HRel: Filter Pruning based on High Relevance between Activation Maps and Class Labels

CH Sarvani[1], Mrinmoy Ghorai[1], Shiv Ram Dubey[2], SH Shabbeer Basha[3]

[1]*Computer Vision Group, Indian Institute of Information Technology, Sri City, Chittoor, Andhra Pradesh- 517646, India.*
[2]*Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology, Allahabad, Uttar Pradesh- 211015, India.*
[3]*DryvAmigo, Banglore, Karnataka, India.*

{sarvani.ch, mrinmoy.ghorai}@iiits.in, srdubey@iiita.ac.in, shabbeer.sh@dryvamigo.com

Abstract

This paper proposes an Information Bottleneck theory based filter pruning method that uses a statistical measure called Mutual Information (MI). The MI between filters and class labels, also called *Relevance*, is computed using the filter's activation maps and the annotations. The filters having High Relevance (HRel) are considered to be more important. Consequently, the least important filters, which have lower Mutual Information with the class labels, are pruned. Unlike the existing MI based pruning methods, the proposed method determines the significance of the filters purely based on their corresponding activation map's relationship with the class labels. Architectures such as LeNet-5, VGG-16, ResNet-56, ResNet-110 and ResNet-50 are utilized to demonstrate the efficacy of the proposed pruning method over MNIST, CIFAR-10 and ImageNet datasets. The proposed method shows the state-of-the-art pruning results for LeNet-5, VGG-16, ResNet-56, ResNet-110 and ResNet-50 architectures. In the experiments, we prune 97.98 %, 84.85 %, 76.89%, 76.95%, and 63.99% of Floating Point Operation (FLOP)s from LeNet-5, VGG-16, ResNet-56, ResNet-110, and ResNet-50 respectively. The proposed HRel pruning method outperforms recent state-of-the-art filter pruning methods. Even after pruning the filters from convolutional layers of LeNet-5 drastically (*i.e.*, from 20, 50 to 2, 3, respectively), only a small accuracy drop of 0.52% is observed. Notably, for VGG-16, 94.98% parameters are reduced, only with a drop of 0.36% in top-1 accuracy. ResNet-50 has shown a 1.17% drop in the top-5 accuracy after pruning 66.42% of the FLOPs. In addition to pruning, the Information Plane dynamics of Information Bottleneck theory is analyzed for various Convolutional Neural Network architectures with the effect of pruning. The code is available at https://github.com/sarvanichinthapalli/HRel.
This paper is published by Neural Networks, Elsevier. The final paper is available at:
https://www.sciencedirect.com/science/article/pii/S0893608021004962.

1. Introduction

Deep Convolutional Neural Networks (CNN) are being used to provide successful and reliable solutions in various domains [2, 9, 15, 51, 52, 64]. In the applications of deep neural networks, the requirement for higher memory and power consumption hinders their deployment on low-end devices such as mobiles, drones. Hence, it is necessary to decrease energy consumption and memory footprint. To solve it, two types of methods have been found in literature, namely network compression and Neural Architecture Search (NAS).

Network compression is an area that accelerates the inference by reducing the Floating Point Operations (FLOPs) and decreases the memory requirement by pruning trainable parameters using various techniques. Network compression can be performed by different techniques such as *network quantization, knowledge distillation, low-rank factorization* and *network pruning*. *Network quantization* reduces the number of bits required to represent the weights [69]. Binarization is an extreme case of this, where only 1 bit is used for representing weights [10, 11]. In *knowledge distillation* methods, a larger teacher model transfers its knowledge to a computationally less expensive student model [27, 53]. *Low-rank factorization* methods aim at reducing the computational requirement by representing the convolution weight matrix as a product of low-rank matrices [29]. *Network (Parameter) pruning* methods, prune the filters in two different ways, *i.e.*, weight pruning [21, 36] and filter pruning [40, 60]. In weight pruning, the least important weights across the network are pruned. Therefore, only a few weights of the filters are pruned by weight pruning methods. Special hardware libraries are required to accelerate the network compressed by the weight pruning method. On the other hand, filter pruning methods prune the complete filter and do not require the support of any special hardware and libraries. Hence, they are widely used in the research community in recent years.

NAS based compression techniques [14, 19, 41, 44, 61, 71] are focused on finding the compact structure of neural network architecture, rather than using a criteria for computing the importance of convolutional filters. NAS methods include channel configuration *i.e.*, number of channels in each layer into the search space. Thereby the best channel configuration under various computational budgets (eg. FLOPS) is selected with less human interference.

This paper is focused on filter pruning methods which are broadly classified into two types, namely, *Data free* - which use the weight matrices of filters [3, 5, 23, 24, 25, 26, 39, 43, 59, 60,

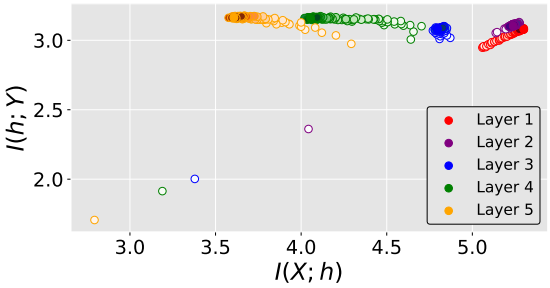


Figure 1: Information Plane dynamics of LeNet-5 architecture trained on MNIST dataset for 20 epochs. The layers are represented with different colors and saturation of each color indicates the progress of training.

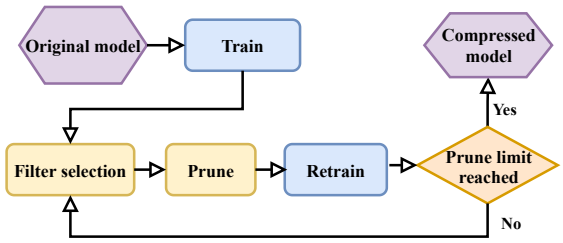


Figure 2: Illustration of complete pruning process: The original heavy model is initially trained before pruning. The pruning starts by selecting filters that are least important, followed by a retrain step. The process of pruning and retraining continues till the desired pruning limit is achieved.

65, 66] and *Data driven* - which use the activation maps generated by the respective filters [1, 13, 16, 28, 31, 37, 40, 45, 46]. A primitive data free filter pruning approach is proposed by Li *et al*. [39] that uses the filter's ℓ_1 norm to determine the significance of filters. The filters with the least ℓ_1 norm are considered to be less important and pruned from the model.The correlation measure between the filters is used to identify and prune the redundant filters [59, 65]. Singh *et al*. [60] employed a custom regularizer based on an orthogonality constraint such that the remaining filters after pruning based on ℓ_1 norm were independent and designed a mechanism to transfer the knowledge from the filters to be pruned to the remaining filters. Ayinde *et al*. [3] pruned the redundant filters based on the relative cosine distance among the filters. In the *data driven* category, Hu *et al*. [28] proposed a filter pruning method that prunes filters having a greater average number of zeroes in their activations. Lin *et al*. [40] pruned filters based on the rank of their respective activation maps. The rank of a matrix gives the maximum number of linearly independent column vectors. Both [28, 40] have identified the significance of filters directly from their activation maps without considering the class labels (ground truths). Jordao *et al*. [31] pruned filters by considering the linear relationship between filters and class labels. In data-free methods, it is difficult to capture the amount of relevant information retrieved by a filter about the class labels. Only in data-driven methods, the relation between transformed input (the input after applying non-linear transformations) at each hidden layer and the ground truth can be captured by an information theoretic quantity called Mutual Information (MI). MI can capture both linear and non-linear relationships. The pruning techniques [1, 13, 16, 37] which use Mutual Information for their filter pruning criteria are data-driven methods.

Using Mutual Information measure, Tishby *et al*. proposed Information Bottleneck (IB) theory [63] and applied it in the context of neural networks [56]. IB theory analyzes the learning of a neural network using the network's Information Plane (each hidden layer's MI with input and true class labels plotted on X-axis and Y-axis respectively) during the training as shown in Fig. 1. From the Information Plane (IP) dynamics of IB theory, each hidden layer's MI with input and class labels increase gradually and saturate during the training.

This paper proposes a data-driven filter pruning technique for neural network classifiers based on IB theory which defines the significance of filters using their Relevance. The filters with the least Relevance from each layer are pruned iteratively. The overall pruning process is depicted in Fig. 2. Among the other existing methods that use MI for defining filter's importance, only the method [1] observes the significance of individual neurons purely based on their Relevance. However, only fully connected and smaller neural networks are pruned by this strategy. Contrary to the method [1], our method can prune filters of convolutional layers in deeper architectures like VGG-16, ResNet-56, ResNet-110 and ResNet-50. The proposed method also relies on a non-parametric estimator [67] that is more stable than the binning method used in [1] for the estimation of MI.
Our contributions are summarized as follows:

- Based on the IB theory, MI between filter's activation maps and class labels is proposed as the criterion to decide the filter's importance.
- The Information Plane dynamics of IB theory is shown along with the effect of pruning, which justifies the proposed filter selection criterion for pruning.
- Extensive experiments show the efficacy of the proposed approach, with a considerable improvement over the recent state-of-the-art methods [3, 5, 13, 14, 16, 23, 24, 25, 31, 37, 39, 40, 41, 42, 44, 59, 60, 61].

The rest of the article is organized as follows- Section 2 reviews the background of IB theory and MI estimation by investigating the limitations in the prior art. Section 3 discusses proposed HRel method. Section 4 illustrates the experimental results and Section 5 concludes with the future directions.

2. Related Works

This section discusses the significance of Information Plane (IP) dynamics in IB theory, limitations of the methods for MI estimation, and the limitations of the existing works that used MI for pruning.

In IB theory [56], the learning process of neural networks is analyzed using the IP dynamics. During the training of neural networks, two quantities, MI of every hidden layer h with input X represented as $I(X;h)$ and MI of every hidden layer h with

label Y represented as $I(h;Y)$, keep increasing. At a point during training, the quantity $I(X;h)$ starts decreasing, while $I(h;Y)$ continues to increase as shown in Fig. 1. This is called as compression phase [56]. However, both the quantities settle at a value and do not change on further training of the neural network. There are conflicting views [4, 18, 55] and supporting views [8, 30, 49] regarding the existence of the compression phase. The proposed method selects filters using MI between their activation maps and class labels for pruning based on IB theory.

MI estimation plays an important role in IB theory, and several works based on IB theory [6, 18, 55, 56] use different MI estimation methods. MI calculation in deep neural networks requires the joint and marginal probabilities of high dimensional variables, which are difficult to compute. Hence various non-parametric estimators [6, 32, 33, 38, 49, 67, 70] have been proposed. The basic method uses binning [50] to estimate MI, where the neurons' outputs are discretized. However, the binning estimate highly depends on the bin size. The non-parametric estimators based on K-Nearest Neighbours [33], and kernel density estimation [32, 38] were being widely used before Mutual Information Neural Estimation (MINE) [6], which solved the problem of scaling with the sample size and dimension. An Rényi's alpha entropy estimator [67] has been proposed using the matrices or tensors, which are basic entities in deep learning. It has also shown the IP dynamics on larger architecture (from the perspective of MI estimation) like VGG-16. The proposed HRel method uses matrix based Rényi's alpha entropy estimator [67] for MI estimation.

In MI based filter pruning methods, Dai *et al*. [13] used an upper bound of Relevance as a part of the loss function. With this modified loss function, MI between every hidden layer and the corresponding class labels increases, and MI between the consecutive layers decreases. Ganesh *et al*. [16] pruned the filters in a hidden layer with lower Mutual Information with all the other filters of the subsequent layer. Amjad *et al*. [1] have shown that in fully connected neural networks, MI between neurons in hidden layers and the corresponding class labels is a good selector for layer-wise neuron importance. Min *et al*. [46] used the entropy of activations conditioned on the loss as a criterion for filter's significance. The filters with higher conditional entropy, which implies a lower MI, are pruned. Recently, Lee *et al*. [37] utilized gradients of MI between the activation maps of BatchNorm layers and final score vectors to the scaling factor of Batch-Normalization during back propagation to decide the filter's importance. The network architecture is augmented with an MI-subnet, which is responsible for the MI estimation.

Compared to the existing methods, the proposed HRel method captures MI between filters and class labels (*i.e*., Relevance) using a matrix based estimator [67] and uses it for filter pruning criterion. To the best of our knowledge, effect of pruning on Information plane of various CNNs is analyzed for the first time. Also, the HRel method is not employing additional architecture or changes in the loss function, unlike the MI based filter pruning methods [13, 37].

3. Proposed HRel Pruning Approach

In this section, we propose an HRel filter pruning approach for convolutional neural networks. The filters are pruned depending on their Relevance in corresponding hidden layers. The Relevance criterion is chosen based on the IB theory using the Mutual Information metric. This section describes the basic definitions and notations, computation of the Relevance followed by steps of filter pruning.

3.1. Basic Definitions and Notations

Mutual Information (MI) between two random variables U, V i.e., $I(\mathrm{U};\mathrm{V})$ quantifies the amount of information that can be inferred about a random variable U by observing the other random variable V or vice versa, which is expressed as

$$I(\mathrm{U};\mathrm{V}) = \mathrm{H}(\mathrm{U}) + \mathrm{H}(\mathrm{V}) - \mathrm{H}(\mathrm{U},\mathrm{V}) \tag{1}$$

where H(U) and H(V) denote *entropy* [12], H(U, V) denotes *joint entropy* [12].

Assume a CNN model having c convolutional layers, in which L_i is the i^{th} convolutional layer. The filters of a convolutional layer L_i can be represented as $\mathcal{F}_{L_i} = \{f_{i,1}, f_{i,2}, ..., f_{i,n_i}\}$, where n_i is the number of filters in layer L_i, $f_{i,j} \in \mathbb{R}^{n_{i-1}\times d_i\times d_i}$, d_i is the kernel size and n_{i-1} is the number of channels in each filter which is same as the depth of the activation input. Suppose there are m mini-batches of input for training the network. For the k^{th} mini-batch, the activation maps of filters from i^{th} hidden layer is denoted by $\mathcal{A}_i^k = \{A_{i,1}^k, A_{i,2}^k, ... A_{i,n_i}^k\} \in \mathbb{R}^{n_i\times s\times h_i\times w_i}$, where n_i is the number of filters, s is the mini-batch size, h_i and w_i are the height and width of the activation maps, respectively. Notably, $A_{j,1}^k \in \mathbb{R}^{s\times h_i\times w_i}$ is the activation map generated by $f_{i,j}$ for all the samples in k^{th} mini-batch. During pruning, filters in layer L_i are split into Pruned filters $\mathcal{P}_{L_i} = \{f_{i,P_1}, f_{i,P_2}, ..., f_{i,P_{p_i}}\}$ and Remaining filters $\mathcal{R}_{L_i} = \{f_{i,R_1}, f_{i,R_2}, ..., f_{i,R_{r_i}}\}$, where p_i and r_i are the number of pruned and remaining filters of layer L_i. P_j and R_k denote j^{th}, k^{th} filters in pruned and remaining filter set, respectively. Notably, $\mathcal{P}_{L_i} \cap \mathcal{R}_{L_i} = \varnothing$, $\mathcal{P}_{L_i} \cup \mathcal{R}_{L_i} = \mathcal{F}_{L_i}$, and $p_i + r_i = n_i$.

3.2. The Relevance as Filter Selection Criteria

The proposed HRel pruning method utilizes the Relevance, which is a key component of Information Plane dynamics used in IB theory. Though there is an ongoing debate on the existence of the compression phase, there is no ambiguity in the increment and saturation of hidden layers' Relevance ($I(L_i;Y)$) *i.e*., MI between each hidden layer's (L_i) activation maps and the class labels (Y) in neural networks, during training. It is also observed that initially, all the layers have less Relevance at the beginning of training. But as the training progresses, the Relevance of each layer also gradually increases and gets saturated, as shown in Fig. 1. So, a higher Relevance gained by the hidden layers during the training implies that the hidden layers learned more relevant information about the class labels. Similar to hidden layers, individual filter's Relevance ($I(f_{i,j};Y)$) *i.e*., MI between each filter's activation maps and the class labels, also determines the amount of relevant information extracted by

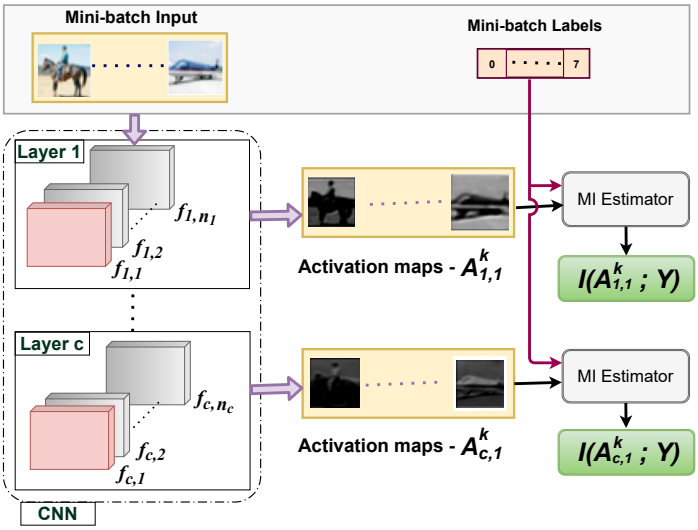


Figure 3: The steps involved in calculating the Relevance of $A^k_{i,j}$, *i.e.*, activation map of the j^{th} filter of i^{th} layer from the k^{th} mini-batch.

filter about the class labels. Hence, the Relevance of filters is employed in the proposed method to determine the significance of the filters across each layer. For k^{th} mini-batch of training data, the Relevance between the activation maps of filters and class labels Y given by $I(A^k_{i,j}; Y)$ is obtained as shown in Fig. 3.

The proposed method estimates two (Relevance) quantities $I(L_i; Y)$ and $I(f_{i,j}; Y)$, which are utilized in IP dynamics and filter pruning, respectively. The computation of $I(f_{i,j}; Y)$ uses the activation maps generated by each filter from a hidden layer, whereas computation of $I(L_i; Y)$ uses the complete output of a hidden layer. Estimation of MI between the activation maps and class labels in the proposed method is similar to [67].

For a given mini-batch of size s with the activation maps generated $X = \{x_1, x_2..., x_s\}$, the Gram matrix G of size $s \times s$ is calculated using Gaussian kernel as $G_{i,j} = e^{-\frac{1}{\sigma^2}\|x_i - x_j\|_F^2}$ where $G \in \mathbb{R}^{s \times s}$ for all i, $j \in [1, s]$, σ denotes kernel width, and $\| \cdot \|_F$ denotes Frobenius Norm. As presented in [48], entropy is consequently calculated using the Eigen values of the normalized Gram matrix N as

$$H(N) = -\sum_{i=1}^{m} \lambda_i \log_2 \lambda_i \tag{2}$$

Where $N_{i,j} = \frac{1}{s}\frac{G_{i,j}}{\sqrt{G_{i,i}G_{j,j}}}$, $N \in \mathbb{R}^{s \times s}$ for all i, j $\in [1, s]$, and λ_i is the i^{th} eigen value of N.

The joint entropy between random variables U, V is calculated using Hadamard product ($\circ$) of their normalized Gram matrices N_U, N_V, respectively [17] as

$$H(U, V) = H(N_U \circ N_V) \tag{3}$$

The Relevance is calculated using the Equations 1, 2 and 3. Unlike rank, the Relevance of activation maps generated by filters changes with the mini-batches, as each filter can share a different amount of information with different classes as shown in Fig. 4. Slight variations in the Relevance values of filters over mini-batches in an epoch can also be observed. Therefore the mean value of the Relevance of filters measured across the mini-batches of training data is considered. The averaged Relevance values across the mini batches for the filters in Fig. 4 are discussed in subsection 4.G and depicted in Fig. 7. It is also observed that in Fig. 4 for each architecture except ResNet-50 from top to bottom, the color saturation of the plots gradually moved towards the brighter side.

Algorithm 1 : HRel pruning of a layer L_i for a pruning iteration

Inputs: $\mathcal{R}_{L_i}$- Set of remaining filters in L_i, $prune\_ratio_i$ - The percentage of filters to prune in each pruning iteration, r_i - The number of remaining filters of layer L_i, $limit_i$-Number of filters to be retained in layer L_i of final compressed model

Output: Updated $\mathcal{R}_{L_i}$

if first *pruning iteration* **then**
  $\mathcal{R}_{L_i} \leftarrow \mathcal{F}_{L_i}$
  $r_i \leftarrow n_i$
end if
if $r_i > limit_i$ **then**
  for each mini-batch k of total m mini-batches, $k \in$ 1,2,...,m **do**
    for each filter $f_{i,j} \in \mathcal{R}_{L_i}$ **do**
      compute $I(A^k_{i,j}; Y)$ using subsection 3.2
    end for
  end for
  for Each filter $f_{i,j} \in \mathcal{R}_{L_i}$ **do**
    $I(f_{i,j}; Y) \leftarrow \frac{\sum_{k=1}^{m} I(A^k_{i,j}; Y)}{m}$
  end for
  $t_i \leftarrow \lceil r_i \times prune\_ratio_i / 100 \rceil$
  Sort $\mathcal{R}_{L_i}$ in ascending order ;
  $\mathcal{R}^{sorted}_{L_i} \leftarrow \{f_{i,R_1}, f_{i,R_2}, ..., f_{i,R_{r_i}}\}$; $R_1, R_2, ...R_{r_i}$ is a permutation of filters in $\mathcal{R}_{L_i}$: $I(f_{i,R_1}; Y) \leq I(f_{i,R_2}; Y) \leq ... \leq I(f_{i,R_{t_i}}; Y) \leq ... \leq I(f_{i,R_{r_i}}; Y)$
  $\mathcal{R}^{prune}_{L_i} \leftarrow \{f_{i,R_1}, f_{i,R_2}, ..., f_{i,R_{t_i}}\}$
  $\mathcal{R}_{L_i} \leftarrow \mathcal{R}_{L_i} - \mathcal{R}^{prune}_{L_i}$
  $r_i \leftarrow r_i - t_i$
end if

3.3. Filter Pruning Steps

The pruning of filters involves three main stages. Firstly, the neural network is trained till the baseline accuracy is achieved. Secondly, each filter's Relevance is computed and filters with low Relevance are pruned. Thirdly, the network is retrained. The filter pruning and retraining are done iteratively.

3.3.1. Initial Training

The network parameters (Θ) are initialized and updated until the model convergence. Training of network with each mini-batch of data is called *training iteration*. During network training, the kernel width σ_i for each hidden layer L_i is computed for all mini-batches.

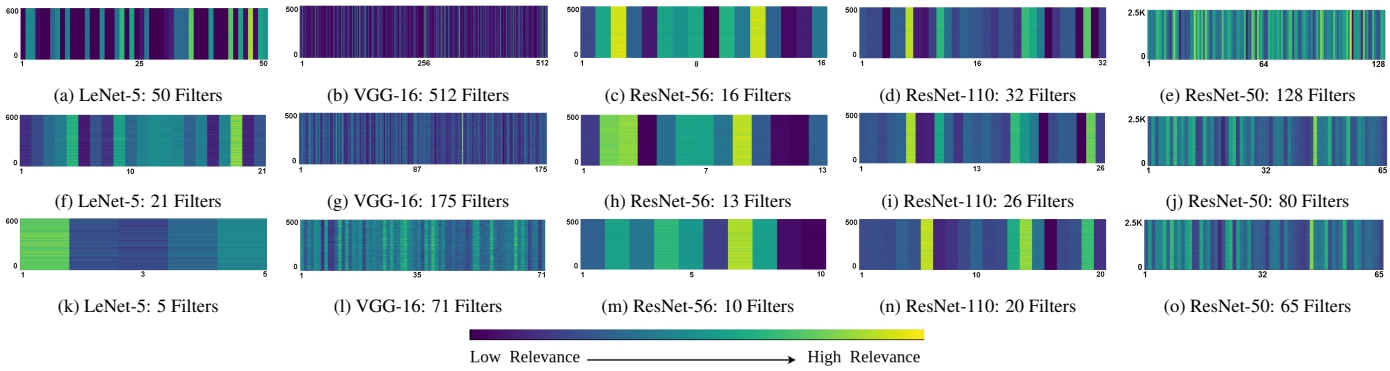


Figure 4: The Relevance of the remaining filters from convolution layers of different architectures (the number of remaining filters are specified for each sub-figure). For each sub-figure, X-axis denotes all the remaining filters in a convolutional layer at different pruning iterations. Y-axis (bottom to top) denotes the mini-batches of the training data. The first row depicts the architectures before pruning. Rows 2, 3 indicate the Relevance of filters during specific pruning iterations. Convolutional layers 2, 9, 13, 37 and 34 of LeNet-5, VGG-16, ResNet-56, ResNet-110, and ResNet-50 respectively are used.

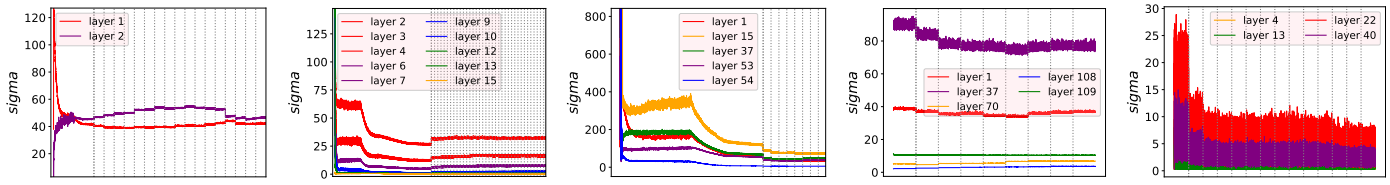


Figure 5: Kernel width σ_i for different layers denoted by sigma across the pruning iterations for different *architecture* + *dataset* combinations. The X-axis denotes training iterations from left to right (each vertical line in the sub-figures denotes a pruning iteration).

3.3.2. Filter Selection and pruning

After initial training, pruning, and retraining of the network are performed iteratively, where each iteration is called *pruning iteration*. In each pruning iteration, the filters selected using Algorithm 1 are pruned, and the network is retrained. To select the filters to be pruned from each layer L_i, two hyperparameters are required. One is the final number of filters to be retained denoted by $limit_i$, and the other is the percentage of remaining filters to be pruned at each iteration of pruning identified by $prune\_ratio_i$. The training data is processed batch-wise and the Relevance of each filter in a given mini-batch k *i.e.*, $I(A^k_{i,j}; y)$, is calculated using steps 2-6 in Algorithm 1. $I(f_{i,j}; y)$ is then obtained by averaging $I(A^k_{i,j}; y)$ across the mini-batches, for all filters in each layer as shown in steps 7-9. Next, the filters in each layer are sorted based on $I(f_{i,j}; y)$ value and top $prune\_ratio_i$ % of the filters in each layer L_i given by t_i are pruned. Step 11 in Algorithm 1 can be skipped if a fixed number of filters are removed from each layer by directly specifying t_i value. The remaining filters and their count are updated using steps 14-16 in Algorithm 1.

3.3.3. Retraining

This section focuses on improving the model performance lost due to the pruning of filters. After pruning the selected filters from the network with c convolutional layers, the model contains a reduced set of the trainable parameters Θ' compared to the original model Θ.

$$\Theta' = \Theta \setminus \{\mathcal{P}_{L_i} \mid i \in 1, 2, \ldots, c\} \tag{4}$$

The network is then fine-tuned by retraining each architecture for a certain number of epochs to regain the accuracy drop.

It has been observed that the kernel width σ_i calculated for each layer along with the initial training saturates after a few epochs as shown in Fig. 5 (i.e., before the pruning iterations begin). Therefore, its calculation is deterred in further training. In our work, we observed σ_i values across the pruning iterations and found that despite pruning, the values did not fluctuate much, as shown in Fig. 5. Hence the computation of σ_i is not done during the retraining of the network after pruning. The final obtained kernel width of hidden layers during pruning is used for both hidden layers and the corresponding filters across the pruning iterations.

4. Experiments and Analysis

To demonstrate the potency of the proposed HRel pruning method, the following dataset + architecture combinations - MNIST [35] + LeNet-5 [35], CIFAR10 [34]+ VGG-16 [58], CIFAR10 + ResNet-56 [22], CIFAR10 + ResNet-110 [22], ImageNet[54] + ResNet-50 [22] are used. Experimental results are analyzed in terms of accuracy, IP dynamics, and the Relevance distribution. Subsections *4.A*, *4.B*, *4.C*, *4.D* and *4.E* compare the pruning results with state-of-the-art methods. Furthermore, the subsection *4.F* analyzes the IP dynamics related

to IB theory and finally, the subsection *4.G* examines the distribution of the Relevance values of filters across the pruning iterations.

The percentage of filters to be removed from each layer and the number of filters to be retained in the final network are the hyperparameters for each of the architectures. The same CNNs as in the recent work based on the rank of activation maps [40], are utilized for verifying the efficacy of the proposed filter pruning method. The batch size is modified to 80 in ResNet-50 and 100 in rest of the architectures. Nesterov momentum [7] is used for ResNet-56 and ResNet-110. Same settings as in [67] are used for MI estimation by using an input kernel of width 8 and a label kernel of width 0.1. During the calculation of filters' Relevance batch size of 128 in ResNet-50 and 100 in rest of the architectures are used. The kernel width of each hidden layer for every dataset + architecture combination is evaluated until the baseline accuracy is achieved. The hyperparameters such as learning rate, learn rate schedule for training and pruning iterations are specified for each architectures in subsections *4.A*, *4.B*, *4.C*, *4.D* and *4.E*.

In our experimental setting, we have evaluated the proposed method in terms of *FLOPs*- Floating Point Operations and *Trainable Parameters* for all the models. For a fair comparison with the other methods, parameters and FLOPs corresponding to Convolutional and Fully Connected layers alone are considered. FLOPs and Params in the results table (Table 1, 2, 3, 4, and 5) denote remaining FLOPs and remaining parameters after pruning, respectively. M denotes Millions (10^6) and B denotes Billions (10^9) in the columns of FLOPs and Params. The percentage of pruned FLOPs and pruned Parameters are denoted by $P_f\%$ and $P_p\%$, respectively. The baseline accuracy and the accuracy after pruning (in percentage) is denoted by $\text{Acc}_{baseline}\%$ and $\text{Acc}_{pruned}\%$ respectively. Accuracy Drop is denoted by Acc↓. For ResNet-50 Top-1,Top-5 implies Top-1 and Top-5 baseline accuracies. Top-$\#_{pruned}\%$ and Top-#↓ denote corresponding accuracy and accuracy drop, after pruning.

4.A. LeNet-5 on MNIST Dataset

MNIST is a handwritten digits dataset, that contains 60,000 training images and 10,000 test images, each of size $28\times28\times1$. LeNet-5 architecture contains 2 convolutional layers, having 20 and 50 filters with the spatial dimension of 5×5, followed by 3 fully connected layers with 800, 500, and 10 neurons, respectively. The network is trained for 40 epochs with the initial learning rate of 0.1, which is divided by 10 at epoch numbers 20 and 30 to achieve the baseline accuracy. For the proposed HRel method, rather than pruning an equal percentage of filters from each layer, better results are observed empirically if initial layers are pruned at a lower rate compared to final layers. Thus, in each pruning iteration, 4% and 12% of the filters are pruned from the first and second convolutional layers, respectively. After pruning, the network is retrained for 40 epochs beginning with a learning rate of 0.1, which is divided by 10 at epochs 10 and 20. LeNet-5 pruning results are compared with benchmark methods in Table 1. Note that HRel-# represents HRel at different pruning limits.

HRel method achieves a higher FLOPs reduction percentage, i.e., 97.98%, with the accuracy of 98.78%, and accuracy drop of 0.52, outperforming CFP [59] and HBFP [5] methods, for an equal FLOPs reduction percentage. While PP-OC [60] has the least accuracy drop, the HRel method achieved the higher test accuracy when $\{20, 50\}$ filters are pruned to $\{4, 5\}$ filters in the first and second convolution layers, respectively. Though VIB [13] method had a lesser number of remaining FLOPs, *i.e.*, 0.09M, the authors have mentioned that they considered half the number of FLOPs. Hence, it would account for 0.18M to compare with all the other methods. Inspite of having more baseline accuracy than CFP and HBFP the accuracy drop is considerably less.

4.B. VGG-16 on CIFAR-10 Dataset

CIFAR-10 dataset consists of 50,000 training images and 10,000 test images belonging to 10 classes. The image size is $32\times32\times3$. The proposed HRel method is applied to VGG-16 architecture 64-64-128-128-256-256-256-512-512-512-512-512-512-512-10 with 13 convolutional layers and 2 fully connected layers to prune the filters from convolutional layers. The network is trained for 300 epochs with the initial learning rate of 0.1 divided by 10 at epoch numbers 80, 140, and 230 to achieve the baseline accuracy. Similar to LeNet-5, a lower pruning ratio is used for initial layers compared to final layers. Consequently, 2% of filters from layers 1 and 2 (layers with 64 filters initially), 4% of filters from layers 3 and 4 (layers with 128 filters initially), 5% of filters from layers 5, 6 and 7 (layers with 256 filters initially), and 10% of filters from the rest of the layers (layers with 512 filters initially) are pruned in each pruning iteration. After pruning, the network is retrained for 90 epochs beginning with a learning rate of 0.01, which is divided by 10 at epochs 40, and 70. The pruning results for HRel-1 and HRel-2 (specified in Table 2) are obtained for VGG-16 with 21-48-64-64-95-107-107-175-71-71-44-44-56 and 20-48-64-64-95-107-107-175-71-71-44-44-56 remaining filters, from each convolutional layer respectively.

The HRel method achieves the accuracy of 93.54% when 84.70% of the FLOPS pruned%, which is better than all the other methods. In terms of accuracy drop HRel is observed to have second best result next to PP-OC. Also, a very promising trade-off is observed between the accuracy and number of remaining FLOPs using the proposed HRel method as compared to the existing methods. It shows the capability of the proposed pruning method to prune a deeper plain model.

4.C. ResNet-56 on CIFAR-10 Dataset

ResNet-56 is a deeper and complex architecture compared to VGG-16. ResNet-56 has 55 convolutional layers and 1 Fully connected layer in total. Except for the first one, all convolutional layers are grouped into three different blocks, with each block having 18 convolutional layers. The number of filters in 1^{st}, 2^{nd} and 3^{rd} blocks is 16, 32 and 64, respectively. The network is trained for 180 epochs with the initial learning rate of 0.1, which is divided by 10 at epoch numbers 91 and 136 to achieve the baseline accuracy. For pruning ResNet-56, we

Table 1: Pruning results of LeNet-5 architecture over MNIST dataset. F here denotes number of remaining filters in convolutional layers 1 and 2 respectively.

Method	$Acc_{baseline}\%$	$Acc_{pruned}\%$	Acc↓	F	FLOPs	$P_f\%$
VIB [13]	-	99.00	-	-	0.09M	-
GAL [42]	99.20	98.99	0.21	2, 15	0.10M	95.60
PP-OC [60]	99.17	99.20	-0.03	4, 5	0.19M	95.56
HRel-1(ours)	**99.30**	**99.23**	**0.07**	**4, 5**	**0.19M**	**95.56**
HRel-2(ours)	**99.30**	**99.16**	**0.14**	**3, 5**	**0.15M**	**96.41**
HRel-3(ours)	**99.30**	**98.99**	**0.31**	**3, 4**	**0.13M**	**96.84**
CFP [59]	99.17	98.23	0.94	2, 3	0.08M	97.98
HBFP [5]	99.17	98.60	0.57	2, 3	0.08M	97.98
HRel-4(ours)	**99.30**	**98.78**	**0.52**	**2, 3**	**0.08M**	**97.98**

follow the same approach as in [59], i.e., pruning 1, 2 and 4 filters from every convolutional layer belonging to 1^{st}, 2^{nd} and 3^{rd} blocks, respectively. After pruning, the network is retrained for 100 epochs beginning with a learning rate of 0.01, which is divided by 10 at epochs 20 and 70. The remaining number of filters in convolutional layers of each block are 10, 20, 38 for HRel-1 and 8, 15, 30 for HRel-2. As shown in Table 3, after pruning 62.06% of the FLOPs, the proposed HRel method achieves 93.19% accuracy, higher than GAL [42] method. In HRel-2 the highest percentage of parameters *i.e.*, 77.83% and FLOPs *i.e.*, 76.89% are pruned, and the accuracy can be observed to be better than HRank, CFP, and HBFP methods. PP-OC has the least accuracy drop. CFP and HRel methods have the next best accuracy drops with higher $P_f\%$ than PP-OC. Also, HRel-2 has a lesser accuracy drop compared to Hrank and HBFP methods.

4.D. ResNet-110 on CIFAR-10 Dataset

ResNet-110 contains 109 convolutional layers and 1 Fully connected layer in total. Similar to ResNet-56, except for the first convolutional layer, all the remaining convolutional layers are grouped into three different blocks, but each block contains 36 convolutional layers, with 16, 32 and 64 filters, respectively. The network is trained for 240 epochs with the initial learning rate of 0.1, which is divided by 10 at epoch numbers 88, 160, and 190 to achieve the baseline accuracy. Similar to ResNet-56, 1, 2, and 4 filters are pruned from each convolutional layer of 1^{st}, 2^{nd}, and 3^{rd} blocks, respectively. Note that similar to other methods, the first convolutional layer is not pruned. After pruning, the network is retrained for 70 epochs beginning with a learning rate of 0.01, which is divided by 10 at epochs 30 and 50. The remaining filters in the convolutional layer of each block for HRel-1 are 10, 20, and 38 and for HRel-2 are 8,15, and 30. HRel-1 reduces 62.1% of the FLOPs and achieves an accuracy of 93.03%, while Jordao *et al.* [31] and ABCPruner obtained better accuracies of 93.75% and 93.79% by pruning a slightly lesser percentage of filters *i.e.*, 60.17% and 60.30 respectively. NAS based method TAS achieves the highest accuracy of 94.33%. ABCPruner and LFPC have less accuracy drop compared to other methods with nearby $P_f\%$ values. In HRel-2, with 76.95% pruned FLOPs and 77.86% pruned parameters, higher accuracy and lower accuracy drop than HRank and HBFP methods are observed.

4.E. ResNet-50 on ImageNet

ImageNet dataset consists of 1.2 million training images and 50,000 test images belonging to 1000 classes. ResNet-50 has 49 convolutional layers and 1 Fully connected layer in total. Except for the first one, all convolutional layers are grouped into four different blocks, with each block having two 1×1 convolutional layers and one regular convolutional layer (*i.e.*, 3×3 kernel size). The network is initialized with the pre-trained weights on the ImageNet dataset and trained for 3 epochs with a learning rate of 0.0001 (to learn the kernel width required for estimating MI at different layers). The 8% of the convolutional layers from the first 3 blocks and 9% from the last block are pruned in every pruning iteration. Similar to other approaches first two convolutional layers from every block are pruned. After pruning, the network is retrained for 33 epochs beginning with a learning rate of 0.001, divided by 10 at epochs 10 and 25. The remaining filters for HRel-1, HRel-2 and HRel-3 from the convolutional layers in each block are [41,80,158,288], [33,60,117,203] and [27,48,92,154], respectively. After pruning 58.88% of the FLOPs, the proposed HRel method achieves 74.54% Top-1 accuracy and 92.12% Top-5 accuracy, with the least accuracy drop of 0.68 among the network compression methods shown in Table 5. In HRel-3 the highest percentage of parameters *i.e.*, 64.40% and FLOPs *i.e.*, 66.42% are pruned and Top-1 accuracy of 73.67% and Top-5 accuracy of 91.70% are observed. HRel method shows comparable performance with MetaPruning and LFPC in terms of Top-1 and Top-5 accuracies. However, it achieves higher Top-1 and Top-5 accuracies than SFP, ASFP, GAL, HRank, PP-OC, CFP and ABCPruner for comparable $P_f\%$. In terms of accuracy drop, the HRel method has the lowest Top-1 accuracy drop and second-best Top-5 accuracy drop than other network compression methods with comparable $P_f\%$. Compared to all the methods, HRel has the second-best accuracy drop next to DMCP.

The results observed using ResNet-56, ResNet-110, and ResNet-50 on CIFAR-10 and ImageNet datasets point out that the proposed HRel pruning method can prune the residual networks with very promising performance in terms of the accuracy as well as pruned FLOPs and parameters. Note that the proposed method has also shown very appealing performance for shallow (LeNet-5) and deep (VGG-16) plain models over different datasets. Overall, it can be deduced from the above experimental results that the proposed HRel pruning approach can retain the filters that are having high Relevance which leads to better accuracy even after pruning. The proposed pruning method is robust since high Relevance is used as the criterion in modeling the proposed HRel pruning strategy with the help of the IB theory.

4.F. Analysis of Information Plane Dynamics

IP dynamics for the architectures LeNet-5, VGG-16, and ResNet-56 are plotted using $I(X;L_i)$ and $I(L_i;Y)$ from the beginning of the training. Whereas for ResNet-110, values from the last ten epochs before reaching the baseline accuracy are observed due to its complex architecture. For ResNet-50, the values are observed only from the last epoch during the initial

Table 2: Pruning results of VGG-16 architecture over CIFAR-10 dataset. Note that the entries are sorted based on the $P_f\%$ in increasing order.

Method	**Acc**$_{baseline}$**%**	**Acc**$_{pruned}$**%**	**Acc↓**	**FLOPs**	$P_f\%$	**Params**	$P_p\%$
ℓ_1-norm [39]	93.25	93.40	-0.15	206.00M	34.30	5.40M	64.00
Ayinde *et al.* [3]	93.80	93.67	0.13	-	40.50	-	78.10
GAL [42]	93.96	90.78	3.18	171.89M	45.20	2.67M	82.20
CPGMI [37]	-	93.86	-	151.00M	51.80	1.99M	86.70
ABCPruner [41]	93.02	93.08	-0.06	82.81M	73.68	1.67M	88.68
CafeNet-E [61]	-	93.67	-	76.00M	-	1.40M	-
HRank [40]	93.96	91.23	2.73	73.70M	76.50	1.78M	92.00
VIB [13]	-	91.50	-	70.63M	77.48	-	-
MINT [16]	93.98	93.43	0.55	-	-	-	83.43
CFP [59]	93.49	92.90	0.59	56.70M	81.93		-
HBFP [5]	93.96	91.99	1.97	51.90M	83.42	2.40M	83.77
PP-OC [60]	93.49	93.43	0.06	48.80M	84.50	0.86M	94.30
HRel-1(ours)	**93.90**	**93.54**	**0.36**	**47.98M**	**84.70**	**0.75M**	**94.98**
HRel-2(ours)	**93.90**	**93.40**	**0.50**	**47.51M**	**84.85**	**0.75M**	**94.98**
Jordao *et al.* [31]	93.30	91.80	1.50	-	90.66	-	-

Table 3: Pruning results of ResNet-56 architecture over CIFAR-10 dataset.

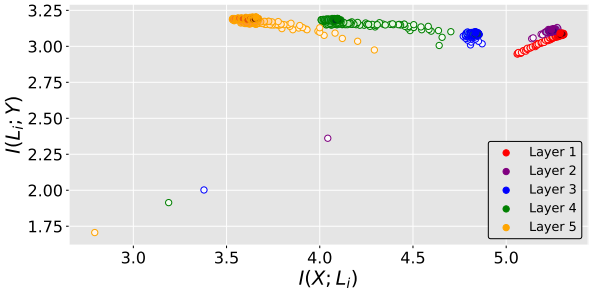
Method	**Acc**$_{baseline}$**%**	**Acc**$_{pruned}$**%**	**Acc↓**	**FLOPs**	$P_f\%$	**Params**	$P_p\%$
ℓ_1-norm [39]	93.04	93.06	-0.02	90.90M	27.60	0.73M	14.10
Ayinde *et al.* [3]	93.39	93.12	0.27	90.70M	27.90	0.65M	23.70
MINT [16]	92.55	93.02	-0.47	-	-	-	55.39
SFP [25]	93.59	92.26	1.33	59.40M	52.60	-	-
ASFP [24]	93.59	92.44	1.15	59.40M	52.60	-	-
TAS [14]	-	93.69	0.77	59.50M	52.70	-	-
LFPC [23]	93.59	93.34	0.25	59.10M	52.90	-	-
ABCPruner [41]	93.26	93.23	0.03	58.54M	54.13	0.39M	54.20
Jordao *et al.* [31]	-	93.71	-	-	57.06	-	-
GAL[42]	93.26	90.36	2.90	49.99M	60.20	0.29M	65.90
HRel-1(ours)	**93.80**	**93.19**	**0.61**	**47.57M**	**62.06**	**0.30M**	**63.76**
PP-OC [60]	93.10	93.15	-0.05	-	68.40	-	-
Hrank [40]	93.26	90.72	2.54	32.52M	74.10	0.27M	68.10
CFP [59]	93.57	92.63	0.93	29.50M	76.59	-	-
HBFP [5]	93.26	91.42	1.84	27.10M	78.43	0.19M	76.97
HRel-2(ours)	**93.80**	**92.70**	**1.10**	**28.99M**	**76.89**	**0.18M**	**77.83**

Table 4: Pruning results of ResNet-110 architecture over CIFAR-10 dataset.

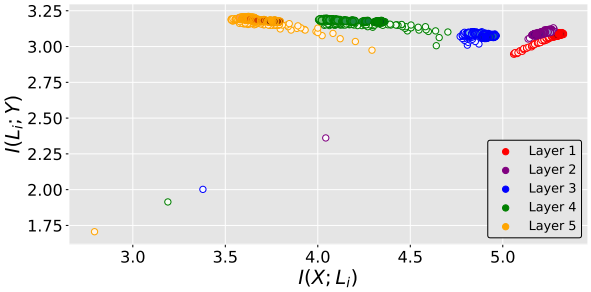
Method	**Acc**$_{baseline}$**%**	**Acc**$_{pruned}$**%**	**Acc↓**	**FLOPs**	$P_f\%$	**Params**	$P_p\%$
ℓ_1 - norm [39]	93.53	93.30	0.23	155.00M	38.70	1.16M	32.60
Ayinde *et al.* [3]	93.65	93.27	0.38	154.00M	39.10	1.13M	34.20
SFP [24]	93.68	93.38	0.30	150.00M	40.80	-	-
GAL [42]	93.35	92.55	0.80	130.20M	48.50	0.95M	44.80
ASFP [24]	93.68	93.10	0.58	121.00M	52.30	-	-
Jordao *et al.* [31]	-	93.75	-	-	60.17	-	-
TAS [14]	-	94.33	0.64	119.00M	53.00	-	-
LFPC [23]	93.68	93.79	-0.11	101.00M	60.30	-	-
HRel-1(ours)	**93.50**	**93.03**	**0.47**	**095.72M**	**62.14**	**0.62M**	**63.80**
ABCPruner [41]	93.50	93.58	-0.08	089.87M	65.04	0.56M	67.41
HRank [40]	93.50	92.65	0.85	079.30M	68.60	0.53M	68.70
HBFP [5]	93.50	91.96	1.54	063.30M	74.95	0.43M	74.92
HRel-2(ours)	**93.50**	**92.71**	**0.79**	**058.20M**	**76.95**	**0.38M**	**77.86**

Table 5: Pruning results of ResNet-50 architecture over ImageNet dataset.

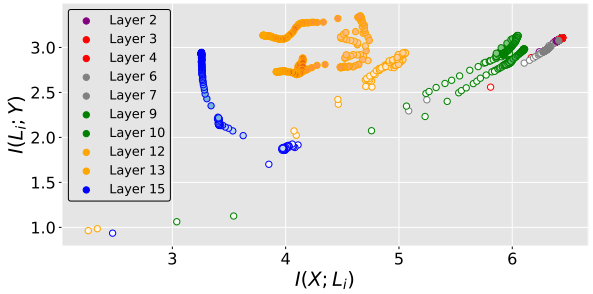
Method	**Top-1**	**Top-1**$_{pruned}$**%**	**Top-1↓**	**Top-5**	**Top-5**$_{pruned}$**%**	**Top-5↓**	**FLOPs**	$P_f\%$	**Params**	$P_p\%$
SFP [25]	76.15	62.14	14.01	92.87	84.60	08.27	-	41.80	-	-
ASFP [24]	76.15	75.53	00.62	92.87	92.73	00.14	-	41.80	-	-
GAL [42]	76.15	71.95	04.20	92.87	90.94	01.93	02.33B	43.03	21.20M	16.86
HRank [40]	76.15	74.98	01.17	92.87	92.33	00.54	02.30B	43.76	16.15M	36.80
TAS [14]	-	76.20	01.26	-	93.07	0.48	02.31B	43.50	-	-
DMCP [19]	76.60	76.20	00.40	-	-	-	02.20B	46.47	-	-
HRel-1(ours)	**76.15**	**75.47**	**00.68**	**92.87**	**92.60**	**00.27**	**02.11B**	**48.66**	**13.23M**	**48.24**
CafeNet-E [61]	77.80	76.90	00.90	-	93.10	-	02.00B	51.33	18.40M	27.84
MetaPruning [44]	76.60	75.40	01.20	-	-	-	02.00B	51.33	-	-
PP-OC [60]	-	-	-	92.20	92.10	00.10	-	-	15.70M	44.10
CFP [59]	-	-	-	92.20	91.40	00.80	-	-	-	49.60
ABCPruner [41]	76.01	73.52	02.49	92.96	91.51	01.45	01.79B	56.61	11.24M	56.01
HRel-2(ours)	**76.15**	**74.54**	**01.61**	**92.87**	**92.12**	**00.75**	**01.69B**	**58.88**	**10.82M**	**57.67**
LFPC [23]	76.15	74.46	01.69	92.87	92.04	00.83	-	60.80	-	-
HRel-3(ours)	**76.15**	**73.67**	**02.48**	**92.87**	**91.70**	**01.17**	**01.38B**	**66.42**	**09.10M**	**64.40**



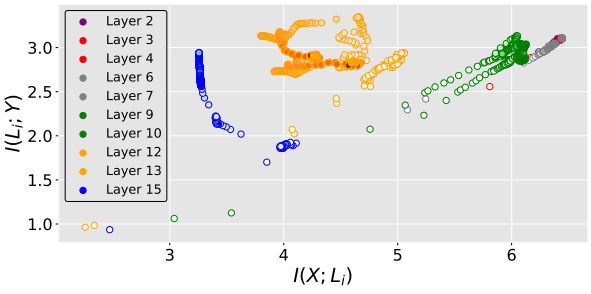
(a) LeNet-5 Without pruning



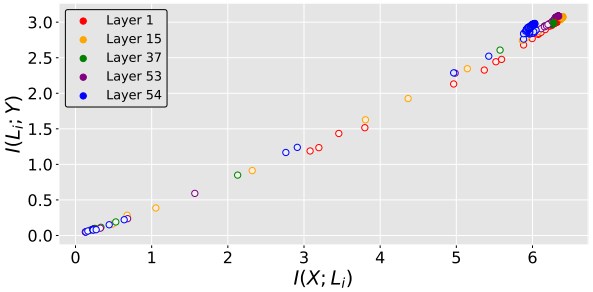
(b) LeNet-5 With pruning



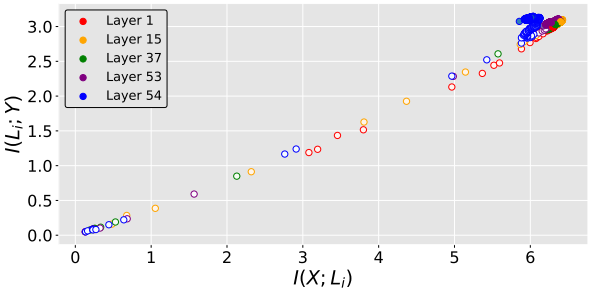
(c) VGG-16 Without pruning



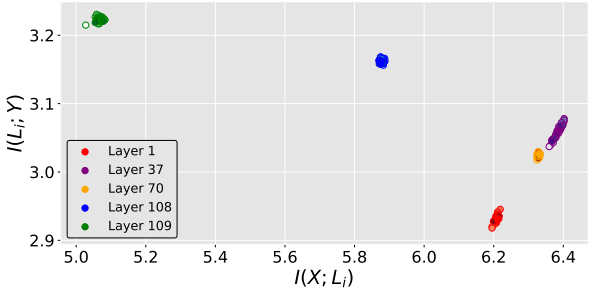
(d) VGG-16 With pruning



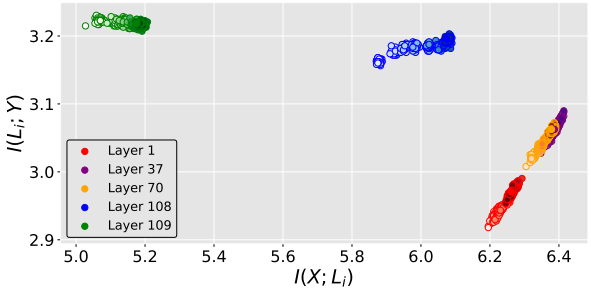
(e) ResNet-56 Without pruning



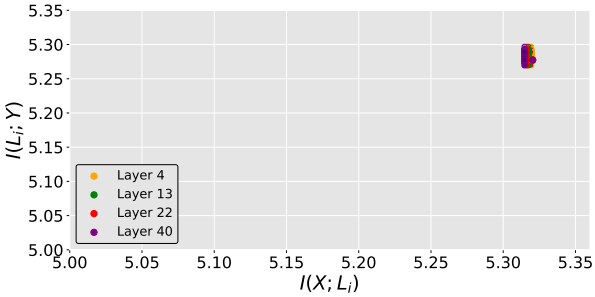
(f) ResNet-56 With pruning



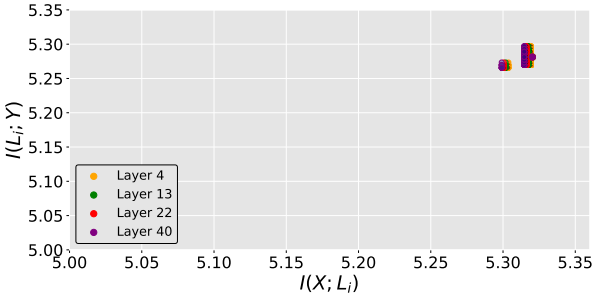
(g) ResNet-110 Without pruning



(h) ResNet-110 With pruning

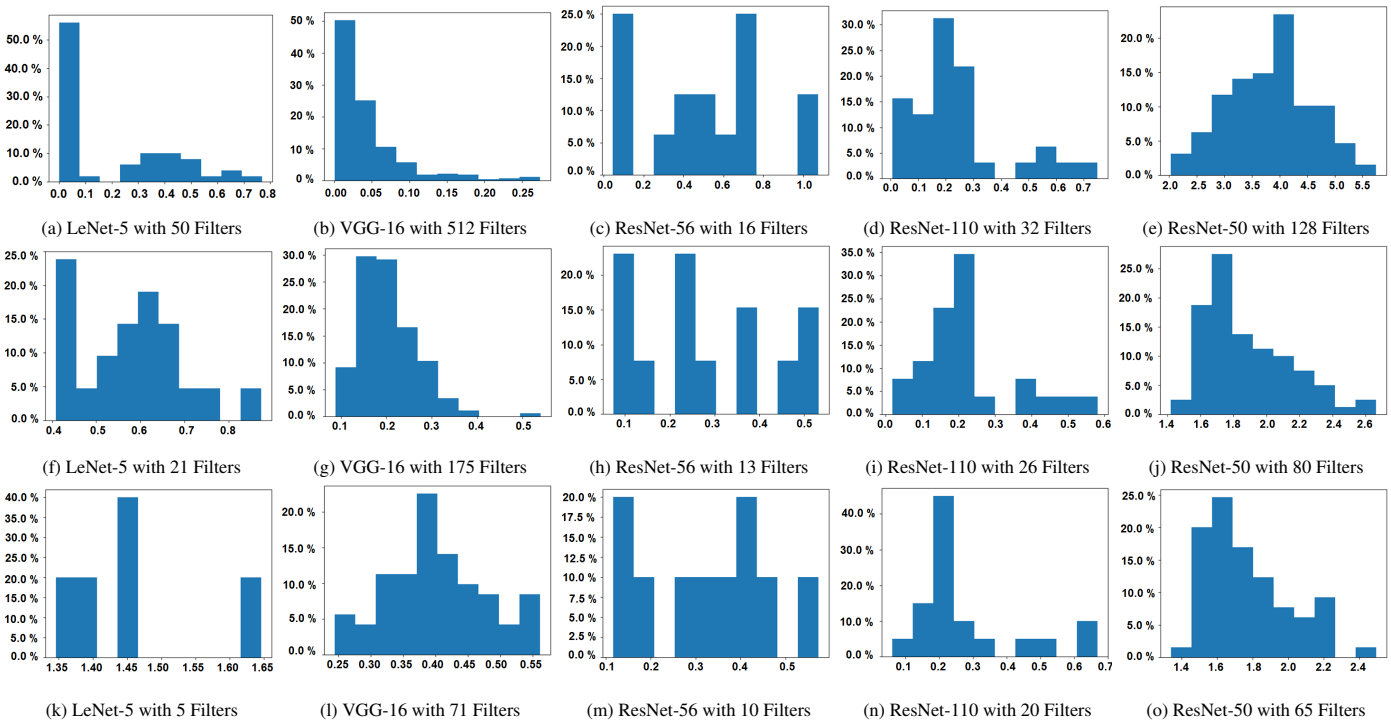


(i) ResNet-50 Without pruning



(j) ResNet-50 With pruning

Figure 6: Information Plane (IP) dynamics of LeNet-5, VGG-16, ResNet-56, ResNet-110, and ResNet-50 architectures. The left column corresponds to the IP dynamics of each architecture without pruning, and the right column shows the IP dynamics after pruning for the corresponding architecture. The layers are represented with different colors and saturation of each color indicates the progress of training.



(a) LeNet-5 with 50 Filters (b) VGG-16 with 512 Filters (c) ResNet-56 with 16 Filters (d) ResNet-110 with 32 Filters (e) ResNet-50 with 128 Filters

(f) LeNet-5 with 21 Filters (g) VGG-16 with 175 Filters (h) ResNet-56 with 13 Filters (i) ResNet-110 with 26 Filters (j) ResNet-50 with 80 Filters

(k) LeNet-5 with 5 Filters (l) VGG-16 with 71 Filters (m) ResNet-56 with 10 Filters (n) ResNet-110 with 20 Filters (o) ResNet-50 with 65 Filters

Figure 7: Distribution of the Relevance values of remaining filters from convolutional layers of different architectures across the pruning iterations (the number of remaining filters is specified for each sub-figure). For each sub-figure, X-axis denotes the range of the Relevance values. Y-axis denotes the percentage of filters having a corresponding Relevance value range. Convolutional layers 2, 9, 15, 37, and 34 of LeNet-5, VGG-16, ResNet-56, ResNet-110, and ResNet-50 respectively are used.

training due to comparatively more mini-batches for the ImageNet dataset. After initial training, for every pruning iteration, the last one epoch for ResNet-50 and the last ten epochs for the rest of the architectures are used to estimate the values $I(X;L_i)$ and $I(L_i;Y)$. From the IP dynamics of each architecture after pruning, *i.e.*, in the second column in Fig. 6, a slight decrease in the Relevance value of the final layers is observed compared to the architecture's IP dynamics without pruning (which is very less in the case of ResNet-50), is observed. This means that the network layers lose slight information concerning the class labels. This can be related to the small accuracy drop resulted from the pruning of filters. Though we preserve the filters with high Relevance, based on "Partial information decomposition" of MI [68, 73], the unique information (i.e., the information provided individually by few pruned filters) and their synergy (i.e., joint information provided only by the combination of few filters) is lost.

The MI estimator [67] used in the HRel method highly depends on the optimal kernel bandwidth of the dataset [62]. The original work [67] and a few related works [72, 73] using this estimator have used only the smaller datasets like CIFAR-10, MNIST, Fruits 360 [47], MADELON [20] etc. In these datasets, kernel bandwidth is chosen either by Silverman's rule of thumb [57] or by empirical evaluation over a range of values. Due to high dimensionality and more data samples in the ImageNet dataset, it is not feasible to obtain optimal kernel bandwidth for ImageNet using these methods. Hence, the same kernel bandwidth values specified in the MI estimator are used for input and class labels of the ImageNet dataset. Though similar and saturated values are observed for all layers in the Information Plane before and after pruning ResNet-50, the filters' Relevance values have shown enough variation among them as shown in Fig. 4, facilitating the selection of filters during pruning iterations. The optimal bandwidth for input and class labels of the ImageNet dataset can produce a better projection of the Information Plane of ResNet-50.

Overall, the IP dynamics indicate some information loss while pruning the filters, which is minimal for the proposed HRel method, as indicated by the experimental results. Thus, minimal information loss is acceptable as the complexity of the model is reduced drastically to facilitate the deployment of deep learning models over resource-constrained devices.

4.G. Analysis of Progression of Pruning using the Relevance Distribution

The distribution of the Relevance values of each architecture at the beginning of certain pruning iterations is shown in Fig. 7. Each column represents the Relevance value distribution for all the remaining filters in a given layer of the architecture. The first row is the distribution of the Relevance value before the beginning of the first pruning iteration. Subsequent rows can be identified by the remaining filters mentioned for each architecture. Note that the plot in Fig. 7 shows the percentage (%) of remaining filters for different ranges of Relevance values. From the Fig. 7a - 7d, 7f - 7n and 7k - 7n it is observed that in each column, with the increase in the pruning iterations, the

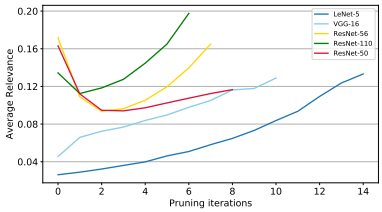


Figure 8: The average Relevance of all the layers across the pruning iterations for various architectures.

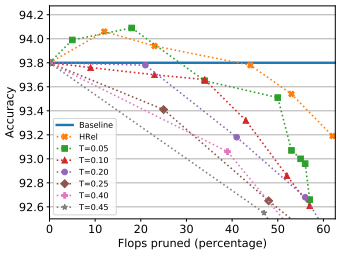


Figure 9: Accuracy of ResNet-56 architecture on CIFAR10 dataset when filters are globally pruned.

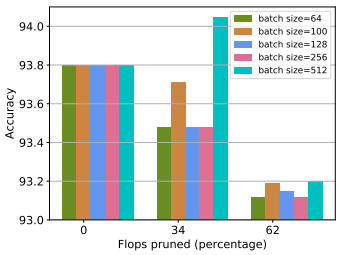


Figure 10: Accuracy of ResNet-56 architecure on CIFAR10 dataset for various batchsizes.

lowest Relevance value among the remaining filters increased. Also, in few architectures such as ResNet-56 and ResNet-110, though the lowest Relevance value did not change much, the percentage of filters having the lowest Relevance value is comparatively decreased after pruning.

It can be noticed that the distribution is slightly shifted towards the right side in most of the cases across the pruning iterations, which shows that the Relevance of the majority of the remaining filters is high. From Fig. 8. it can be observed that the average Relevance across the pruning iterations increased continuously for LeNet-5 and VGG-16. However, for ResNet-56 and ResNet-110 the average Relevance decreased initially for few pruning iterations and then increased. This observation further supports the proposed idea of utilization of high Relevance in the HRel pruning method. For ResNet-50 there is no much increment observed even after few pruning iterations in Fig. 8. The Relevance distribution values are also shifted to the left for ResNet-50 in Fig. 7. The Relevance distribution values can also be more accurate if the optimal kernel bandwidth is used.

4.H. Ablation study

An ablation study is conducted to understand the effect of global filter pruning based on Relevance values and the effect of batch size during the estimation of filters' Relevance.

1) Global pruning: The filters are compared globally based on their Relevance values in the global pruning method. As shown in Fig. 7, filters' Relevance values keep changing with the pruning iterations. Thus, in every pruning iteration, the estimated Relevance values of all the remaining filters across the layers are sorted and the maximum value from the least T% of the values is considered as the threshold. A new threshold value based on the filters' Relevance is used at every pruning iteration. Consequently, the filters with Relevance below the threshold are pruned. The experiments are conducted on ResNet-56 architecture using CIFAR-10 dataset with the values 5, 10, 20, 25, 40, and 45 for T. In the global pruning method, the lower T values resulted in better accuracy, as shown in Fig. 9. However, it can be observed that none of the global pruning results have achieved comparable accuracy with the proposed HRel method, where filters are ranked layer-wise based on their Relevance. Even the recent work [1] concludes that it is not suggested to compare the filters' Relevance across layers. However, it is a good estimator for the filter's importance when compared layer-wise. The global pruning method in [1] prunes the filters at a higher rate from the layers with relatively low Relevance filters. A similar observation is found in our ablation study. Filters are selectively pruned from a few layers at a higher rate, due to which more accuracy drop is observed, as depicted in Fig. 9.

2) Effect of batch size: Here, the effect of batch size during the computation of Relevance of filters is analyzed. Batch sizes of 64, 100, 128, 256, and 512 are used with ResNet-56 on CIFAR-10 dataset. The results obtained using the HRel method with a batch size of 100 for ResNet-56 are reported in Table 3. As illustrated in Fig. 10, it is observed that the batch size 64 resulted in comparatively lower accuracy. At the initial pruning iterations, the first and second-best performances are obtained with the batch size of 512 and 100, respectively. However, for higher pruned FLOPs, all the batch sizes produced similar results. Hence, the batch size during the estimation of filters' Relevance has less effect on the final performance of the HRel method.

5. Conclusion

In this paper, filters in CNNs are pruned based on their Relevance value. The Relevance measure is chosen based on IB theory, which is measured using the mutual information (MI) between the activations maps of the respective filters and the ground truths. The proposed HRel pruning method is evaluated on MNIST, CIFAR-10, and ImageNet datasets using LeNet-5, VGG-16, ResNet-56, ResNet-110, and ResNet-50 models. The pruning results obtained using the HRel method are superior compared to the current state-of-the-art pruning methods. The IP dynamics show the significance of the pruning criteria. The analysis of IP plane dynamics before and after pruning for the different CNNs suggests that the information loss after

pruning is negligible. The filters' Relevance is observed to increase from initial pruning iteration to final iteration except for ResNet-50 on ImageNet. The deployment of lightweight models that are pruned using the HRel method on edge devices such as drones, mobiles may be potential future research.

Acknowledgements

We thank Nvidia for donating two Titan X GPUs, which are used to perform the experiments of this research.

References

[1] RA Amjad, K Liu, and BC Geiger. Understanding neural networks and individual neuron importance via information-ordered cumulative ablation. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 2, 3, 11

[2] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016. 1

[3] Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, 118:148–158, 2019. 1, 2, 8

[4] Emilio Rafael Balda, Arash Behboodi, and Rudolf Mathar. An information theoretic view on learning of artificial neural networks. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–8. IEEE, 2018. 3

[5] SH Basha, Mohammad Farazuddin, Viswanath Pulabaigari, Shiv Ram Dubey, and Snehasis Mukherjee. Deep model compression based on the training history. *arXiv preprint arXiv:2102.00160*, 2021. 1, 2, 6, 7, 8

[6] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR, 2018. 3

[7] Aleksandar Botev, Guy Lever, and David Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1899–1903. IEEE, 2017. 6

[8] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. *arXiv preprint arXiv:1902.09037*, 2019. 3

[9] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015. 1

[10] Matthieu Courbariaux, Yoshua Bengio, and J. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015. 1

[11] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3123–3131, Cambridge, MA, USA, 2015. MIT Press. 1

[12] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. 3

[13] Bin Dai, Chen Zhu, Baining Guo, and David Wipf. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, pages 1135–1144. PMLR, 2018. 2, 3, 6, 7, 8

[14] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. *arXiv preprint arXiv:1905.09717*, 2019. 1, 2, 8

[15] Haytham M Fayek, Margaret Lech, and Lawrence Cavedon. Evaluating deep learning architectures for speech emotion recognition. *Neural Networks*, 92:60–68, 2017. 1

[16] Madan Ravi Ganesh, Jason J Corso, and Salimeh Yasaei Sekeh. Mint: Deep network compression via mutual information-based neuron trimming. *arXiv preprint arXiv:2003.08472*, 2020. 2, 3, 8

[17] Luis Gonzalo Sanchez Giraldo, Murali Rao, and Jose C Principe. Measures of entropy from data using infinitely divisible kernels. *IEEE Transactions on Information Theory*, 61(1):535–548, 2014. 4

[18] Ziv Goldfeld, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. Estimating information flow in deep neural networks. *arXiv preprint arXiv:1810.05728*, 2018. 3

[19] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1539–1547, 2020. 1, 8

[20] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. *Advances in neural information processing systems*, 17, 2004. 10

[21] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[23] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2009–2018, 2020. 1, 2, 8

[24] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, Chenggang Yan, and Yi Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE transactions on cybernetics*, 50(8):3594–3604, 2019. 1, 2, 8

[25] Y He, G Kang, X Dong, Y Fu, and Y Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018. 1, 2, 8

[26] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 1

[27] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 1

[28] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016. 2

[29] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *CoRR*, abs/1405.3866, 2014. 1

[30] Hlynur Jónsson, Giovanni Cherubini, and Evangelos Eleftheriou. Convergence behavior of dnns with mutual-information-based regularization. *Entropy*, 22(7):727, 2020. 3

[31] Artur Jordao, Fernando Yamada, and William Robson Schwartz. Deep network compression based on partial least squares. *Neurocomputing*, 406:234–243, 2020. 2, 7, 8

[32] Artemy Kolchinsky and Brendan D Tracey. Estimating mixture entropy with pairwise distances. *Entropy*, 19(7):361, 2017. 3

[33] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004. 3

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[35] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. 5

[36] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 1

[37] Min Kyu Lee, Seunghyun Lee, Sang Hyuk Lee, and Byung Cheol Song. Channel pruning via gradient of mutual information for light-weight convolutional neural networks. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1751–1755. IEEE, 2020. 2, 3, 8

[38] Nikolai Leonenko, Luc Pronzato, Vippal Savani, et al. A class of rényi information estimators for multidimensional densities. *Annals of statistics*, 36(5):2153–2182, 2008. 3

[39] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 2, 8

[40] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020. 1, 2, 6, 8

[41] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 673 – 679, 2020. 1, 2, 8

[42] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019. 2, 7, 8

[43] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 1

[44] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3296–3305, 2019. 1, 2, 8

[45] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 2

[46] Chuhan Min, Aosen Wang, Yiran Chen, Wenyao Xu, and Xin Chen. 2pfpce: Two-phase filter pruning based on conditional entropy. *arXiv preprint arXiv:1809.02220*, 2018. 2, 3

[47] Horea MURESAN and Mihai OLTEAN. Fruit recognition from images using deep learning. *Acta Univ. Sapientiae*, 10(1):26–42, 2018. 10

[48] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002. 4

[49] Morteza Noshad, Yu Zeng, and Alfred O Hero. Scalable mutual information estimation using dependence graphs. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2962–2966. IEEE, 2019. 3

[50] Sri Purwani, Julita Nahar, and Carole Twining. Analyzing bin-width effect on the computed entropy. In *AIP Conference Proceedings*, volume 1868, page 040008. AIP Publishing LLC, 2017. 3

[51] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[52] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[53] A. Romero, Nicolas Ballas, S. Kahou, Antoine Chassang, C. Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2015. 1

[54] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 5

[55] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019. 3

[56] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017. 2, 3

[57] Bernard W Silverman. Monographs on statistics and applied probability. *Density estimation for statistics and data analysis*, 26, 1986. 10

[58] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[59] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay Namboodiri. Leveraging filter correlations for deep model compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 835–844, 2020. 1, 2, 6, 7, 8

[60] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P Namboodiri. Acceleration of deep convolutional neural networks using adaptive filter pruning. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):838–847, 2020. 1, 2, 6, 7, 8

[61] Xiu Su, Shan You, Tao Huang, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. Locally free weight sharing for network width search. In *International Conference on Learning Representations*, 2020. 1, 2, 8

[62] Nicolás I Tapia and Pablo A Estévez. On the information plane of autoencoders. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. 10

[63] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000. 2

[64] Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS computational biology*, 13(1):e1005324, 2017. 1

[65] Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. Cop: Customized deep model compression via regularized correlation-based filter-level pruning. *arXiv preprint arXiv:1906.10337*, 2019. 2

[66] Liangjian Wen, Xuanyang Zhang, Haoli Bai, and Zenglin Xu. Structured pruning of recurrent neural networks through neuron selection. *Neural Networks*, 123:134–141, 2020. 2

[67] Kristoffer Wickstrøm, Sigurd Løkse, Michael Kampffmeyer, Shujian Yu, Jose Principe, and Robert Jenssen. Information plane analysis of deep neural networks via matrix-based renyi's entropy and tensor kernels. *arXiv preprint arXiv:1909.11396*, 2019. 2, 3, 4, 6, 10

[68] P. L. Williams and R. Beer. Nonnegative decomposition of multivariate information. *ArXiv*, abs/1004.2515, 2010. 10

[69] J. Wu, C. Leng, Yuhang Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4820–4828, 2016. 1

[70] Salimeh Yasaei Sekeh and Alfred O Hero. Geometric estimation of multivariate dependency. *Entropy*, 21(8):787, 2019. 3

[71] Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019. 1

[72] Shujian Yu, Luis Gonzalo Sanchez Giraldo, Robert Jenssen, and Jose C. Principe. Multivariate extension of matrix-based renyi's α-order entropy functional, 2019. 10

[73] Shujian Yu, Kristoffer Wickstrøm, Robert Jenssen, and José C Príncipe. Understanding convolutional neural networks with information theory: An initial exploration. *IEEE transactions on neural networks and learning systems*, 2020. 10