Babeş-Bolyai University
Faculty of Mathematics and Computer Science

Optimization of Convolutional Neural Networks by Pruning and Architecture Design

Summary of the PhD Thesis

Author:
Csanád Sándor

Supervisor:
Prof. Dr. Horia F. Pop

2024

Keywords: structured pruning, architecture design, convolutional neural networks, filter importance, hierarchical convolutional networks

Contents

Thesis Table of Contents

List of publications

All rankings are listed according to the classification of conferences[1] in Computer Science.

- Category A:
  - **Csanád Sándor**, Szabolcs Pável and Lehel Csató. Pruning cnn's with linear filter ensembles. In Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020), volume 325, series: Frontiers in Artificial Intelligence and Applications, pages 1435–1442. IOS Press, 2020.
- Category B:
  - **Csanád Sándor**, Szabolcs Pável and Lehel Csató. Neural network pruning based on filter importance values approximated with monte carlo gradient estimation. In Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - volume 5, pages 315–322. INSTICC, SciTePress, 2022.
  - Levente Tempfli and **Csanád Sándor**. HierNet: Image Recognition with Hierarchical Convolutional Networks. In Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024), volume 2, pages 147-155, SciTePress, 2024.
- Category D:
  - **Csanád Sándor**. Finding dense supermasks in randomly initialized neural networks. In Proceedings of the 11th International Conference on Applied Informatics (ICAI 2020), volume 2650, pages 288–295. Ceur Workshop Proceedings, 2020.

Publications score: **17** points.

[1] `https://www.core.edu.au/`

Chapter 1

Introduction

Over the past decade, deep neural networks have revolutionized many areas of computer science, including computer vision, speech recognition, machine translation and natural language processing. By applying deep networks, we have gained the ability to address a wide variety of highly complex real-world problems.

As deep neural networks can solve more and more problems, there is an increasing demand to deploy them not only on servers with "unlimited" computational and energy resources (with powerful GPUs and tens or hundreds of gigabytes of RAM), but also on various energy and computational constrained edge devices such as smartphones, watches, various embedded and IoT systems. To meet this demand, continuous development is essential, not only on the hardware side, but also on the software side. On the hardware side, specialized domain-specific architectures are being introduced that are capable of efficiently executing the core operations found in deep neural networks. At the same time, on the software side, various compression algorithms, such as quantization and network pruning, are being developed to reduce the size of the network, thus minimizing memory access and floating-point or fixed-point operations during inference.

Among these compression techniques, the primary focus of this thesis is on network pruning which involves the identification and removal of redundant parameters from the network (see Figure 1.1a). The process of pruning addresses the following issues when deploying neural networks on devices with limited computational resources:

- **Memory requirements**: During inference, network parameters are stored in memory, which is often limited on various embedded devices. A smaller network has fewer parameters, which requires less memory. In addition, structured pruning removes entire filters from the network, which means fewer feature maps during inference. Reducing the number of feature maps generated by a layer can significantly reduce the memory requirements of the

network.

- **Memory bandwidth**: Network parameters are read from memory, multiplied by the layer's input, and the results are written back to memory. If the network contains fewer parameters, fewer memory reads are required. Since memory access is generally an expensive operation, this can speed up the inference process significantly. In addition, as mentioned earlier, structured pruning removes entire channels from the network, which means that additional memory accesses are eliminated since these feature maps should not be accessed.

- **Computational complexity**: Removing parameters from the network results in fewer multiply-add operations, which can speed up inference.

- **Power consumption**: As mentioned before, a smaller network requires less memory access and fewer floating point operations. Of these, memory access in particular is a power hungry operation (in comparison to 32-bit multiplications, there is a nearly two-order-of-magnitude discrepancy. [19]) and it is important to reduce as much as possible.

- **Privacy**: If networks are small enough to fit into the device's memory, the various user data can be processed locally without sending it to the cloud. This can be crucial in different applications where the network has to process sensitive data.

In addition to reducing network size through parameter reduction or bit-width reduction, alternative approaches aim to improve network performance by *optimizing the architecture* or a specific layer within the architecture for greater efficiency [82, 61, 6]. For example, the field of compact architecture design focuses on creating highly efficient architectures. This is achieved by replacing the conventional convolution operation with more efficient alternatives, such as using depthwise separable convolution [6, 61]. This alternative operation can extract an equivalent number of feature maps while using fewer parameters and fewer floating point operations. Other approaches attempt to redesign the conventional single-branch structure of networks by introducing additional branches [80] (see Figure 1.1b) that perform operations depending on the network input. Although this approach typically results in larger networks with more parameters and more floating-point operations, requiring longer training times, only a single (or a few) branch(es) needs to be evaluated during inference. As a result, the network requires a comparable amount of memory and computational resources as a traditional single-branch network, but has an enhanced feature extraction capability that contributes to more accurate predictions. In this thesis, we also explore the

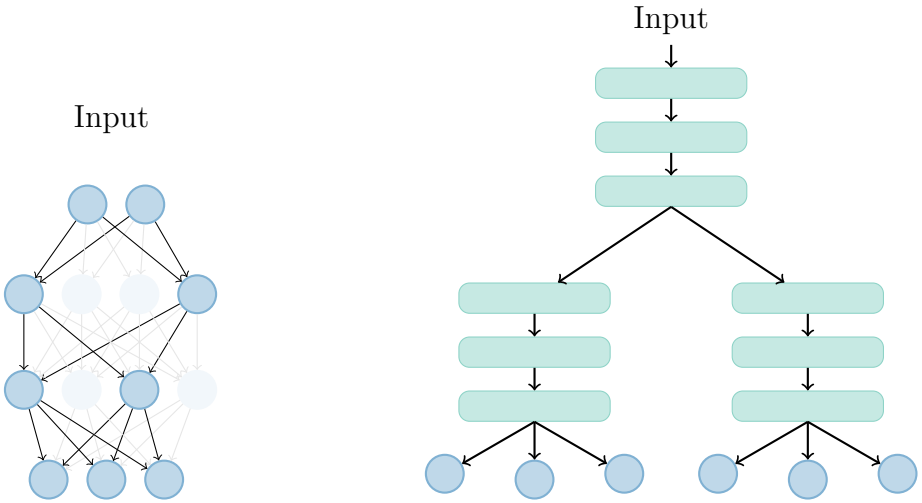


Figure 1.1: (a): Example of a pruned network, where neurons and their connection are removed; (b): Example of a convolutional neural network with branching.

concept of hierarchical networks, where the network is organized as a quasi-decision tree, with the edges representing the feature-extracting layers and the nodes representing the classifiers.

1.1 Contributions to the field

The contribution of this thesis to the field of deep learning is threefold:

1. Two approaches are proposed to estimate the filter or neuron importance values in convolutional as well as in fully connected layers. Based on these importance values, the least important filters are removed from the network, resulting in a smaller network with a minimal loss of accuracy:

    - The first approach is based on the linear filter ensembles (LFE) method, which estimates the importance of filters and iteratively removes the least important ones. See Section 2.1 and [63].
    - The second approach is based on a method that constructs a probability distribution over the presence or absence of network filters. The probabilities associated with filters are inferred by optimizing various energy functions using the log-derivative trick and Monte Carlo gradient estimation. See Section 2.2 and [64].

- Experiments are conducted with the pruning methods on the ResNet architecture trained using the CIFAR-10 dataset. The results demonstrate that the methods can eliminate approximately $30 - 70\%$ of the parameters, depending on the model size. This level of parameter reduction is comparable to that achieved by state-of-the-art methods [25, 47, 24, 45] employing structured pruning techniques (Section 2.3).

2. It is shown that random subnetworks with high accuracy are present in randomly initialized networks not just in the sparse form, but also in the dense form (see Chapter 3 and [62]):

   - By applying the developed structured pruning methods, it is shown that randomly initialized networks contain *dense subnetworks* with an accuracy far from chance: the subnetwork of the *randomly initialized* LeNet architecture achieves more than 50% accuracy on the MNIST dataset.
   - It is also shown that an untrained, wide LeNet network has a subnetwork with 80% accuracy on the MNIST dataset.

3. A novel convolutional neural network architecture is presented, constructed as a quasi-decision tree to exploit the hierarchy between the classes (see Chapter 4 and [77]):

   - A tree-like architecture is introduced, where the edges represent the feature-extracting layers and the nodes represent the classifiers. The structure of the decision tree corresponds to the hierarchical relationships between the label classes, meaning that the visually similar classes are located in the same subtree.
   - A semi-supervised method is presented for determining the hierarchical relationships between a large number of classes.
   - It is shown that this method outperforms the accuracy of the ResNet architecture by $1-3\%$, demonstrating the effectiveness of incorporating input hierarchy into CNNs.

Chapter 2

Structured Pruning

Modern neural networks, despite their efficacy, often suffer from redundancy and inefficiency due to the vast number of parameters they employ [9]. Structured pruning addresses these issues by selectively removing entire filters or neurons, as opposed to individual weights (unstructured pruning), thereby maintaining the regular structure of the network and ensuring compatibility with existing deep learning hardware and software [42, 36].

Pruning not only reduces the model size and computational costs but also often leads to better generalization by eliminating overfitting [19]. In this chapter, we propose two advanced methods for structured pruning: Linear Filter Ensembles (LFE) [63] and Probabilistic Gradient-Based Pruning (PGBP) [64].

2.1 Linear Filter Ensembles

The LFE method estimates the importance of filters in a convolutional neural network (CNN) by considering the impact on network performance when multiple filters are simultaneously disabled. The method uses a linear model to predict the importance of filters based on their performance in different filter ensembles. Filters are ranked based on their importance, and the least important ones are pruned.

2.1.1 Proposed Method

Given a dataset $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ and a convolutional network $f(\boldsymbol{x}|\mathcal{W})$ with input $\boldsymbol{x}$ and parameters $\mathcal{W} = \{W^1, \ldots, W^L\}$ in the layers $\{1, 2, \ldots, L\}$, the empirical loss is:

$$\mathcal{L}(f(\cdot|\mathcal{W}), \mathcal{D}_{train}) = \frac{1}{N} \sum_{i=1}^{N} C(f(\boldsymbol{x}^{(i)}; \mathcal{W}), y^{(i)}), \tag{2.1}$$

where $C(\cdot, \cdot)$ is the error function, such as cross-entropy loss, N is the number of samples in training set $\mathcal{D}_{train}$ and $y^{(i)}$ is the ground truth label for the input sample $\boldsymbol{x}^{(i)}$.

We introduce binary mask vectors $\mathbf{z} \in \{0,1\}^{N_l}$ for each layer l, where N_l represents the number of filters or neurons in that layer. These vectors indicate which units are active. The network's performance with a mask $\mathbf{z}^{(i)}$ is scored as:

$$s^{(i)} = 1 - \frac{\mathcal{L}_i - \mathcal{L}_{\min}}{\mathcal{L}_{\max} - \mathcal{L}_{\min}}, \tag{2.2}$$

where $\mathcal{L}_i$ is the loss when $\mathbf{z}^{(i)}$ mask is applied to the network, $\mathcal{L}_{\min}$ and $\mathcal{L}_{\max}$ are the minimum and maximum losses observed with the different masks, respectively.

The importance of each filter is computed by solving the equation

$$\mathbf{Z} \cdot \boldsymbol{\theta} = \boldsymbol{s}, \tag{2.3}$$

where $\mathbf{Z} = [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}]^T$ is the matrix of binary masks (each row of the matrix is a $\mathbf{z}^{(i)}$ mask vector), $\boldsymbol{s} = [s^{(1)}, \ldots, s^{(M)}]^T$ is a column vector of scores, and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{N_l}]^T$ is a column vector of filter importance values.

The pruning process involves removing the least important filters based on their computed importance values. Starting from either the first or the last layer, we iteratively prune filters and retrain the network until a predefined stopping criterion is met.

Given the importance values $\boldsymbol{\theta}$, we sort the filters and determine an optimal pruning threshold by evaluating the network accuracy on a validation set $\mathcal{D}_{val}$. The threshold is tuned to achieve a balance between pruning effectiveness and accuracy preservation.

After pruning, the network is fine-tuned on the training set $\mathcal{D}_{train}$ to recover any loss in performance due to the removal of filters.

2.1.2 Experiments with LFE

Experiments were conducted on a synthetic XOR dataset and the CIFAR-10 dataset using ResNet architectures. For the XOR dataset, a multilayer perceptron with 10 neurons in the hidden layer was pruned. The success rate of pruning was measured by how often the optimal network structure (3 neurons in the hidden layer) was achieved. For the CIFAR-10 dataset, ResNet models of varying depths (20, 32, 56, 110) were pruned. The results of these experiments are presented in Section 2.3.

2.2 Probabilistic Gradient-Based Pruning

The LFE method assumes that structural units (neurons or filters) within the same layer are independent of each other. While this assumption may hold for a single layer, it does not hold when evaluating the importance of these units across multiple layers.

Pruning the network layer by layer can partially address this issue, but it cannot fully capture the true influence of the units on each other. To address this, we introduced a more generalized method. In this approach, a probability distribution over the network units is learned with the goal of decreasing the empirical loss (or increasing the empirical score).

While a simple Bernoulli distribution was used in this work, targeting a single layer at a time, the optimization method based on Monte Carlo gradient estimation is flexible enough to allow the use of more complex probability models that can model cross-layer dependencies. Extending this approach with a more complex probability distribution model will allow modeling connections between layers, thereby estimating the importance of units not only within a single layer, but also across multiple successive layers.

2.2.1 Proposed Method

We define $\mathbf{z}$ as a vector of *binary* random variables, where each z_i random variable indicates whether the associated network unit is active or not. z_i is treated as a random variable with a Bernoulli distribution parameterized by θ_i:

$$P(z_i = 1) = p_i = \sigma(\theta_i), \tag{2.4}$$

where σ is the sigmoid function.

The objective is to optimize the joint probability distribution $P_{\boldsymbol{\theta}}(\mathbf{z})$ of $\mathbf{z}$ (see Equation 2.12), parameterized by $\boldsymbol{\theta}$, to maximize the expected network score:

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \underset{\boldsymbol{z} \sim P_{\boldsymbol{\theta}}(\mathbf{z})}{\mathbb{E}} [s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))] = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, S(P_{\boldsymbol{\theta}}), \tag{2.5}$$

where $\mu_{\mathcal{W}}(\boldsymbol{x})$ is the network, parameterized by $\mathcal{W}$, $\boldsymbol{x}$ is the input image, and $s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))$ is the network score when $\boldsymbol{z}$ mask applied on it.

We use gradient ascent to optimize the parameters of the probability distribution $P_{\boldsymbol{\theta}}(\mathbf{z})$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}_k}. \tag{2.6}$$

Given the infeasibility of evaluating all $2^{|\mathcal{W}|}$ mask combinations to calculate $\nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}_k}$, we use Monte Carlo gradient estimation with the log-derivative trick to approximate the gradient. Namely, from the log-derivative trick we have that:

$$\nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\boldsymbol{z}) = P_{\boldsymbol{\theta}}(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}). \tag{2.7}$$

Using this, the gradient of the expected score can be reformulated as:

$$\begin{aligned} \nabla_{\theta} S(P_{\boldsymbol{\theta}}) &= \int_{\boldsymbol{z}} \nabla_{\theta} P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z})) d\boldsymbol{z} && (2.8) \\ &= \int_{\boldsymbol{z}} \nabla_{\theta} P_{\boldsymbol{\theta}}(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z})) d\boldsymbol{z} && (2.9) \\ &= \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_{\boldsymbol{\theta}}(\mathbf{z})} [\nabla \log P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))], && (2.10) \end{aligned}$$

Next, we approximate the gradient using Monte Carlo estimation with N samples:

$$\nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}}) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}^{(i)})). \tag{2.11}$$

Variance reduction techniques, such as subtracting the moving average of the score and normalizing by the score variance, enhance convergence. Various score functions (loss-score, acc-score, exp-acc-score) are tested for effectiveness, adjusting the score calculations to improve convergence.

The probability distribution $P_{\boldsymbol{\theta}}(\mathbf{z})$ is modeled as a product of Bernoulli distributions, assuming independence between units:

$$P_{\boldsymbol{\theta}}(\boldsymbol{z}) = \prod_i p_i^{z_i} (1 - p_i)^{1 - z_i}, \tag{2.12}$$

where $p_i = \sigma(\theta_i)$ is the probability of $z_i = 1$.

The network pruning algorithm iteratively optimizes the probability distribution for each layer, prunes the layer based on $P_{\boldsymbol{\theta}}(\mathbf{z})$, and fine-tunes the network. This layer-by-layer pruning accounts for interdependencies within layers, enhancing the efficiency and effectiveness of the pruning process.

2.2.2 Experiments with PGBP

PGBP was tested on a VGG-like architecture and ResNet architectures trained on CIFAR-10. For the VGG-like network, random filters were added to the layers, and the algorithm's ability to prune these filters was evaluated. In the case of ResNet architectures, the method was applied to ResNet-32, 56, and 110. The results showed that PGBP could effectively prune filters while maintaining high accuracy (see Section 2.3).

2.3 Structured Pruning Results

Both LFE and PGBP were compared to other pruning methods from the literature. The comparisons were based on the reduction in parameters and FLOPs and the corresponding impact on accuracy.

Table 2.1 summarizes the results for both methods applied to ResNet architectures and compares them with other state-of-the-art methods.

ResNet	Method	Accuracy (%)			↓(%)	
		Baseline	Pruned	Diff. ↓	FLOPs	Params.
32	SFP [25]	92.63	92.08	0.55	41.5	41.24
	FPGM [24]	92.63	92.82	-0.19	**53.2**	**53.2***
	LFE	92.97	92.42	0.55	46.4	49.35
	PGBP	92.97	92.29	0.68	50.22	43.65
56	PFEC[42]	93.04	93.06	-0.02	27.6	13.7
	SFP [25]	93.59	93.35	0.1	47.14	52.6
	ThiNet [47]	93.8	92.98	0.82	49.78	49.67
	FPGM [24]	93.59	93.49	0.1	47.14	52.6
	LFE	93.44	93.18	0.26	57.64	**68.14**
	Adapt-DCP [45]	93.74	93.77	-0.03	**68.48**	54.80
	PGBP	93.44	93.08	0.36	64.22	57.79
110	PFEC[42]	93.53	93.3	0.23	38.6	32.40
	SFP [25]	93.68	93.86	-0.18	40.8	40.72*
	FPGM [24]	93.68	93.85	-0.17	52.3	52.7*
	LFE	94.05	93.48	0.57	63.68	60.08
	PGBP	94.05	93.45	0.6	**72.53**	**68.89**

Table 2.1: Comparison of the pruned ResNet architecture (trained on the CIFAR-10 dataset) with results from the literature.

2.4 Conclusion

This chapter introduced two structured pruning methods, LFE and PGBP, and demonstrated their effectiveness on CIFAR-10 with ResNet architectures. Both methods showed comparable results to state-of-the-art techniques, with PGBP achieving particularly high compression rates. Future work includes exploring these methods on larger datasets and more complex architectures to further validate their efficacy and robustness.

Chapter 3

Dense Supermasks in Randomly Initialized Neural Networks

This chapter shows the concept of *dense supermasks* within randomly initialized, untrained neural networks, building on the foundational work presented in [62].

Previous research has established the presence of sparse subnetworks within randomly initialized networks that perform comparably to trained networks [83, 57]. Inspired by these findings, we aim to explore dense subnetworks, termed supermasks, which demonstrate significantly higher accuracy than chance levels in untrained networks.

3.1 Pruning Methods

We employed several pruning methods to identify these dense subnetworks:

1. L_1 **and** L_2 **Norms**: These norms measure the importance of neurons based on their parameter values. Neurons with smaller norms are considered less important and are pruned first.

2. **Linear Filter Ensembles (LFE)**: This method evaluates the importance of neurons by analyzing the network loss when different filter ensembles are applied.

3. **Probabilistic Gradient-Based Pruning (PGBP)**: This approach maximizes the expected score of the network by sampling masks from a probability distribution and refining these probabilities using Monte Carlo gradient estimation.

4. As a control case we also consider **random pruning**, where neurons are pruned randomly.

3.2 Experimental Setup

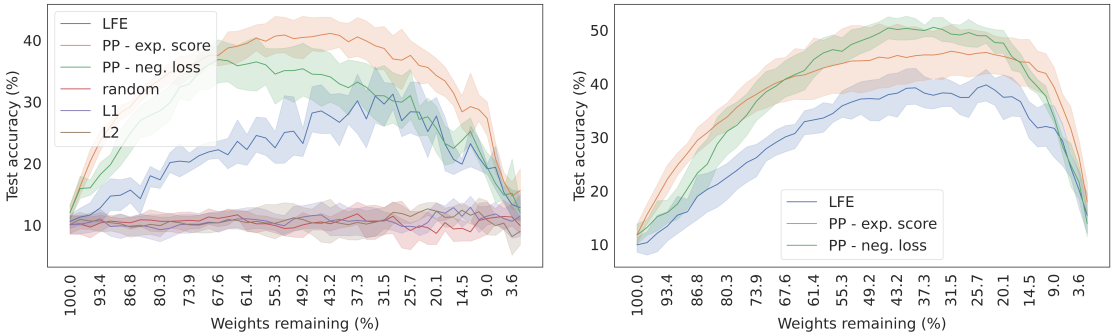
We conducted experiments on the MNIST dataset using the LeNet-300-100 architecture. This network comprises an input layer with 784 units, followed by two hidden layers with 300 and 100 units, respectively, and an output layer with 10 units. The network parameters were initialized from a normal distribution, and no training was applied. Pruning was performed in both one-shot and iterative manners, progressively reducing the network's size by removing the least important neurons.

We also experiment with the Wide-LeNet architecture, an expanded version of the traditional LeNet model, featuring two fully connected hidden layers with 3010 and 1010 neurons, respectively. This increased width enhances the probability of identifying high-performing subnetworks within a larger parameter space.

3.3 Results

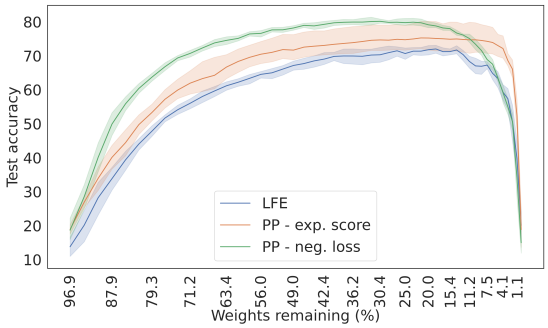
Figure 3.1 shows the accuracy of the pruned networks as more parameters are removed. The results are summarized as follows:

1. **LeNet-300-100 Architecture**:
    - **Random Pruning**: Accuracy remained at 10%, consistent with random chance.
    - L_1 **and** L_2 **Pruning**: No significant accuracy improvement was observed.
    - **LFE Pruning**: Achieved a maximum accuracy of 34.2% with 70% of the parameters removed.
    - **Probabilistic Pruning**: With the negative loss score, accuracy reached 36.86% after removing 33% of the parameters, and 41.08% with the exponential score function after 67% parameter removal.
2. **Iterative Pruning**:
    - **LFE Method**: Reached a maximum accuracy of 39.8%.
    - **Probabilistic Pruning**: Achieved an accuracy of 46.82% with the exponential score function and 50.52% with the negative loss score function.
3. **Wide-LeNet Architecture**:



(a) LeNet, one-shot pruning accuracy

(b) LeNet, iterative pruning accuracy



(c) Wide-LeNet, iterative pruning accuracy

Figure 3.1: Randomly initialized (untrained), pruned LeNet-300-100 and Wide-LeNet evaluated on the MNIST dataset. Change in accuracy as more parameters are removed from the models. Figure (a) shows the results of the one-shot pruning experiment, Figure (b) shows the results of the iterative pruning experiment, and Figure (c) shows the results of the Wide-LeNet iterative pruning experiment.

- **Iterative Pruning**: The network, initially with 3010 and 1010 neurons in its first and second hidden layers, achieved over 80% accuracy after 70% of the parameters were pruned using the probabilistic method with the negative loss score function. Even with 10% of its original size, the network maintained an accuracy of 73.3%.

3.4 Conclusions

Our experiments confirmed that dense subnetworks with significantly higher accuracy than random chance exist within randomly initialized, untrained neural networks. Using linear filter ensembles and probabilistic pruning methods, we demonstrated the presence of these supermasks in both the LeNet-300-100 and

Wide-LeNet architectures. This discovery challenges the traditional view of neural network initialization and training, suggesting that the potential for high-performance subnetworks exists inherently within the initial random parameters.

These findings provide new avenues for the efficient design of networks, where the focus may shift from extensive training to the identification of inherent subnetworks that already possess high accuracy.

Chapter 4

HierNet: Hierarchical Convolutional Networks

Traditional CNNs follow a sequential construction, stacking convolutional layers from the input layer to fully connected layers. This setup allows the network to capture fine-grained details in the initial layers and progressively higher-level features in later stages, providing a comprehensive visual representation of an object. However, these models treat all classes equally, neglecting the inherent hierarchy among data classes. For example, some classes share visual similarities, like dogs and cats, or flowers that look more similar to each other than to animals.

To leverage this inherent class hierarchy, we introduce *HierNet*, a hierarchical CNN architecture resembling a decision tree (see Figure 4.1). In HierNet, edges represent convolutional operations for feature extraction, while nodes perform classification tasks to determine the next route based on extracted features. Final class predictions are generated by the leaf nodes. The entire tree is trained, but during inference, only a single path from the root to a leaf node is evaluated.

4.1 HierNet Architecture and Training

The architecture consists of nodes (V) and edges (E), forming a tree with a unique root node. Each edge has feature-extracting operations, and nodes contain classifier functions to route samples to the appropriate child node. The root node forwards the input image to the first edge. Classifier functions at the nodes include flattening or pooling, linear transformations, and softmax operations.

HierNet utilizes a sequence of operations similar to standard CNNs, such as ResNet, as the backbone. As Figure 4.2 shows, the backbone's layers are distributed among the edges from root to leaf, maintaining the same order as in the original model. The primary difference is that HierNet has classifiers in every node

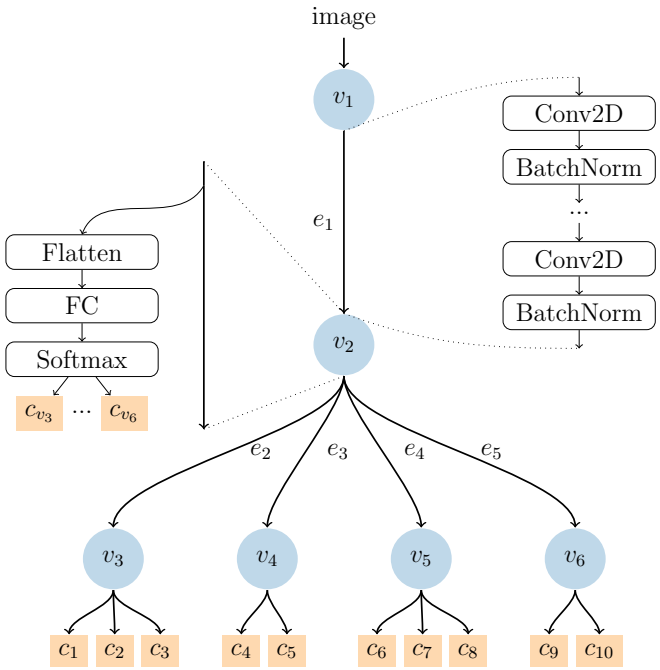


Figure 4.1: HierNet architecture with a tree topology, feature extraction operations in the edges, and classification operations in the nodes.

(except the root), unlike the single classification layer in traditional models.

HierNet can be trained end-to-end, with the output probability of each leaf node being the product of its own output and the probabilities of all ancestor nodes. The hierarchical tree's order necessitates reordering dataset labels to match the hierarchy. Categorical cross-entropy loss is used for training, and the training settings mirror those of the backbone model. Metrics include "conditional accuracy" for the concatenated output probabilities and "routing accuracy" for the correct path from root to leaf.

Inference involves evaluating a single path from the root to a leaf node. This process includes selecting the first edge, running feature extraction, feeding the output into the classifier, computing probabilities, and determining the next route until reaching a leaf node.

4.2 Hierarchy Construction and Grouping Algorithm

The hierarchy of classes is represented by the tree's topology. While manual construction is possible for small datasets, we introduce an automated method using a

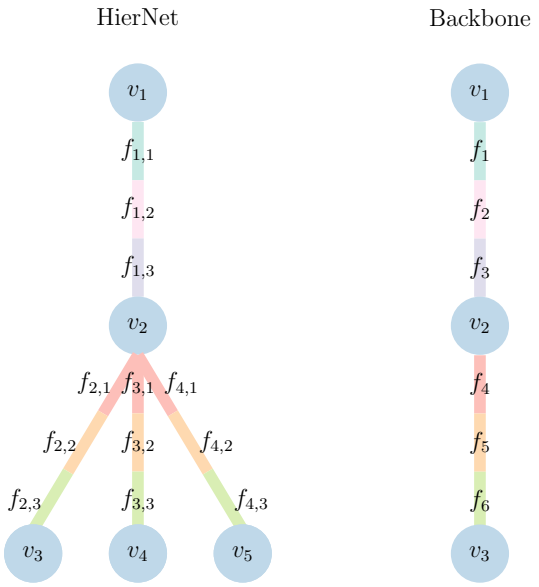


Figure 4.2: The layers in HierNet and the backbone model are highlighted in different colors, indicating specific parameter sets. As shown, HierNet closely mirrors the layers of the backbone model.

confusion probability matrix (CPM) to group visually similar classes. A CPM captures misclassification probabilities among classes, which helps define the proximity between classes. Classes are grouped based on their proximity, with constraints on group size and minimum similarity to ensure meaningful hierarchies.

4.3 Experiments and Results

We experimented with HierNet on the CIFAR-100 dataset, which has 100 classes and is suitable for testing our grouping algorithm. Key hyperparameters include the split point of the backbone CNN, the number of additional classifier layers, minimum proximity of group members, and maximum group size. Optimal values were found through extensive testing. We augmented the dataset with random horizontal flipping and translation, and evaluated the models using accuracy as the primary metric.

The results demonstrate the significant advantage of HierNet over its backbone models (see Table 4.1). HierNet consistently outperforms the corresponding ResNet and ELU ResNet models across all tested network sizes. For instance, the 32-layer HierNet model achieves an accuracy of 70.45%, surpassing the 68.38% accuracy of the 56-layer ResNet. Similarly, the 32-layer HierNet based on the ELU ResNet backbone attains 70.43% accuracy, compared to the 69.03% accuracy of

#layers	ResNet	HierNet	ELU ResNet	HierNet
20	65.96	**68.08**	65.54	**68.16**
32	67.08	**70.45**	67.88	**70.43**
44	68.12	**70.75**	68.79	**70.79**
56	68.38	**72.01**	69.03	**72.29**
110	71.33	**73.27**	72.93	**74.15**

Table 4.1: Comparison of the accuracy of our HierNet and the backbone ResNet and ELU ResNet for different network sizes

the 56-layer ELU ResNet. This highlights HierNet's efficiency and superior performance despite its smaller size.

4.4 Conclusion

This chapter introduced HierNet, a hierarchical CNN that leverages visual similarities and class hierarchies to improve classification accuracy. HierNet outperformed traditional ResNet models, demonstrating the effectiveness of exploiting class hierarchies. Future work includes refining the grouping algorithm to balance class distributions better and optimizing training configurations specific to HierNet.

Chapter 5

Conclusions and future work

This thesis contributes to the field of deep learning by introducing novel structured pruning methods and a new architecture designed to improve the efficiency of convolutional neural networks (CNNs).

To reduce the number of filters and neurons in CNNs, the linear filter ensembles (LFE) method assign importance values to filters in convolutional layers and neurons in fully connected layers by building and evaluating linear filter ensembles. The experiments, conducted on models trained on a small XOR-like dataset as well as on the CIFAR-10 dataset, demonstrated the ability of this method to identify and remove redundant filters from the network. In particular, when the pruning technique was applied to different ResNet architectures, the results obtained were comparable to those obtained by various state-of-the-art methods. The probabilistic gradient-based pruning (PGBP) method was introduced as an alternative approach to estimate the importance of filters and neurons by constructing a probability distribution over the filters using the log-derivative trick and Monte Carlo gradient estimation. The experiments demonstrated the effectiveness of this method in identifying random filters added to pre-trained networks and in pruning the ResNet-110 architecture trained on the CIFAR-10 dataset, removing approximately 70% of the parameters.

Next, the existence of dense subnetworks in randomly initialized, untrained networks was investigated that achieve accuracy far from chance. With the use of the LFE and PGBP methods, it was shown that the randomly initialized LeNet-300-100 architecture contains a subnetwork that achieves 50% accuracy on the MNIST dataset, while pruning the wider version of this architecture finds a subnetwork that achieves 80% accuracy.

Finally, a novel tree-like architecture, HierNet, was introduced that can exploit the hierarchical relationships between classes. In this network, the edges represent the feature extraction layers, while the nodes perform the classifications to first classify images into superclasses and then perform fine-grained classification on the

selected branch. This approach achieved $2-3\%$ more accuracy on the CIFAR-100 dataset compared to the baseline ResNet model, with only a few more floating-point operations required during inference.

Possible future research directions from the pruning side include the development of probability models that can estimate the importance of units not just within a single layer, but also across multiple layers. This is possible, since the PGBP method is flexible enough to allow the use of more complex probability models. Another interesting direction is to experiment with pruning during the training phase, similar to the dropout method, which randomly deactivates neurons during training to increase network robustness and improve generalization. With the use of PGBP, the network could continuously adapt to the evolving structure by selectively activating or deactivating units according to the learned probabilities.

HierNet could be further improved by automating the construction of the model structure. This can be achieved by dynamically constructing and continuously evaluating the HierNet model, beginning with a smaller CNN architecture without intermediate decision nodes and subsequently adding new layers and decision nodes based on the confusion probability matrix. One advantage of this method is that it inherently generates an asymmetric graph based on class similarity. This concentrates the resource-intensive feature extraction operations where they are most needed, resulting in more cost-effective processing.

Bibliography

[1] Kambiz Azarian et al. "Learned Threshold Pruning". In: *CoRR* abs/2003 00075 (2020). arXiv: 200300075.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *CoRR* abs/1308.3432 (2013). arXiv: 1308.3432. URL: http://arxiv.org/abs/1308.3432.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[4] Davis Blalock et al. "What is the State of Neural Network Pruning?" In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 129–146. URL: https://proceedings.mlsys.org/paper/2020/file/d2ddea18f00665ce8623e36bd4e3c7c5-Paper.pdf.

[5] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[6] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[7] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289* (2015).

[8] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[9] Misha Denil et al. "Predicting Parameters in Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf.

[10] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[11] Xiaohan Ding et al. "Repvgg: Making vgg-style convnets great again". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13733–13742.

[12] Alexander Finkelstein, Uri Almog, and Mark Grobman. *Fighting Quantization Bias With Bias*. 2019. arXiv: 1906.03193 [cs.LG].

[13] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *ICLR'2019*. 2019. URL: https://openreview.net/forum?id=rJl-b3RcF7.

[14] Jonathan Frankle et al. *Stabilizing the Lottery Ticket Hypothesis*. 2020. arXiv: 1903.01611 [cs.LG].

[15] Amir Gholami et al. "A Survey of Quantization Methods for Efficient Neural Network Inference". In: *CoRR* abs/2103.13630 (2021). arXiv: 2103.13630. URL: https://arxiv.org/abs/2103.13630.

[16] Aidan N. Gomez et al. "Learning Sparse Networks Using Targeted Dropout". In: *CoRR* abs/1905.13678 (2019). arXiv: 1905.13678. URL: http://arxiv.org/abs/1905.13678.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[18] Shixiang Shane Gu et al. "MuProp: Unbiased Backpropagation for Stochastic Neural Networks". In: *CoRR* abs/1511.05176 (2016).

[19] Song Han, Huizi Mao, and William J. Dally. "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding". In: *CoRR* abs/1510.00149 (2015). arXiv: 1510.00149. URL: http://arxiv.org/abs/1510.00149.

[20] Song Han et al. "Learning Both Weights and Connections for Efficient Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1135–1143.

[21] Babak Hassibi et al. "Optimal Brain Surgeon: Extensions and Performance Comparisons". In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. NIPS'93. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1993, pp. 263–270. URL: http://dl.acm.org/citation.cfm?id=2987189.2987223.

[22] Kaiming He and Jian Sun. "Convolutional neural networks at constrained time cost". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 5353–5360.

[23] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[24] Yang He et al. "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[25] Yang He et al. "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 2234–2240. DOI: 10.24963/ijcai.2018/309. URL: https://doi.org/10.24963/ijcai.2018/309.

[26] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. 5th ed. Amsterdam: Morgan Kaufmann, 2012. ISBN: 978-0-12-383872-8.

[27] Torsten Hoefler et al. "Sparsity in Deep Learning: Pruning and Growth for Efficient Inference and Training in Neural Networks". In: *J. Mach. Learn. Res.* 22.1 (Jan. 2021). ISSN: 1532-4435.

[28] Hengyuan Hu et al. *Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures*. 2016. arXiv: 1607.03250 [cs.NE].

[29] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[30] Ruyi Ji et al. "Attention convolutional binary neural tree for fine-grained visual categorization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10468–10477.

[31] John M. Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589.

[32] Khronos OpenCL Working Group. *The OpenCL Specification, Version 1.1*. Ed. by Aaftab Munshi. 2011. URL: https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf.

[33] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *ICLR'2015*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[34] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. Faculty of Computer Science, University of Toronto, 2009. URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105.

[36] Aditya Kusupati et al. "Soft Threshold Weight Reparameterization for Learnable Sparsity". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[37] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]* 2 (2010).

[38] Yann LeCun, John S. Denker, and Sara A. Solla. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1990, pp. 598–605. URL: http://papers.nips.cc/paper/250-optimal-brain-damage.pdf.

[39] Yann Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*. 1998, pp. 2278–2324.

[40] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. "Snip: single-Shot Network Pruning based on Connection sensitivity". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=B1VZqjAcYX.

[41] Bowen Li et al. "DFQF: Data Free Quantization-aware Fine-tuning". In: *Proceedings of The 12th Asian Conference on Machine Learning*. Ed. by Sinno Jialin Pan and Masashi Sugiyama. Vol. 129. Proceedings of Machine Learning Research. PMLR, Nov. 2020, pp. 289–304. URL: https://proceedings.mlr.press/v129/li20a.html.

[42] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=rJqFGTslg.

[43] Yuhang Li et al. "{BRECQ}: Pushing the Limit of Post-Training Quantization by Block Reconstruction". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=POWv6hDd9XH.

[44] Ji Lin et al. "Runtime Neural Pruning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/a51fb975227d6640e4fe47854476d133-Paper.pdf.

[45] Jing Liu et al. "Discrimination-aware Network Pruning for Deep Model Compression". In: *TPAMI'2021* PP (2021), (early access). DOI: 10.1109/TPAMI.2021.3066410.

[46] Christos Louizos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through L0 Regularization". In: *ArXiv* abs/1712.01312 (2017).

[47] J. Luo, J. Wu, and W. Lin. "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Vol. 00. Oct. 2018, pp. 5068–5076. DOI: 10.1109/ICCV.2017.541. URL: doi.ieeecomputersociety.org/10.1109/ICCV.2017.541.

[48] Jian-Hao Luo and Jianxin Wu. "AutoPruner: An End-to-End Trainable Filter Pruning Method for Efficient Deep Model Inference". In: *CoRR* abs/1805 08941 (2018). arXiv: 180508941.

[49] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[50] Eldad Meller et al. "Same, Same But Different: Recovering Neural Network Quantization Error Through Weight Factorization". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 4486–4495.

[51] Pavlo Molchanov et al. "Pruning Convolutional Neural Networks for Resource Efficient Inference". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJGCiw5gl.

[52] MICHAEL C. MOZER and PAUL SMOLENSKY. "Using Relevance to Reduce Network Size Automatically". In: *Connection Science* 1.1 (1989), pp. 3–16. DOI: 10.1080/09540098908915626. eprint: https://doi.org/10.1080/09540098908915626. URL: https://doi.org/10.1080/09540098908915626.

[53] Markus Nagel et al. "A White Paper on Neural Network Quantization". In: *ArXiv* abs/2106.08295 (2021).

[54] Markus Nagel et al. "Up or Down? Adaptive Rounding for Post-Training Quantization". In: *CoRR* abs/2004.10568 (2020). arXiv: 2004.10568. URL: https://arxiv.org/abs/2004.10568.

[55] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide*. Version 3.2. 2010.

[56] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[57] Vivek Ramanujan et al. *What's Hidden in a Randomly Weighted Neural Network?* 2019. arXiv: 1911.13299 [cs.CV].

[58] Minsoo Rhu et al. "Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks". In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2018, pp. 78–91. DOI: 10.1109/HPCA.2018.00017.

[59] Minsoo Rhu et al. "Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks". In: *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2018, pp. 78–91. DOI: 10.1109/HPCA.2018.00017.

[60] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer Publishing Company, Incorporated, 2010. ISBN: 1441919392.

[61] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[62] Csanád Sándor. "Finding Dense Supermasks in Randomly Initialized Neural Networks". In: *Proceedings of the 11th International Conference on Applied Informatics (ICAI)* (Eger, Hungary, Jan. 29–31, 2020). Ed. by István Fazekas, Gergely Kovásznai, and Tibor Tómács. CEUR Workshop Proceedings 2650. Aachen, 2020, pp. 288–295. URL: http://ceur-ws.org/Vol-2650/#paper30.

[63] Csanád Sándor, Szabolcs Pável, and Lehel Csató. "Pruning CNN's with Linear Filter Ensembles". In: *ECAI 2020 - 24th European Conference on Artificial Intelligence*. 2020, pp. 1435–1442. DOI: 10.3233/FAIA200249. URL: https://doi.org/10.3233/FAIA200249.

[64] Csanád Sándor., Szabolcs Pável., and Lehel Csató. "Neural Network Pruning based on Filter Importance Values Approximated with Monte Carlo Gradient Estimation". In: *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*. INSTICC. SciTePress, 2022, pp. 315–322. ISBN: 978-989-758-555-5. DOI: 10.5220/0010786700003124.

[65] Victor Sanh, Thomas Wolf, and Alexander Rush. "Movement Pruning: Adaptive Sparsity by Fine-Tuning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 20378–20389. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/eae15aabaa768ae4a5993a8a4f4fa6e4-Paper.pdf.

[66] Shibani Santurkar et al. "How Does Batch Normalization Help Optimization?" In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[67] Pedro Savarese, Hugo Silva, and Michael Maire. *Winning the Lottery with Continuous Sparsification*. 2020. URL: https://openreview.net/forum?id=BJe4oxHYPB.

[68] John R. Searle. "Minds, Brains, and Programs". In: *Mind Design*. Cambridge, MA, USA: MIT Press, 1985, pp. 282–307. ISBN: 0262580527.

[69] Anish Shah et al. "Deep Residual Networks with Exponential Linear Unit". In: *Proceedings of the Third International Symposium on Computer Vision and the Internet*. VisionNet'16. Jaipur, India: Association for Computing Machinery, 2016, pp. 59–65. ISBN: 9781450343015. DOI: 10.1145/2983402.2983406. URL: https://doi.org/10.1145/2983402.2983406.

[70] Sietsma and Dow. "Neural net pruning-why and how". In: *IEEE 1988 International Conference on Neural Networks*. 1988, 325–333 vol.1. DOI: 10.1109/ICNN.1988.23864.

[71] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529 (2016), pp. 484–503. URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[72] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.1556.

[73] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[74] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. “Highway Networks”. In: *CoRR* abs/1505.00387 (2015). arXiv: 1505.00387. URL: http://arxiv.org/abs/1505.00387.

[75] Chong Min John Tan and Mehul Motani. “DropNet: Reducing Neural Network Complexity via Iterative Pruning”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 9356–9366. URL: https://proceedings.mlr.press/v119/tan20a.html.

[76] Ryutaro Tanno et al. “Adaptive neural trees”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6166–6175.

[77] Levente Tempfli. and Csanád Sándor. “HierNet: Image Recognition with Hierarchical Convolutional Networks”. In: *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*. INSTICC. SciTePress, 2024, pp. 147–155. ISBN: 978-989-758-680-4. DOI: 10.5220/0012321100003636.

[78] Georg Thimm and Emile Fiesler. “Evaluating pruning methods”. In: *1995 International Symposium on Artificial Neural Networks (ISANN’95)*. National Chiao-Tung University, Hsinchu, Taiwan, Republic of China, 1995, A2 20–25. URL: http://infoscience.epfl.ch/record/82305.

[79] Stijn Verdenius, Maarten Stol, and Patrick Forré. “Pruning via Iterative Ranking of Sensitivity Statistics”. In: *CoRR* abs/2006.00896 (2020). arXiv: 2006.00896. URL: https://arxiv.org/abs/2006.00896.

[80] Zhicheng Yan et al. “HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2740–2748. DOI: 10.1109/ICCV.2015.314.

[81] Shuochao Yao et al. "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework". In: *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. SenSys '17. Delft, Netherlands: ACM, 2017, 4:1–4:14. ISBN: 978-1-4503-5459-2. DOI: 10.1145/3131672.3131675. URL: http://doi.acm.org/10.1145/3131672.3131675.

[82] Xiangyu Zhang et al. "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[83] Hattie Zhou et al. "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 3592–3602.

[84] Barret Zoph et al. "Learning Transferable Architectures for Scalable Image Recognition". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 8697–8710.