A Cooperation Index for Model Pruning

Do-Hoon Kim
Hanyang University
kik941002@hanyang.ac.kr

Jay I. Myung
Ohio State University
myung.1@osu.edu

Yung-Kyun Noh[*]

Hanyang University
Korea Institute for Advanced Study
nohyung@hanyang.ac.kr

Abstract

In complex models, tools for measuring parameter importance identify its core functional element and improve both generalizability and interpretability by pruning redundant ones. Effective pruning relies on these tools, which serve as decision-making criteria. The SHAP Value (SV) has recently been considered such a criterion, interpreted as measuring the average marginal contribution across all possible paths of parameter accumulation. However, we find that this averaging process of SV systematically overweights redundant parameters. Instead, we propose that measuring the speed of decay of the marginal contribution can serve as a more effective decision-making criterion. Specifically, we quantify the number of cooperative contribution for each parameter and show that this criterion is more effective for parameter pruning in backward elimination, leading to a more optimal set of remaining parameters.

1 Introduction

Neuroplasticity is one of the fundamental properties of neural networks [9, 13]. As human brain adapts by reinforcing useful connections and compensates for a deficiencies or losses through neuroplasticity, a similar process can benefit artificial neural networks by pruning unnecessary parameters and optimizing the remaining ones [8, 14, 16, 18, 23, 7]. The interactions among parameters, however, are often complex and non-additive, making it challenging to assess the individual contribution of each parameter within a complex model.

One widely adopted and influential tool for measuring feature contribution is the SHAP value (SV) defined as [15, 24, 4, 22, 13, 17]:

$$\phi_i(f) = \frac{1}{n} \sum_{S \subseteq \mathcal{P} \backslash \{i\}} \begin{pmatrix} n-1 \\ |S| \end{pmatrix}^{-1} \left(f(S \cup \{i\}) - f(S) \right). \tag{1}$$

Given a set of features, $\mathcal{P}$, of size n, SV $\phi_i(f)$ represents the weighted average contribution of feature i by considering all possible cases in which feature i is additionally applied to the feature subset $S \subseteq \mathcal{P} \backslash \{i\}$. Here, the term $(f(S \cup \{i\}) - f(S))$ in Eq 1 is dubbed as marginal contribution of feature i. Note that this term is not fixed; instead, it changes depending on the feature subset S. The SV is predominantly used as sensitive, high-risk decision-making agents across broad real-world problems due to its game-theoretically principled nature [15, 24, 4, 22, 17]. However, our main

[*]Corresponding author

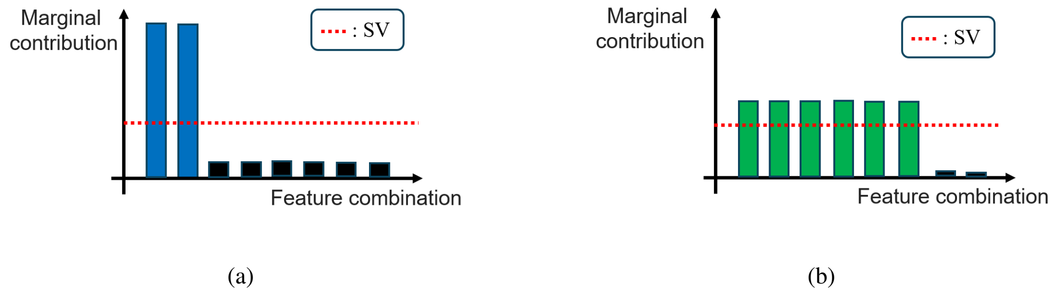


Figure 1: A conceptual illustration of decay patterns of the marginal contribution for two different parameters. Each bar represents a marginal contribution corresponding to different parameter combinations. With black bars indicating insignificant contributions.

observations show that SV's core concept—averaging—overweights redundant parameters and can lead to counter-intuitive decisions.

For the two graphs illustrated in Figure 1, when the marginal contribution of a parameter is represented across various combinations of other parameters, and the contribution values are sorted in descending order from the left to right. The average contributions (SVs mentioned in Eq. 1) of both are same, but the **decay patterns** of the marginal contribution differ significantly. In the left graph Fig. 1a, when the parameter is additionally applied, the decay pattern of the marginal contribution depicts high contributions but only in limited combinations of other parameters, showing the speed of decay is fast. In contrast, the right graph Fig. 1b illustrates that although the parameter's marginal contributions are not exceptionally high, they consistently contribute across numerous combinations of other parameters, showing the speed of decay is slow. Pruning the parameter on the left can be easily offset by others, whereas pruning the one on the right may lead to a significant loss in overall performance. The SV, by averaging contributions, fails to distinguish between these two fundamentally different roles.

These observations point to a critical consideration in model pruning: not all parameters with a similar average contribution play equivalent roles within the networks. Instead, Our motivation is that parameters consistently contributing in conjunction with various combinations of other parameters are unlikely to be replaced by other parameters and must therefore be retained within the networks. Recognizing these difference of the roles is crucial, yet effective methods for leveraging this information remain largely underdeveloped. To address this gap, we propose a simple index that quantifies the speed of decay of the marginal contribution when trained alongside different combinations of other parameters.

2 Marginal contribution and parameter importance

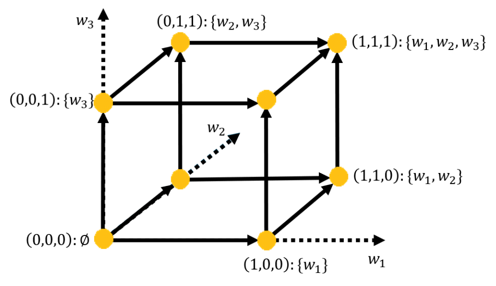
The contribution of a parameter is inherently tied to its interaction with other parameters. Accordingly, we consider the accumulation of contributions by arranging the parameters in a set of parameter $\mathcal{P}$ into all possible permutations. Along the permutation sequence, we sequentially add parameters one at a time. At each step of the sequence, the model consisting of all parameters selected up to that step is jointly optimized on the data.

Let $\Pi(\mathcal{P})$ denote the set of all possible permutations of the parameter indices $i \in \mathcal{P}$ and define S_π^i as the subset of parameters that appear right before i in the ordering $\pi \in \Pi(\mathcal{P})$. We then define $f^*(S_\pi^i)$ as the objective function—such as the log likelihood or the negative loss function—optimized with respect to the data over the parameters in S_π^i.

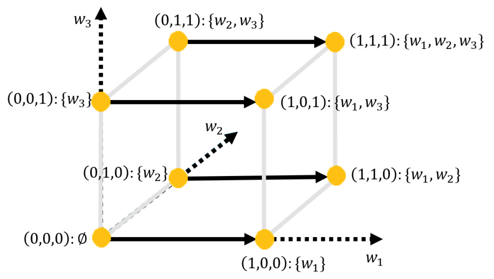
Marginal Contribution. Given the permutation $\pi \in \Pi(\mathcal{P})$, the marginal contribution of parameter $i \in \mathcal{P}$ is defined as:

$$\Delta_{\pi,i} = f^*(S_\pi^i \cup \{i\}) - f^*(S_\pi^i), \tag{2}$$

similar to [19]. The marginal contribution $\Delta_{\pi,i}$ quantifies the performance gain achieved by adding parameter i along that permutation π. The $\Delta_{\pi,i}$ is nonnegative because, with the addition of parameter i, the worst-case performance of $f^*(S_\pi^i \cup \{i\})$ is $f^*(S_\pi^i)$, which occurs when parameter i is not utilized.



(a)



(b)

Figure 2: (a) A three-dimensional cube representing different edge paths from $(0, 0, 0)$ to $(1, 1, 1)$ for a vector of three parameters (w_1, w_2, w_3). (b) Visualization of all paths in which w_1 can contributes in conjunction with the parameters encountered before w_1.

Geometrically, for a given parameter permutation, the process of sequential parameter addition can be interpreted as a shortest path, traversing various intermediate vertices along the edges of an n-dimensional hypercube, as shown in Fig. 2a from the origin $\mathbf{0}_n = (0, \ldots, 0)^\top$ to $\mathbb{1}_n = (1, \ldots, 1)^\top$ [1, 21]. Each vertex represent the parameter subsets S, each with an performance value $f^*(S)$ trained with the parameter subset S. The marginal contribution $\Delta_{\pi,i}$ is then precisely the change in f^* along an edge of this hypercube, corresponding to the addition of parameter i (Fig. 2b). Note that

$$\sum_{i \in \mathcal{P}} \Delta_{\pi_1,i} = \sum_{i \in \mathcal{P}} \Delta_{\pi_2,i} = f^*(\mathcal{P}), \tag{3}$$

for any $\pi_1, \pi_2 \in \Pi(\mathcal{P})$ because the sum of marginal contributions of every parameters along any permutation path $\pi \in \Pi(\mathcal{P})$ equals the value of $f^*(\mathcal{P})$ optimized with all parameters, corresponding to the vertex $\mathbb{1}_n$. The main point is that different permutations path may give different marginal contributions for the same parameter, which gives non-uniform decay pattern as shown in Fig. 1.

SHAP Value (SV) [5, 15, 19]: The SV in Eq. (1) can be rewritten a simple average of the parameter's individual marginal contributions across all possible permutation paths on the n-dimensional hypercube:

$$\phi_i = \frac{1}{n!} \sum_{\pi \in \Pi(\mathcal{P})} \Delta_{\pi,i}, \tag{4}$$

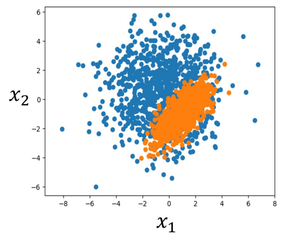
where the f in Eq. (1) is now replaced by the optimized objective f^*. A simple example, pruning parameter choice under the vertex values detailed in Appendix A, illustrates that SV-based pruning can result in clearly wrong decisions.

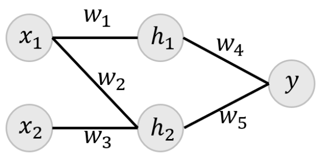
3 Cooperation Index

3.1 Definition

We now formally define the Cooperation Index (CI). The definition is based on classifying the characteristic of a parameter's marginal contribution within each permutation path $\pi \in \Pi(\mathcal{P})$.

- **Cooperative contribution:** Parameter $i \in \mathcal{P}$ is said to be cooperative if marginal contribution of parameter i satisfies $\Delta_{\pi,i} > \phi_i$. In this case, its marginal contribution is said to be cooperative contribution. If the number of cooperative contribution is high, parameter i consistently contributes more than its expected value, resulting in the speed of decay of marginal contribution is slow.
- **Replaceable contribution:** Parameter $j \in \mathcal{P}$ is said to be replaceable if marginal contribution of parameter j satisfies $\Delta_{\pi,j} < \phi_j$. If the number of replaceable contribution is high, the SV assigned to this parameter can be easily compensated by other parameters, indicating that this parameter is replaceable by other parameters. In this case, the speed of decay of marginal contribution is fast.


(a)



(b)

Table 1: Importance scores for the initial pruning

Params	SV	CI
w_1	0.0790	0.25
w_2	0.0836	0.25
$\boldsymbol{w_3}$	**0.0350**	**0.38**
w_4	0.0790	0.25
w_5	0.1134	0.25

Figure 3: (a) Two-class 2-D training data. (b) A toy neural network consisting of five weight parameters $w_1, \ldots, w_5$ and a single hidden layer with hidden units h_1 and h_2.

Based on the number of cooperative contributions on the n-dimensional hypercube, we define the Cooperation Index (CI) for parameter i as follows:

$$\mathrm{CI}(i) = \frac{|\{\pi \in \Pi(\mathcal{P}) : \Delta_{\pi,i} > \phi_i\}|}{|\Pi(\mathcal{P})|}. \tag{5}$$

Here, $\Pi(\mathcal{P})$ denotes the set of all permutation paths over the parameter set $\mathcal{P}$. If the total number of parameters is n, the total number of permutation path $|\Pi(\mathcal{P})|$ is $n!$. Note that SV and CI are used for parameter ranking purposes, and both methods prune low-ranking parameters. The computational cost for calculating both is the same.

3.2 An Illustrative Experiment with Synthetic Data

Here, we present a simple toy example to illustrate the central concept of the CI. The neural network model in Figure 3b has five parameters, $w_1, \ldots, w_5$, which yields $2^5 = 32$ parameter subsets and $5! = 120$ permutations. In each permutation path, the subset designated as learning parameters is trained on the training data in Figure 3a to predict the labels, while the remaining parameters are fixed as zero and retained. After training, the model's negative cross-entropy was evaluated on training data. Initially, the SV and CI scores for all five parameters are presented in Table 1. We note that parameter w_3 exhibits the *largest* CI, while the others exhibit the *lowest*. Once parameter w_3 is removed, the classification information from x_2 is lost. Thus, in the initial pruning, w_3 is never pruned by the CI criterion. These cooperative behavior of w_3 appear to high CI value in analyzing the given training data. Detailed dynamics of CI are provided in appendix B.

3.3 Experiment

We extended experiments to the real-world cases by using VGG-16 [20] and ResNet-18 [10] architectures on MNIST [12] and CIFAR-10 [11] datasets. The experiments were designed to confirm generalization improvement stemming from the initial, delicate phase of filter removal. We intentionally overfit the models by reducing training data and performed model pruning at the individual filter level in the all experiments. The core of our experiment is to confirm for generalization performance gains at the start of the pruning process, involving minimal filter removal, rather than high-ratio pruning. The experiments were conducted five realizations with different random seeds, and we report the best performing result in terms of accuracy. The pruning ratio indicates the percentage of the removed filters and weights of pruned filters are set as zero and retained in the networks.

Results. The results of our large-scale experiments in Table 2 demonstrate the superiority of CI and highlights CI's ability to preserve the model's core functional elements in comparison of SV. The interesting result of the experiment of VGG model on MNIST data shows the improvement of generalization performance of the model by pruning unnecessary parameters (filters). Experimental detail and comparison with other baselines are provided in appendix C and D.

Table 2: Comparison of Pruning Methods over five realizations with different random seeds.

Model, Datasets	**Method**	**Original Accuracy (%)**	**Accuracy (%) at Pruning Ratio**
VGG-16, CIFAR-10	SV (Baseline)	79.84	76.44 at 3%
	CI (Ours)	79.84	**73.49** at 3%
VGG-16, MNIST	SV (Baseline)	88.22	87.65 3%
	CI (Ours)	88.22	**92.54** at 3%
ResNet-18, CIFAR-10	SV (Baseline)	76.40	69.04 at 3%
	CI (Ours)	76.40	**71.41** at 3%
ResNet-18, MNIST	SV (Baseline)	87.63	77.09 at 3%
	CI (Ours)	87.63	**83.01** at 3%

4 Conclusion

This study introduces a novel and simple criterion for measuring parameter importance. The Cooperation Index (CI) quantifies the speed of decay of the marginal contribution and addresses the limitation of SHAP value. This approach is more effective for model pruning, revealing model's core functional elements and improving the generalizability of the model. A key challenge moving forward lies in adapting Cooperation Index to larger models and more diverse datasets across a range of applications.

Acknowledgments and Disclosure of Funding

This work was partly supported by NRF/MSIT (No. RS-2024-00421203) and IITP/MSIT (IITP-2021-0-02068, RS-2020-II201373, RS-2023-00220628).

References

[1] O. Candogan, I. Menache, A. Ozdaglar, and P. A. Parrilo. Flows and decompositions of games: Harmonic and potential games. *Mathematics of Operations Research*, 36(3):474–503, Aug. 2011.

[2] J. Castro, D. Gómez, and J. Tejada. Polynomial calculation of the shapley value based on sampling. *Comput. Oper. Res.*, 36:1726–1730, 2009.

[3] A. Catav, B. Fu, Y. Zoabi, A. L. W. Meilik, N. Shomron, J. Ernst, S. Sankararaman, and R. Gilad-Bachrach. Marginal contribution feature importance - an axiomatic approach for explaining data. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1324–1335. PMLR, 18–24 Jul 2021.

[4] S. Cohen, G. Dror, and E. Ruppin. Feature selection via coalitional game theory. *Neural Computation*, 19:1939–1961, 07 2007.

[5] I. Covert, S. M. Lundberg, and S. I. Lee. Understanding global feature contributions with additive importance measures. advances in neural information processing systems. volume 33, pages 17212–17223, 2020.

[6] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[7] A. Ghorbani and J. Zou. Neuron shapley: Discovering the responsible neurons, 2020.

[8] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[9] B. Hassibi, D. Stork, and G. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[12] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[13] Y. Lecun, J. Denker, and S. Solla. Optimal brain damage. volume 2, pages 598–605, 01 1989.

[14] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets, 2017.

[15] S. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions, 2017.

[16] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression, 2017.

[17] W. E. Marcílio and D. M. Eler. From explanations to feature selection: assessing shap values as feature selection mechanism. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 340–347, 2020.

[18] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference, 2017.

[19] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton, 1953.

[20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[21] A. Stern and A. Tettenhorst. Hodge decomposition and the shapley value of a cooperative game. *Games and Economic Behavior*, 113:186–198, Jan. 2019.

[22] S. Tripathi, N. Hemachandra, and P. Trivedi. Interpretable feature subset selection: A shapley value based approach, 2021.

[23] C.-K. Yeh, C.-Y. Hsieh, A. S. Suggala, D. I. Inouye, and P. Ravikumar. On the (in)fidelity and sensitivity for explanations, 2019.

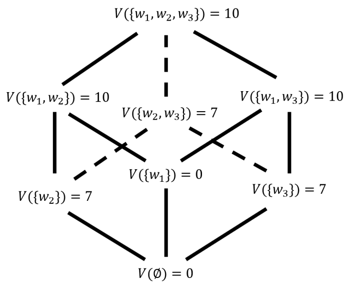
[24] M. Zaeri-Amirani, F. Afghah, and S. Mousavi. A feature selection method based on shapley value to false alarm reduction in icus, a genetic-algorithm approach, 2018.

A Appendix A

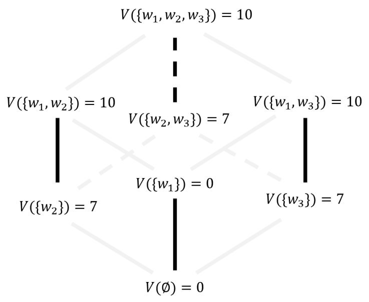
An Example of SV-Based Pruning Failure. Three parameters, denoted by w_1, w_2, w_3, produce a total gain of 10 when all are used. Let they have an f^*-hypercube with the following vertex values:

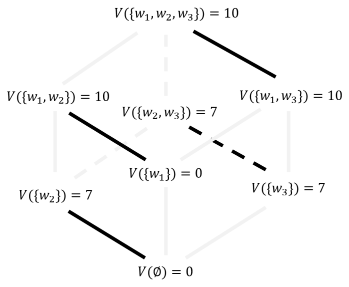
$$f^*(S) = \begin{cases} 10, & \text{if } S = \{w_1, w_2, w_3\} \\ 10, & \text{if } S = \{w_1, w_2\}, \\ 10, & \text{if } S = \{w_1, w_3\}, \\ 7, & \text{if } S = \{w_2, w_3\}, \\ 7, & \text{if } S = \{w_2\}, \\ 7, & \text{if } S = \{w_3\}, \\ 0, & \text{if } S = \{w_1\}, \\ 0, & \text{if } S = \emptyset. \end{cases} \tag{6}$$

The corresponding f^* values represent the incremental gains associated with transitions along vertex paths. The SVs assigned to each parameter are $(w_1, w_2, w_3) = (2, 4, 4)$, indicating that w_1 should be pruned first. On the other hand, CI yields $(w_1, w_2, w_3) = \left(\frac{2}{3}, \frac{1}{2}, \frac{1}{2}\right)$ reflecting the redundancy among parameters. According to the CI, w_2 or w_3 should be pruned, as they are replaceable by each other, preserving the total gain after pruning.



(a) geometric view of all possible path





(b) All possible path of w_1 (c) All possible path of w_2

Figure 4: Visualization of the f^*-hypercube in Appendix A.

B Appendix B

According to Figure 5 and 6, the order of parameter pruning by CI is $w_4 \to w_1 \to w_3 \to w_2 \to w_5$. Figure 6 shows that after w_4 is removed in the first stage, w_1 loses all cooperation and becomes the next selected parameter, while w_2 and w_5 strengthen their cooperation, leading to an increase in both of their CI scores. Figure 5 illustrates the dynamics of parameter scores in the SV-CI space as each parameter is pruned.

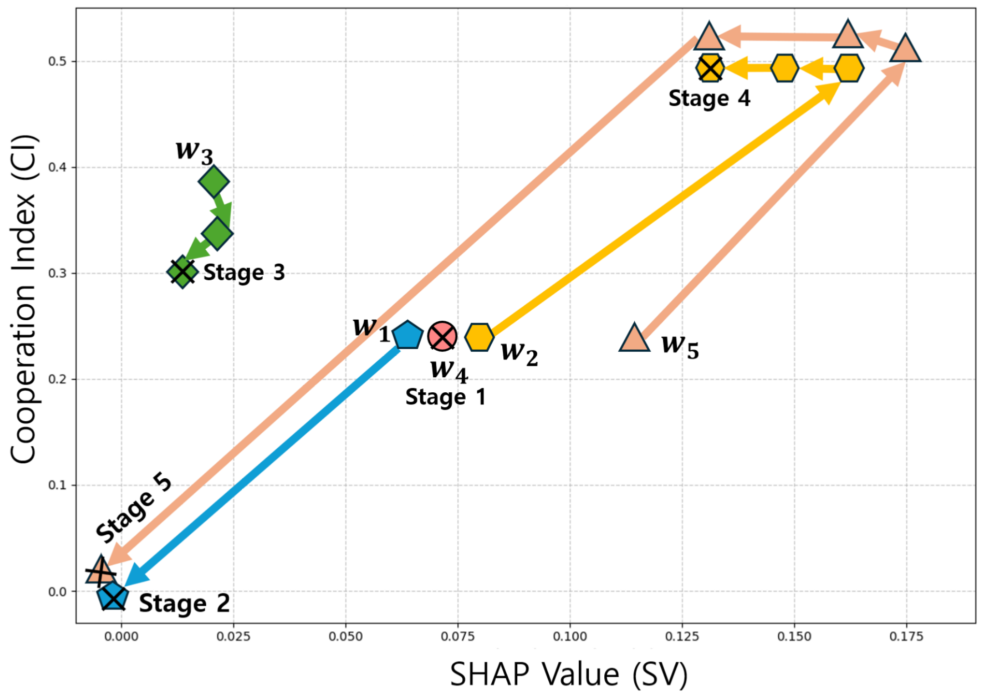
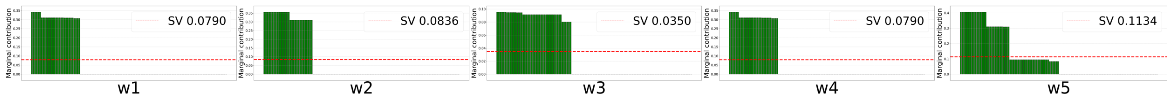
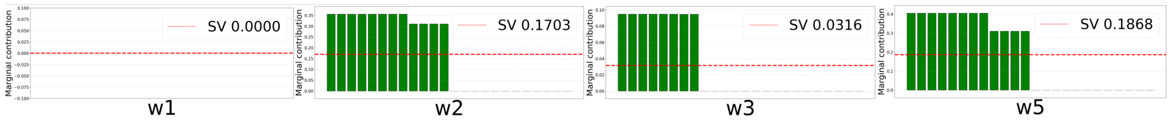


Figure 5: Trajectories of the SV and CI values of the parameters during the parameter pruning process.



(a)



(b)

Figure 6: Marginal contributions across all permutations and SV for the five parameters at the initial and second pruning stages. (a) Initial stage. (b) Second stage after w_1 has been pruned.

C Appendix C

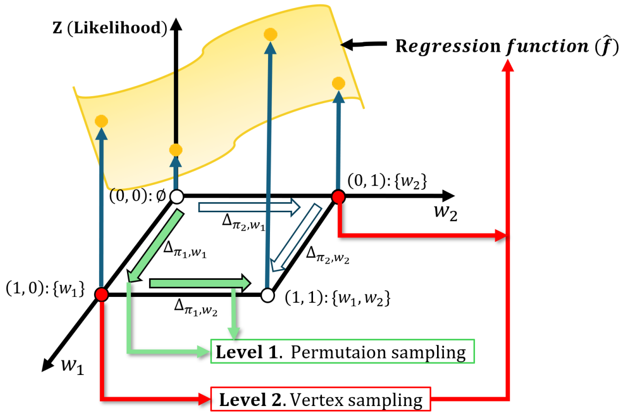


Figure 7: Illustration of the two-level marginal contribution estimation scheme. See the text for details.

Two-Level Approximation Scheme and algorithm for the CI Estimation As in the conventional calculation of SV, we approximate the computation of marginal contributions over all subsets by sampling from $n!$ permutations to avoid exponential complexity of $O(2^n)$[15, 3]. In addition to permutation sampling, it would require evaluating the model performance $f^*(S)$ for every subset in every sampled permutation. To make this feasible, we apply regression on the f^*-hypercube to approximate the marginal contributions (Level 2). Therefore, we introduce a two-level approximation scheme as illustrated in Figure 7.

Level 1: Permutation sampling. For the random sampling of the permutations $\Pi_{\text{samples}} \subseteq \Pi(\mathcal{P})$ [15, 2], we calculate the marginal contribution for all parameter indexes $i \in \mathcal{P}$,

$$\widehat{\Delta}\pi, i = \widehat{f}(S_\pi^i \cup i) - \widehat{f}(S_\pi^i), \tag{7}$$

for $\pi \in \Pi_{\text{samples}}$, using the regression function $\widehat{f}(S)$ trained for approximating $f^*(S)$ in the level 2 approximation. The complexity of level 1 is $O(N \cdot S)$, where M is the number of permutations and S is the number of parameters.

Level 2: Vertex sampling. As mentioned, directly calculating $f^*(S)$ for every required subset is also computational bottleneck. The purpose of Level 2 is to create the regression model $\widehat{f}(S)$ to overcome this challenge. First, we sample the m number of vertices of the f^*-hypercube. Using the training data, we then train the model which only uses parameters corresponding to the subset S to obtain $f^*(S)$. Those vertices and its $f^*(S)$ are used to train a regression model $\widehat{f}(S)$ as illustrated in Figure 7. The complexity of level 2 is $O(m \cdot T_{eval})$, where m is the number of sampled vertices (subsets) and T_{eval} is the time to evaluate one subset.

Using the estimated marginal contribution $\widehat{\Delta}_{\pi,i}$, SV is calculated as follows:

$$\widehat{\phi}_i = \frac{\sum_{\pi \in \Pi_{\text{sample}}} \widehat{\Delta}_{\pi,i}}{|\Pi_{\text{sample}}|}, \tag{8}$$

and the CI measures how rapidly the marginal contributions decay using

$$\widehat{\text{CI}}(i) = \frac{\sum_{\pi \in \Pi_{\text{sample}}} \mathbf{1}(\hat{\Delta}_{\pi,i} > \hat{\phi}_i)}{|\Pi_{\text{sample}}|}. \tag{9}$$

Algorithm 1 Detailed CI Calculation via Two-Level Approximation

```
Require:
 1: $N$: The original, trainable neural network.
 2: $P$: The set of all prunable parameters, total number $|P|$.
 3: $N_v$: Number of vertex (parameter subset) samples.
 4: $N_p$: Number of permutation samples.
 5: $\hat{f}$: Trained regression model (from Level 2 Approximation).
Ensure:
 6: $CI_{scores}$: Dictionary of CI scores for each parameter.

    Part 1: Calculate all marginal contributions
 7: $C_{\text{full}} \leftarrow$ Initialize empty list for each parameter $p \in P$.
 8: for $j \leftarrow 1$ to $N_p$ do
 9:     $\pi \leftarrow$ Randomly permute the set $P$.
10:     $S_{\text{prev}} \leftarrow \emptyset$
11:     for $k \leftarrow 1$ to $|P|$ do
12:         $p_k \leftarrow$ The $k$-th parameter in permutation $\pi$.
13:         $S_{\text{curr}} \leftarrow S_{\text{prev}} \cup \{p_k\}$
14:         $\Delta \leftarrow \hat{f}(S_{\text{curr}}) - \hat{f}(S_{\text{prev}})$
15:         Add $\Delta$ to the list $C_{\text{full}}[p_k]$
16:         $S_{\text{prev}} \leftarrow S_{\text{curr}}$
17:     end for
18: end for

    Part 2: Compute CI scores from contributions
19: $CI_{scores} \leftarrow$ Initialize empty dictionary.
20: for each parameter $p$ in $P$ do
21:     $SV_p \leftarrow$ Mean($C_{\text{full}}[p]$)
22:     $count_{\text{cooperative}} \leftarrow 0$
23:     for each contribution $\Delta$ in $C_{\text{full}}[p]$ do
24:         if $\Delta > SV_p$ then
25:             $count_{\text{cooperative}} \leftarrow count_{\text{cooperative}} + 1$
26:         end if
27:     end for
28:     $CI_{scores}[p] \leftarrow count_{\text{cooperative}}/|C_{\text{full}}[p]|$
29: end for

30: return $CI_{scores}$
```

D Appendix D

Experiments details. We employed a fully connected Multi-Layer Perceptron (MLP) as the regression model. Specifically, the architecture consists of two hidden layers, each with 4096 neurons, using ReLU activation functions. In level 2 (vertex sampling), we sampled 100 vertex of hypercube and used for training regression model ($\widehat{f}$). The hyperparameter includes the learning rate (1e-4), weight decay (1e-5), and training epochs (100). The pruning ratio is a user-defined hyperparameter, not determined by our method. In our implementation, pruning is performed in a single step for each target ratio. To achieve a desired pruning ratio, we first rank all parameters by their CI scores in ascending order. We then select the batch of parameters with the lowest CI scores corresponding to that ratio and prune them all at once. Additionally, the experiments took approximately 4 GPU-hours on a single NVIDIA A6000 GPU and required about 2GB of GPU memory.

Datasets details. For MNIST dataset experiments, Both models were trained on the dataset scaled down to 1/100 of original training dataset.

For CIFAR-10 dataset experiments, VGG-16 model was trained on the dataset scaled down to 2/10 of original training dataset and ResNet-18 model was trained on the dataset scaled down to 3/10 of original training dataset.

Stability of CI. We observed that how CI value can differ by the number of permutation sampling. We sampled permutations between 100 and 1000 times from different random seeds. When sampling 1000 permutations, the CI almost converged to a single value empirically. Therefore, we performed 1000 permutation samples in all experiments.

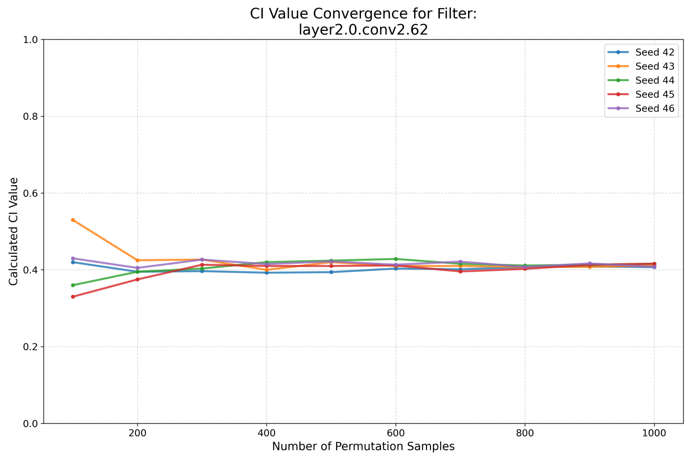


Figure 8: Convergence of CI.

Other baselines. (Table 3) Other baseline show the results no generalization improvement.

Table 3: Comparison of Other Pruning Methods.

Model, Datasets	**Method**	**Original Accuracy (%)**	**Accuracy (%) at Pruning Ratio**
VGG-16, CIFAR-10	L1-Norm [8, 6]	79.84	10 at 3%
	Taylor [13, 9, 18]	79.84	79.58 at 3%
VGG-16, MNIST	L1-Norm	88.22	9.82 at 3%
	Taylor	88.22	88.22 at 3%
ResNet-18, CIFAR-10	L1-Norm	76.40	29.13 at 3%
	Taylor	76.40	76.32 at 3%
ResNet-18, MNIST	L1-Norm	87.63	9.80 at 3%
	Taylor	87.63	86.31 at 3%