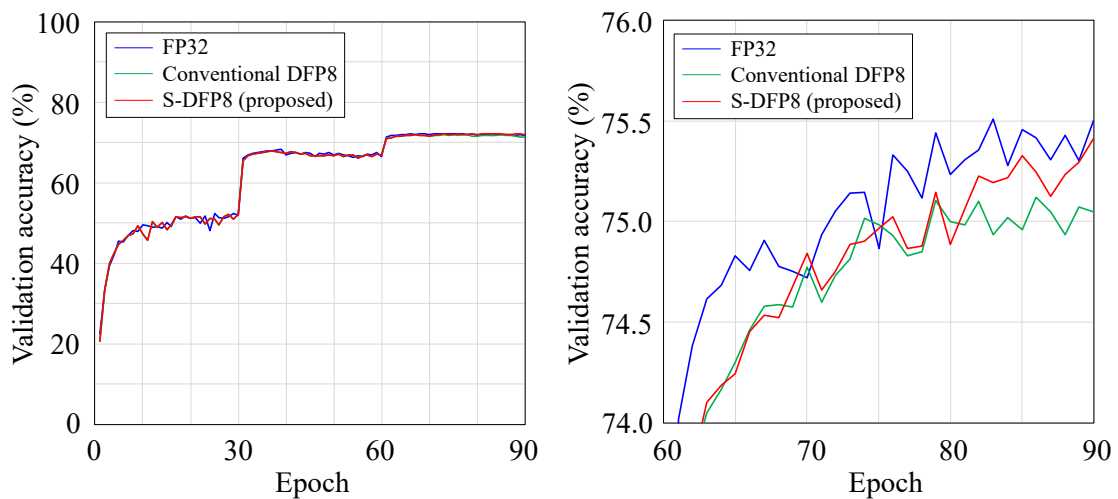


Fig. 3.10 Experimental results on the ImageNet classification task with ResNet-50.

## ResNet-101

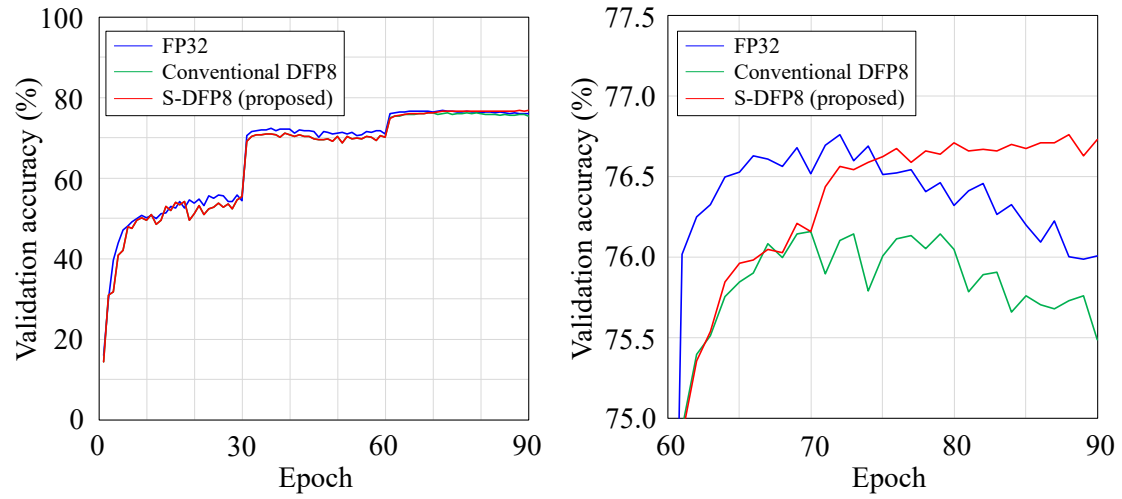


Fig. 3.11 Experimental results on the ImageNet classification task with ResNet-101.

## ResNet-152

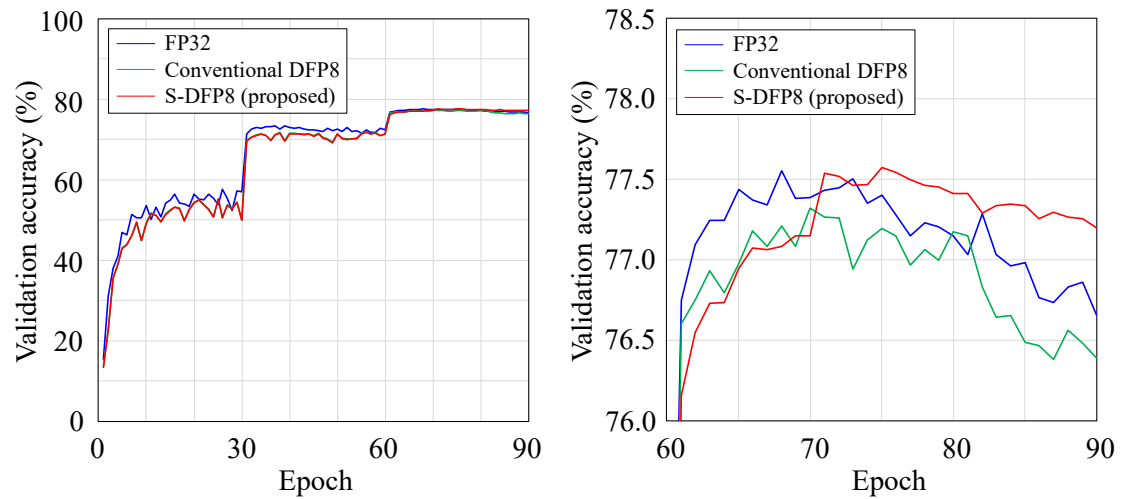


Fig. 3.12 Experimental results on the ImageNet classification task with ResNet-152.

Table 3.2 Comparison of Top-1 validation accuracies with related works on ImageNet task with ResNet-50.

| Method           | Format       | Bit-width | Accuracy of FP32 (%) | Accuracy of reduced precision (%) |
|------------------|--------------|-----------|----------------------|-----------------------------------|
| Das et al. [19]  | DFP          | 16        | 75.7                 | 75.7                              |
| Yang et al. [23] | DFP          | 8         | 74.7                 | 68.0                              |
| <b>Ours</b>      | <b>S-DFP</b> | <b>8</b>  | 75.5                 | <b>75.4</b>                       |

## 3.2 Summary

In this chapter, we describe our proposed quantization method using S-DFP format for DNNs training. To prevent accuracy degradation due to quantized DNN training when using conventional DFP format, S-DFP shifts the data representable range of DFP from a large value area to a small value area by adding bias to the exponent of DFP. Using the proposed S-DFP format for quantized DNNs training, the quantized model can be trained using 8-bit fixed-point precision without significant accuracy degradation on the ImageNet task with ResNet-32, ResNet-50, ResNet-101, and ResNet-152. Since the validation accuracy of the quantized model with proposed S-DFP was achieved equivalent accuracy with FP32 regardless of the depth in ResNet, the proposed S-DFP is expected to significantly improve the accuracy of the quantized model regardless of the DNN model type, such as transformer other than ResNet. In addition, the proposed method achieved the equivalent accuracy with conventional 16-bit DFP instead of using 8-bit S-DFP. Therefore, the proposed method is more suitable for accelerating DNN training while maintaining accuracy in comparison with conventional DFP format.

## Chapter 4 Automated hyperparameter search for DNN acceleration

This chapter describes an automatic hyperparameter search for DNNs acceleration methods such as pruning to avoid inefficient manual hyperparameter assignment. For this purpose, we studied the following contents:

- 1) We propose an automatic pruning rate search method for structured pruning of DNNs to reduce the workload of inefficient manual pruning rate assignment for achieving the target accuracy of the pruned model.
- 2) We demonstrate that our proposed method can reduce the model parameters and the number of floating-point operations (FLOPs) on CIFAR-10 and ImageNet with various ResNets while maintaining good accuracy.
- 3) Because our method utilizes the pruning error (the difference in weight magnitude before and after pruning) to determine the pruning rate, it is applicable to quantization.

Furthermore, we described a structured pruning method without degrading the prediction accuracy of complex DNNs with shortcut connection. In addition, we demonstrate the effectiveness of our proposed pruning method with zero-padding layers:

- 4) To improve the inference speed and the compression ratio for the DNN model with a shortcut connection such as ResNet, we propose structured pruning method that inserts zero-padding layers into all input paths to the addition operator.
- 5) Using our proposed pruning method, the compression ratio and the FLOPs of the pruned model is considerably improved without significant accuracy degradation on the ImageNet task with ResNet-34 and ResNet-50, and CIFAR-10 with ResNet-32, ResNet-56, and ResNet-110.

Structured pruning is utilized to compress the neural network model size. Existing methods require manual assignment of the pruning rate, which can lead to significant accuracy degradation if the pruning rate is not assigned appropriately. To address this issue, we propose GAPRS (Gradient-Aware Automatic Pruning Rate Search), an automatic pruning rate search method for structured pruning. Our method reduces the

workload associated with manual pruning rate assignment and ensures the target accuracy of the pruned model. Our method initially sets multiple pruning rate candidates and then automatically searches for the appropriate pruning rate using the pruning error and the gradient of the pruned weights. By applying an iterative pruning approach, our method can significantly prune the entire model without being constrained by the initial pruning rate candidates. Additionally, we propose a technique to enhance the compression ratio for models with shortcut connections, such as ResNet, by inserting zero-padding layers into all input paths to the addition operator. We evaluated GAPRS against various state-of-the-art methods on CIFAR-10 and ImageNet using various ResNet architectures. For ResNet-50, our method achieved a 60.1% reduction in parameters while maintaining 76.17% accuracy on ImageNet. These results demonstrate the effectiveness of GAPRS in achieving high compression ratios without model accuracy degradation.

## 4.1 GAPRS: Gradient-aware automatic pruning rate search for structured pruning

### 4.1.1 Related works

#### 4.1.1.1 Importance for pruning

When applying weight pruning to a DNN model, the weights with small importance below a threshold determined by the pruning rate are removed. The importance of structured pruning is calculated in terms of high granularity, such as channels. The magnitude of weights is widely utilized for pruning importance [24, 33, 52, 54, 59], as are the pruning error of feature maps [25, 27, 53] and the scaling factor of the batch normalization layer [31]. Previous works [28, 55-58] utilize the gradient of the loss function with respect to the weights targeted for pruning for importance to reflect how the pruned weights affect the accuracy of the pruned model. The gradient indicates the change in the loss function of the pruned model. This reflects the impact of the pruned weights on the accuracy of the pruned model. Therefore, by utilizing the gradient for importance, we can expect to perform pruning while assessing loss change. However, another work [32] has reported that using the gradient for pruning importance results in lower performance of the pruned model than that obtained by using the magnitude.

#### 4.1.1.2 Local pruning and global pruning

Local pruning and global pruning are the two main methods for setting the pruning rate. Local pruning [24, 45, 60-62] sets the pruning rate individually for each layer of the target model. Typically, the number of combinations of pruning rates is extremely high in local pruning, making it difficult to find the optimal combination of rates. In contrast, global pruning [33, 56, 63, 64] sets only one pruning rate for the entire model, and the pruning rate for each layer is generally different from the pruning rate of the entire model. Although only one pruning rate is set in global pruning, it is still manually determined. Therefore, the pruning rate must be manually tuned through trial and error while checking the accuracy of the pruned model.

#### 4.1.1.3 One-shot pruning and iterative pruning

The pruned model is often fine-tuned to mitigate the accuracy degradation caused by pruning. Earlier works [25, 34] have proposed a one-shot pruning, where the model is pruned at a high pruning rate only once and then fine-tuned. Alternatively, there is an iterative pruning [25, 26, 35, 66-68] that iteratively prunes the model at a low pruning rate and fine-tunes it. Other works [33, 65, 68] have reported that the accuracy degradation of the pruned model is less with iterative pruning than with one-shot pruning.

In our proposed method, the magnitude of weights is used as the importance for pruning to mitigate the accuracy degradation of the pruned model. Unlike conventional local and global pruning, where the pruning rates are manually set, the proposed method automatically derives the pruning rate for each layer that does not degrade the accuracy of the pruned model by utilizing the gradients of the pruned weights. Therefore, even if the model is complex and the number of pruning rate combinations is large, it is easier to find the optimal pruning rate for each layer compared to conventional methods (details of the derivation method are described in Section 4.1.2.1). Furthermore, the proposed method iteratively performs pruning and fine-tuning using the automatically derived pruning rates to mitigate the accuracy degradation of the pruned model.

### 4.1.2 Automatic pruning rate search

An overview of our proposed automatic pruning rate search method is presented in Fig. 4.1. First, we construct the model for pruning, as shown in Fig. 4.1(a). Next, the pruning rate for each layer is chosen from the pruning rate candidates by comparing the pruning error with the threshold (Fig. 4.1(b)). The pruning error is defined by the L1-norm of the difference between the weight before and after pruning. Hence the magnitude of the pruning error depends on the pruning rate, as shown in Fig. 4.1(b). The threshold is derived from the gradient of the pruning target weight. (Details of how to derive the threshold are described in Section 4.1.2.1) The largest pruning rate among the candidates exhibiting a pruning error less than the threshold is used for pruning. In Fig. 4.1(b), the pruning rate candidates are set to 50%, 40%, 30%, 20%, 10%, 0%. Note that, the threshold is derived by Taylor expansion in our method. Therefore, the threshold includes the

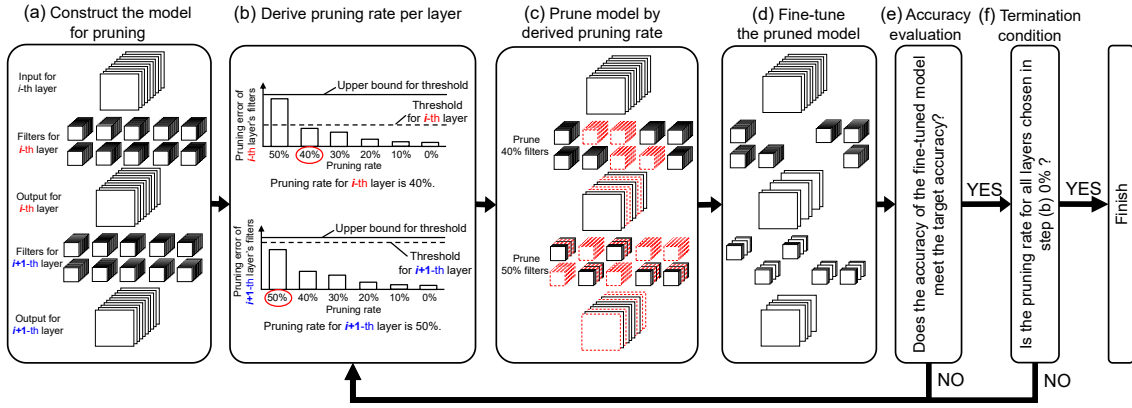


Fig. 4.1 Overview of our proposed pruning search. The pruning rate for each layer is chosen from the pruning rate candidates by comparing the pruning error with the threshold. The model is then pruned using the chosen pruning rates, and the pruned model is fine-tuned. If the accuracy of the fine-tuned model meets the target, the pruned fine-tuned model is moved to the next step (f); otherwise, the model is discarded and the pruning rates are re-searched. Our method iteratively searches for pruning rates until the pruning rate for all layers chosen in step (b) is 0%.

approximation error. To avoid an incorrect pruning rate search stemming from the approximation error included in the threshold by the Taylor expansion, we implement the trust region method [51, 69]. Trust region method is an optimization method for parameters that include errors. In the trust region method, the upper bound of the parameters including errors is limited. In our method, since the threshold includes the

approximation error by Taylor expansion, the upper bound of the threshold is set. After deriving the pruning rate, the pruning target model is pruned, as shown in Fig. 4.1(c), and then the pruned model is fine-tuned, as shown in Fig. 4.1(d). The fine-tuned model is then evaluated (Fig. 4.1(e)) to determine if it achieves the target accuracy. If the accuracy of the fine-tuned model meets the target, the pruned fine-tuned model is moved to the next step; otherwise, it is discarded. We continue to apply the model pruning iteratively until the pruning rate for all layers chosen in step Fig. 4.1(b) is 0% (Fig. 4.1(f)). By applying the model pruning iteratively, the entire model can be pruned by a lower granularity pruning rate than the pruning rate candidates. The details for each step in Fig.3 are described in the following subsections.

#### 4.1.2.1 Pruning rate search by gradient-aware threshold

The loss function of the model pruned by the ideal pruning rate is expected not to be degraded. We set this ideal situation as the following constraint: *The loss function of the model pruned by the ideal pruning rate is less than or equal to the loss function of the model before pruning.* When only one element of the pruning target weight vector is pruned, this constraint is presented as

$$L(w_1 + \Delta w_1, w_2, \dots, w_n) \leq L_b, \quad (4.1)$$

where  $w_i$  is the  $i$ -th element of a pruning target weight vector ( $i = 1, 2, \dots, n$ ),  $n$  is the number of elements of the pruning target weight vector, and  $\Delta w_i$  is a pruning error of  $w_i$ .  $\Delta w_i$  is defined as the difference between the weight magnitude before and after pruning of the  $i$ -th element. Thus,  $\Delta w_i = -w_i$ .  $L(w_1 + \Delta w_1, w_2, \dots, w_n)$  is a loss function of the pruned model.  $L_b$  is the loss function of the model before pruning. By applying first-order Taylor expansion to the loss function of the pruned model presented in Eq. 4.1, we can derive

$$L(w_1 + \Delta w_1, w_2, \dots, w_n) \approx L(W) + \frac{\partial L(W)}{\partial w_1} \cdot \Delta w_1 \leq L(W) + \left| \frac{\partial L(W)}{\partial w_1} \right| \cdot |\Delta w_1| \leq L_b, \quad (4.2)$$

where  $W = [w_1, w_2, \dots, w_n]^T$  is the weight vector without pruning. Here, since  $W$  is

the weight vector without pruning,  $L(W)$  represents the loss function of the model before pruning. Therefore, as the value of  $L(W)$  is the same as the value of  $L_b$ , Eq. 4.2 cannot be satisfied. To satisfy Eq. 4.2, we relax the constraint by introducing a margin of the loss function, namely, *the loss function of the model pruned by the ideal pruning rate is less than or equal to the loss function of the model before pruning “with a margin”*. We modify Eq. 4.2 in accordance with the relaxed constraint, as

$$L(W) + \left| \frac{\partial L(W)}{\partial w_1} \right| \cdot |\Delta w_1| \leq L_b + L_b \cdot L_m \quad (4.3)$$

Where  $L_b \cdot L_m$  is the margin of the loss function introduced by the relaxed constraint, and  $L_m$  is the hyperparameter to control the value of the loss function margin. By isolating  $|\Delta w_1|$  on the left-hand side of Eq. 4.3 and rearranging the other terms to the right-hand side, the threshold of the pruning error  $|\Delta w_1|$  that satisfies the relaxed constraint is derived as follows.

$$|\Delta w_1| \leq \frac{L_b + L_b \cdot L_m - L(W)}{\left| \frac{\partial L(W)}{\partial w_1} \right|}. \quad (4.4)$$

Equation 4.4 represents the upper bound of the pruning error  $|\Delta w_1|$  when only the element  $w_1$  of the weight vector  $W$  is pruned. Therefore, we can determine whether  $w_1$  can be pruned or not in terms of the loss function degradation by pruning using Eq. 4.4.

Next, we expand Eq. 4.4 for higher granularity pruning, such as channel-wise pruning. A similar equation to Eq. 4.4 holds for each element  $w_i$  included in  $W$ :

$$|\Delta w_i| \leq \frac{L_b + L_b \cdot L_m - L(W)}{\left| \frac{\partial L(W)}{\partial w_i} \right|}. \quad (4.4')$$

Equation 4.4' represents the upper bound of  $|\Delta w_i|$  when each element  $w_i$  is pruned. In channel-wise pruning, we remove an entire channel, which removes all elements  $w_1$  to  $w_n$  of  $W$  associated with that channel. Therefore, since the total pruning error for a

channel is the sum of the individual weight pruning errors within that channel, the upper bound of the pruning error in channel-wise pruning is represented by the sum of the absolute values of the errors when each element in  $W$  is pruned,  $\sum_{i=1}^n |\Delta w_i|$ . Thus, by summing the inequality of Eq. 4.4' from  $i = 1$  to  $n$ , we can obtain the upper bound of the pruning error in channel-wise pruning:

$$\sum_{i=1}^n |\Delta w_i| \leq L_b + L_b \cdot L_m - L(W) \cdot \sum_{i=1}^n \frac{1}{|\partial L(W)/\partial w_i|}.$$

We obtain the upper bound of the average pruning error per weight by normalizing both sides by the number of elements  $n$  in the weight vector  $W$ :

$$\Delta W = \frac{1}{n} \cdot \sum_{i=1}^n |\Delta w_i| \leq \frac{L_b + L_b \cdot L_m - L(W)}{n} \cdot \sum_{i=1}^n \frac{1}{|\partial L(W)/\partial w_i|} = T_h, \quad (4.5)$$

where  $\Delta W$  is the L1-norm of the pruned weight normalized by the number of elements of the pruned weight  $n$ . The right side of Eq. 4.5 is the threshold of  $\Delta W$ ,  $T_h$ . The loss function of the pruned model does not degrade when the pruning error  $\Delta W$  is less than or equal to the threshold  $T_h$ . Therefore, the pruning rate where  $\Delta W$  is less than or equal to  $T_h$  can be found by comparing  $\Delta W$  and  $T_h$  while changing the pruning rate. In our method, the changed pruning rate is chosen from the pruning rate candidates  $Rate_{cand}$ , and the largest pruning rate chosen from  $Rate_{cand}$  where  $\Delta W$  is less than or equal to  $T_h$  is used for the model pruning, as shown in Fig. 4.1(b).

In Eq. 4.5, since the hyperparameter  $L_m$  affects the value of  $T_h$ ,  $L_m$  may affect the result of the chosen pruning rate from  $Rate_{cand}$ . However, in our proposed method, the pruning rate utilized for model pruning is chosen only when the accuracy of the model pruned by the chosen pruning rate meets the target accuracy, as shown in Fig. 4.1(e). In other words, even if an extreme value is given to  $L_m$ , the chosen pruning rate is not used for model pruning unless the pruned model's accuracy meets the target accuracy. As a result, the accuracy of the model pruned by the pruning rate derived with our method is robust against the value of  $L_m$ . We experimentally demonstrate the robustness of the accuracy and the compression ratio of the model pruned by our method against  $L_m$  in Section 4.1.3.

Both  $\Delta W$  and  $T_h$  in Eq. 4.5 are derived from each  $W$ . Therefore, our proposed method derives the pruning rate for each  $W$  in the same way as local pruning. In other words, the proposed method can derive the pruning rate for any pruning granularity: layer-wise, channel-wise, or element-wise. It can thus also derive the pruning rate for unstructured pruning by calculating the pruning error at the level of element-wise granularity, not just for structured pruning. Algorithm 4.1 shows how the pruning rate is chosen from  $Rate_{cand}$ .

Algorithm 4.1 Pruning rate search by comparing pruning error  $\Delta W$  and threshold  $T_h$ .

```

Input: Pruning target weight  $W$ 
Input: Threshold  $T_h$ 
Input: Pruning rate candidates  $Rate_{cand}$ 
           # for example,  $Rate_{cand} = [50\%, 40\%, 30\%, 20\%, 10\%, 0\%]$ 
Output: Derived pruning rate

1:   $i = 0$ 
2:   $Rate_{tmp} = Rate_{cand}[i]$  # Set maximum pruning rate  $Rate_{cand}$ .
3:  Calculate the pruning error  $\Delta W$  with  $Rate_{tmp}$ .
4:  while  $i \leq \text{number of } Rate_{cand}$  do
5:    if  $\Delta W \leq T_h$  then
6:      break
7:    else
8:       $i = i + 1$ 
9:       $Rate_{tmp} = Rate_{cand}[i]$ 
           # Set the pruning rate as one lower than the last iteration.
10:   Calculate pruning error  $\Delta W$  with  $Rate_{tmp}$ .
11:   end if
12: end while
13: return  $Rate_{tmp}$ 

```

Iterative pruning proposed in [26, 35] is a method for suppressing the pruning-related accuracy degradation. We also apply iterative pruning to the proposed pruning rate search. Specifically, the proposed method iteratively executes model pruning with the derived pruning rate and fine-tuning of the pruned model. The threshold at  $k$  times during iterative pruning  $T_{h,k}$  is represented as

$$T_{h,k} = \frac{L_b + L_b \cdot L_m - L W_k}{n} \cdot \sum_{i=1}^n \frac{1}{\partial L(W_k) / \partial w_{i,k}}, \quad (4.6)$$

where  $W_k$  is the pruned model weight after searching for the pruning rate  $k$  times and  $w_{i,k}$  is the  $i$ -th element of  $W_k$ . Since  $k$  represents the number of times a pruning rate search is implemented, when  $k=0$ ,  $W_0$  is the same as the weight before model pruning. Therefore, the value of  $L(W_0)$  is the same as that of  $L_b$ . As a result, the initial value of threshold  $T_{h,0}$  depends on the value of  $L_b \cdot L_m$ , because  $L_b - L(W_0)$  in the numerator of the right side of Eq. 4.6 becomes zero.

$T_{h,k}$  includes an approximation error because it is derived by Taylor expansion. Thus, the pruning rate derived by Eq. 4.6 also includes an approximation error. To mitigate the impact of approximation errors from the Taylor expansion on the pruning rate search, we implement the trust region method [51, 69], which is an optimization method that utilizes parameters including errors. Specifically, we optimize the threshold  $T_{h,k}$  derived from the Taylor expansion within the trust region. The trust region is defined as the range from zero to the upper bound of the parameters to be optimized, including error. In the proposed method, since  $T_{h,k}$  includes the approximation error,  $T_{h,k}$  for all pruning target weight tensors is limited by the upper bound  $U_b$ :

$$U_b \geq \|T_{h,k,all}\|_2. \quad (4.7)$$

Here,  $T_{h,k,all} = [T_{h,k,1}, T_{h,k,2}, \dots, T_{h,k,m}]$  is a vector of the thresholds.  $T_{h,k,i}$  ( $i = 1, 2, \dots, m$ ) is the threshold for each pruning target weight  $W_{k,i}$ , and  $m$  represents the number of the pruning target weight. At each step of the pruning rate search,  $U_b$  is dynamically updated based on whether the accuracy of the pruned model meets the target accuracy. The details of how to update  $U_b$  are described in Section 4.1.2.2. The method for setting the initial value of  $U_b$  is described in Section 4.1.2.3.

#### 4.1.2.2 Model pruning, fine-tuning, and accuracy evaluation

After deriving the pruning rate, the pruning target model is pruned by the derived pruning rate. Our proposed method utilizes the L1-norm of the weight value for pruning importance, and then we fine-tune the pruned model to compress the accuracy degradation by pruning. We utilize the remaining weights of the pruned model for fine-tuning.

To obtain a pruned model with the target accuracy, the proposed method evaluates the accuracy of the pruned model after fine-tuning  $Acc_f$ , and then compares  $Acc_f$  with the target accuracy. To control the target accuracy, we introduce a parameter  $Acc_{ctrl}$ . When comparing the accuracy, we first set the baseline accuracy  $Acc_b$ , then set the target accuracy  $Acc_b - Acc_{ctrl}$ .  $Acc_b$  is set to the accuracy of the model before pruning since the pruned model accuracy typically is expected to be the same as the accuracy of the model before pruning. When  $Acc_f$  exceeds the target accuracy  $Acc_b - Acc_{ctrl}$ , fine-tuning is terminated before reaching the maximum epoch, and it proceeds to the next step, Fig. 4.1(f). The maximum epoch for fine-tuning is set as an initial value before fine-tuning. Otherwise, the fine-tuned model is discarded.

The parameter  $U_b$  that limits the upper bound of the threshold is also updated by using the accuracy comparison result. When  $Acc_f$  meets the target accuracy  $Acc_b - Acc_{ctrl}$ , the approximation error included in the threshold is expected to be small and the pruned model is likely to still contain redundant parts.  $U_b$  is thus increased so that a higher pruning rate can be chosen for the next pruning rate search and the remaining redundant parts can be removed. In contrast, when  $Acc_f$  does not exceed the target accuracy  $Acc_b - Acc_{ctrl}$ , the approximation error is expected to be large and  $U_b$  is decreased.

The proposed method iteratively executes pruning rate selection, model pruning, fine-tuning, and accuracy comparison until the pruning rate of 0% is chosen in the step of Fig. 4.1(b) for all pruning target weights.

#### 4.1.2.3 Initializing and updating the upper bound of threshold

When the initial value of the upper bound for threshold  $U_{b,init}$  is small, the initial threshold  $T_{h,0}$  is set to a small value at the first search, as the upper bound of  $T_h$  is limited by  $U_b$  due to the constraint in Eq. 4.7. Consequently, small pruning rates are

chosen at the first pruning rate search. This can lead to early termination of the search. In extreme case, if  $U_{b,init}$  is zero, since all pruning rates are set to 0%, pruning rate search is terminated in the first search. To avoid this, the maximum pruning rate in  $Rate_{cand}$  should be chosen at the first search. In our method, according to Eq. 4.5, the maximum pruning rate in  $Rate_{cand}$  can be chosen when  $T_h$  is set to more than the value of the maximum pruning error calculated by the maximum pruning rate. Additionally, from Eq. 4.7, the L2-norm of  $T_h$  is limited by  $U_b$ . Therefore, when  $U_b$  is set to the L2-norm of the maximum pruning rate, the limitation of pruning rate selection by  $U_b$  does not occur. Consequently, to set the maximum pruning rate in  $Rate_{cand}$ , we set the value of  $U_{b,init}$  to the L2-norm of the maximum pruning rate in the pruned model.

The proposed method updates  $U_b$  to a larger value when the accuracy of the finetuned model meets the target accuracy during the pruning rate search. However, when the value of  $U_b$  is extremely large, convergence of the pruning rate search takes a long time. We thus limit the upper bound of  $U_b$  by  $U_{b,init}$ .

Algorithm 4.2 summarizes the overall procedure of the proposed pruning rate search.

Algorithm 4.2 Proposed automatic pruning rate search.

**Input:** Pre-trained model  
**Input:** Pruning rate candidates:  $Rate_{cand}$   
**Input:** Loss margin control parameter  $L_m$   
**Input:** Accuracy control parameter  $Acc_{ctrl}$   
**Output:** Derived pruning rate

- 1: Calculate  $Acc_b$  using the pre-trained model.
- 2: Calculate loss function and gradient by pre-trained model.
- 3: Target model  $\leftarrow$  pre-trained model
- 4: Initialize upper bound of threshold  $U_b$  to  $U_{b,init}$
- 5: **while** pruning rate for all layers is 0%. **do**
- 6:     Calculate threshold  $T_{h,k}$  using Eq. 4.6
- 7:     **if** L2-norm of  $T_{h,k,all} > U_b$  **then**
- 8:         Scale all  $T_{h,k}$  so that L2-norm  $T_{h,k,all}$  equals  $U_b$ .
- 9:     **end if**
- 10:    Derive the pruning rate per layer by Algorithm 5.1.
- 11:    Prune Target model with derived pruning rate.
- 12:    Fine-tune the pruned Target model.
- 13:    Calculate  $Acc_f$  using the fine-tuned model.
- 14:    **if**  $Acc_f \geq Acc_b - Acc_{ctrl}$  **then**
- 15:        Target model  $\leftarrow$  fine-tuned model
- 16:        Calculate loss function and gradient by Target model.
- 17:         $U_b = \min(2 \times U_b, U_{b,init})$
- 18:    **else**
- 19:         $U_b = 0.5 \times U_b$
- 20:    **end if**
- 21: **end while**
- 22: **return** Derived pruning rates, Target model

#### 4.1.3 Structured pruning around shortcut connection with zero-padding layers

Recent state-of-the-art DNN models such as ResNet often use a shortcut connection with an addition operator shown in Fig. 2.3. When multiple feature maps are input to the addition operator from multiple layers, the elements of the same index in the feature maps are added by the addition operator. Thus, the number of input feature maps must be the same. Here, when applying the pruning, the layers of the pruning target model are pruned independently, which means the number of input feature maps to the addition operator may not align, as shown in Fig. 2.3(c). As a result, the addition operator cannot work. Therefore, to guarantee the calculation in the addition operator, the number of multiple input feature maps to the addition operator must be equal. In the original ResNet model [29], a zero-padding layer is inserted only into the shortcut connection to align the number of input feature maps, and no zero-padding layer is inserted into the residual block path because the number of feature maps from the shortcut connection is always less than the number of feature maps from the residual block. However, when all layers are candidates for pruning, the output channels of the residual block may be pruned, which leads to cases where there are fewer feature maps from the residual block path than from the shortcut connection, i.e., the numbers of each do not align.

To improve the compression ratio around the addition operator, we insert zero-padding layers into all input paths to the addition operator to align the number of feature maps from multiple paths, as presented in Fig. 4.2. The inserted zero-padding layers append the zero matrices to the output of all layers other than the layer that has the largest number of channels, thus ensuring that the number of channels of all input layers to the addition operator is aligned. For example, in Fig. 4.3, one channel is pruned in layer A, and three channels are pruned in layer B. Thus, the number of remaining channels is three in layer A and one in layer B. Hence, to align the number of feature maps from layer A and layer B, the inserted zero-padding layer for layer B appends two zero matrices. Now that the number of feature maps from multiple layers is aligned, the addition operator can execute the operation using the input feature maps.

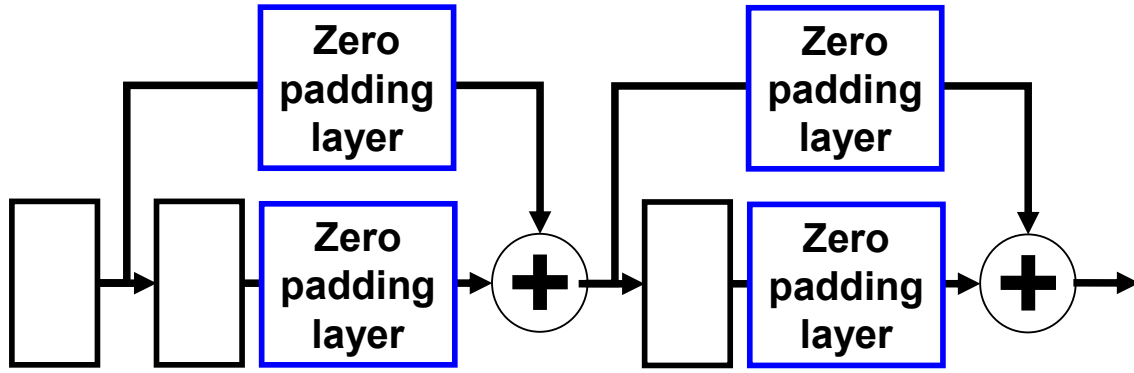


Fig. 4.2 Proposed pruning method around shortcut connection. Zero-padding layers are inserted into all input paths to the addition operator.

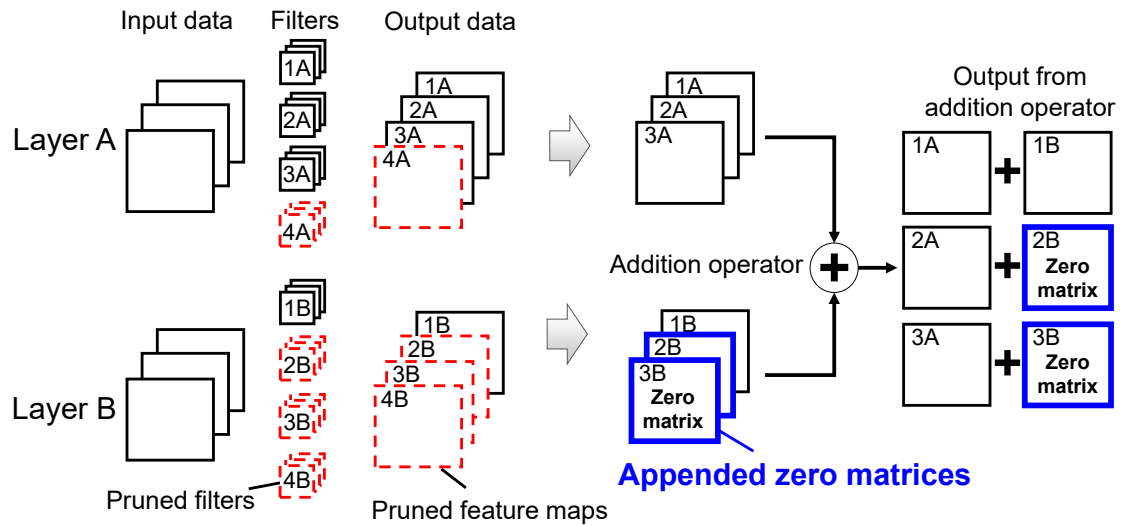


Fig. 4.3 Details of the proposed pruning method around shortcut connection. Inserted zero-padding layers append zero matrices to the output of the layers with fewer channels. The zero matrices are appended so that the number of channels of all input feature maps to the addition operator is aligned.

#### 4.1.4 Experiments

We evaluate the proposed method on CIFAR-10 [29] and ImageNet (ILSVRC-2012) [30] training and validation datasets with various ResNets to determine its effectiveness with different difficulty levels and depth models. For the experiment with CIFAR-10, we use ResNet-32, ResNet-56, and ResNet-110. The model pre-trained with CIFAR-10 published in [36] is utilized here. The mini-batch size is 128. The maximum fine-tuning duration is 250 epochs. The initial learning rate is 0.1 and is multiplied by 0.1 at epochs 100, 150, and 200. For the experiment with ImageNet, we use ResNet-34 and ResNet-50. The published model pre-trained with ImageNet provided by PyTorch [37] is utilized here. The mini-batch size is 256. The maximum fine-tuning duration is 90 epochs. The initial learning rate is 0.01 and is multiplied by 0.1 at epochs 30 and 60. The common conditions of both experiments with CIFAR-10 and ImageNet are as follows. We use cross-entropy loss functions to train the model both before and after pruning. The evaluated models are fine-tuned by momentum stochastic gradient descent (SGD) with L2 weight decay. The momentum coefficient is 0.9. The weight decay coefficient is  $10^{-4}$ . The control parameter of the loss function margin  $L_m$  is set to 0.1. The pruning rate candidates  $Rate_{cand}$  are set to  $\{50\%, 40\%, 30\%, 20\%, 10\%, 0\%\}$ . The scaling factors for increasing and reducing  $U_b$  are set to 2 and 0.5, respectively, as referenced in [69]. We use NVIDIA V-100 GPU for all experiments.

##### 4.1.4.1 Performance comparisons

We compare the accuracy of the pruned model and the compression ratio of the parameters and FLOPs among the proposed method and earlier state-of-the-art works. Table 4.1 shows the performance comparisons on CIFAR-10 with ResNet-32, ResNet-56, and ResNet-110. Table 4.2 shows the performance comparisons on ImageNet with ResNet-34 and ResNet-50. In the proposed method, we set  $Acc_{ctrl}$  to achieve a similar pruned model accuracy to the earlier works. Our findings show that the proposed method achieves the best compression ratio of the parameters and FLOPs with a similar pruned model accuracy to earlier works in all evaluated cases.

Table 4.1 Comparison of experimental results on CIFAR-10.

| Model      | Method                             | Top-1 accuracy (%) |              | Compression ratio(%) |             |
|------------|------------------------------------|--------------------|--------------|----------------------|-------------|
|            |                                    | Baseline           | Pruned model | Parameters           | FLOPs       |
| ResNet-32  | SFP [40] 2015                      | 92.63              | 90.63        | N/A                  | 28.8        |
|            | FPGM [41] 2019                     | 92.63              | 91.93        | N/A                  | 53.2        |
|            | COP [34] 2019                      | 92.64              | 91.97        | 57.5                 | 53.9        |
|            | SCOP [38] 2020                     | 92.66              | 92.13        | 56.2                 | 55.8        |
|            | BATUDE [48] 2022                   | 92.49              | 92.18        | 64.2                 | N/A         |
|            | KPGP [70] 2022                     | 92.65              | 92.68        | 43.4                 | 43.1        |
|            | Chen et al. [71] 2024              | 92.63              | 92.64        | 40.3                 | 46.8        |
|            | Zheng et al. [72] 2024             | 92.86              | 92.39        | N/A                  | 55.6        |
|            | Liu et al. [73] 2024               | 92.65              | 92.31        | 56.5                 | 57.3        |
|            | <b>Ours</b> ( $Acc_{cnt}=0.0$ )    | 92.63              | <b>92.70</b> | <b>77.5</b>          | <b>63.2</b> |
| ResNet-56  | PaG [26] 2016                      | 93.04              | 93.12        | 47.9                 | N/A         |
|            | HRank [43] 2020                    | 93.26              | 93.17        | 42.4                 | 50.0        |
|            | GAL [44] 2019                      | 93.26              | 93.38        | 11.8                 | 37.6        |
|            | SCOP [38] 2020                     | 93.70              | 93.64        | 56.3                 | 56.0        |
|            | CHIP [47] 2021                     | 93.26              | 92.05        | 71.8                 | 72.3        |
|            | ISP [49] 2022                      | 93.56              | 93.74        | N/A                  | 54.0        |
|            | EACP [76] 2023                     | 93.20              | 93.11        | 63.5                 | 68.0        |
|            | Chen et al. [71] 2024              | 93.39              | 93.60        | 40.3                 | 45.5        |
|            | Zheng et al. [72] 2024             | 93.64              | 93.04        | N/A                  | 55.8        |
|            | SFP-D [74] 2024                    | 93.26              | 93.29        | 63.4                 | 63.3        |
|            | Liu et al. [73] 2024               | 93.20              | 93.59        | 69.4                 | 65.5        |
|            | <b>Ours</b> ( $Acc_{cnt} = -0.3$ ) | 93.39              | <b>93.72</b> | <b>77.8</b>          | <b>72.2</b> |
| ResNet-110 | HRank [43] 2020                    | 93.50              | 92.65        | 68.7                 | 68.6        |
|            | GAL [44] 2019                      | 93.50              | 92.74        | 44.8                 | 48.5        |
|            | PF [25] 2016                       | 93.53              | 93.30        | 32.4                 | 38.6        |
|            | PaG [26] 2019                      | 93.53              | 93.56        | 53.1                 | N/A         |
|            | CHIP [47] 2021                     | 93.50              | 93.63        | 68.3                 | 71.6        |
|            | DECORE [75] 2022                   | 93.50              | 93.50        | 64.8                 | 61.8        |
|            | EACP [76] 2023                     | 93.30              | 93.35        | 65.9                 | 68.3        |
|            | Liu et al. [73] 2024               | 93.54              | 93.21        | 80.4                 | 79.0        |
|            | <b>Ours</b> ( $Acc_{cnt} = 0.1$ )  | 93.68              | <b>93.62</b> | <b>91.5</b>          | <b>86.7</b> |

Table 4.2 Comparison of experimental results on ImageNet.

| Model     | Method                            | Top-1 accuracy (%) |              | Compression ratio(%) |             |
|-----------|-----------------------------------|--------------------|--------------|----------------------|-------------|
|           |                                   | Baseline           | Pruned model | Parameters           | FLOPs       |
| ResNet-34 | NISP [42] 2018                    | 73.23              | 72.31        | 43.7                 | 43.8        |
|           | PaG [26] 2019                     | 73.30              | 72.35        | 45.3                 | N/A         |
|           | FPGM [41] 2018                    | 73.92              | 72.63        | N/A                  | 41.1        |
|           | SCOP [38] 2020                    | 73.31              | 72.93        | 39.7                 | 39.1        |
|           | Zhang et al. [70] 2022            | 73.31              | 72.74        | 36.3                 | 45.2        |
|           | <b>Ours</b> ( $Acc_{cnt} = 0.3$ ) | 73.27              | <b>73.01</b> | <b>50.6</b>          | <b>43.4</b> |
| ResNet-50 | ThiNet [45] 2017                  | 72.88              | 72.04        | 33.7                 | 36.8        |
|           | Taylor [46] 2019                  | 76.18              | 74.50        | 44.5                 | 45.0        |
|           | DCP [39] 2018                     | 76.01              | 74.95        | 51.5                 | 55.6        |
|           | HRank [43] 2020                   | 76.15              | 74.98        | 36.6                 | 43.7        |
|           | Adapt-DCP [50] 2022               | 76.01              | 75.15        | 55.0                 | 52.4        |
|           | Taylor [46] 2019                  | 76.18              | 75.48        | 30.1                 | 35.0        |
|           | FPGM [41] 2019                    | 76.15              | 75.59        | N/A                  | 42.2        |
|           | PaG [26] 2019                     | 76.15              | 75.64        | 45.7                 | N/A         |
|           | SCOP [38] 2020                    | 76.15              | 75.95        | 42.8                 | 45.3        |
|           | CHIP [47] 2021                    | 76.15              | 76.15        | 44.2                 | 48.7        |
|           | ISP [49] 2022                     | 76.15              | 75.97        | N/A                  | 56.6        |
|           | CRP [77] 2023                     | 76.15              | 75.15        | N/A                  | 64.3        |
|           | MOS [78] 2024                     | 76.13              | 75.04        | N/A                  | 60.4        |
|           | EACP [76] 2023                    | 75.82              | 73.95        | 54.7                 | 53.8        |
|           | Zheng et al. [72] 2024            | 76.12              | 75.12        | N/A                  | 53.5        |
|           | SFP-D [74] 2024                   | 76.15              | 75.24        | 50.3                 | 52.7        |
|           | Lu et al. [79] 2023               | 76.15              | 75.77        | N/A                  | 44.9        |
|           | Liu et al. [73] 2024              | 76.15              | 75.68        | 52.5                 | 55.6        |
|           | Chen et al. [71] 2024             | 76.13              | 75.29        | 44.2                 | 50.4        |
|           | <b>Ours</b> ( $Acc_{cnt} = 0.0$ ) | 76.15              | <b>76.17</b> | <b>60.1</b>          | <b>59.8</b> |

#### 4.1.4.2 Robustness evaluation for hyperparameter $L_m$

In our method, the pruning rate is derived by comparing the pruning error and the threshold. The threshold is derived by using the hyperparameter  $L_m$  that controls the loss function margin, as shown in Eq. 5.6. To investigate the robustness of the pruned model performance against  $L_m$ , we evaluate the pruned model accuracy and the compression ratio of the pruned model while changing  $L_m$  on CIFAR-10 with ResNet-32.

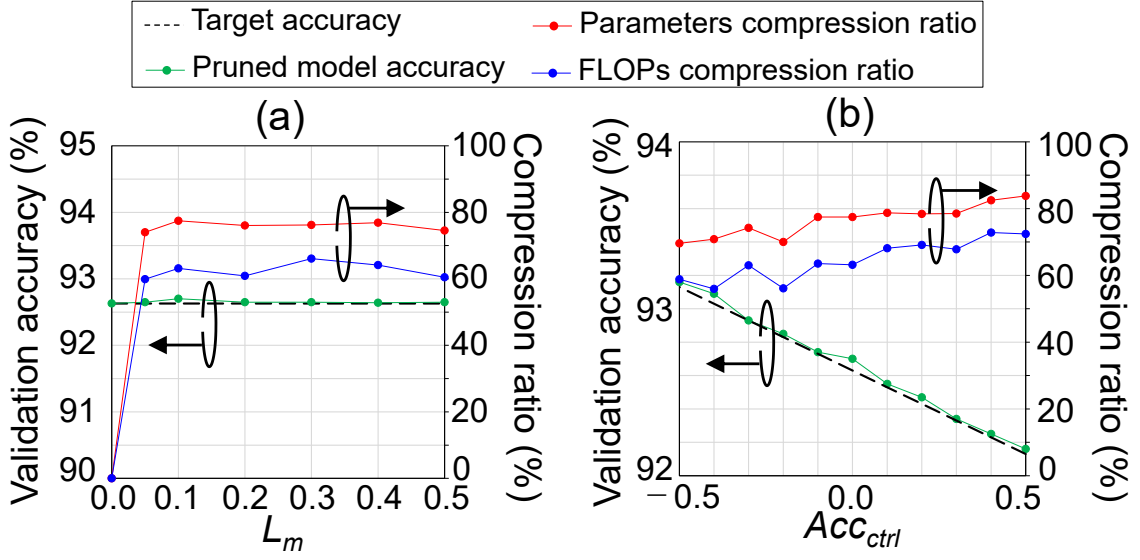


Fig. 4.4 Pruned model performance against the hyperparameters on CIFAR-10 with ResNet-32. (a) Performance against  $L_m$  ( $Acc_{ctrl} = 0\%$ ). The accuracy and compression ratios of the pruned model are robust for the  $L_m$  choice when  $L_m$  is greater than zero. (b) Performance against  $Acc_{ctrl}$  ( $L_m = 0.1$ ). The pruned model accuracy can be effectively controlled by  $Acc_{ctrl}$ .

The result is presented in Fig. 4.4(a). As we can see, when  $L_m$  is greater than zero, since the accuracy and the compression ratio of the pruned model are not changed significantly, the choice of  $L_m$  does not significantly affect the pruned model performance.

In contrast, when  $L_m$  is zero, the compression ratio of the pruned model becomes 0%. In our method, the threshold at  $k$  times during the pruning rate search  $T_{h,k}$  is derived by Eq. 4.6. So when  $L_m$  is zero, the threshold  $T_{h,k}(L_m = 0)$  is represented as

$$T_{h,k}(L_m = 0) = \frac{L_b - L(W_k)}{n} \cdot \sum_{i=1}^n \frac{1}{\partial L(W_k) / \partial w_{i,k}}. \quad (4.8)$$

When  $k$  is zero,  $T_{h,0}(L_m = 0)$  represents the initial value of the threshold used for the first pruning rate search.

$$T_{h,0}(L_m = 0) = \frac{L_b - L(W_0)}{n} \cdot \sum_{i=1}^n \frac{1}{\partial L(W_0) / \partial w_{i,0}}. \quad (4.9)$$

Where  $W_0$  is the weight of the model before pruning. Hence  $L(W_0)$  represents a loss

function of the model before pruning. Here,  $L_b$  is also defined as a loss function of the model before pruning. Therefore, as  $L(W_0)$  and  $L_b$  are the same value,  $T_{h,0}(L_m = 0)$  becomes zero. That is, the threshold used for the first pruning rate search becomes zero. In our method, the largest pruning rate among the candidates exhibiting a pruning error less than the threshold is used for pruning. In addition, our method terminates the pruning rate search when 0% is chosen as the pruning rate of all layers as shown in Fig. 4.1(b). Therefore, when  $T_{h,0}$  is zero, since the pruning rate for all layers is set to 0% at the first search, then the pruning rate search is terminated at the first search. As a result, since the model is not pruned when  $L_m$  is set to zero, the compression ratio of the pruned model becomes 0%.

#### 4.1.4.3 Accuracy control of pruned model

The proposed method controls the accuracy of the pruned model using the accuracy control parameter  $Acc_{ctrl}$ . We expect the accuracy of the pruned model to be the target accuracy  $Acc_b - Acc_{ctrl}$ . To see whether  $Acc_{ctrl}$  can effectively control the pruned model accuracy, we investigate the relationship between the pruned model performance and  $Acc_{ctrl}$  on CIFAR-10 with ResNet-32. The result is presented in Fig. 4.4(b), where we can see that the accuracy of the pruned model is almost the same value as the target accuracy. This demonstrates that the pruned model's accuracy is effectively controlled by  $Acc_{ctrl}$ . In Fig. 4.4(b), the compression ratios increase as the accuracy is reduced, which is consistent with the findings presented in earlier reports [33, 35].

#### 4.1.4.4 Temporary pruning rate during pruning rate search

We next investigate whether our method can automatically search the pruning rate in accordance with the pruned model accuracy. Fig. 4.5 shows the temporary pruning rate of the entire model, the accepted pruning rate of the entire model, and the accuracy of the fine-tuned model pruned by the temporary pruning rate during pruning rate search on CIFAR-10 with ResNet-32. The temporary pruning rate means the pruning rate chosen by our method at step (b) in Fig. 4.1. The accepted pruning rate means the pruning rate that is used for the fine-tuned model which satisfies the target accuracy. The pink-colored area in Fig. 4.5 represents the period during which the accuracy of the fine-tuned model pruned by the temporary pruning rate is less than the target accuracy.

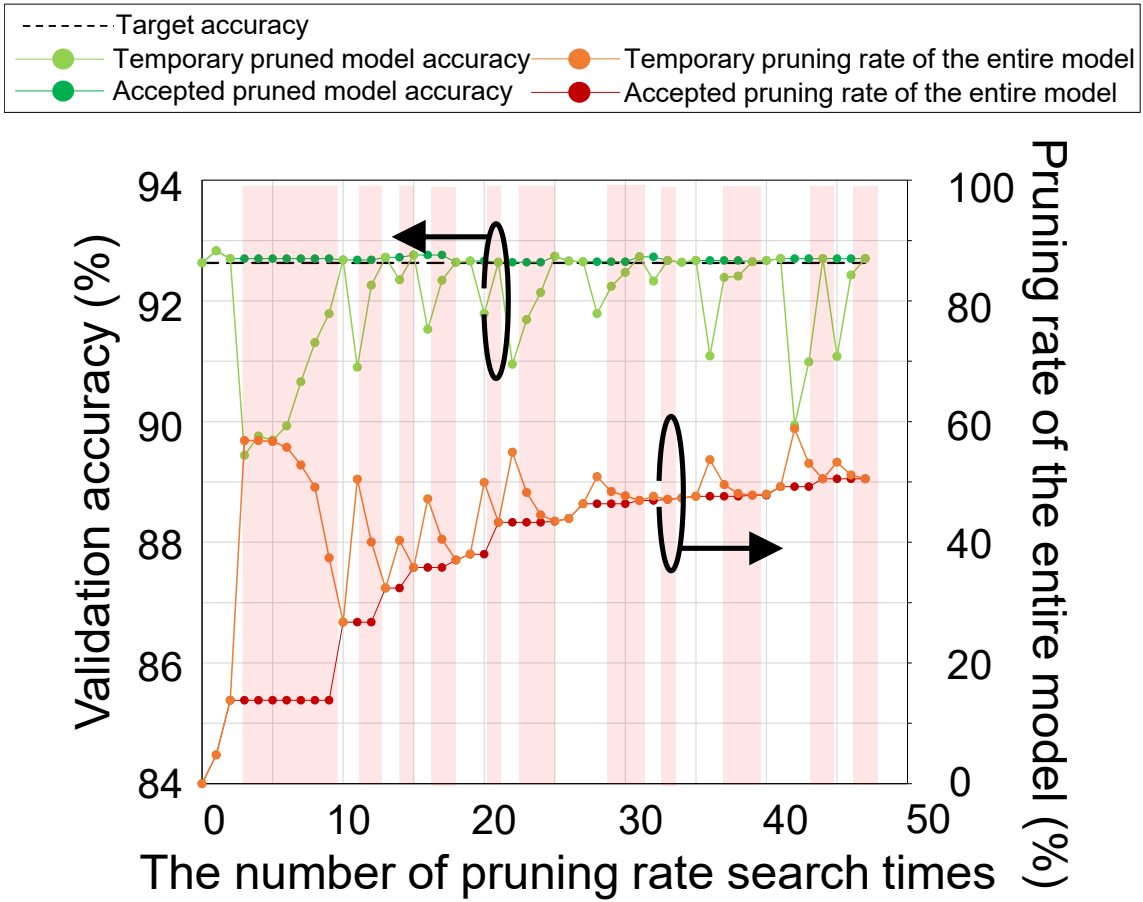


Fig. 4.5 Pruning rate of the entire model and the pruned model accuracy during pruning rate search on CIFAR-10 with ResNet-32. The temporary pruning rate is the pruning rate chosen by our method at the step in Fig. 4.1(b), and the temporary pruned model accuracy is the accuracy pruned by the temporary pruning rate. The accepted pruning rate is the pruning rate that is used for the fine-tuned model which satisfies the target accuracy, and the accepted pruned model accuracy is the accuracy pruned by the accepted pruning rate.

In Fig. 4.5, when our method chooses a high pruning rate, the accuracy of the fine-tuned model does not meet the target accuracy (pink-colored period). Then our method discards the fine-tuned model pruned by the chosen pruning rates, and the accepted pruning rate is not updated. On the other hand, when the accuracy of the fine-tuned model meets the target accuracy (no colored period), our method accepts the fine-tuned model then the accepted pruning rate is updated to the chosen pruning rate. These results show that our method automatically searches for the pruning rates that meet the target accuracy.

#### 4.1.4.5 Convergence of pruning rate search

As the proposed method iteratively searches for the pruning rate, the convergence of the pruning rate search is also important. Fig. 4.6 presents the pruning rate of the entire model during the pruning rate search. The pruning rate increases rapidly at beginning of the search and then gradually converges.

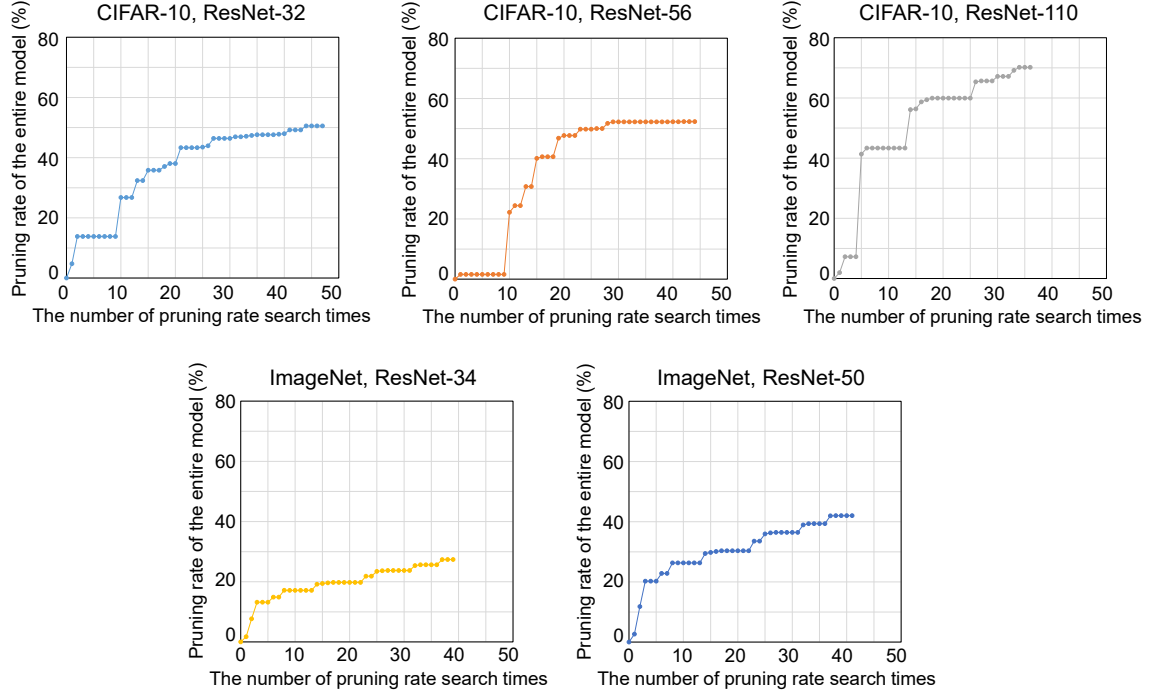


Fig. 4.6 Pruning rate of the entire model during pruning rate search.

#### 4.1.4.6 Effectiveness of inserting zero-padding layers

Table 4.3 shows the experimental results of inserting the zero-padding layers. Our method considerably improves the compression ratio in all cases without any marked accuracy degradation. Fig. 4.7 shows the number of output channels of the pruned models for each layer. As we can see, the number of output channels for each layer is significantly different for the cases with and without zero-padding layers. Specifically, when the zero-padding layers are not applied, the number of output channels varies greatly depending on the layer. In the case of pruning the residual block, if the zero-padding layers are not applied, then the channel size of the multiple feature maps input to the addition operator cannot be aligned. Therefore, since the residual block cannot be pruned, the number of output channels varies greatly depending on the layer. In contrast, when the zero-padding

layers are applied, the output channels for each layer are pruned regardless of the layer. Because the channel size of the multiple feature maps input to the addition operator can be aligned, the residual block can be pruned. Therefore, the compression ratio of the pruned model using our proposed zero-padding layers is improved.

Table 4.3 Pruned model accuracy, compression ratio, and FLOPs with and without zero-padding layers.

| Dataset  | Model      | Applying zero-padding layer | Top-1 accuracy (%) |              | Compression ratio(%) |             |
|----------|------------|-----------------------------|--------------------|--------------|----------------------|-------------|
|          |            |                             | Baseline           | Pruned model | Parameters           | FLOPs       |
| CIFAR-10 | ResNet-32  | No                          | 92.63              | 92.69        | 64.8                 | 62.2        |
|          |            | Yes                         | 92.63              | 92.70        | <b>77.5</b>          | <b>63.2</b> |
|          | ResNet-56  | No                          | 93.39              | 93.71        | 66.3                 | 65.4        |
|          |            | Yes                         | 93.39              | 93.72        | <b>77.8</b>          | <b>72.2</b> |
|          | ResNet-110 | No                          | 93.68              | 93.60        | 84.4                 | 79.5        |
|          |            | Yes                         | 93.68              | 93.62        | <b>91.5</b>          | <b>86.7</b> |
| ImageNet | ResNet-34  | No                          | 73.27              | 72.98        | 44.0                 | 29.2        |
|          |            | Yes                         | 73.27              | 73.01        | <b>50.6</b>          | <b>43.4</b> |
|          | ResNet-50  | No                          | 76.15              | 76.15        | 54.2                 | 59.5        |
|          |            | Yes                         | 76.15              | 76.17        | <b>60.1</b>          | <b>59.8</b> |

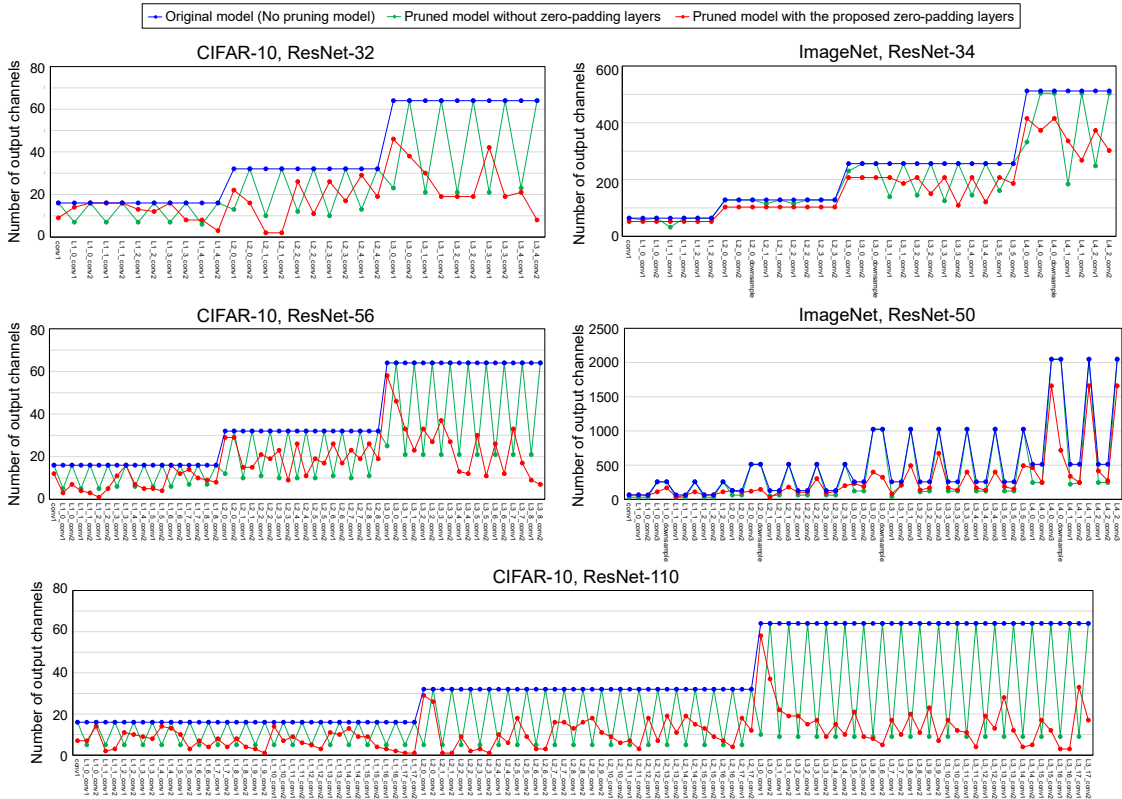


Fig. 4.7 Number of output channels before and after pruning.

## 4.2 Summary

In this chapter, to reduce the workload of inefficient manual pruning rate assignment, we proposed an automatic pruning rate search method for structured pruning. In the proposed method, by selecting a pruning rate in which the pruning error is smaller than a pre-defined threshold, the accuracy degradation of the pruned model is suppressed. In addition, to improve the compression ratio of the layers around the addition operator, zero-padding layers are inserted into all input paths to the addition operator. The experimental results demonstrate that the pruned model accuracy can be effectively controlled by the proposed method. In addition, the pruning rate derived by the proposed method can be converged. We experimentally showed the superiority of our method using various ResNets on CIFAR-10 and ImageNet.

## Chapter 5 Cost-effective DNN training for materials discovery

This chapter describes cost-effective DNN training methods for materials discovery to accelerate fine-tuning and to reduce labeled training data. For this purpose, we studied the following contents:

- 1) We propose a mask-less self-supervised learning method using atom replacement, to generate the unlabeled catalyst data; this is helpful in improving the catalyst energy prediction accuracy of GNNs, as it is usually difficult to obtain a large amount of labeled data.
- 2) we demonstrate that our proposed method can accelerate the fine-tuning after pre-training, in addition to the improvement of energy prediction accuracy.
- 3) Furthermore, we propose an active learning method using expanded features including the features of material structure and the predicted energy. The expanded features are used for calculating the Euclidean distances to sample the training data for GNNs.
- 4) We demonstrate that compared to existing active learning methods, our proposed method improves energy prediction accuracy for various GNNs and evaluation datasets using less training data.

### 5.1 Self-supervised learning with atom replacement for catalyst energy prediction

Recently, materials discovery utilizing the material informatics (MI) technique is considered to be an efficient approach to discovering new materials. Energy or environment-related materials are deeply related to chemical reactions, and the reactivity of materials is determined by the electronic energy (or internal energy) of materials and reactant systems. The electronic energy is often calculated by first-principles calculation which is based on the Schrödinger equation. Among them, a method called density functional theory (DFT) is widely used nowadays. Since the computational cost of the DFT calculation is relatively high, many graph neural networks (GNNs) for energy

prediction have been proposed to obtain the energy values quickly. However, it is difficult to obtain a large amount of labeled data for GNNs training, because a large amount of the DFT data is necessary for training. Although self-supervised learning (SSL) methods using unlabeled data, i.e., no DFT energy information is needed, have been proposed to improve the GNNs’ energy prediction accuracy, existing SSL methods use a non-existent atom of “mask”; this may lead to the inefficient GNNs training. In this work, we propose a mask-less SSL method by replacing some atoms in the target material systems, aiming to improve the energy prediction accuracy of GNNs. Our target systems are catalysts, and here the atoms in unlabeled catalysts are replaced with other actual existing atoms. Then, GNNs predict whether the atoms have been replaced or not for all atoms composing the material. After that, GNNs pre-trained by the proposed SSL are fine-tuned by downstream tasks. We demonstrate the superiority of our proposed method on the Open Catalyst dataset and Poisoned Catalyst dataset using the energy prediction GNNs, CGCNN and PaiNN. The GNNs pre-trained by the proposed SSL achieve the best mean absolute error of predicted energy rather than that of the existing mask-based SSL in all experimental conditions.

### 5.1.1 Related works

These years, many GNNs have been proposed to predict the energy of materials such as catalysts that can be described by molecular structures [89, 90, 100]. Molecular structures can be represented by graph structures with atoms as nodes and bonds as edges. Therefore, the proposed GNNs can treat molecular structures as input data by mapping molecular structures to graph structures. These GNNs for materials energy prediction proposed in [89, 90, 100] are trained by supervised learning, and the energies contained in the labeled data used by supervised learning are calculated by DFT. However, because the energy calculation time by DFT is generally too long [88], it is difficult to obtain a large amount of labeled data for supervised learning.

Recently, various SSL methods using unlabeled data have been proposed for GNN-based material energy prediction to overcome the scarcity of labeled data [95, 96, 97, 99]. The previous studies [95, 96, 97] have proposed a SSL method that masks the atoms and the bonds composing molecules. Then GNN predicts the masked atoms and bonds of the molecule. Another work [99] proposes contrastive learning using data augmented

molecular structure data by using a mask. Since all of these existing SSL methods for energy prediction GNNs [95, 96, 97, 99] are mask-based methods, GNNs learn the representation of the “mask”. However, the atoms and the bonds represented by the mask are artificially generated for SSL. In other words, the data of the “mask” are not contained in the real data of fine-tuning. Therefore, a mismatch occurs between the pre-training by the SSL task and the fine-tuning by the downstream task. In the field of natural language processing (NLP), prior work [101] has shown that when neural network models are pre-trained by mask-based SSL, a large amount of training data and training epochs are required to achieve high-accuracy neural network predictions. Another work in NLP [102] points out the problems of SSL with masks. For example, when multiple tokens in the input sentence are masked, there is an inability to utilize the information of masked tokens to predict the other masked tokens.

These previous studies of SSL in the field of NLP indicate that existing mask-based SSL for material energy prediction may not be able to effectively utilize information from other masked atoms to predict masked atoms or may not learn the relationship between the masked atoms and the other actual existing atoms very well. In other words, if we do not use the mask in SSL for pre-training, the energy prediction accuracy of GNNs is expected to be improved.

### 5.1.2 Self-supervised learning with atom replacement for energy prediction GNNs

Figure 5.1 shows the proposed SSL method. First, unlabeled catalyst structure data without energy information are prepared. The unlabeled catalyst structure data contains information on the element composition of the catalyst. Next, the replacement target atoms are specified from the atoms composing the catalyst according to the corresponding atom replacement ratio. The atom replacement ratio is a hyperparameter. The element of the specified atoms is replaced with an element randomly selected from the elements in the given atom library. The atom library consists only of elements that actually compose the material. That is, the element library only contains the actual existing elements, and does not contain the un-existing elements like “mask”. The unlabeled atom-replaced

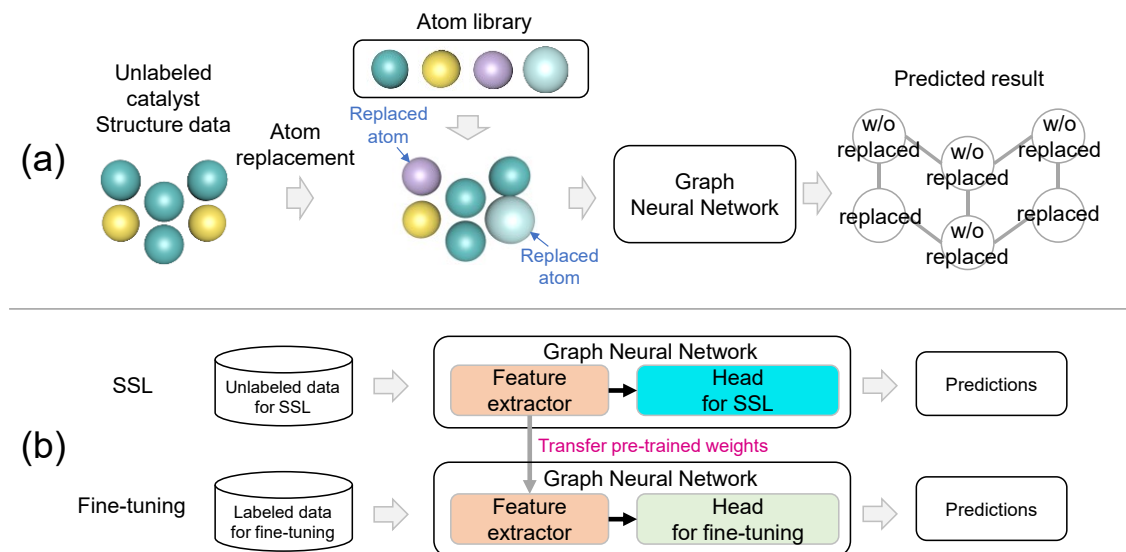


Fig. 5.1 Overview of training framework with proposed SSL. (a) Our proposed method replace the atoms of input unlabeled data to realize mask-less SSL. (b) In our method, the pre-trained weights of the feature extractor are transferred to the GNN for fine-tuning. Then, the GNN is fine-tuned by labeled data for the downstream task.

catalyst structure data is input to the GNN, then the GNN predicts whether the atoms have been replaced or not for all atoms in the structure data. The loss is calculated by comparing the predicted result and the answer generated from the unlabeled data, and the calculated loss is backpropagated to GNN. After pre-training by the SSL task, the head of the pre-trained GNN is changed to match the downstream task. Then, the GNN is fine-tuned by the downstream task.

### 5.1.3 Experiments

We evaluate the proposed SSL method using energy prediction GNNs of CGCNN and PaiNN by three datasets for catalyst energy prediction: Open Catalyst, Poisoned Catalyst, and Expanded Poisoned Catalyst.

#### 5.1.3.1 Dataset

##### 1) Open Catalyst

Table 5.1 shows the overview of the three evaluation datasets. Open Catalyst [11] is the largest published catalyst dataset. Labeled data in Open Catalyst dataset consists of the structure of various solid catalysts with adsorbate molecules and the energy of the

Table 5.1 Overview of the evaluation dataset.

| Dataset                                                | Open Catalyst 2022 [93]                                               | Poisoned Catalyst [82]                                                     | Expanded Poisoned Catalyst [82]               |
|--------------------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------------|-----------------------------------------------|
| <b>Number of datapoints</b><br>(Training / Validation) | 45,890 / 2,624                                                        | 3,811 / 900                                                                | 13,453 / 1,000                                |
| <b>Chemical formula of Catalyst</b>                    |                                                                       |                                                                            |                                               |
| Unary (AxOy)                                           | 4,695                                                                 | 4,711                                                                      | 14,453                                        |
| Binary (AxByOz)                                        | 43,639                                                                | 0                                                                          | 0                                             |
| <b>Sampled Elements</b>                                |                                                                       |                                                                            |                                               |
| Alkali                                                 | 9,348                                                                 | 0                                                                          | 0                                             |
| Alkaline                                               | 11,536                                                                | 0                                                                          | 0                                             |
| p-block metals                                         | 12,202                                                                | 0                                                                          | 0                                             |
| Metallids                                              | 6,750                                                                 | 0                                                                          | 0                                             |
| Transition metals                                      | 47,603                                                                | 4,711                                                                      | 14,453                                        |
| <b>Including adsorbate</b>                             | O, H, N, C, OH, OOH, H <sub>2</sub> O, CO, O <sub>2</sub>             | H, OH, N <sub>2</sub> , NH <sub>2</sub> , NNH                              | H, OH, N <sub>2</sub> , NH <sub>2</sub> , NNH |
| <b>Including poisoning atoms</b>                       | None                                                                  | H                                                                          | H, O, OH                                      |
| <b>Task</b>                                            | Predict the relaxed state total energy from the non-relaxed structure | Predict the relaxed state adsorption energy from the non-relaxed structure |                                               |

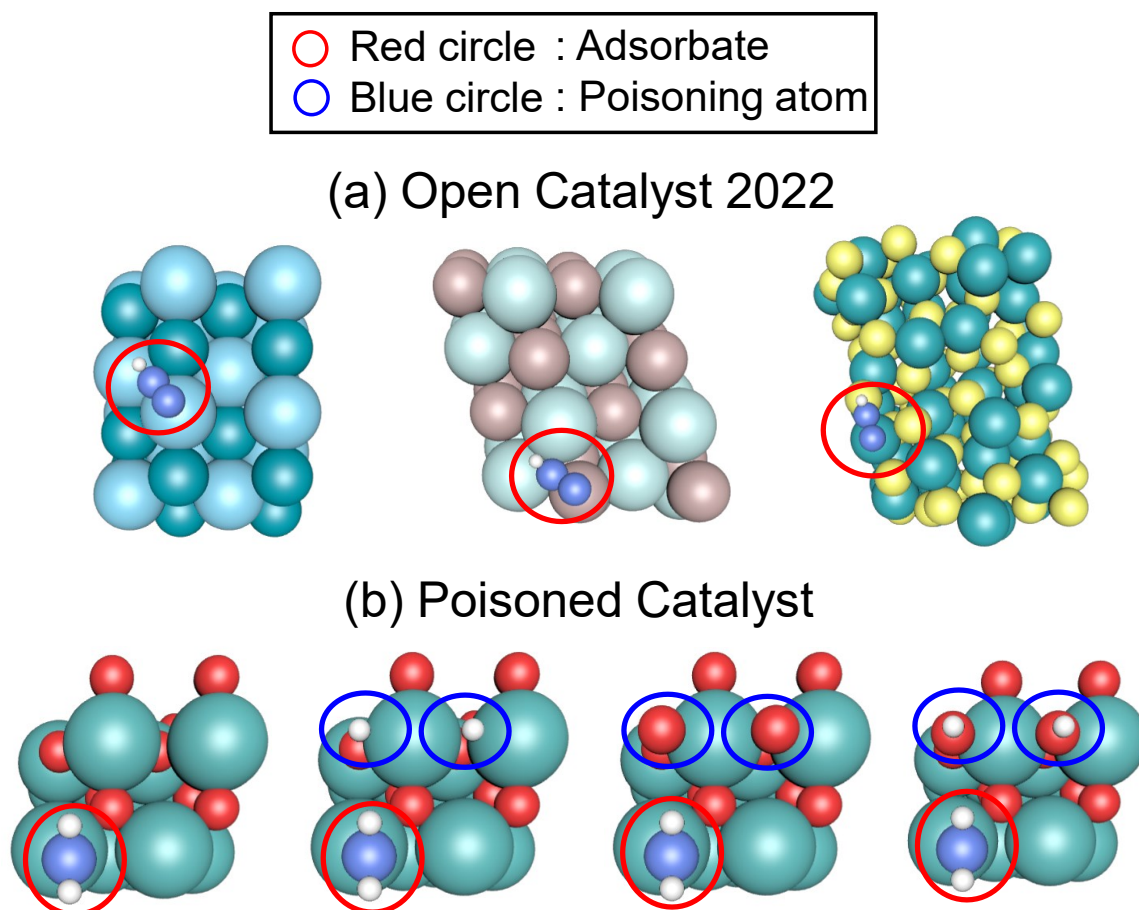


Fig. 5.2 3D view of the catalyst dataset for evaluation. OC22 dataset has various structures of catalysts. On the other hand, the poisoned catalyst dataset contains poisoning atoms causing catalyst poisoning.

catalyst-adsorbate structure. Open Catalyst published two datasets, Open Catalyst 2020 (OC20) [92] and Open Catalyst 2022 (OC22) [93]. OC20 contains the adsorption energy of the catalyst-adsorbate system. OC22 contains the total energy of the catalyst-adsorbate system. To investigate the effectiveness of the SSL in the transfer learning framework, we use OC20 for pre-training and use O22 for fine-tuning. In the Open Catalyst benchmark, they provide various tasks. In this experiment, we use the task called “initial structure to relaxed energy” (IS2RE). IS2RE is used to predict the energy of the relaxed structure from the non-relaxed structure. For the evaluation using the Open Catalyst dataset, we use OC22 In-Domain validation data for the IS2RE.

## 2) *Poisoned Catalyst & Expanded Poisoned Catalyst*

In the search for a catalyst, it is important to understand the effect called “catalyst poisoning” where various products on the catalyst surface prevent the desired reactions on the catalyst [82, 83, 84]. In the Open Catalyst datasets, because there are only adsorbates on the surface of the catalyst, as shown in Fig. 5.2(a), the “clean” Open Catalyst datasets are not suitable for studying the effect of catalyst poisoning. In order to obtain the energies which take into account the effect of catalyst poisoning, we have created two types of catalyst structure datasets including poisoning atoms on the catalyst surface by referring to [82] as shown in Fig. 5.2(b). The dataset named “Poisoned Catalyst” uses the hydrogen atom for the poisoning atoms. Another dataset, “Expanded Poisoned Catalyst”, uses not only the hydrogen atom but also the oxygen atom and the hydroxyl as the poisoning atoms. For all Poisoned Catalyst dataset and Expanded Poisoned Catalyst dataset, the labeled data were generated by calculating the adsorption energy between adsorbate and catalyst with poisoning atom using DFT. For the downstream task using the Poisoned Catalyst dataset and the Expanded Poisoned Catalyst dataset, GNNs predict the adsorption energy of the relaxed state from the non-relaxed structure. For the Poisoned Catalyst dataset, we generated 4,711 labeled data. From the whole 4,711 data, 900 data were randomly selected as the validation dataset. The remaining 3,811 data were used for SSL in pre-training and fine-tuning. For the Expanded Poisoned Catalyst dataset, we generated 14,453 labeled data. Then, we randomly selected 1,000 data from the 14,453 to be used as validation data. The remaining 13,453 samples were all used for SSL and fine-tuning.

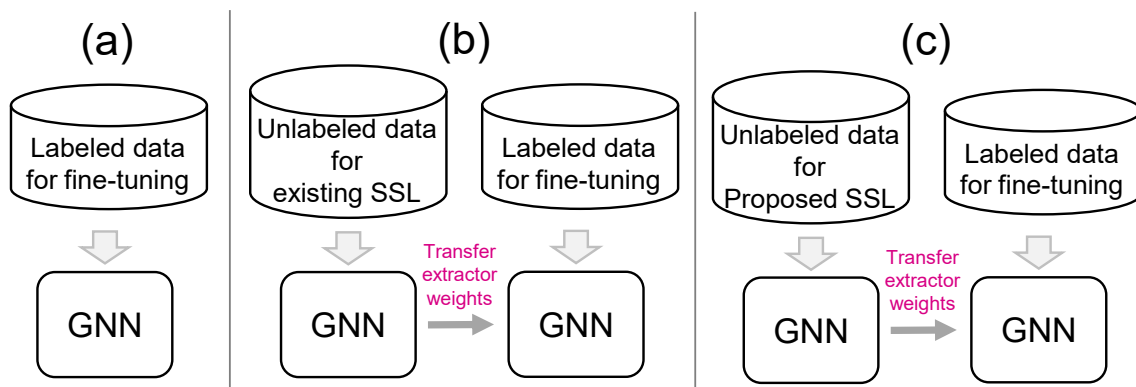


Fig. 5.3 GNN training flow in the experiments of this work. (a) Only fine-tune GNN (not using SSL). (b) Pre-train GNN by the existing mask-based SSL, then the pre-trained GNN is fine-tuned. (c) Pre-train GNN by the proposed mask-less SSL, then the pre-trained GNN is fine-tuned.

Table 5.2 Hyperparameters in training. Other hyperparameters are the same as the published Open Catalyst setting in [105] and [106]

| Model | Dataset                    | Batch size | Initial learning rate | Training epoch |
|-------|----------------------------|------------|-----------------------|----------------|
| CGCNN | Poisoned Catalyst          | 128        | $10^{-3}$             | 80             |
|       | Expanded Poisoned Catalyst | 128        | $10^{-3}$             | 80             |
| PaiNN | Open Catalyst 2022         | 256        | $10^{-4}$             | 200            |
|       | Poisoned Catalyst          | 32         | $10^{-4}$             | 80             |
|       | Expanded Poisoned Catalyst | 32         | $10^{-4}$             | 80             |

### 5.1.3.2 Training flow

To evaluate the effectiveness of the proposed methods, we train the GNN by three different training methods, as displayed in Fig. 5.3; (a) only fine-tune GNN (not using SSL), (b) pre-train GNN by existing mask-based SSL proposed in [97], then fine-tune the pre-trained GNN, and (c) pre-train GNN by proposed mask-less SSL, then fine-tune the pre-trained GNN. The prior existing SSL methods using mask for energy prediction GNNs [95, 96, 97, 99] have not applied their proposal to the catalyst datasets such as OC20 and OC22. To evaluate the performance of the existing mask-based SSL method for catalyst energy prediction, we implement one of the latest existing mask-based SSL method [97] with PyTorch by referring to their published code on GitHub [103].

In this work, we use the energy prediction GNN models of CGCNN [89] and PaiNN [90]. For CGCNN training, we first train CGCNN by OC20 IS2RE dataset. Then we apply SSL to the CGCNN trained by OC20. After SSL, the CGCNN trained by the SSL task is fine-tuned by the downstream task. For PaiNN training using OC22 dataset, we first train the scratch PaiNN model by SSL. Then we fine-tune the PaiNN trained by the SSL task using OC22 dataset. When using the Poisoned Catalyst dataset and the Expanded Poisoned Catalyst dataset, we first apply SSL to the pre-trained PaiNN model with OC20 IS2RE dataset published in [104]. After SSL, we fine-tune the pre-trained PaiNN by the downstream task. The unlabeled data for all SSL tasks in this work are generated by removing the energy information from the labeled downstream task datasets, OC22, Poisoned Catalyst, and Expanded Poisoned Catalyst.

For the hyperparameters in pre-training with SSL and fine-tuning, we use almost the same hyperparameters published in Open Catalyst GitHub (CGCNN [105], PaiNN [106]). We only change the batch size, the initial learning rate, and the training epochs from the published hyperparameters setting, as shown in Table 5.2. The existing and proposed SSL require the hyperparameters of the mask ratio and the replacement ratio. We set the mask and the replacement ratio to  $\{10\%, 20\%, 30\%\}$  for each SSL method. After SSL, we fine-tune the GNNs pre-trained by the three ratios of the mask and the replacement. In the performance comparison, we use the best prediction accuracy of the fine-tuned models pre-trained by the three ratios.

For the validation using OC22, we use the dataset called the In-Domain validation dataset for IS2RE. For the validation using the Poisoned Catalyst dataset and the Expanded Poisoned Catalyst dataset, we execute 5-fold cross-validation. We run all experiments three times and take the average and the standard deviation of mean absolute error (MAE) of predicted energy for performance comparison. In other words, for the experiment using OC22, we run the experiment three times, and for the experiment using the Poisoned Catalyst and the Expanded Poisoned Catalyst, we run the experiment fifteen times by running the 5-fold cross-validation three times.

Table 5.3 Comparison of MAE of predicted energy on the evaluation dataset.

| Dataset                    | Model | Mean absolute error (eV)         |                                            |                                       |
|----------------------------|-------|----------------------------------|--------------------------------------------|---------------------------------------|
|                            |       | Without using SSL<br>Fig. 6.3(a) | Existing mask-based SSL [17]<br>Fig 6.3(b) | Proposed mask-less SSL<br>Fig 6.3(c)  |
| Open Catalyst 2022         | PaiNN | $1.9945 \pm 0.0161$              | $1.9720 \pm 0.0124$                        | <b><math>1.9331 \pm 0.0178</math></b> |
| Poisoned Catalyst          | CGCNN | $0.3406 \pm 0.0158$              | $0.3128 \pm 0.0251$                        | <b><math>0.3026 \pm 0.0232</math></b> |
|                            | PaiNN | $0.2878 \pm 0.0205$              | $0.2691 \pm 0.0183$                        | <b><math>0.2579 \pm 0.0175</math></b> |
| Expanded Poisoned Catalyst | CGCNN | $0.3031 \pm 0.0184$              | $0.3009 \pm 0.0242$                        | <b><math>0.2880 \pm 0.0266</math></b> |
|                            | PaiNN | $0.2766 \pm 0.0195$              | $0.2664 \pm 0.0257$                        | <b><math>0.2583 \pm 0.0188</math></b> |

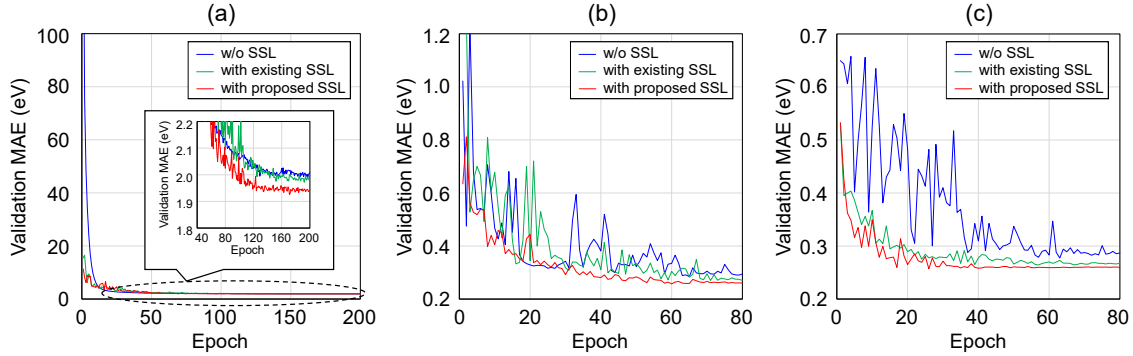


Fig. 5.4 Learning curve of validation MAE of predicted energy during fine-tuning in PaiNN using (a) OC22 dataset, (b) Poisoned Catalyst dataset, and (c) Expanded Poisoned Catalyst dataset.

### 5.1.3.3 Experimental results

To evaluate the effectiveness of the proposed method in various datasets and various GNN models, we compared the MAE of predicted energy by the proposed method and the existing method. Table 5.3 presents the MAE comparisons obtained from different validation datasets (OC22 dataset, Poisoned Catalyst dataset, and Expanded Poisoned Catalyst dataset) using CGCNN and PaiNN. By applying SSL, the MAE of the predicted energy is improved. Furthermore, in all cases, our proposed method achieves a better MAE than that of applying existing SSL. This result indicates that our proposed method has the potential to improve energy prediction accuracy in various datasets and GNN models.

Figure 5.4 shows the learning curve of validation MAE of the predicted energy using PaiNN during fine-tuning. In all cases, by applying SSL, the convergence speed of the learning curve becomes faster than not applying the SSL, in addition to the improvement of the MAE. Furthermore, when applying the proposed SSL, the convergence speed of the learning curve becomes even faster than when applying the existing SSL method.

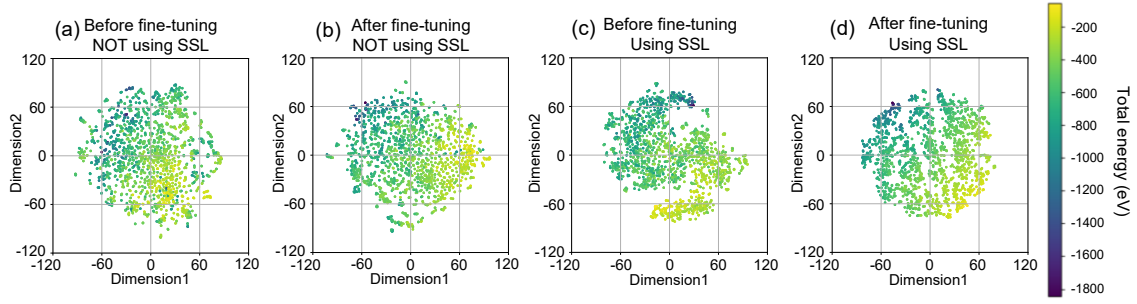


Fig. 5.5 Two-dimensional visualization of extracted vector embeddings from PaiNN by t-SNE. The embeddings are obtained by OC22 IS2RE validation data. The color bar shows the total energy of each point in the t-SNE projection. (a) t-SNE projection from the PaiNN before fine-tuning without SSL, i.e. scratch PaiNN model. (b) t-SNE projection from fine-tuned PaiNN without SSL. (c) t-SNE projection from PaiNN pre-trained by SSL before fine-tuning. (d) t-SNE projection from fine-tuned PaiNN pre-trained by SSL.

Table 5.4 Euclidean distance of vector embeddings of PaiNN before and after fine-tuning calculated using the results of Fig. 6.5.

| Euclidean distance          |              |
|-----------------------------|--------------|
| Without SSL <sup>1</sup>    | 79.94        |
| <b>With SSL<sup>2</sup></b> | <b>47.62</b> |

1. The distance of "Without using SSL" is calculated by the embeddings of the PaiNN before and after fine-tuning without using SSL.

2. The distance of "With SSL" is calculated by the embeddings of the PaiNN pre-trained by the proposed SSL before and after fine-tuning.

To clarify why the fine-tuning is accelerated when using SSL, we reduce the dimension of GNN vector embeddings before and after fine-tuning, with or without applying SSL, by the t-distributed Stochastic Neighbor Embedding (t-SNE) [107]. Figure 5.5 shows the two-dimensional t-SNE projections of the extracted vector embeddings of PaiNN by using OC22 validation data. Each point is colored by the total energy. Table 5.4 shows the Euclidean distance calculated by the vector embeddings of PaiNN before and after fine-tuning. By applying the proposed SSL, Euclidean distance before and after fine-tuning is decreased. That is, the change in the value of vector embeddings of PaiNN before and after fine-tuning is decreased by the proposed SSL.

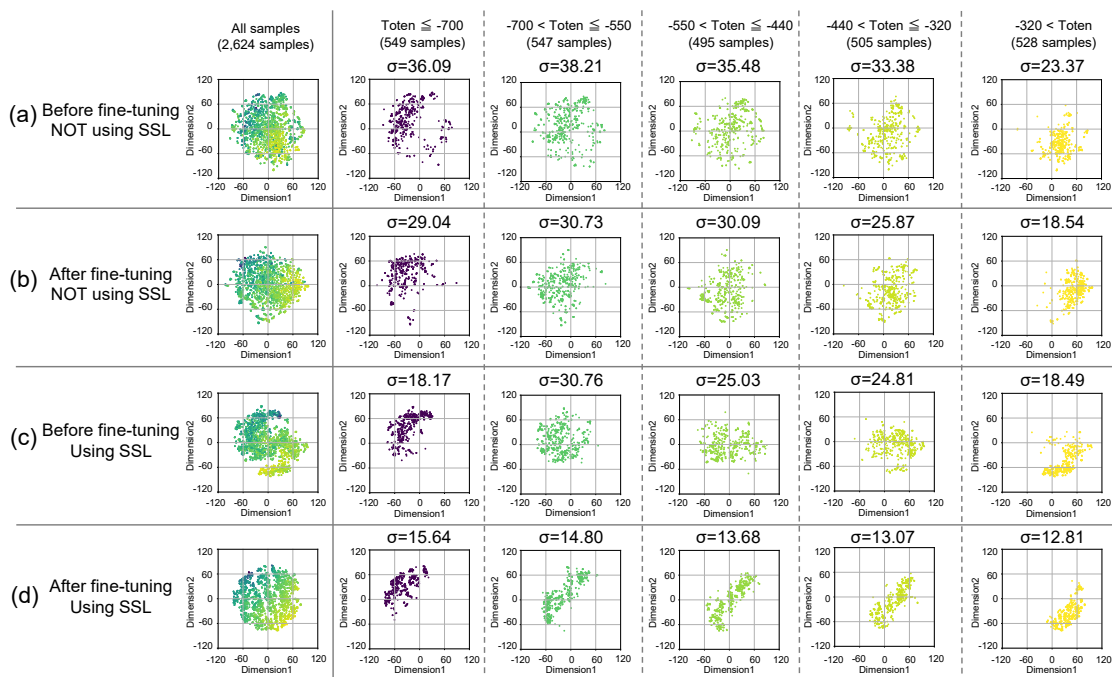


Fig. 5.6 Scatter plots created by classifying all points obtained from the OC22 validation data with the specified range of total energy. “Toten” means total energy. Standard deviation  $\sigma$  for each cluster is calculated by the residuals among the regression line by orthogonal distance regression and all points in each cluster. That is,  $\sigma$  represents the variance of the points in the cluster separated by similar total energy.

The reason why the change of vector embeddings before and after fine-tuning is decreased when using SSL may be that the GNNs pre-trained by our proposed SSL learn a part of the representation which will be learned during fine-tuning. In fine-tuning, since GNNs are trained by labeled catalyst structure data with energy, the fine-tuned GNNs are expected to learn the representation of the catalyst structure and energy. Figure 6 shows the scatter plots which are created by classifying all points obtained from the OC22 validation data, as shown in Fig. 5.5, with the specified range of total energy. In Fig. 5.6, Toten means total energy. Standard deviation  $\sigma$  for each cluster, displayed in Fig. 5.6, is calculated by the residuals among the regression line obtained by orthogonal distance regression, i.e. Deming regression, and all points in each cluster. In other words,  $\sigma$  represents the variance of the points in the cluster separated by similar total energy. By comparing Fig. 5.6(a) and Fig. 5.6(b), we can see that the variance of the points in the cluster of similar total energy is decreased by fine-tuning. These results indicate that GNNs learn the representations of the structure and total energy of the catalyst through

fine-tuning. On the other hand, in SSL, GNNs are trained by unlabeled catalyst structure data without energy. Hence, the GNNs pre-trained by SSL task are expected to learn only the representation of the catalyst structure like the element composition and so on. Note that, in the field of material chemistry, it is well known that materials with similar element composition have similar total energy. The total energy of a material depends on the number of electrons in the atoms composing the material. That is, the total energy of a material depends on the element composition of the material. From this fact in the material chemistry, the vector embeddings of similar energy of the GNNs pre-trained by SSL are expected to be placed close. By comparing Fig. 5.6(a) and Fig. 5.6(c), we can see that  $\sigma$  is decreased by SSL even when the fine-tuning is not applied. Hence, GNNs seem to learn the representation of the element composition through SSL.

From these results, the GNNs pre-trained by SSL may learn a part of the representations of the element composition which was intended to be learned by the later fine-tuning step. Therefore, the change of vector embeddings before and after fine-tuning become small by SSL, as shown in Fig.5. As a result, GNNs pre-trained by the proposed SSL must learn a fewer number of representations during fine-tuning, and thus, the fine-tuning is accelerated by SSL, as shown in Fig. 5.4.

## 5.2 Active learning for graph neural network training in catalyst energy prediction

As described in previous section, energy prediction methods using graph neural networks (GNNs) have been proposed to reduce the computational cost of energy calculations utilizing DFT [90, 113, 114]. However, computationally heavy DFT calculations are still required to generate labeled training data to create highly accurate GNNs.

Active learning for neural network training has been proposed to achieve highly accurate neural networks with a smaller size of labeled training data [115]. The active learning algorithm chooses the data for training from the candidates of the unlabeled training data. Previous works [123, 124, 125] use the geometric distances calculated by the features of the unlabeled data aiming to sample the training datasets that capture the diversity of the unlabeled data. In materials discovery, promising materials are discovered from a vast array of material structure data. Another work [126] applied the active learning method using geometric distances to sample the training datasets that capture the diversity of material structures for material energy prediction. The energies of the materials, however, are very different even when the structures of materials are similar. In addition, the energy prediction accuracy of GNNs is lowered when there is a large variability in the energies of the structures in the training dataset. Therefore, when training a GNN to predict the energies of different materials, the training data should capture the energy diversity, not only the structural diversity.

To improve the accuracy of GNNs for material energy prediction with less training data, we propose an active learning method that can sample the data capturing structural and energetic diversity. Our method first prepares the unlabeled material structure data that serve as the basis for the labeled training data. Then, the structural features and the energy value of the unlabeled data are obtained from GNN. Next, we make the expanded features by concatenating the structural features and the energy value. After that, the Euclidean distances among the expanded features are calculated. Finally, a specified number of structural data for GNN training are sampled from the candidates in the order of the longest of the minimum distances calculated for each feature. The proposed method samples data in the expanded feature space while preserving the mutual distances.

Consequently, the sampled data captures the diversity of the entire feature space, i.e., the structural and energy diversity, composed of the unlabeled data. We evaluate the proposed active learning method with PaiNN [90] and EquiformerV2 [113] by using three catalyst datasets, Open Catalyst 2022 [93], catalysts for nitrogen reduction reaction [110], and catalysts for oxygen reduction reaction. The proposed method shows the best prediction accuracy with less training data under all experimental conditions.

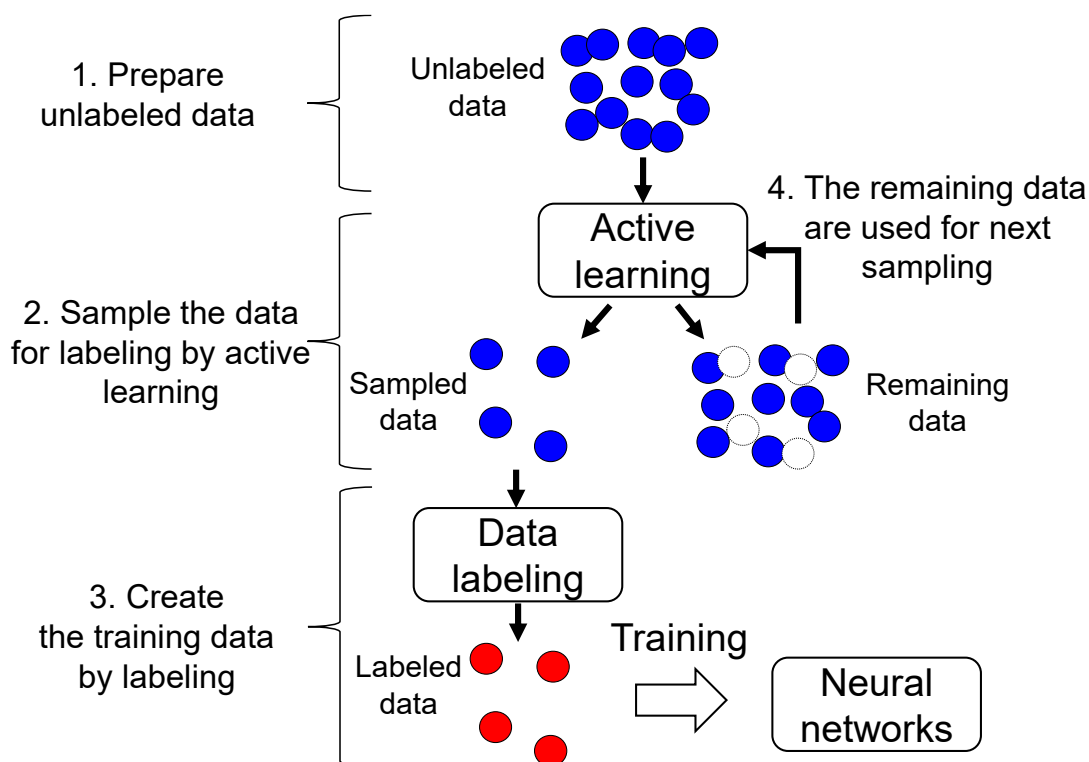


Fig.5.7 Training flow of neural network with active learning.

### 5.2.1 Related works

Data labeling and annotation, which are necessary for creating training data for neural networks, generally require significant time and cost. To create highly accurate neural networks with less training data, active learning has been widely utilized to sample the data used for training from large amounts of unlabeled data [115]. Figure 5.7 shows an overview of neural network training flow with active learning. A large amount of unlabeled data is first prepared. Next, the data are sampled based on an active learning algorithm from the unlabeled data. The training data are created by labeling the sampled data. Then the neural network is trained using the labeled training data, while the

remaining unlabeled data are used in the next sampling step.

Previous studies have proposed various active learning methods. Uncertainty sampling is one of the common strategies in active learning [116, 117, 118]. The policy of uncertainty sampling is to reduce model uncertainty, i.e., to improve the model accuracy, by learning from highly uncertain data first. Thus, this method chooses the data from the unlabeled data by calculating the uncertainty of all unlabeled data. Various uncertainties have been proposed for uncertainty sampling such as entropy [116], least confident [117], and the variance of the predicted values by Gaussian process regression (GPR) based on Bayesian optimization [118]. Expected model change [127] chooses data that will result in a larger change in the expected value of the gradient generated during backpropagation. In recent years, many GNNs have been proposed to predict the energy of materials [90, 113, 114]. Because the annotation cost for material data is generally high due to the computational effort required for DFT calculation, active learning for material energy prediction by GNNs has been proposed. Previous works [119, 120, 121, 122] apply GPR to sample training data from the unlabeled material structure data. Another work [129] samples the data based on the variance of the multiple predicted values obtained by training multiple models, following the uncertainty sampling policy. However, when multiple data are sampled by these existing active learning methods, similar data will be sampled [123]. Therefore, if there is diversity in the unlabeled data, the sampled data cannot capture the diversity of the unlabeled data.

Various active learning methods have been proposed to sample the data that capture the diversity of the unlabeled data. Early works [129, 130] have proposed the use of a variational autoencoder (VAE) and generative adversarial network (GAN) to generate training data. Other works [131, 132] use the mutual information between the model predictions and the model parameters based on Bayesian optimization. Meanwhile, previous studies [123, 124, 125] have proposed the data sampling method using the geometric distances calculated by features of the unlabeled data, without using VAE, GAN, and Bayesian model. Another work [126] applies the active learning method using geometric distance to sample training data for material energy prediction by neural networks. The sampled data by these methods capture the diversity of material structure of unlabeled data.

## 5.2.2 Active learning for material energy prediction GNNs

In this section, we introduce the existing active learning method [126] used for neural network training of material energy prediction, before highlighting its drawbacks. Then, we describe our proposed active learning method.

### 5.2.2.1 Existing active learning for energy prediction GNNs and the problems

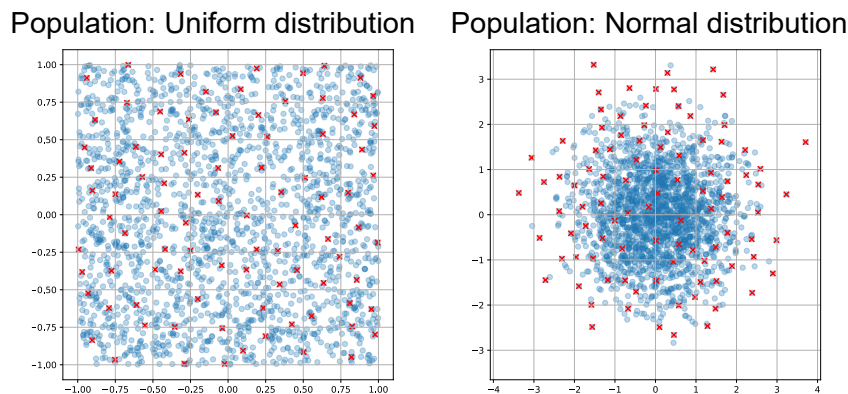


Fig.5.8 Example of sampled data by existing active learning [126]. Blue points are the feature map of the training data candidates. Red crosses are the sampled data by active learning. Since the data are sampled broadly from the population, independent of the distribution of the population, the sampled data capture the diversity of the population's features.

In the materials discovery, promising materials are explored from a large pool of material candidates. In addition, since the structures of the material candidates vary, the neural networks for materials discovery require generalization across different material structures. Previous work [126] has proposed the active learning method to create the training data for neural networks which captures the diversity of the material structures. This method first feeds the unlabeled structure data to the neural network as inputs. Then the features of the material structures are extracted from the output of the hidden layer of the neural network. The dimensions of the extracted structural features are reduced to two to make a feature map. Next, Euclidean distances among the features are calculated. After that, the data used for training are sampled in the order of the longest of the minimum distances calculated for each feature. The data sampled by this method capture the diversity of the structure of the unlabeled structure data, independent of the distribution of the unlabeled data, as shown in Fig. 5.8.

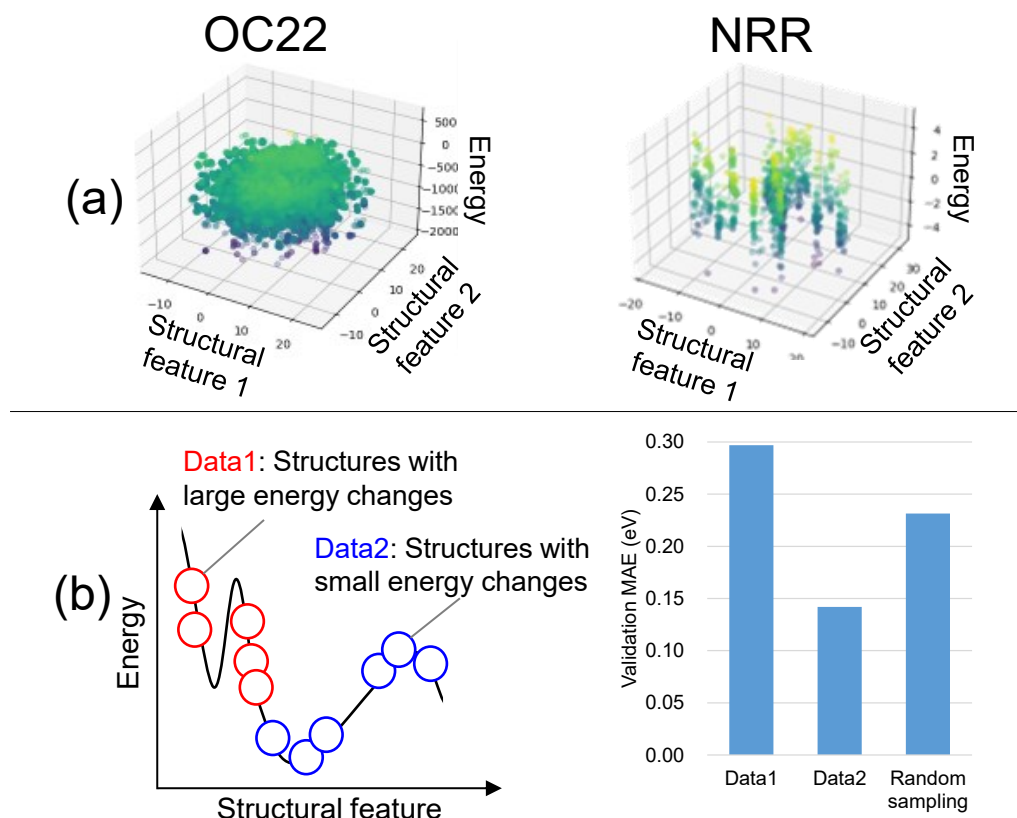


Fig.5.9 (a) Feature map of neural networks training data for catalyst energy prediction [93, 110]. The x-axis and y-axis show the features of material structure. The z-axis shows the energy of catalysts. Even if the structural features are similar, i.e. the material structures are similar, the energies among the materials are different. (b) MAE of the predicted energy for the three validation datasets. The evaluated model is a published PaiNN model [104]. The energy prediction error increases when the validation data is composed of structures with large energy changes.

However, because the existing method primarily uses the features of the material structures for data sampling, it is difficult to capture the diversity in the energy of the materials. Fig. 5.9(a) shows the feature maps of catalyst datasets, Open Catalyst 2022, and catalysts for nitrogen reduction reaction (NRR). In the catalyst search, even though the structures among different materials are similar, the energies of the materials are very different. Fig. 5.9(b) shows the mean absolute error (MAE) of the predicted energy for three validation datasets. To create the three validation datasets, we extract 100 data points from the validation data of NRR by sampling the structure data with large energy changes (Data1), by sampling the data with small energy changes (Data2), and by random

sampling. We evaluate the MAE using the published GNN model of PaiNN [104], trained by Open Catalyst 2020 data [92]. The obtained MAEs indicate that the energy prediction accuracy degrades when the validation data is composed of structures with large energy variations. From these observations, the neural networks for catalyst search require generalization not only of the structure but also of the energy of materials. In other words, if we obtain the training data that captures both the structural and energy diversity of the materials, the energy prediction accuracy of neural networks is expected to improve, even with less training data.

### 5.2.2.2 Proposed active learning using expanded features

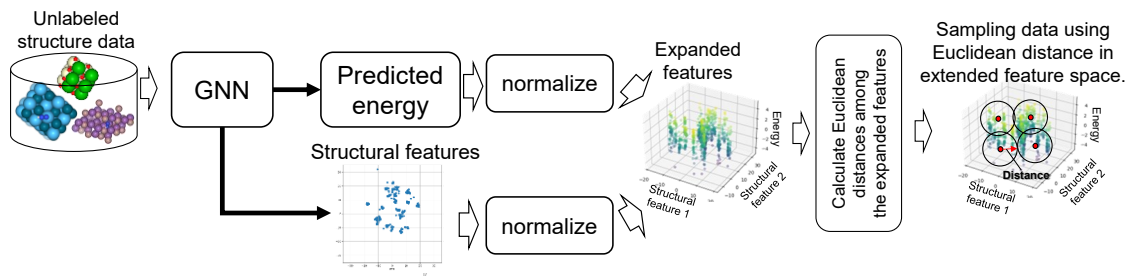


Fig.5.10 Overview of proposed active learning method.

To obtain training data that captures both the structural and energy diversity of the materials, we propose an active learning method that utilizes expanded features, which include information on the structure and energy of the materials. Figure 5.10 shows an overview of our proposed method. Our method first prepares the unlabeled material structure data with unknown energy for the candidates of GNNs training data. Next, N-dimensional features of the structure are extracted from the feature extractor of GNN, by inputting the unlabeled structure data, where N is the number of atoms per input material structures. In addition, the energy prediction values of the unlabeled structure data are obtained from the GNN. The dimensions of the extracted structural features are reduced to two by densMAP [133]. Next, the structural features and the predicted energies are normalized. Then the expanded features are generated by concatenating the normalized structural features and energies. Although there is a large scatter in the structural features of the catalyst data, there is a lot of overlap, as shown in Fig. 5.9(a). To create the GNNs with generalization ability for the structure data, the training data for GNNs require

capturing the diversity of the whole extended feature space, i.e., the structural and energy diversity, independent of the distribution of the unlabeled data. To exclude similar features, i.e., similar structures and energies, we calculate the Euclidean distances of the expanded features by following the equation,

$$Distance_{i,j} = \sqrt{(SF1_i - SF1_j)^2 + (SF2_i - SF2_j)^2 + (EN_i - EN_j)^2} \quad (5.1)$$

where  $i$  and  $j$  label distinct structures and  $Distance_{ij}$  is the distance between the structure  $i$  and  $j$ .  $SF1$  and  $SF2$  represent the structural features, and  $EN$  is the predicted energy. When the number of unlabeled data is  $M$ , we obtain an  $M - 1 \times M - 1$  distance table. Using the distance table, a specified number of structure data are sampled from the candidates in the order of the longest of the minimum distances calculated for each feature. In this work, we set the sampling size to 10% of the total number of unlabeled data. We create the labeled training data using the sampled data by calculating the energy with DFT. Then, the GNN is trained using the labeled training data. The remaining unlabeled structure data are used for the next sampling step. Data sampled in the second and subsequent sampling steps are added to the existing training data.

### 5.2.3 Experiment

We evaluate the proposed active learning method with the energy prediction GNNs of PaiNN and EquiformerV2, using three datasets for catalyst energy prediction.

#### 5.2.3.1 Dataset

We describe an overview of the three datasets used for evaluating our proposed method.

*a) Open Catalyst 2022:* Open Catalyst 2022 (OC22) is one of the largest published catalyst datasets for predicting the total energy of catalysts. OC22 consists of the structure of various solid catalysts with adsorbate molecules and the total energy of the catalyst-adsorbate system. The Open Catalyst benchmark provides three tasks called S2EF (structure to energy and forces), IS2RS (initial structure to relaxed structure), and IS2RE (initial structure to relaxed energy) [91]. In this work, we utilize the IS2RE task for predicting the total energy of the relaxed state structure based on the non-relaxed state structure. For the evaluation using the OC22 data, we use OC22 In-Domain validation

data for IS2RE.

*b) NRR: Catalysts data for nitrogen reduction reaction:* In recent years, numerous materials have been explored as potential electrocatalysts for electrochemical nitrogen reduction reaction (NRR) [81, 82, 83, 110]. Since OC22 data does not include some adsorbates involved in the NRR, we have created catalyst-adsorbate system datasets including those adsorbates (e.g. N<sub>2</sub>, NH<sub>2</sub>, and NNH), by referring to [110] as shown in Table I. Since the adsorption energy is a useful metric when searching for promising catalysts [110], we use the DFT calculated adsorption energy for the labeled data in NRR data. Moreover, GNNs predict the adsorption energy of the relaxed state structure from the non-relaxed state structure.

*c) ORR: Catalysts data for oxygen reduction reaction:* In fuel cell development, transition metal-based catalysts have been explored to efficiently realize oxygen reduction reaction (ORR) to extract electricity from oxygen and hydrogen [111, 112]. We have created catalyst-adsorbate system datasets for ORR in the same manner as for the NRR data by referring to [110]. We design the task for ORR to predict the adsorption energy of the relaxed state structure from the non-relaxed state structure.

### 5.2.3.2 Models and hyperparameters for training

In this work, we use the energy prediction GNN models of PaiNN and EquiformerV2. For the hyperparameters in GNNs training, we use almost the same hyperparameters as those published on GitHub for PaiNN [106] and EquiformerV2 [134]. We modify the batch size, the initial learning rate, and the training epochs from the published hyperparameters setting. For PaiNN training, we set the batch size to 4, the initial learning rate to  $5 \times 10^{-5}$ , and the training epochs to 80. For EquiformerV2 training, we set the batch size to 4, the initial learning rate to  $10^{-4}$ , and the training epochs to 50.

### 5.2.3.3 Training flow with active learning

To evaluate the effectiveness of the proposed methods, we apply four data sampling methods to sample the GNN training data from the evaluation data. The applied methods are random sampling, uncertainty sampling based on the variance of the predicted energy obtained by Gaussian process regression (GPR) [120], the existing active learning using

distances calculated by structural features [126], and our proposed method. We use densMAP [133], a dimensionality reduction method, to reduce the dimensions of the structural features extracted from GNN when applying the existing active learning method and our proposed method. We sample the training data using evaluated sampling methods by 10% of the total number of unlabeled data for each sampling event. Training data sampled during the second and subsequent sampling events are added to the existing training data. After sampling the training data, we train the scratch GNN model using all available training data to avoid catastrophic forgetting [135], a phenomenon where the model forgets previously learned information. For validation, we calculate the MAE of the predicted energy to evaluate the GNN accuracy. We run all experiments three times and take the average of the MAE of three runs.

#### 5.2.3.4 Experimental results

We first evaluate our active learning method on OC22 data for predicting the total energy of the catalysts. Figure 5.11 shows the results of active learning for GNN training on OC22 data. As shown in Fig. 5.11, our proposed method consistently achieves the lowest MAE among all methods, even with a limited amount of training data.

To investigate the effectiveness of our method when using metrics other than total energy for feature expansion, we evaluate our method using NRR and ORR data. These datasets are used for predicting the adsorption energy in the catalyst-adsorbate system. Figure 5.12 and Figure 5.13 show the results of active learning for GNN training using NRR data and ORR data. As shown in Fig. 5.12 and Fig. 5.13, even in this scenario, our proposed method also achieves the lowest MAE among all methods, with a limited amount of training data.

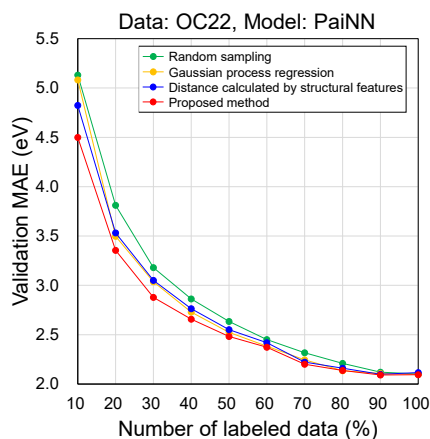


Fig.5.11 Comparison of the MAE using different active learning on OC22 data.

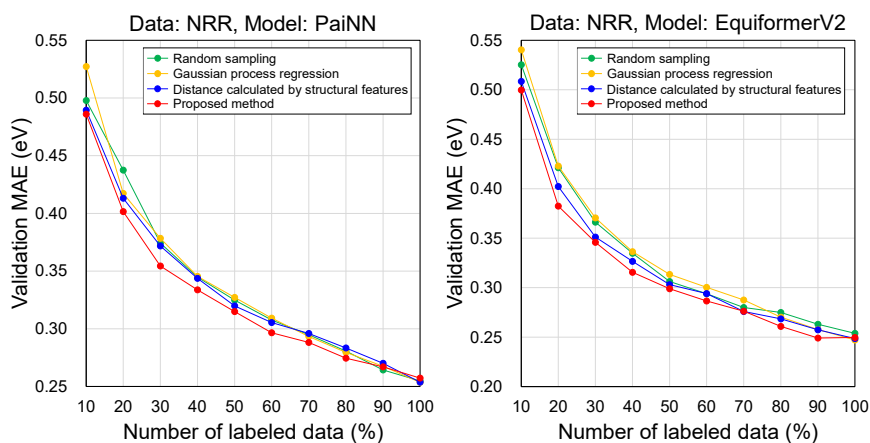


Fig.5.12 Comparison of the MAE using different active learning on NRR data.

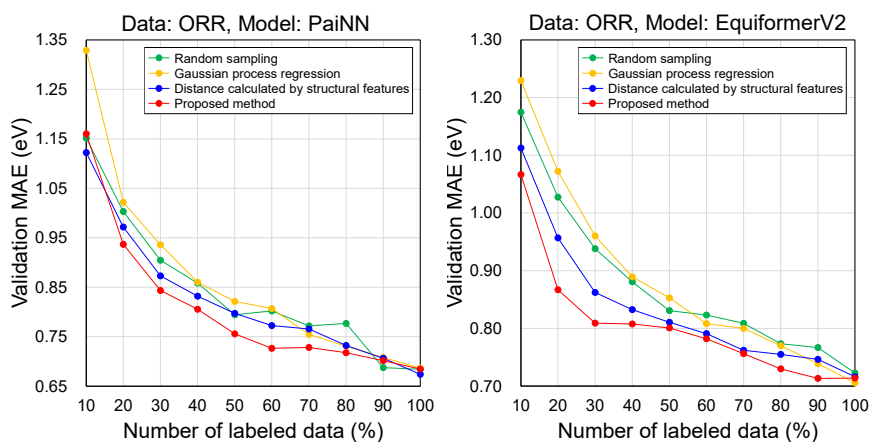


Fig.5.13 Comparison of the MAE using different active learning on ORR data.

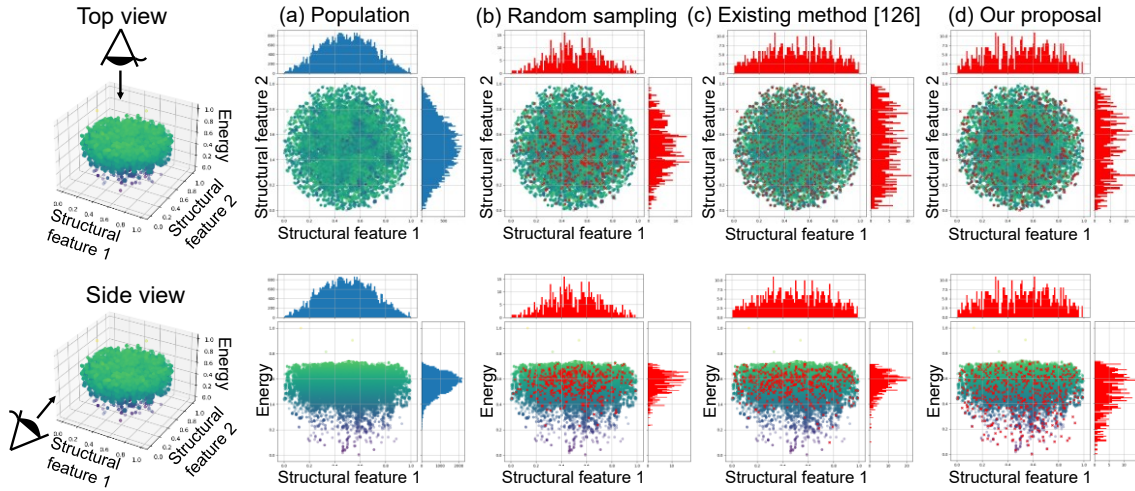


Fig.5.14 The feature space of the proposed expanded features generated using PaiNN with OC22 training data. Red crosses show 500 sampled data points by various sampling methods. The top row 2-D feature maps represent the top view of 3-D feature space. The bottom row 2-D feature maps represent the side view. The distributions of the data sampled by our proposed method show a more even spread in terms of all dimensions, i.e., the structural features and the energy, compared to those sampled by random sampling and the existing active learning method, which are heavily influenced by the population distribution.

Table 5.5 Standard deviation of the population and the sampled data by various sampling method in Fig. 5.14.

|                            | Standard deviation of population | Standard deviation of sampled data |                   |                 |
|----------------------------|----------------------------------|------------------------------------|-------------------|-----------------|
|                            |                                  | Random sampling                    | Existing AL [126] | Proposed method |
| Structural feature 1 (1-D) | 0.2028                           | 0.2018                             | <b>0.2484</b>     | <b>0.2473</b>   |
| Structural feature 2 (1-D) | 0.2008                           | 0.1937                             | <b>0.2548</b>     | <b>0.2377</b>   |
| Energy (1-D)               | 0.0900                           | 0.0877                             | 0.0905            | <b>0.1606</b>   |
| Expanded feature (3-D)     | 0.2993                           | 0.2931                             | 0.3672            | <b>0.3787</b>   |

Fig. 5.14 shows the feature spaces of the expanded features obtained by PaiNN with OC22 data, and the sampled data points by various sampling methods. The dimensions of the extracted structural features are reduced by densMAP. The blue distributions are the distributions of the population, i.e., the expanded features. The red crosses show 500 sampled data points, and the red distributions are the distribution of the sampled data

points. Fig. 5.14(a) shows the population is generated by OC22 data. In this example, the x-axis and the y-axis are the structural features, and the z-axis is the energy. Fig. 5.14(b) is the distributions of the sampled data when applying random sampling for all dimensions, i.e., structural feature1, structural feature2, and energy, depend on the distributions of the population. Fig. 5.14(c) is the distributions of the sampled data when applying the existing active learning method [126]. Although the distribution of the sampled data for structural features resembles a uniform distribution, the distribution of the sampled data for energy depends on the distribution of the population. Thus, the data sampled by the existing method do not capture the diversity of the energy of the population. Fig. 5.14(d) represents the distributions of the sampled data when applying the proposed method. By applying our method, the distributions of the sampled data for the structural features and the energy are broad compared to the distributions obtained by other sampling methods. Therefore, the sampled data by our method capture both the structural and energy diversity. Table 5.5 shows the standard deviation calculated by the population data and the sampled data. The standard deviations for the structural features and the energy are calculated using a 1-dimensional vector that represents the values of the structural features and the energy, respectively. The standard deviation of the expanded features is calculated using a 3-dimensional vector that includes the values of the structural features and the energy. The data sampled by our proposed method have the largest standard deviation calculated using the expanded features, indicating that our method samples the data more evenly from the population, regardless of the population distribution. This is in contrast to random sampling and the existing active learning method. Therefore, the data sampled by our method can better capture the diversity of the feature spaces.

### 5.3 Summary

In this chapter, we proposed a mask-less SSL method with atom replacement using unlabeled data to improve the catalyst energy prediction accuracy of GNNs. Obtaining a large amount of labeled data for material's energy prediction GNNs training is generally difficult due to the DFT's long energy computation time. The proposed SSL aims for efficient learning by replacing atoms by only using the actual existing elements, whereas the existing SSL methods use non-existing atoms of "mask". We evaluate the proposed method on the Open Catalyst dataset, the Poisoned Catalyst dataset, and the Expanded Poisoned Catalyst dataset using energy prediction GNNs of CGCNN and PaiNN. The evaluation results show that the proposed SSL improves the MAE of predicted energy in all cases. We also show that the proposed method not only improves the energy prediction accuracy but also accelerates the fine-tuning after pre-training. By analyzing the vector embedding of GNNs using t-SNE, we find that the GNNs pre-trained by the proposed SSL using unlabeled catalyst data may learn the representation of the element composition of the catalyst. Furthermore, we suggest that the GNNs pre-trained by SSL learn in advance a part of the representations of the element composition intended to be learned by fine-tuning. Therefore, the GNNs pre-trained by the proposed SSL must learn a fewer number of representations during fine-tuning. As a result, the fine-tuning is accelerated when using the proposed SSL.

Furthermore, we proposed an active learning method to improve the catalyst energy prediction accuracy of GNNs with limited training data. Obtaining a large amount of training data for material energy prediction is generally difficult due to the long energy computation time required by DFT. Our proposed active learning method aims to reduce the number of training data by leveraging all available metrics, including material data such as structure and energy. This is in contrast to existing active learning methods, which use only the structural features. We evaluated our method on three datasets, OC22, NRR, and ORR, using energy prediction GNNs of PaiNN and EquiformerV2. The evaluation results show that our method improves the MAE of the predicted energy with limited training data among all evaluated sampling methods.

## Chapter 6 Conclusion

This dissertation presents DNN computational cost reduction techniques, including automated hyperparameter search, not only for established fields such as image processing but also for emerging fields such as materials discovery.

In Chapter 2, the intrinsic issues of the acceleration method of DNNs are introduced as follows:

- Trade-off between prediction accuracy and computation speed
- Manual hyperparameter tuning in acceleration method of DNNs
- DNNs computational cost for materials discovery

In this study, the solutions to these issues are presented in Chapter 3 to Chapter 5:

- 1) 8-bit shifted dynamic fixed-point format for quantized DNNs (Chapter 3)
- 2) Structured pruning with zero-padding layers (Chapter 4)
- 3) Gradient-aware automatic pruning rate search (Chapter 4)
- 4) Self-supervised learning with atom replacement (Chapter 5)
- 5) Active learning utilizing expanded feature (Chapter 5)

In Chapter 3, a quantization method using shifted dynamic fixed-point format for DNNs training is proposed. To prevent accuracy degradation due to quantized DNN training when using conventional DFP format, S-DFP shifts the data representable range of DFP from a large value area to a small value area by adding bias to the exponent of DFP. Using the proposed S-DFP format for quantized DNNs training, the quantized model can be trained using 8-bit fixed-point precision without significant accuracy degradation on the ImageNet task with ResNet-32, ResNet-50, ResNet-101, and ResNet-152. Since the validation accuracy of the quantized model with proposed S-DFP was achieved equivalent accuracy with FP32 regardless of the depth in ResNet, the proposed S-DFP is expected to significantly improve the accuracy of the quantized model regardless of the DNN model type. In addition, the proposed method achieved the equivalent accuracy with conventional 16-bit DFP instead of using 8-bit S-DFP.

In Chapter 4, to reduce the workload of inefficient manual pruning rate assignment, we describe our proposed automatic pruning rate search method named GAPRS for structured pruning. In the proposed method, by selecting a pruning rate in which the pruning error is smaller than a pre-defined threshold, the accuracy degradation of the

pruned model is suppressed. The experimental results demonstrate that the pruned model accuracy can be effectively controlled by the proposed method. In addition, the pruning rate derived by the proposed method can be converged. Furthermore, to improve the inference speed and the compression ratio for the DNN model with a shortcut connection such as ResNet, zero-padding layers are inserted into all input paths to the addition operator. We evaluated our proposed method on CIFAR-10 and ImageNet using various ResNet architectures. We experimentally showed the superiority of GAPRS against various state-of-the-art methods on CIFAR-10 and ImageNet using various ResNet architectures. For ResNet-50, our method achieved a 60.1% reduction in parameters and a 59.8% reduction in FLOPs while maintaining 76.17% accuracy on ImageNet. These results demonstrate the effectiveness of GAPRS and the proposed inserting zero-padding layers in achieving high compression ratios without model accuracy degradation.

Chapter 5 presents novel SSL and active learning methods to reduce the computational cost of materials discovery. Our proposed mask-less SSL method employs atom replacement with unlabeled data to improve DNN-based catalyst energy prediction accuracy. Unlike existing SSL methods that utilize non-existing "mask" atoms, our approach leverages only real elements, promoting efficient learning. We evaluated this method on three benchmark datasets (OC22, Poisoned Catalyst, and Expanded Poisoned Catalyst) using CGCNN and PaiNN GNNs, observing improved energy prediction accuracy across all experiments. Furthermore, we introduce a novel active learning method to enhance DNN-based catalyst energy prediction accuracy with limited training data. This method leverages all available material properties (structure and energy) to minimize the required training data. Evaluations on three benchmark datasets (OC22, NRR, and ORR) using PaiNN and EquiformerV2 GNNs demonstrate that our method achieves the lowest prediction error among all evaluated sampling methods, even with limited data.

Finally, the conclusion of this study is presented in this chapter. This dissertation presents methods for achieving low-cost DNN computation and automatic hyperparameter optimization for DNN acceleration, addressing not only established applications such as image processing but also emerging fields such as materials discovery. Our work contributes to bridging the DNN computational demand-supply gap in diverse fields, including image processing and materials discovery, while avoiding

inefficient manual hyperparameter tuning for DNN acceleration.



## References

- [1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- [2] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).
- [3] Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.
- [4] Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE access*, 6, 64270-64277.
- [5] 坂井 靖文, “深層学習の高性能化と高速化・軽量化技術の関係”, *Software Design* 2021年10月号
- [6] 坂井 靖文, “深層学習のモデル軽量化技術”, *Software Design* 2022年3月号
- [7] Gholami, A., Yao, Z., Kim, S., Hooper, C., Mahoney, M. W., & Keutzer, K. (2024). AI and memory wall. *IEEE Micro*.
- [8] Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., & Villalobos, P. (2022, July). Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.
- [9] Liang, T., Glossner, J., Wang, L., Shi, S., & Zhang, X. (2021). Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461, 370-403.
- [10] Gong, C., Chen, Y., Lu, Y., Li, T., Hao, C., & Chen, D. (2020). VecQ: Minimal loss DNN model compression with vectorized weight quantization. *IEEE Transactions on Computers*, 70(5), 696-710.
- [11] Zhu, M., & Gupta, S. (2017). To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.
- [12] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., & Zou, Y. (2016). DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.
- [13] Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016, September). XNOR-Net: ImageNet classification using binary convolutional neural networks. In *European conference on computer vision* (pp. 525-542). Cham: Springer International Publishing.

- [14] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., & Wu, H. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.
- [15] Chen, C. Y., Choi, J., Gopalakrishnan, K., Srinivasan, V., & Venkataramani, S. (2018, March). Exploiting approximate computing for deep learning acceleration. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 821-826). IEEE.
- [16] Wang, N., Choi, J., Brand, D., Chen, C. Y., & Gopalakrishnan, K. (2018). Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31.
- [17] Sun, X., Choi, J., Chen, C. Y., Wang, N., Venkataramani, S., Srinivasan, V. V., Vijayalakshmi, V., Cui, X., Zhang, Wei., & Gopalakrishnan, K. (2019). Hybrid 8-bit floating point (HFP8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32.
- [18] Horowitz, M. (2014, February). Computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)* (pp. 10-14). IEEE.
- [19] Das, D., Mellempudi, N., Mudigere, D., Kalamkar, D., Avancha, S., Banerjee, K., Sridharan, S., Vaidyanathan, K., Kaul, B., Georganas, E., Heinecke, A., Dubey, P., Corbal, J., Shustrov, N., Dubtsov, R., Fomenko, E., & Pirogov, V. (2018). Mixed precision training of convolutional neural networks using integer operations. *arXiv preprint arXiv:1802.00930*.
- [20] Wu, S., Li, G., Chen, F., & Shi, L. (2018). Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*.
- [21] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [22] Xu, K., Shao, X., Tian, Y., Yang, S., & Zhang, X. (2024). AutoMPQ: Automatic Mixed-Precision Neural Network Search via Few-Shot Quantization Adapter. *IEEE Transactions on Emerging Topics in Computational Intelligence*.
- [23] Yang, Y., Deng, L., Wu, S., Yan, T., Xie, Y., & Li, G. (2020). Training high-performance and large-scale deep neural networks with full 8-bit integers. *Neural Networks*, 125, 70-82.
- [24] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.
- [25] Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning

filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

[26] Salama, A., Ostapenko, O., Klein, T., & Nabi, M. (2019). Pruning at a glance: Global neural pruning for model compression. *arXiv preprint arXiv:1912.00200*.

[27] He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389-1397).

[28] Lee, N., Ajanthan, T., & Torr, P. H. (2018). SNIP: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

[29] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[30] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255).

[31] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision* (pp. 2736-2744).

[32] Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Gutttag, J. (2020). What is the state of neural network pruning?. *Proceedings of machine learning and systems*, 2, 129-146.

[33] Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*.

[34] Wang, W., Fu, C., Guo, J., Cai, D., & He, X. (2019). COP: Customized deep model compression via regularized correlation-based filter-level pruning. *arXiv preprint arXiv:1906.10337*.

[35] Kim, W., Kim, S., Park, M., & Jeon, G. (2020). Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, 33, 585-595.

[36] [https://github.com/akamaster/pytorch\\_resnet\\_cifar10](https://github.com/akamaster/pytorch_resnet_cifar10)

[37] <https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py>

[38] Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., & Xu, C. (2020). SCOP: Scientific control for reliable neural network pruning. *Advances in Neural Information Processing Systems*, 33, 10936-10947.

[39] Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., & Zhu, J. (2018). Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems*, 31.

[40] He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*.

- [41] He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4340-4349).
- [42] Yu, R., Li, A., Chen, C. F., Lai, J. H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., & Davis, L. S. (2018). NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9194-9203).
- [43] Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., & Shao, L. (2020). HRank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1529-1538).
- [44] Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., & Doermann, D. (2019). Towards optimal structured CNN pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2790-2799).
- [45] Luo, J. H., Wu, J., & Lin, W. (2017). ThiNet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision* (pp. 5058-5066).
- [46] Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11264-11272).
- [47] Sui, Y., Yin, M., Xie, Y., Phan, H., Aliari Zonouz, S., & Yuan, B. (2021). CHIP: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34, 24604-24616.
- [48] Yin, M., Phan, H., Zang, X., Liao, S., & Yuan, B. (2022). BATUDE: Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 36, No. 8, pp. 8874-8882).
- [49] Ganjdanesh, A., Gao, S., & Huang, H. (2022). Interpretations steered network pruning via amortized inferred saliency maps. In *European Conference on Computer Vision* (pp. 278-296).
- [50] Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., & Tan, M. (2021). Discrimination-aware network pruning for deep model compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8), 4035-4051.
- [51] Yuan, Y. X. (2000). A review of trust region algorithms for optimization. In *Iciam* (Vol. 99, No. 1, pp. 271-282).
- [52] Dai, X., Yin, H., & Jha, N. K. (2019). NeST: A neural network synthesis tool based on a grow-and-prune paradigm. *IEEE Transactions on Computers*, 68(10),

1487-1497.

- [53] Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.
- [54] See, A., Luong, M. T., & Manning, C. D. (2016). Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*.
- [55] Fang G, Ma X, Wang X (2023) Structural pruning for diffusion models. *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 16716-16728.
- [56] Liu, L., Zhang, S., Kuang, Z., Zhou, A., Xue, J. H., Wang, X., Wang, X., Chen, Y., Yang, W., Liao, Q., & Zhang, W. (2021, July). Group fisher pruning for practical network compression. In *International Conference on Machine Learning* (pp. 7021-7032). PMLR.
- [57] De Jorge, P., Sanyal, A., Behl, H. S., Torr, P. H., Rogez, G., & Dokania, P. K. (2020). Progressive skeletonization: Trimming more fat from a network at initialization. *arXiv preprint arXiv:2006.09081*.
- [58] Wang, C., Zhang, G., & Grosse, R. (2020). Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*.
- [59] Guo, Y., Yao, A., & Chen, Y. (2016). Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29.
- [60] Wang, H., & Fu, Y. (2022). Trainability preserving neural pruning. *arXiv preprint arXiv:2207.12534*.
- [61] Ding, X., Ding, G., Han, J., & Tang, S. (2018). Auto-balanced filter pruning for efficient convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).
- [62] Wang, H., Qin, C., Zhang, Y., & Fu, Y. (2020). Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243*.
- [63] Chin, T. W., Ding, R., Zhang, C., & Marculescu, D. (2020). Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1518-1528).
- [64] Tiwari, R., Bamba, U., Chavan, A., & Gupta, D. K. (2021). ChipNet: Budget-aware pruning with heaviside continuous approximations. *arXiv preprint arXiv:2102.07156*.
- [65] Cheng, H., Zhang, M., & Shi, J. Q. (2024). A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [66] You, Z., Yan, K., Ye, J., Ma, M., & Wang, P. (2019). Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in neural information processing systems*, 32.

- [67] Mehta, R. (2019). Sparse transfer learning via winning lottery tickets. *arXiv preprint arXiv:1905.07785*.
- [68] Liu, N., Yuan, G., Che, Z., Shen, X., Ma, X., Jin, Q., Ren J, Tang J, Liu S, & Wang, Y. (2021, July). Lottery ticket preserves weight correlation: Is it desirable or not?. In *International Conference on Machine Learning* (pp. 7011-7020). PMLR.
- [69] Nocedal, J., & Wright, S. J. (Eds.). (1999). *Numerical optimization*. New York, NY: Springer New York.
- [70] Zhang, G., Xu, S., Li, J., & Guo, A. J. (2022). Group-based network pruning via nonlinear relationship between convolution filters. *Applied Intelligence*, 52(8), 9274-9288.
- [71] Chen, Y., & Wang, Z. (2024). An Effective Information Theoretic Framework for Channel Pruning. *arXiv preprint arXiv:2408.16772*.
- [72] Zheng, Y., Sun, P., Ren, Q., Xu, W., & Zhu, D. (2024). A novel and efficient model pruning method for deep convolutional neural networks by evaluating the direct and indirect effects of filters. *Neurocomputing*, 569, 127124.
- [73] Liu, Y., Fan, K., & Zhou, W. (2024). Iterative filter pruning with combined feature maps and knowledge distillation. *International Journal of Machine Learning and Cybernetics*, 1-15.
- [74] Li, G., Li, R., Li, T., Shen, C., Zou, X., Wang, J., Wang, C., & Li, N. (2025). SFP: Similarity-based filter pruning for deep neural networks. *Information Sciences*, 689, 121418.
- [75] Alwani, M., Wang, Y., & Madhavan, V. (2022). DECORE: Deep compression with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12349-12359).
- [76] Liu, Y., Wu, D., Zhou, W., Fan, K., & Zhou, Z. (2023). EACP: An effective automatic channel pruning for neural networks. *Neurocomputing*, 526, 131-142.
- [77] Chen, Y., Li, R., Li, W., Wang, J., & Li, R. (2023). Three-Stage Global Channel Pruning for Resources-Limited Platform. *IEEE Transactions on Neural Networks and Learning Systems*.
- [78] Lian, Y., Peng, P., Jiang, K., & Xu, W. (2024). Multi-objective compression for CNNs via evolutionary algorithm. *Information Sciences*, 661, 120155.
- [79] Lu, X., Dong, W., Li, X., Wu, J., Li, L., & Shi, G. (2023). Adaptive search-and-training for robust and efficient network pruning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8), 9325-9338.
- [80] Cai, Y., Hua, W., Chen, H., Suh, G. E., De Sa, C., & Zhang, Z. (2022). Structured pruning is all you need for pruning cnns at initialization. *arXiv preprint arXiv:2203.02549*.

- [81] Skúlason, E., Bligaard, T., Gudmundsdóttir, S., Studt, F., Rossmeisl, J., Abild-Pedersen, F., Vegge, T., Jónsson, H., & Nørskov J. K. (2012). A theoretical evaluation of possible transition metal electro-catalysts for N<sub>2</sub> reduction. *Physical Chemistry Chemical Physics*, 14(3), 1235-1245.
- [82] Höskuldsson, Á. B., Abghoui, Y., Gunnarsdóttir, A. B., & Skúlason, E. (2017). Computational screening of rutile oxides for electrochemical ammonia formation. *ACS Sustainable Chemistry & Engineering*, 5(11), 10327-10333.
- [83] Höskuldsson, Á. B., Abghoui, Y., Gunnarsdóttir, A. B., & Skúlason, E. (2021). Correction to “Computational Screening of Rutile Oxides for Electrochemical Ammonia Formation”. *ACS Sustainable Chemistry & Engineering*, 9(23), 8012-8014.
- [84] Gudmundsson, M., Ellingsson, V., Skúlason, E., & Abghoui, Y. (2022). Optimizing nitrogen reduction reaction on nitrides: a computational study on crystallographic orientation. *Topics in Catalysis*, 65(1), 252-261.
- [85] Ishikawa, A., & Tateyama, Y. (2021). A first-principles microkinetics for homogeneous–heterogeneous reactions: application to oxidative coupling of methane catalyzed by magnesium oxide. *ACS Catalysis*, 11(5), 2691-2700.
- [86] Ishikawa, A. (2022). Heterogeneous catalyst design by generative adversarial network and first-principles based microkinetics. *Scientific Reports*, 12(1), 11657.
- [87] Kim, M., Yeo, B. C., Park, Y., Lee, H. M., Han, S. S., & Kim, D. (2019). Artificial intelligence to accelerate the discovery of N<sub>2</sub> electroreduction catalysts. *Chemistry of Materials*, 32(2), 709-720.
- [88] Mohr, S., Ratcliff, L. E., Genovese, L., Caliste, D., Boulanger, P., Goedecker, S., & Deutsch, T. (2015). Accurate and efficient linear scaling DFT calculations with universal applicability. *Physical Chemistry Chemical Physics*, 17(47), 31360-31370.
- [89] Xie, T., & Grossman, J. C. (2018). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14), 145301.
- [90] Schütt, K., Unke, O., & Gastegger, M. (2021, July). Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning* (pp. 9377-9388). PMLR.
- [91] Zitnick, C. L., Chanussot, L., Das, A., Goyal, S., Heras-Domingo, J., Ho, C., Hu, W., Lavril, T., Palizhati, A., Rivière, M., Shuaibi, M., Sriram, A., Tran, K., Wood, B., Yoon, J., Parikh, D., & Ulissi, Z. (2020). An introduction to electrocatalyst design using machine learning for renewable energy storage. *arXiv preprint arXiv:2010.09435*.
- [92] Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K.,

- Heras-Domingo, J., Ho, C., Hu, W., Palizhati, A., Sriram, A., Wood, B., Yoon, J., Parikh, D., Zitnick, C. L., & Ulissi, Z. (2021). Open catalyst 2020 (OC20) dataset and community challenges. *Acs Catalysis*, *11*(10), 6059-6072.
- [93] Tran, R., Lan, J., Shuaibi, M., Wood, B. M., Goyal, S., Das, A., Heras-Domingo, J., Kolluru, A., Rizvi, A., Shoghi, N., Sriram, A., Therrien, F., Abed, J., Voznyy, O., Sargent, E. H., Ulissi, Z., & Zitnick, C. L. (2023). The Open Catalyst 2022 (OC22) dataset and challenges for oxide electrocatalysts. *ACS Catalysis*, *13*(5), 3066-3084.
- [94] <https://github.com/Open-Catalyst-Project/ocp>
- [95] Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., & Leskovec, J. (2019). Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*.
- [96] Zhang, Z., Liu, Q., Wang, H., Lu, C., & Lee, C. K. (2021). Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems*, *34*, 15870-15882.
- [97] Zhang, Z., Liu, Q., Zhang, S., Hsieh, C. Y., Shi, L., & Lee, C. K. (2021). Graph self-supervised learning for optoelectronic properties of organic semiconductors. *arXiv preprint arXiv:2112.01633*.
- [98] Kenton, J. D. M. W. C., & Toutanova, L. K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT* (Vol. 1, p. 2).
- [99] Magar, R., Wang, Y., & Barati Farimani, A. (2022). Crystal twins: self-supervised learning for crystalline material property prediction. *npj Computational Materials*, *8*(1), 231.
- [100] Gasteiger, J., Becker, F., & Günnemann, S. (2021). GemNet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, *34*, 6790-6802.
- [101] Clark, K. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- [102] Yang, Z. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.
- [103] [https://github.com/zaixizhang/SSL\\_OSC](https://github.com/zaixizhang/SSL_OSC)
- [104] [https://dl.fbaipublicfiles.com/opencatalystproject/models/2022\\_05/is2re/painn\\_h1024\\_bs4\\_x8\\_is2re\\_all.pt](https://dl.fbaipublicfiles.com/opencatalystproject/models/2022_05/is2re/painn_h1024_bs4_x8_is2re_all.pt)
- [105] <https://github.com/Open-Catalyst-Project/ocp/blob/main/configs/is2re/all/cgcnn/cgcnn.yml>
- [106] [https://github.com/Open-Catalyst-Project/ocp/blob/main/configs/is2re/all/painn/painn\\_h1024\\_bs8x4.yml](https://github.com/Open-Catalyst-Project/ocp/blob/main/configs/is2re/all/painn/painn_h1024_bs8x4.yml)
- [107] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, *9*(11).
- [108] MacFarlane, D. R., Cherepanov, P. V., Choi, J., Suryanto, B. H., Hodgetts, R.

- Y., Bakker, J. M., Ferrero Vallana, F. M., & Simonov, A. N. (2020). A roadmap to the ammonia economy. *Joule*, 4(6), 1186-1205.
- [109] Smith, C., Hill, A. K., & Torrente-Murciano, L. (2020). Current and future role of Haber–Bosch ammonia in a carbon-free energy landscape. *Energy & Environmental Science*, 13(2), 331-344.
- [110] Höskuldsson, Á. B., Dang, T., Sakai, Y., Ishikawa, A., & Skúlason, E. (2023). High-throughput computational screening of doped transition metal oxides as catalysts for nitrogen reduction. *Cell Reports Physical Science*, 4(10).
- [111] Nørskov, J. K., Rossmeisl, J., Logadottir, A., Lindqvist, L. R. K. J., Kitchin, J. R., Bligaard, T., & Jonsson, H. (2004). Origin of the overpotential for oxygen reduction at a fuel-cell cathode. *The Journal of Physical Chemistry B*, 108(46), 17886-17892.
- [112] Gíslason, P. M., & Skúlason, E. (2019). Catalytic trends of nitrogen doped carbon nanotubes for oxygen reduction reaction. *Nanoscale*, 11(40), 18683-18690.
- [113] Liao, Y. L., Wood, B., Das, A., & Smidt, T. (2023). Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. *arXiv preprint arXiv:2306.12059*.
- [114] Sakai, Y., Dang, T., Fukuta, S., Shirahata, K., Ishikawa, A., Inoue, A., Kawaguchi, H., Höskuldsson, Á. B., & Skúlason, E. (2023). Self-Supervised Learning with Atom Replacement for Catalyst Energy Prediction by Graph Neural Networks. *Procedia Computer Science*, 222, 458-467.
- [115] Settles, B. (2009). Active learning literature survey. *Computer Sciences Technical Report 1648, University of Wisconsin–Madison*.
- [116] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3), 379-423.
- [117] Settles, B., & Craven, M. (2008, October). An analysis of active learning strategies for sequence labeling tasks. In *proceedings of the 2008 conference on empirical methods in natural language processing* (pp. 1070-1079).
- [118] Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.
- [119] Koncewicz, W., Moździerz, M., & Brus, G. (2023). A fast Gaussian process-based method to evaluate carbon deposition during hydrocarbons reforming. *International Journal of Hydrogen Energy*, 48(31), 11666-11679.
- [120] Tamura, R., Takei, Y., Imai, S., Nakahara, M., Shibata, S., Nakanishi, T., & Demura, M. (2021). Experimental design for the highly accurate prediction of material properties using descriptors obtained by measurement. *Science and*

- Technology of Advanced Materials: Methods*, 1(1), 152-161.
- [121] Lookman, T., Balachandran, P. V., Xue, D., & Yuan, R. (2019). Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. *npj Computational Materials*, 5(1), 21.
- [122] Deringer, V. L., Bartók, A. P., Bernstein, N., Wilkins, D. M., Ceriotti, M., & Csányi, G. (2021). Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16), 10073-10141.
- [123] Sener, O., & Savarese, S. (2017). Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.
- [124] Hwang, M., Jeong, Y., & Sung, W. (2020, September). Data distribution search to select core-set for machine learning. In *The 9th International Conference on Smart Media and Applications* (pp. 172-176).
- [125] Paul, M., Ganguli, S., & Dziugaite, G. K. (2021). Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34, 20596-20607.
- [126] Shimizu, K., Arguelles, E. F., Li, W., Ando, Y., Minamitani, E., & Watanabe, S. (2021). Phase stability of Au-Li binary systems studied using neural network potential. *Physical Review B*, 103(9), 094112.
- [127] Cai, W., Zhang, Y., & Zhou, J. (2013, December). Maximizing expected model change for active learning in regression. In *2013 IEEE 13th international conference on data mining* (pp. 51-60). IEEE.
- [128] Musielewicz, J., Wang, X., Tian, T., & Ulissi, Z. (2022). FINETUNA: fine-tuning accelerated molecular simulations. *Machine Learning: Science and Technology*, 3(3), 03LT01.
- [129] Sinha, S., Ebrahimi, S., & Darrell, T. (2019). Variational adversarial active learning. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5972-5981).
- [130] Kim, K., Park, D., Kim, K. I., & Chun, S. Y. (2021). Task-aware variational adversarial active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8166-8175).
- [131] Kirsch, A., Van Amersfoort, J., & Gal, Y. (2019). BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32.
- [132] Kirsch, A., Farquhar, S., Atighehchian, P., Jesson, A., Branchaud-Charron, F., & Gal, Y. (2021). Stochastic batch acquisition: A simple baseline for deep active learning. *arXiv preprint arXiv:2106.12059*.
- [133] Narayan, A., Berger, B., & Cho, H. (2020). Density-preserving data

visualization unveils dynamic patterns of single-cell transcriptomic variability. *bioRxiv*, 2020-05.

[134] [https://github.com/Open-Catalyst-](https://github.com/Open-Catalyst-Project/ocp/blob/main/configs/s2ef/all/equiformer_v2/equiformer_v2_N@8_L@4_M@2_31M.yml)

[Project/ocp/blob/main/configs/s2ef/all/equiformer\\_v2/equiformer\\_v2\\_N@8\\_L@4\\_M@2\\_31M.yml](https://github.com/Open-Catalyst-Project/ocp/blob/main/configs/s2ef/all/equiformer_v2/equiformer_v2_N@8_L@4_M@2_31M.yml)

[135] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113, 54-71.

[136] Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C. J., Kornbluth, M., & Kozinsky, B. (2023). Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1), 579.

[137] Chen, C., & Ong, S. P. (2022). A universal graph deep learning interatomic potential for the periodic table. *Nature Computational Science*, 2(11), 718-728.

[138] Deng, B., Zhong, P., Jun, K., Riebesell, J., Han, K., Bartel, C. J., & Ceder, G. (2023). CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 5(9), 1031-1041.

[139] Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., & Tang, J. (2021). Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1), 857-876.



# List of Publications and Presentations

## Publications in journals

- 1) Sakai, Y., Eto, Y., & Teranishi, Y. (2022). Structured pruning for deep neural networks with adaptive pruning rate derivation based on connection sensitivity and loss function. *Journal of Advances in Information Technology*, 1.
- 2) Höskuldsson, Á. B., Dang, T., Sakai, Y., Ishikawa, A., & Skúlason, E. (2023). High-throughput computational screening of doped transition metal oxides as catalysts for nitrogen reduction. *Cell Reports Physical Science*, 4(10).
- 3) Tsuji, S., Yamada, F., Kawaguchi, H., Inoue, A., & Sakai, Y. (2022). Greedy search algorithm for partial quantization of convolutional neural networks inspired by submodular optimization. *Neural Computing and Applications*, 1-11.
- 4) Sakai, Y., & Tamiya, Y. (2021). S-DFP: shifted dynamic fixed point for quantized deep neural network training. *Neural Computing and Applications*, 1-8.
- 5) Sakai, Y., Pedroni, B. U., Joshi, S., Tanabe, S., Akinin, A., & Cauwenberghs, G. (2019). DropOut and DropConnect for reliable neuromorphic inference under communication constraints in network connectivity. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(4), 658-667.

## Presentations at international conferences

- 1) Sakai, Y., Matsumura, N., Inoue, A., Kawaguchi, H., Thang, D., Ishikawa, A., Höskuldsson, Á. B., & Skúlason, E. (2024, June). Active Learning for Graph Neural Networks Training in Catalyst Energy Prediction. In *2024 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.
- 2) Matsumoto, K., Matsuda, T., Inoue, A., Kawaguchi, H., & Sakai, Y. (2024, February). A Mixed-Precision Quantization Method without Accuracy Degradation Using Semilayers. In *Asian Conference on Machine Learning* (pp. 882-894). PMLR.
- 3) Matsuda, T., Matsumoto, K., Inoue, A., Kawaguchi, H., & Sakai, Y. (2023, September). Semilayer-Wise Partial Quantization Without Accuracy Degradation or Back Propagation. In *International Conference on Artificial Neural Networks* (pp. 283-295). Cham: Springer Nature Switzerland.

- 4) Sakai, Y., Dang, T., Fukuta, S., Shirahata, K., Ishikawa, A., Inoue, A., Kawaguchi, H., Höskuldsson, Á. B., & Skúlason, E. (2023). Self-Supervised Learning with Atom Replacement for Catalyst Energy Prediction by Graph Neural Networks. *Procedia Computer Science*, 222, 458-467.
- 5) Sakai, Y., Iwakawa, A., Tabaru, T., Inoue, A., & Kawaguchi, H. (2022, August). Automatic pruning rate derivation for structured pruning of deep neural networks. In *2022 26th international conference on pattern recognition (ICPR)* (pp. 2561-2567). IEEE.
- 6) Sakai, Y., Iwakawa, A., & Tabaru, T. (2022, April). Structured Pruning with Automatic Pruning Rate Derivation for Image Processing Neural Networks. In *Proceedings of the 2022 6th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence* (pp. 100-104).
- 7) Okado, K., Matsumoto, K., Inoue, A., Kawaguchi, H., & Sakai, Y. (2022, March). Channel-wise quantization without accuracy degradation using  $\Delta$ loss analysis. In *Proceedings of the 2022 7th International Conference on Machine Learning Technologies* (pp. 56-61).
- 8) Tsuji, S., Kawaguchi, H., Inoue, A., Sakai, Y., & Yamada, F. (2021, November). Greedy search algorithm for mixed precision in post-training quantization of convolutional neural network inspired by submodular optimization. In *Asian Conference on Machine Learning* (pp. 886-901). PMLR.
- 9) Yamada, F., Tsuji, S., Kawaguchi, H., Inoue, A., & Sakai, Y. (2021). A High-Speed Neural Architecture Search Considering the Number of Weights. In *KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27–October 1, 2021, Proceedings 44* (pp. 109-115). Springer International Publishing.
- 10) Tsuji, S., Yamada, F., Kawaguchi, H., Inoue, A., & Sakai, Y. (2020, November). GPQ: Greedy Partial Quantization of Convolutional Neural Networks Inspired by Submodular Optimization. In *2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI)* (pp. 106-109). IEEE.
- 11) Sakai, Y. (2020, November). Quantization for deep neural network training with 8-bit dynamic fixed point. In *2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI)* (pp. 126-130). IEEE.

- 12) Tsunoda, Y., Notsu, T., Sakai, Y., Hamada, N., Mori, T., Ishihara, T., & Inoue, A. (2019, July). Neural-network assistance to calculate precise eigenvalue for fitness evaluation of real product design. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 405-406).
- 13) Sakai, Y., Pedroni, B. U., Joshi, S., Akinin, A., & Cauwenberghs, G. (2019, March). DropOut and DropConnect for reliable neuromorphic inference under energy and bandwidth constraints in network connectivity. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 76-80). IEEE.

## Awards

- 1) Y. Sakai, Best Presentation Award, ISMSI, Apr. 2022.

## Publications in Press Release

- 1) “スーパーコンピュータを活用した AI モデルの開発において、世界トップクラスの高速演算性能と認識精度を実現できるソフトウェア技術を開発”, 2021 年 11 月  
<https://www.fujitsu.com/jp/about/research/article/202111-candar2021-nlp.html>,
- 2) “複雑化・強大化する AI 処理を高速化する Content-Aware Computing 技術とは”, 2021 年 4 月  
<https://www.fujitsu.com/jp/about/resources/publications/technicalreview/topics/article009.html>,
- 3) “計算精度を自動調整する「Content-Aware Computing」により、AI 処理を 10 倍高速化, 2020 年 2 月  
<https://www.fujitsu.com/jp/about/research/article/202002-cac.html>
- 4) “計算の厳密性を自動調整し AI 処理を最大 10 倍高速化するコンピューティング技術を開発”, 2019 年 10 月  
<https://pr.fujitsu.com/jp/news/2019/10/25-1.html>

## **Publications in texts and books**

- 1) 坂井 靖文, “深層学習のモデル軽量化技術”, Software Design 2022 年 3 月号, 技術評論社
- 2) 坂井 靖文, “深層学習の高性能化と高速化・軽量化技術の関係”, Software Design 2021 年 10 月号, 技術評論社

## **Publications in Open-Source Software**

- 1) Y. Sakai, [https://github.com/FujitsuResearch/automatic\\_pruning](https://github.com/FujitsuResearch/automatic_pruning)
- 2) Y. Sakai, <https://github.com/FujitsuLaboratories/CAC/tree/main/cac/pruning>



## Acknowledgement

I would like to express my greatest appreciation to Professor Hiroshi Kawaguchi of Kobe University for giving me insightful suggestions, invaluable comments, and grateful encouragement to prepare this dissertation. I would like to show my deepest gratitude to Visiting Professor Atsuki Inoue of Kobe University for providing me constructive comments and warm encouragement related to this research. My deepest appreciation goes to them.

I would like to thank my colleagues of the quantization and pruning project: Dr. Hisakatsu Yamaguchi, Mr. Yutaka Tamiya, Mr. Tetsutaro Hashimoto, Mr. Yukihiro Kawabe, Dr. Tsuguchika Tabaru, Mr. Akinori Iwakawa, Dr. Akihiko Kasagi, Mr. Satoshi Tanabe, Mr. Haderbach Amir, Mr. Satoki Tsuji, Ms. Fuyuka Yamada, Mr. Kazuki Okado, and Mr. Kengo Matsumoto.

Furthermore, I really would like to appreciate in SSL and active learning project: Mr. Yudai Sato, Mr. Yuki Shinto, and Mr. Hiroki Yaemori, Mr. Shigeki Fukuta, Dr. Koichi Shirahara, Dr. Yuta Yoshimoto, Dr. Naoki Matsumura, Dr. Dang Thang, Prof. Atsushi Ishikawa, Dr. Árni Björn Höskuldsson, and Prof. Egill Skúlason.

I would like to express my sincere gratitude to Dr. Kohta Nakashima, Dr. Yasumoto Tomita, Dr. Eiji Yoshida, Dr. Naoki Akaboshi, and Ms. Hada Naoko for their support in obtaining my doctoral degree through the company's doctoral program.

Finally, I would like to express my heartfelt gratitude to my family, Aki Sakai, Yasuaki Sakai, and Asaya Sakai, for giving me their persistent help, moral support, and considerable encouragement.

Yasufumi Sakai