Slimming Neural Networks Using Adaptive Connectivity Scores

Madan Ravi Ganesh, Dawsin Blanchard, Jason J. Corso, *Senior Member, IEEE*, and Salimeh Yasaei Sekeh, *Member, IEEE*

***Abstract*— In general, deep neural network (DNN) pruning methods fall into two categories: 1) weight-based deterministic constraints and 2) probabilistic frameworks. While each approach has its merits and limitations, there are a set of common practical issues such as trial-and-error to analyze sensitivity and hyper-parameters to prune DNNs, which plague them both. In this work, we propose a new single-shot, fully automated pruning algorithm called *slimming neural networks using adaptive connectivity scores* (SNACS). Our proposed approach combines a probabilistic pruning framework with constraints on the underlying weight matrices, via a novel connectivity measure, at multiple levels to capitalize on the strengths of both approaches while solving their deficiencies. In SNACS, we propose a fast hash-based estimator of *adaptive conditional mutual information* (ACMI), that uses a weight-based scaling criterion, to evaluate the connectivity between filters and prune unimportant ones. To automatically determine the limit up to which a layer can be pruned, we propose a set of operating constraints that jointly define the upper pruning percentage limits across all the layers in a deep network. Finally, we define a novel *sensitivity* criterion for filters that measures the strength of their contributions to the succeeding layer and highlights critical filters that need to be completely protected from pruning. Through our experimental validation, we show that SNACS is faster by over 17× the nearest comparable method and is the state-of-the-art single-shot pruning method across four standard Dataset-DNN pruning benchmarks: CIFAR10-VGG16, CIFAR10-ResNet56, CIFAR10-MobileNetv2, and ILSVRC2012-ResNet50.**

***Index Terms*— Multivariate dependency measure, mutual information (MI), neural network compression, pruning, sensitivity.**

I. Introduction

CRITICAL real-world applications like autonomous vehicle navigation [1], [2] and simultaneous machine translation [3], [4] demand real-time response [5] without any compromise in performance. Proposed solutions in these application domains are implicitly bound to constraints brought forward by restricted space and memory availability on custom hardware implementations. These factors are at odds with the general research design goal of high performance in deep neural networks (DNNs), which is often achieved by increasing the overall size and capacity of the DNN. The trade-off between these constraints has brought increased attention to the field of DNN pruning [6], [7], the main objective of which is to maintain an adequate level of performance, often within a few percent of the original DNN, while only using a fraction of its memory or FLOPs. Of course, the adequacy of any level of performance depends on the specific application, but the general goal nevertheless remains critical.

There are two main approaches to pruning: 1) deterministic constraints on weight matrices [8], [9], [10] and 2) probabilistic frameworks [11], [12], [13]. Methods based on deterministic constraints on weight matrices are straightforward to implement and leverage the underlying structure of the weight matrices, but they often do not account for the downstream impact of pruning filters. On the other hand, probabilistic frameworks focus on reducing the redundancy between layers using information theoretic measures or variational Bayesian inference but are not fast or efficient at modeling the sensitivity of filters at an individual level. In a sense, the two types of methods are converses: one's weakness is remedied by the other. Yet, to the best of our knowledge, there has been no recent work that combines both approaches and improves upon them. Furthermore, there are many unresolved practical issues among both approaches, e.g., the labor-intensive process of analyzing the sensitivity of different layers to pruning or imposing an upper limit on the pruning percentage for each layer and the number of resources and time spent in iteratively pruning DNNs.

To that end, we are able to unify the benefits of both methods while mitigating their respective drawbacks: we propose *slimming neural networks using adaptive connectivity scores* (SNACS) as a hybrid single-shot pruning approach. In SNACS, we introduce the *adaptive conditional mutual information* (ACMI) measure, which incorporates weights as a scaling function within the framework of conditional mutual information (MI) [14], [15]. The ACMI measure evaluates the connectivity between pairs of filters across adjacent layers and prunes unimportant filters. In this work, we explore weight and activation-based scaling functions.

To remove the manual effort involved in setting the upper pruning percentage limit of layers, we define a set of operating

Manuscript received 18 February 2021; revised 18 August 2021 and 15 January 2022; accepted 22 July 2022. Date of publication 23 August 2022; date of current version 1 March 2024. The work of Madan Ravi Ganesh and Jason J. Corso was supported in part by the Google Faculty Research Award and in part by the National Institute of Standards and Technology (NIST) under Grant 60NANB17D191. The work of Salimeh Yasaei Sekeh was supported by NSF under Grant 1920908 and Grant 2053480. *(Corresponding author: Madan Ravi Ganesh.)*

Madan Ravi Ganesh is with the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: madantrg@umich.edu).

Dawsin Blanchard and Salimeh Yasaei Sekeh are with the School of Computing and Information Science, University of Maine, Orono, ME 04469 USA.

Jason J. Corso is with the Departments of Robotics and Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2022.3198580, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2022.3198580

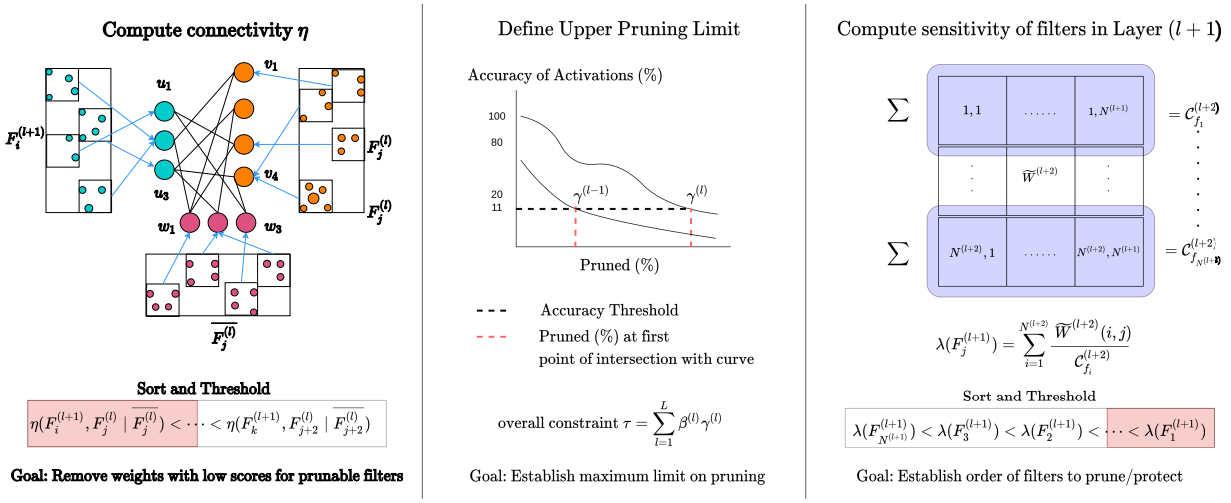


Fig. 1. Illustration of the three major components of SNACS that help prune connections between layer l and $l+1$. First, we propose the hash-based ACMI estimator to compute the connectivity scores between filters in layer $l+1$ and all the filters in layer l. These connectivity scores are thresholded to obtain the set of filters that need to be pruned. Next, to protect the network from being irregularly/excessively pruned, we use a custom set of operating constraints, based on the degradation of activation quality at various pruning levels, to decide on the upper pruning percentage limit for layer $l+1$. Finally, we compute the sensitivity of filters in $l+1$ as the sum of normalized weights between chosen filters in layer $l+1$ and all the filters in layer $l+2$. We sort and threshold the sensitivity values to create a subset of sensitive filters that should be protected from pruning. Combining the information from all three components, we prune layer $l+1$.

constraints to automatically evaluate them. The constraints are based on the degradation in quality of activations at various levels of compression. Additionally, we encapsulate the importance of a filter using our proposed *Sensitivity* criterion, defined as the sum of a filter's contributions (normalized weights) to filters in the succeeding layer. Using this measure, we curate a subset of relatively less sensitive filters that can be pruned based on their connectivity scores, while we protect highly sensitive filters from any form of pruning. We highlight all the main components of SNACS in Fig. 1.

Overall, we summarize our contributions in this work as follows.

1) We propose a hybrid single-shot pruning approach, SNACS, which takes advantage of both a probabilistic pruning framework and simple weight-based constraints.
2) In SNACS, we propose the use of ACMI as a way to measure the connectivity between filters and derive its hash-table-based implementation.
3) In the interest of simplifying the process of defining upper pruning percentage limits of layers in a DNN, we propose a set of operating constraints to help automate their definition.
4) We apply a custom notion of *Sensitivity* in filters, using their contribution to succeeding layers, to prioritize the pruning of largely insensitive filters while protecting highly sensitive ones.

By incorporating our contributions within the SNACS framework, we improve the overall run-time of the pruning algorithm by upward of 17×, increase the accuracy of the estimator, and create an entirely automated pruning pipeline while offering state-of-the-art performance in single-shot pruning of DNNs.

II. Related Works

In Section II, we discuss prior works in pruning and MI estimators, as well as methods at their intersection. Among pruning approaches there are two broad categories: 1) methods that use a deterministic constraint on the weight matrices and 2) methods that use a probabilistic framework to reduce the redundancy and maintain the flow of information between layers. Within the first category of methods, there is a subset that enforces sparsity by modifying the objective function while the remaining directly apply constraints on the weight matrices.

A. Deterministic Constraints on Weight Matrix

1) Direct Constraint on Weight Matrices: Some of the earliest works in pruning use the second-order relationship between the objective function and weights of the network to evaluate and remove unimportant values [16], [17]. Since then, several advancements in the form of directly thresholding weights [10], [18] or using the $||\cdot||_1$ constraint to define the importance of filters [19] have been proposed. A more recent subset of methods have adopted data-driven logic to derive the importance of filter weights. Two such methods are ThiNet [9] and NISP [20], where the reconstruction of outcomes with the removal of weights is posed as a post-training objective. By virtue of how direct constraints are placed on weight matrices, they often do not account for the downstream impact of pruning or are built on the assumption of a purely deterministic relationship between filters. Instead, we use the combination of a weight-based scaling function and filter connectivity within a probabilistic framework to maintain the flow of information between layers and overcome these issues.

2) Modification of Objective Function: Inducing sparsity in weight matrices by modifying the objective function involves imposing a strong constraint on how weights develop during training. Constraints range from simple methods, such as single or multiple l_n norms [21] on channel outputs, simple patterned masks to regulate group sparsity [22], and optimization

over group-lasso-based objective functions [8], [23], to more complicated ideas like balancing individual versus group sparsity constraints [24], [25] and adding discrimination-aware losses at intermediate layers to enhance and easily identify important channels [26].

More recent methods combine the idea of modifying the objective function with more abstract concepts like meta-learning [27], where sparsity-inducing regularizers are used to learn latent vectors that help decide on the weight values directly or GANs, where an adversarial pruned network generator optimizes a loss based on the features derived from the original network [28]. To provide a controlled setup to study and compare the effects of pruning a network against its original counterpart, we avoid strong comparisons against methods that modify the objective function. Apart from optimizing over a fundamentally different objective function, which are harder to optimize, these methods require multiple iterations of pruning and fine-tuning built-in to their setup, while we use only a single pruning and retraining step to optimize a simple objective function.

B. Probabilistic Frameworks

Pruning approaches that use probabilistic frameworks can be divided into Bayesian and non-Bayesian methods. Bayesian methods apply a variational Bayesian inference perspective to pruning, with a focus on estimating the posterior distribution of weights using ELBO [29], [30]. While they offer a theoretically sound perspective to pruning, they require a strong assumption on the prior distribution of weights which induces sparsity across the network. Furthermore, their performances on large-scale datasets have more room to grow.

The non-Bayesian approach to pruning focuses on using information-theoretic measures, with minimal assumptions and widespread applicability when compared to the Bayesian methods. These include Luo and Wu [13], in which entropy of activations is used as a measure of the importance of a filter, VIBNet [12], where the information bottleneck principle is used to minimize the redundancy between adjacent layers and MINT [11], in which geometric conditional MI is used to determine the dependencies between filter pairs in adjacent layers. While they are adept at reducing redundancy and maintaining the flow of information between layers, they are slow and inefficient at modeling the sensitivity of individual filters to pruning. In SNACS, we propose the use of ACMI to improve the speed of dependency computations for MI as well as the accuracy of the estimates. Furthermore, by highlighting sensitive filters that need to remain unpruned and jointly defining the upper pruning percentage limit of layers we obtain additional gains when pruning a DNN.

C. Multivariate Dependency Measures

Approaches for estimating multivariate dependencies using MI can be broadly classified into two categories: plugin and direct estimation. Plugin estimators like Kernel density estimators (KDEs) [31], KNN estimators [32], and others [33], [34] form the bulk of early works in computing multivariate dependency. However, plugin estimators need to accurately estimate the probability density function of input variables. This, when combined with their large run-time complexity, renders them highly un-scalable. To overcome these issues, direct estimators for Renyi-entropy and MI [33], [34], and Henze–Penrose divergence measure [35] have been proposed. They provide manageable run-time complexity while avoiding direct knowledge of the density function. Crucial to the functioning of many direct estimation methods is the use of graph-theoretic ideas, such as the nearest-neighbor ratios [36], which uses the k-NN graph to estimate MI, and the minimum spanning tree used to estimate the GMI [37]. These graph-based approaches help make the evaluation of MI computationally tractable. While most methods fall into either plugin or direct categories, recent work has focused on the development of a hybrid approach [38]. This approach combines the fast run-time implementation of hash-tables with an error convergence rate akin to plugin methods, thus merging the advantages of both the estimation approaches.

III. Algorithm and Component Description

In the following sections, we outline SNACS's algorithm. We follow it by providing details of the ACMI measure and the set of constraints that automatically define the upper pruning percentage limits of layers in a DNN. The final section explains our notion of sensitivity, which identifies and protects important filters from being pruned.

A. Notation

We assume that a given DNN has a total of L layers, where

1) *Sensitive filters()*: Function that returns the indices of a subset of filters that need to be protected from pruning, as computed using sensitivity (Section III-E).
2) $F_i^{(l+1)}$: Activations from the selected filter i in layer $l+1$. We also overload this notation to represent the indexing scheme of the selected filter.
3) $N^{(l+1)}$: Total number of filters in layer $l+1$.
4) $S_{F_i^{(l+1)}}$: The set of filter indices whose values are pruned from the weight vector.
5) $\eta()$: Connectivity score between two filters computed using ACMI (Section III-C).
6) $\overline{F_j^{(l)}}$: Activations from the set of all filters excluding $F_j^{(l)}$ in layer l. We also overload this notation to represent the indexing scheme of the selected filters.
7) δ: Threshold on connectivity scores to ensure only strong connections are retained.
8) $\gamma^{(l+1)}$: Upper limit on pruning percentage for layer $l+1$ defined using the constraints in Section III-D.
9) $W^{(l+2)}$: Weight matrix of layer $l+2$.
10) $\widetilde{W}^{(l+2)}(i, j)$: Indexing element in ith row and jth column of the weight matrix of layer $l+2$ averaged over the height and width dimensions.

B. Algorithm

The overall goal of our algorithm is to find the set of filters that contribute minimally to the flow of information between layers and prune their values from the weight matrix. We apply

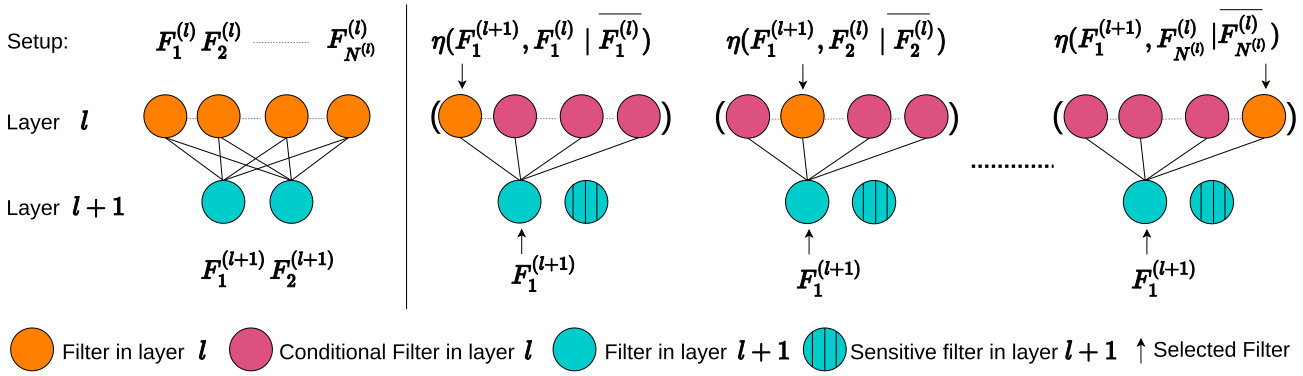


Fig. 2. Illustrative example of computing ACMI, $\eta()$, between activations of filters in layers $l+1$ and l. In each $\eta()$ computation, the arrows indicate the filters between which we compute the connectivity score while taking into consideration the activations from the remaining filters in layer l. We repeat these steps for every possible pair of filters except for highly-sensitive filters in layer $l+1$, where η need not be computed since their connections (lines between filters) will not be pruned.

SNACS between every pair of adjacent layers in a pretrained DNN where

1) We identify a subset of sensitive filters in layer $l+1$ that need to be protected from pruning and iterate over the remaining insensitive filters in layer $l+1$.
2) To measure the connectivity score, η, between filters in layers l and $l+1$, we apply our proposed hash-based ACMI estimator to the activations from each set of filters. Fig. 2 shows an example of this process. The connectivity score evaluates the strength of the relationship between two filters in the context of contributions from all the remaining filters in layer l.
3) If the connectivity score is lower than a threshold level δ, and the number of pruned filters does not exceed the predetermined upper limit, denoted by $\gamma^{(l+1)}$, we add the index of the filter to $S_{F_i^{(l+1)}}$. The weights for retained and protected filters/neurons are untouched, while we zero out the weights for the entire kernel/elements in pruned filters/neurons.

In the practical implementation of Algorithm 1, we determine the value of δ by thresholding η values from a chosen layer to remove sufficient weights and match the predetermined γ^{l+1}. Once we prune the filters that contribute the least across all the layers of the DNN, we proceed to retraining the network using a setup that mirrors the training phase of the pretrained DNN. Across Algorithm 1, we note that SNACS does not contain a continual feedback loop to update weights when pruning. Instead, we take only a single retraining pass after pruning. Compared to iterative pruning approaches, which often continually fine-tune to compensate for the performance lost due to pruning, SNACS falls firmly in the domain of single-shot pruning methods.

Algorithm 1: SNACS Pruning Between Filters of Layers $(l, l+1)$

for *Every pair of layers* $(l, l+1), l \in 1, 2, \ldots, L-1$ **do**
  Compute $\gamma^{(l+1)}$;
  for $F_i^{(l+1)}, i \in \{1, 2, \ldots N^{(l+1)}\} \setminus$ SENSITIVE_FILTERS$\left(\{1, 2, \ldots N^{(l+1)}\}\right)$ **do**
    Initialize $S_{F_i^{(l+1)}} = \emptyset$;
    for $F_j^{(l)}, j \in 1, 2, \ldots N^{(l)}$ **do**
      Compute $\eta(F_i^{(l+1)}, F_j^{(l)} | \overline{F_j^{(l)}})$;
      if $\left(\eta(F_i^{(l+1)}, F_j^{(l)} | \overline{F_j^{(l)}}) \leq \delta$ ***and*** $\sum_i |S_{F_i^{(l+1)}}| / (N^{(l+1)} N^{(l)}) < \gamma^{(l+1)}\right)$ **then**
        $S_{F_i^{(l+1)}} = S_{F_i^{(l+1)}} \cup \text{index}(F_j^{(l)})$
      end
    end
  end
end

C. Adaptive Conditional Mutual Information

In this section, we introduce adaptive MI (AMI), a nonlinear dependency measure that is based on the f-divergence measure [15], [39], extend it to a conditional formulation, and discuss the hash-table-based estimator used to compute ACMI.

1) Definition: Let $\mathcal{X}$ and $\mathcal{Y}$ be Euclidean spaces and let P_{XY} be a probability measure on the space $\mathcal{X} \times \mathcal{Y}$. Here, P_X and P_Y define the marginal probability measures. Similar to [14], for given function $(x, y) \in \mathcal{X} \times \mathcal{Y} \mapsto \varphi(x, y) \geq 0$, the AMI, denoted by $I_\varphi(X; Y)$, is defined as

$$I_\varphi(X; Y) = \mathop{\mathbb{E}}_{P_X P_Y}\left[\varphi(X, Y) g\left(\frac{dP_{XY}}{dP_X P_Y}\right)\right] \tag{1}$$

where $(dP_{XY}/dP_X P_Y)$ is the Radon–Nikodym derivative, and $g : (0, \infty) \mapsto \mathbb{R}$ is a convex function and $g(1) = 0$. Note that when $(dP_{XY}/dP_X P_Y) \to 1$ then $I_\varphi \to 0$. The overall bounds on the AMI measure are given by

$$0 \leq I_\varphi(X, Y) \leq \frac{1}{2} \mathop{\mathbb{E}}_{P_X P_Y}\left[\varphi(X, Y)\left(\frac{dP_{XY}}{dP_X P_Y} + 1\right)\right]. \tag{2}$$

An explanation of how we arrive at these bounds is provided in Appendix A.

2) Adaptive Conditional Mutual Information: Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be Euclidean spaces and let P_{XYZ} be a probability measure on the space $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. We presume $P_{XY|Z}$, $P_{X|Z}$,

and $P_{Y|Z}$ are the joint and marginal conditional probability measures, respectively. P_Z defines the marginal probability measure on the space $\mathcal{Z}$. Following [14], the ACMI, denoted by $I_\varphi(X;Y|Z)$, is defined as

$$I_\varphi(X;Y|Z) = \mathop{\mathbb{E}}_{P_Z P_{X|Z} P_{Y|Z}}\left[\varphi(X,Y,Z) g\left(\frac{dP_{XY|Z}}{dP_{X|Z}P_{Y|Z}}\right)\right]. \quad (3)$$

In this article, we focus on the particular case of $g(t) = ((t-1)^2/2(t+1))$, introduced in [40]. Using this formulation allows for $I_\varphi \in [0,1]$ and symmetric behavior, which are critical to the functioning of our algorithm, as well as a number of other properties which allow for statistical analysis, as highlighted in [41]. Note that when $\varphi = 1$, the ACMI in (3) becomes the conditional geometric MI measure proposed in [37]. Next, we propose a hash-based estimator of ACMI to approximate the connectivity score between filters.

3) Hash-Based Estimator of ACMI: Consider N i.i.d samples $\{(X_i, Y_i, Z_i)\}_{i=1}^N$ drawn from P_{XYZ}, which is defined on the space $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$. We define a dependence graph $G(X,Y,Z)$ as a directed multipartite graph, consisting of three sets of nodes V, U, and W, with cardinalities denoted as $|V|$, $|U|$, and $|W|$, respectively and with the set of all edges E_G. The variable W here is different from the DNN weight matrix. Following similar arguments to [38], we map each point in the sets $\mathbf{X} = \{X_1, \ldots, X_N\}$, $\mathbf{Y} = \{Y_1, \ldots, Y_N\}$, and $\mathbf{Z} = \{Z_1, \ldots, Z_N\}$ to the nodes in the sets V, U, and W, respectively, using the hash function H.

Here, $H(x) = H_2(H_1(x))$, where the vector valued hash function $H_1 : \mathbb{R}^d \mapsto \mathbb{Z}^d$ is defined as $H_1(x) = [h_1(x), \ldots, h_1(x_d)]$, for $x = [x_1, \ldots, x_d]$ and $h_1(x_i) = \lfloor (x_i + b/\epsilon) \rfloor$, for a fixed $\epsilon > 0$, and random variable $b \in [0, \epsilon]$. The random hash function $H_2 : \mathbb{Z}^d \mapsto \mathcal{F}$ is uniformly distributed on the output $\mathcal{F} = \{1, 2, \ldots, F\}$ where for a fixed tunable integer c_H, $F = c_H N$.

After the projection of values on to the dependence graph $G(X,Y,Z)$, we define the following cardinality:

$$N_{ijk} = \#\{(X_t, Y_t, Z_t) \quad \text{s.t. } H(X_t) = i, \; H(Y_t) = j, H(Z_t) = k\} \quad (4)$$

which is the number of joint collisions of the nodes (X_t, Y_t, Z_t) at the triple (v_i, u_j, ω_k). Let N_{ik}, N_{jk}, and N_k be the number of collisions at the vertices (v_i, ω_k), (u_j, ω_k), and ω_k, respectively. By using N_{ijk}, N_{ik}, N_{jk}, and N_k, we define the following ratios:

$$r_{ijk} := \frac{N_{ijk}}{N}, \quad r_{ik} := \frac{N_{ik}}{N}, \; r_{jk} := \frac{N_{jk}}{N}, \; r_k := \frac{N_k}{N}. \quad (5)$$

Finally, using the above ratios we propose the following hash-based estimator of the ACMI measure (3):

$$\widehat{I}_\varphi(X;Y|Z) = \sum_{e_{ijk} \in E_G} \varphi(i,j,k) \, \frac{r_{ik}\, r_{jk}}{r_k} g\left(\frac{r_{ijk}\, r_k}{r_{ik}\, r_{jk}}\right) \quad (6)$$

summed over all edges e_{ijk} of $G(X,Y,Z)$ having nonzero ratios.

Theorem 1: For given $g(t) = ((t-1)^2/2(t+1))$ and under the assumptions: (A1) The support sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ are bounded. (A2) The function φ is bounded. (A3) The continuous marginal, joint, and conditional density functions are belong to Hölder continuous class, [42]. For fixed d_X, d_Y, and d_Y, as $N \to \infty$ we have

$$\widehat{I}_\varphi(X;Y|Z) \longrightarrow I_\varphi(X;Y|Z), \quad a.s. \quad (7)$$

The proof of Theorem 1 is available in the Appendices.

4) Implementation: Overall, X, Y, and Z denote sets of activations derived from different filters, and we obtain a scalar value (connectivity score) as the outcome of the ACMI estimator in (6). The flexibility in defining function φ offers a way to connect the probabilistic framework of MI to existing weight-based pruning approaches. In Section IV, we explore a variety of options for φ and empirically determine that a function defined on the weight matrix helps achieve the highest pruning performance in our experiments.

D. Definition of Upper Pruning Percentage Limit of Layers

To protect different layers of the DNN from being excessively pruned, we propose a set of operating constraints to automate the joint definition of the upper pruning percentage limits of every layer in the DNN. Our approach follows the trends in degradation of the quality of activations when we prune a layer to varying extents. At each layer, we collect the performances of an SVM model with an RBF kernel ($\alpha_c^{(l)}$), trained on a subset of activations from the un-pruned version of the layer and tested on the same subset from the pruned version of the layer at various compression levels c, where $c \in \{1, 2, 3, \ldots, 99\}$. Here, we use the ground-truth labels from the dataset to train the SVM model.

Once we have the performance of SVM models across all layers, we cycle through them, from 100% to 0%, to find the optimal threshold value such that the sum of compression levels of all the layers dictated by the selected threshold adds up to our overall target pruning percentage. Each layer's pruning percentage is dictated by the highest compression level where the SVM model's performance exceeds the chosen threshold. The general trend we observe is higher the compression level, the lower the SVM model's performance. Thus, picking smaller performance thresholds leads to the selection of higher compression levels in a layer. We select the highest compression level from a range of possible values to compensate for noisy and inconsistent behavior in SVM performances. Mathematically, we optimize

$$\tau = \sum_{l=1}^{L} \beta^{(l)} \gamma^{(l)} \quad (8)$$

where $\beta^{(l)}$ is the ratio of the number of parameters in layer l to the total number of parameters across the entire DNN, and τ denotes the desired pruning percentage across the DNN. Fig. 3 illustrates this process using an example of four layers. It is important to note that the statistics computed from the SVM models across all layers can be executed in parallel, at an average of 36 s per SVM model. The speed and parallelism are important differences from prior work where optimization involves computing the permutation of pruning percentages across various layers (order of 99^l). Across each such permutation of pruning percentages, the entire network

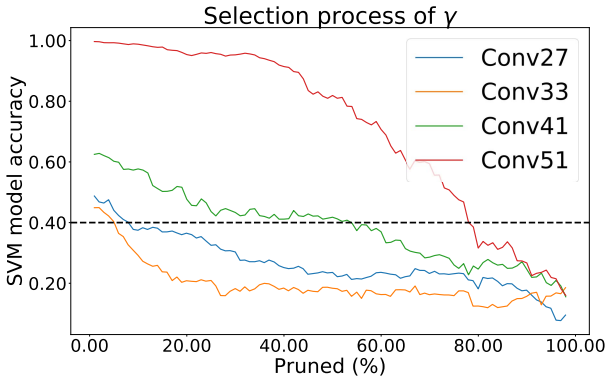


Fig. 3. Selection process for upper pruning percentage limits for each layer of the DNN is based on using a fixed threshold (dotted line) over the SVM models' performances such that the weighted sum of Pruning (%) allocated to each layer, the x-coordinate where the threshold intersects the curve latest, matches the overall sparsity τ.

needs to be retrained/fine-tuned, which can take anywhere from a couple of hours (CIFAR-10) to a week (ILSVRC-2012). This cost is significantly higher when compared to the time taken for a forward pass across the DNN and training an RBF-SVM model. Our core contribution in this work is a systematic approach to deciding the upper pruning percentage limits across all layers of the DNN. Previous works often relegate this information to the final chosen values without disclosing how they arrived at them. We provide the γ-values for all layers of each DNN architecture in our supplementary materials.

E. Sensitivity of Filters

A common assumption made during pruning is that all filters in a layer have the same downstream impact and hence can be characterized solely using the magnitude of their weights. In contrast, probabilistic pruning approaches like MINT [11] aim to maintain the flow of information between a pair of layers, but they consider all filters equally important. Taking into account each filter's impact on succeeding layers is an effective tool to assess their importance and protect filters that contribute the majority of information from being pruned.

We define a sensitivity criterion, $\lambda(F_i^{l+1})$, that can be used to sort filters in their order of importance. Using this, we curate a subset of filters that are critical and hence need to be protected from pruning, while the remaining filters are pruned using the steps in Algorithm 1. To evaluate the sensitivity of filters in layer $l+1$, we look at the weight matrix of its downstream layer $l+2$, $W^{(l+2)}$, and assess the contributions from filters in $l+1$ to those in $l+2$. Here, $W^{(l+2)} \in \mathbb{R}^{N^{(l+2)} \times N^{(l+1)} \times H \times W}$, where H, W are the height and width of the filters in layer $l+2$. For a given filter, the sum of normalized contributions across all the filters in $l+2$ is its overall sensitivity, $\lambda(F_i^{(l+1)})$. It is defined as

$$\lambda\left(F_i^{l+1}\right) = \sum_{f_c=1}^{N^{(l+2)}} \widetilde{W}^{(l+2)}(f_c, i) \big/ \mathcal{C}^{(l+2)}(f_c) \quad (9)$$

$$\text{where } \mathcal{C}^{(l+2)}(f_c) = \sum_{f_p=1}^{N^{(l+1)}} \widetilde{W}^{(l+2)}(f_c, f_p). \quad (10)$$

Here, $\mathcal{C}^{(l+2)}$ is the normalization constant used to relate the weights of filters from $l+1$ contributing to the same filter in $l+2$ and $\widetilde{W}^{(l+2)}$ is the weight matrix of $l+2$ averaged over the height and width dimensions.

Once we obtain the order of sensitivity values for filters in a given layer, we define a threshold of highly sensitive filters that remain untouched after empirically comparing the improvement in performance at similar pruning levels with and without protecting sensitive filters. This comparison is critical to ensure that only sensitive filters, which contribute the majority of the information downstream, remain untouched. This idea also helps improve the overall compression performance since less sensitive filters can be pruned more without compromising the quality of information flowing between layers to a large degree. After empirically comparing the degradation in performance of the SVM model, between the case when all the filters are pruned and the case when we protect a variable percentage of sensitive filters, we determine the set of highly sensitive filters to protect from pruning, and return their indices to Algorithm 1.

IV. Experimental Results

We divide our results into three sections, formatted as an ablative study. Section IV-A focuses on the evaluation of run-time and choice of φ, to highlight the impact of using our ACMI estimator in place of the MST-based estimator used in MINT [11]. Here, the upper pruning limits are manually defined, with the help of artificial limits placed on the SVM model accuracy, to mimic prior work. In Section IV-B, we detail the results of applying SNACS (ACMI + Automated upper pruning percentage limits) across four Dataset-DNN combinations. Within this section, we focus on drawing strong comparisons against single-shot pruning approaches and highlight how competitive SNACS is amongst approaches that use a modified objective function or iterative pruning. Finally, in Section IV-C, we discuss the impact of adding our sensitivity measure as a way to prioritize and fully protect important filters from being pruned.

1) Dataset-DNN: We use four standard Dataset-DNN combinations to evaluate and compare our approach to standard baselines. They are, CIFAR10 [43]-VGG16 [44], CIFAR10-ResNet56 [45], CIFAR10-MobileNetv2 [46], and ILSVRC2012 [47]-ResNet50. A detailed breakdown of each dataset and the experimental setup used in each experiment is included in the supplementary materials.

2) Metric: We use the following metrics to compare performances.

1) Pruning (%): The percentage of parameters removed when compared to the total number of parameters in the un-pruned DNN (Conv and FC only).
2) Test Accuracy (%): The best performance on the testing set, upon training, for baseline networks, and retraining, for pruning methods.
3) FLOPs Reduced (%): The percentage of FLOPs reduced when compared to the un-pruned DNN.

Apart from the above metrics, we also use *run-time* to compare speed of estimators. A high quality method must have high

TABLE I

WE COMPARE THE MAXIMUM COMPRESSION PERFORMANCE OF A VARIETY OF φ FUNCTIONS WHEN MAINTAINING A TEST ACCURACY $\geq$93.43%. $\varphi = \exp((-\text{WEIGHTS}^2/2))$ PERFORMS THE BEST, AND WE USE THIS IN ALL FURTHER EXPERIMENTS

φ function	Pruning (%)
constant $= 1$	84.02
$\|\text{weights}\|_2$	84.12
weights^2	84.17
$\exp(\frac{-\text{weights}^2}{2})$	**84.46**
$\|\text{act}\|_2$	76.13
$\|\text{weights}\|_2 \|\text{act}\|_2$	82.59
$\exp(-\frac{\text{weights}^2 \|\text{act}\|_2^2}{2})$	76.99

compression performance while maintaining a test accuracy relatively close to the baseline.

A. Evaluation of Estimator

1) Run-Time Comparison: We compare the run-times taken to compute the dependency scores across convolution layer 9 in VGG16 using our proposed ACMI estimator and the MST-based estimator used in MINT [11]. For this experiment, we use three distinct estimators, the MST-based estimator from MINT, our ACMI estimator with $\varphi = 1$ and $\varphi = \|\text{weight}\|_2$. Here, weight values are rescaled between $[0, 1]$. To provide a fair comparison, we adopt the grouping concept introduced in MINT. From Fig. 4, we make two important observations: 1) run-time increases with an increase in group size across both estimators and 2) relative to the run-time from the MST-based estimator, our estimator is faster by at least 17×. Thus, we show that our estimator significantly reduces the overall run-time required to compute conditional MI across a DNN. Furthermore, we reduce the run-time for one of the largest computational bottlenecks irrespective of the scaling function used in ACMI.

2) Selection of φ: There are several potential functions that we can associate with φ. In Table I, we illustrate a variety of functions and their performance, w.r.t. the Pruning (%) while maintaining an accuracy $\geq$93.43% in the VGG16-CIFAR10 setup. The main differences between Section IV-A and MINT [11] are the inclusion of ACMI and the manual definition of upper pruning percentage limits using artificially capped SVM model accuracies (0.8). From Table I, we observe that most variants of φ outperform MINT, including $\varphi = 1$. Furthermore, we find that $\varphi = \exp((-\text{weights}^2/2))$ performs the best when compared to all the options for φ we explore. Thus, we set this as the default φ throughout all further experiments.

B. Large-Scale Comparison

When compared to existing single-shot pruning methods, from Table II, we observe that SNACS outperforms all of them by a significant margin to establish new state-of-the-art performances. Our consistently high results establish our hybrid pruning framework as one of the top performing single-shot algorithms. A combination of improved estimates from the hash-based ACMI estimator (Table I) and the joint definition of upper pruning percentage limits for each layer in the DNN are the main contributors to our high performance.

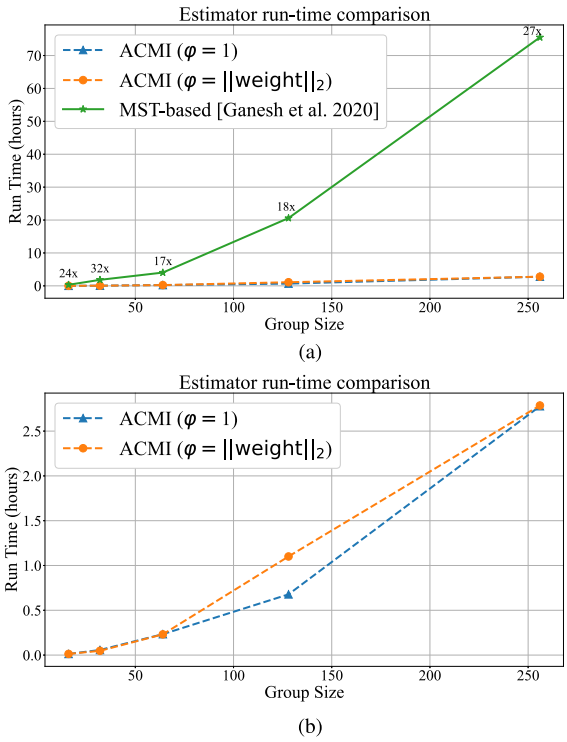


Fig. 4. (a) When comparing run-times between the MST-based estimator used in MINT [11] and our hash-based ACMI estimator, our estimator provides up to 27× speedup in run-time. Run-time comparison across MST and ACMI measures. (b) Across different selections of the scaling function in our estimator, the run-times scale similarly as the number of groups increases. Run-time comparison across various φ.

Fig. 5 helps put SNACS's performance in perspective of pruning approaches that use either sparsity inducing objective functions or iterative retraining setups. In general, we expect a decrease in performance with an increase in the number of parameters pruned. Often, iterative approaches achieve the highest compression while suffering minimal drop in testing accuracy, with methods that use joint optimization sprinkled across the entire range of pruning (%) values. Single-shot methods are often the weakest performers given that they get the fewest attempts to account for the loss in accuracy after pruning. However, across each dataset-DNN combination, our algorithm is highly competitive with the best pruning approaches regardless of variations in optimizers, iterative pruning pipelines, modified objective functions or layer-by-layer fine-tuning. SNACS remains competitive at high pruning levels despite using a single prune-retrain step.

An important distinction between our pruning approach and other single-shot methods we compare against is that we avoid pruning early layers to a large extent, as shown in Fig. 6. Given that a large portion of FLOPs are concentrated in the early portion of the network, the percentage of FLOPs reduced by our SNACS is slightly lower when compared to methods like X-Nets [48], which preemptively prunes the network before training, or SSS [23], which optimizes a different objective function altogether. Interestingly, on closer

TABLE II

USING A **SINGLE** TRAIN-PRUNE-RETRAIN CYCLE, SNACS IS AMONG THE TOP PERFORMERS ACROSS ALL THE DATASET-DNN COMBINATIONS. BASELINES ARE ORDERED ACCORDING TO INCREASING PRUNING (%)

	Method	Pruning (%)	Test Accuracy (%)	FLOPs Reduced (%)
CIFAR-10 VGG16	Baseline	N.A.	93.98	N.A.
	l_1-norm [19]	64.00	93.40	34.18
	Variational Pruning [29]	73.34	93.18	39.29
	SSS [23]	73.80	93.02	41.60
	MINT [11]	83.46	93.43	N.A.
	Network Slimming [21]	88.52	93.80	50.94
	X-Nets [48]	92.33	93.00	N.A.
	Bayesian Compression [30]	94.50	91.00	N.A.
	SNACS ($\tau = 0.96$)	96.16	91.06	67.85
CIFAR-10 ResNet56	Baseline	N.A.	92.55	N.A.
	l_1-norm [19]	13.70	93.06	27.28
	Variational Pruning [29]	20.49	92.26	20.17
	NISP [20]	42.60	93.01	43.61
	FSDP [49]	50.00	92.64	N.A.
	SCOP [50]	56.30	93.64	56.00
	MINT [11]	57.01	93.42	33.67
	SNACS ($\tau = 0.685$)	68.59	93.38	36.89
CIFAR-10 MobileNetv2	Baseline	N.A.	93.66	N.A.
	SCOP [50]	36.10	94.24	40.30
	SNACS ($\tau = 0.55$)	55.00	94.28	28.52
ILSVRC2012 ResNet50	Baseline	N.A.	76.13	N.A.
	SSS [23]	38.82	71.82	43.04
	NISP [20]	43.82	71.99	44.01
	MINT [11]	49.62	71.05	N.A.
	X-Nets [48]	50.00	72.85	50.00
	SCOP [50]	51.80	75.26	54.60
	SNACS ($\tau = 0.60$)	54.99	74.68	34.51
	SNACS ($\tau = 0.65$)	59.67	74.37	39.04
	SNACS ($\tau = 0.70$)	64.51	73.59	45.65
	SNACS ($\tau = 0.75$)	68.93	72.60	51.52

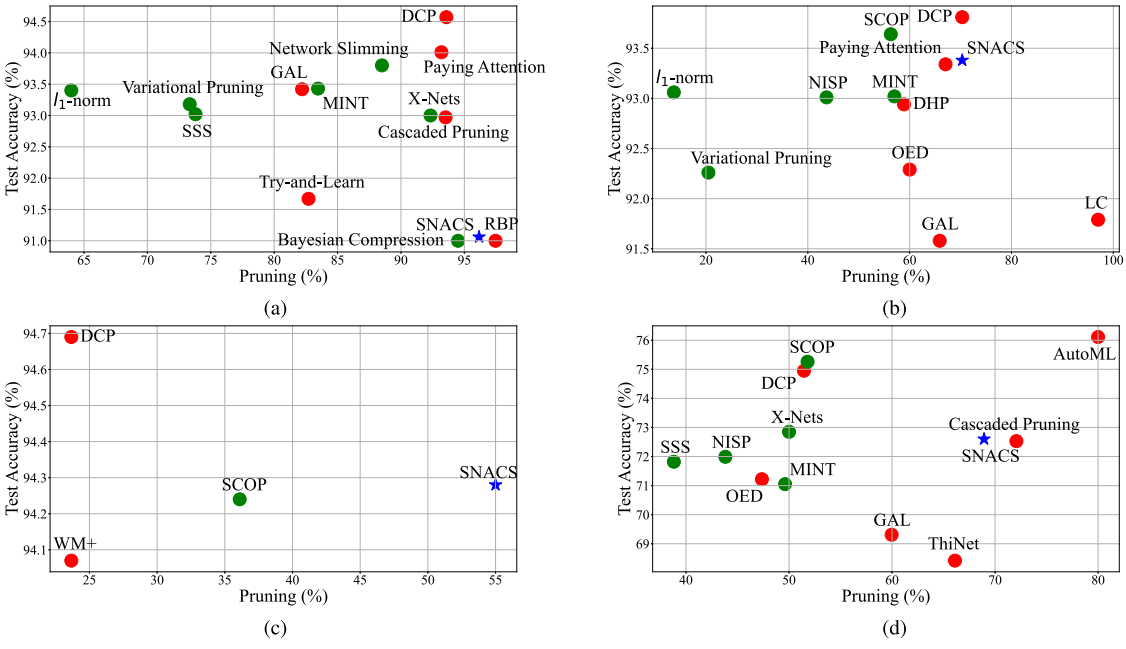


Fig. 5. Comparison of single-shot (green) versus nonsingle-shot (red) pruning approaches across our benchmarks. SNACS, despite being a single-shot approach, is highly competitive with the best performing methods. (a) CIFAR10-VGG16. (b) CIFAR10-ResNet56. (c) CIFAR10-MobileNetv2. (d) ILSVRC2012-ResNet50.

inspection of Fig. 6, we observe minimal correlation between the patterns of high and low γ values achieved in MINT and our work. While MINT showcases minimal pruning in the early and middle set of layers, SNACS focuses on the middle

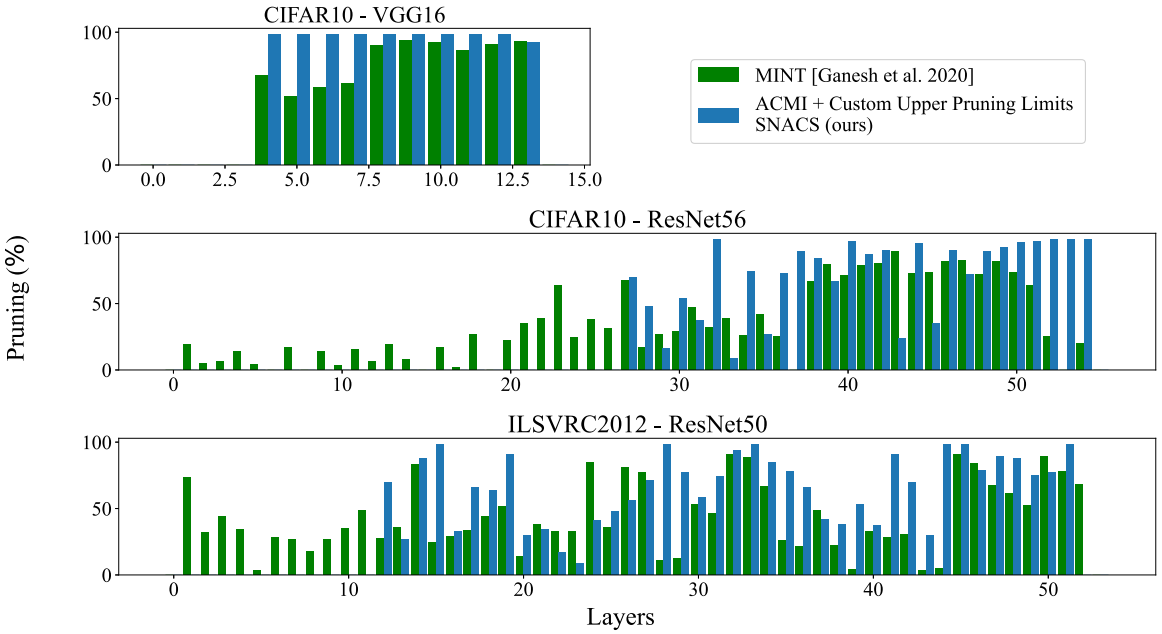


Fig. 6. On observing the compression performance per layer in the ILSVRC2012-ResNet50 experiment, SNACS is able to achieve high Pruning (%) while focusing only on the middle and latter layers, avoiding the early layers. Interestingly, the pattern of pruning in MINT and SNACS is extremely different.

TABLE III

BY SAVING A SMALL PERCENTAGE OF SENSITIVE FILTERS, WE CAN FURTHER IMPROVE THE OVERALL PRUNING (%) WHILE MAINTAINING HIGH TEST ACCURACY (%)

	Method	Pruning (%)	Test Accuracy (%)
CIFAR10 -ResNet56	Baseline	N.A.	92.55
	SNACS (ours)	68.59	93.38
	SNACS + sensitivity (ours)	**68.96**	**93.41**

and final set of layers, avoiding the early layers. We believe this variation stems from the fact that γ values in MINT were co-opted from prior works which focus on individual layers while in SNACS the joint definition of γ s helps capture trends across multiple layers while trying to optimize the performance-sparsity tradeoff.

We observe that when using SNACS, DNNs are more forgiving when pruning layers closer to the output than input since the retraining phase allows them to overcome the loss of abstract concepts learned in later layers but not fundamental structures when compressing the earlier layers of the network. Our observations are matched by the discriminant scores in [49] and the median oracle ranking statistics per layer from [51]. However, these observations are in direct contrast to previous works that identify that portions of the network closer to the input are often pruned first [23], [28]. We hypothesize that their outcomes stem from the modification of the objective function and subsequent training of baseline networks, whereas our approach and those in [50] and [52] focus on removing filters based on a predefined criterion without modifying the loss function.

C. Sensitivity-Based Pruning

Experiments in Sections IV-A and IV-B assume that all filters contribute equally to the information flow downstream and hence, we only use the connectivity scores for pruning. In this section, we highlight the impact of using the sensitivity criterion to prioritize the pruning of relatively weaker filters while protecting more sensitive filters from pruning on the CIFAR10-ResNet56 experimental setup. In Fig. 7(a) and (b), we illustrate the 2-D pruning masks generated by our

TABLE IV

DEVIATING THE % OF FILTERS SAVED FROM OUR OPTIMAL CONSTRAINTS FORCES LOWER SPARSITY LEVELS WITH BAD TESTING PERFORMANCE. OPTIMAL VALUES ARE HIGHLIGHTED IN **BOLD**

Layer	% Saved	Sparsity (%)	Test Accuracy (%)
Layer 28	30	15.03	92.83
	34	15.03	93.05
	38	14.35	92.88
	45	**55.07**	**93.41**
	50	48.92	92.71
	54	45.89	93.24
	60	39.74	93.10
Layer 44	25	26.97	92.97
	30	54.83	93.13
	35	48.55	93.28
	40	**58.17**	**93.41**
	45	53.58	92.96
	50	48.99	93.50
	52.5	45.92	92.86

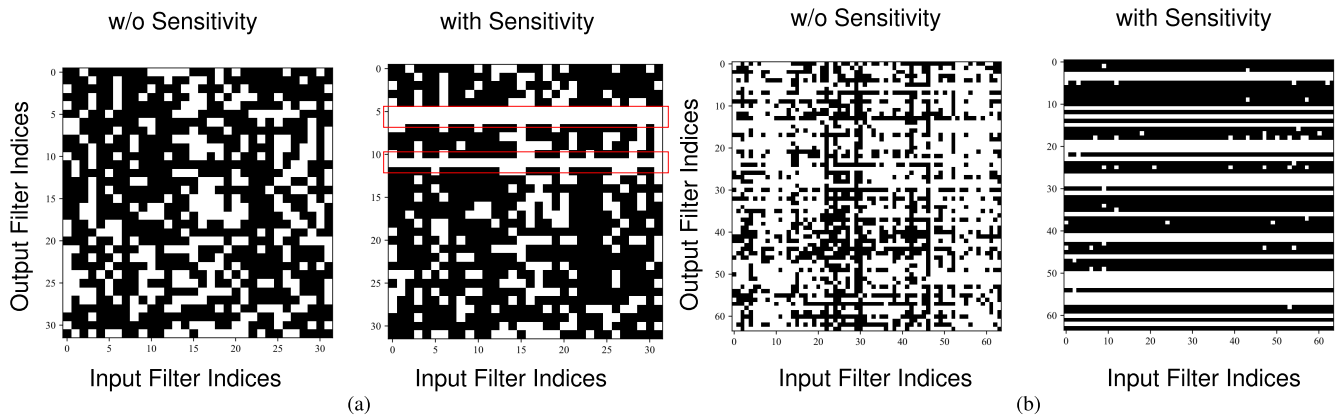


Fig. 7. Illustrations of filters retained (white) and pruned (black) w/o and with sensitivity based pruning. When protecting important filters from pruning, all its associate connections are maintained (red highlight). An interesting impact of sensitivity is that the connections pruned can be completely modified when compared to its counterpart w/o pruning. This is illustrated by the pruning mask of convolution 46. (a) Convolution 35. (b) Convolution 46.

algorithm, where the colors black and white represent filters that are removed and retained, respectively, and we observe three distinct behaviors. First, when we protect a filter from pruning, an entire row representing all of its associated connections is retained. Second, we also observe an increase in the number of weights pruned from filters that are not protected. An overall increase in the number of black pixels illustrates this idea. Finally, when we apply the sensitivity criterion to layers that were previously not pruned to a large extent (Fig. 6 Convolution 32, 34, and many others) we observe a complete restructure in the way filters are pruned. Fig. 7(b) highlights this trend by showcasing an increase in the overall pruning of the layer and a stark difference in how it is pruned. All these observations put together lead to an overall improvement in the Pruning (%) with the inclusion of sensitivity while maintaining high test accuracy (%), as shown in Table III.

Across the results presented in Table III, we maintain the percentage of filters protected from pruning at an optimal level. We determine the optimal combination of high sparsity and accuracy by constraining the % of filters saved to a value such that SVM model performance is higher than the case when no filters are protected. The performance comparison is restricted to the SVM model only, and no retraining is necessary. When we relax this constraint (Table IV), we observe that the performance levels drop by a significant amount and the sparsity level is lower than expected. This trend highlights the necessity of maintaining our constraints to obtain the optimal combination of high sparsity with accuracy.

V. Conclusion

Overall, we propose a novel DNN pruning algorithm called SNACS which uses ACMI to measure the connectivity between filters, a simple set of operating constraints to automate the definition of upper pruning percentage limits of layers in a DNN, and a sensitivity criterion that helps protect a subset of critical filters from pruning. SNACS provides a faster overall run-time, improves accuracy in the estimation process, and offers state-of-the-art levels of compression using a single train-prune-retrain cycle, and the sensitivity criterion can further boosts the compression performance. An important direction of future work is to extend this algorithm to an iterative approach and incorporate it into the training phase. Doing so would help reduce the overall training time while achieving extreme levels of sparsity. Additionally, characterizing the pruned networks using a multitude of events like adversarial attacks, calibration error, and many others could shed light on how close such networks are to being deployed in the real-world.

Appendix A
Bounds on AMI

Recall the definition of AMI (1). For the particular case of g, $g(t) = ((t-1)^2/2(t+1))$, we have

$$I_\varphi(X;Y) = \frac{1}{2}\mathop{\mathbb{E}}_{P_X P_Y}\left[\varphi(X,Y)\left(\frac{dP_{XY}}{dP_X P_Y}+1\right)\right] \tag{11}$$

$$-2\mathop{\mathbb{E}}_{P_X P_Y}\left[\varphi(X,Y)h\left(\frac{dP_{XY}}{dP_X P_Y}\right)\right] \tag{12}$$

where $h(t) = (t/t+1)$. When $(dP_{XY}/dP_X P_Y) = 1$, then the minimum value of I_φ is zero. Further, when P_{XY} and $P_X P_Y$ have no overlapping space then the second term in (11) becomes zero. Therefore, bounds on I_φ is given as

$$0 \le I_\varphi(X,Y) \le \frac{1}{2}\mathop{\mathbb{E}}_{P_X P_Y}\left[\varphi(X,Y)\left(\frac{dP_{XY}}{dP_X P_Y}+1\right)\right]. \tag{13}$$

Appendix B
Proof of Theorem 1

Recall our estimator in Section III-C

$$\widehat{I}_\varphi(X;Y|Z) = \sum_{e_{ijk}\in E_G} \varphi(i,j,k)\,\alpha_{ijk}\,g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right) \tag{14}$$

where $\alpha_{ijk} = (r_{ik}\,r_{jk}/r_k)$. The expectation of $\widehat{I}_\varphi$ is derived as

$$\mathbb{E}\left[\sum_{e_{ijk}\in E_G} \varphi(i,j,k)\,\alpha_{ijk}\,g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle| E_G\right] \tag{15}$$

$$= \sum_{e_{ijk}\in E_G} \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk}\,g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle| E_{ijk}\right] \tag{16}$$

where E_{ijk} is the event that there is an edge between the vertices v_i, u_j, and ω_k in the dependency graph $G(X, Y, Z)$. Let hash function H_1 map the N i.i.d points X_k, Y_k, and Z_k to $\tilde{X}_k$, $\tilde{Y}_k$, and $\tilde{Z}_k$. Following the notations used in [38], we denote $E_i^{=1}$ be the event that there is exactly one vector from $\tilde{X}_i$ that maps to v_i using H_2. Similarly, we define $\overline{E_j^{=1}}$ and $E_k^{=1}$. We denote $E_{ijk}^{=1} := E_i^{=1} \cap E_j^{=1} \cap E_k^{=1}$ and let $\overline{E_{ijk}^{=1}}$ be the complement set of $E_{ijk}^{=1}$.

We simplify (16) by splitting it into two parts: without collision and due to collision. Based on the law of total expectation, we have

$$\begin{aligned} &= \sum_{e_{ijk}\in E_G} P\left(E_{ijk}^{=1}\middle|E_{ijk}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk} g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle| E_{ijk}^{=1}, E_{ijk}\right] \\ &\quad+ \sum_{e_{ijk}\in E_G} P\left(\overline{E_{ijk}^{=1}}\middle|E_{ijk}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|\overline{E_{ijk}^{=1}}, E_{ijk}\right]. \end{aligned} \tag{17}$$

Step 1 Bias on w/o Collision: Similar to [38, Lemma 7.3], we derive

$$P\left(E_{ijk}^{=1}\middle|E_{ijk}\right) = 1 - O\left(\frac{1}{\epsilon^d N}\right), \quad d = d_X + d_Y + d_Z. \tag{18}$$

This is because all three $|V|$, $|U|$, and $|W|$ are upper bounded by $O(\epsilon^{-d})$. Note that ϵ is a function of N. Additionally from [38], we infer the following results:

$$\mathbb{E}[\alpha_{ijk}] = \frac{\mathbb{E}[r_{ik}]\,\mathbb{E}[r_{jk}]}{\mathbb{E}[r_k]} + O\left(\sqrt{\frac{1}{N}}\right). \tag{19}$$

Note that (19) is implied based on the fact that $\mathbb{V}(\alpha_{ijk}) \leq O(1/N)$ which is proven by applying Efron-Stein inequality under assumptions **(A1)** and **(A3)**, similar to arguments in [38, Lemma 7.10]. In addition, we have

$$\mathbb{E}\left[\frac{r_{ijk}}{\alpha_{ijk}}\right] = \frac{\mathbb{E}[r_{ijk}]}{\mathbb{E}[\alpha_{ijk}]} + O\left(\sqrt{\frac{1}{N}}\right) \tag{20}$$

$$\begin{aligned} \mathbb{E}\left[\frac{r_{ijk}}{\alpha_{ijk}}\right] &= P\left(E_{ijk}^{\leq 1}\right)\mathbb{E}\left[\frac{r_{ijk}}{\alpha_{ijk}}\middle|E_{ijk}^{\leq 1}\right] \\ &\quad+ P\left(E_{ijk}^{>1}\right)\mathbb{E}\left[\frac{r_{ijk}}{\alpha_{ijk}}\middle|E_{ijk}^{>1}\right] \end{aligned} \tag{21}$$

where by using similar arguments as in [38, eq. (56)], we have $P(E_{ijk}^{\leq 1}) = 1 - O(\sqrt{1/(\epsilon^d N)})$. Therefore, $P(E_{ijk}^{>1}) = O((1/(\epsilon^d N))^{1/2})$. Further the second term in (21) is the bias because of collision of H, which will be proved later in the current section, that is upper bounded by $O((1/(\epsilon^d N))^{1/2})$.

Let x_D and x_C, respectively, denote the discrete and continuous components of the vector x, with dimensions d_D and d_C. Also let $f_{X_C}(x_C)$ and $p_{X_D}(x_D)$ respectively denote density and pmf functions of these components associated with the probability measure P_X. Let X have d_C and d_D, Y have d_C', d_D', and Z have d_C'', d_D'' as their continuous and discrete components, respectively. Then it can be shown that

$$\begin{aligned} \mathbb{E}\left[r_{ijk}\middle|E_{ijk}^{\leq 1}\right] &= P(X_D = x_D, Y_D = y_D, Z_D = z_D)\epsilon^{d_C+d_C'+d_C''} \\ &\quad\times (f(x_C, y_C, z_C|x_D, y_D, Z_D) + \Delta(\epsilon, q, \gamma)) \end{aligned} \tag{22}$$

where densities have bounded derivatives up to the order $q \geq 0$ and belong to the Hölder continuous class with smoothness parameter γ. Note that $\Delta(\epsilon, q, \gamma) \to 0$ as $N \to \infty$. Now from [38, eqs. (50), (51), and (53)] and from (19), (20) above, under assumptions **(A1)** and **(A3)**, we derive

$$\mathbb{E}\left[\frac{r_{ijk}}{\alpha_{ijk}}\middle|E_{ijk}^{\leq 1}\right] = \frac{dP_{XYZ}\,P_Z}{dP_{XZ}\,P_{YZ}} + \widetilde{\Delta}(\epsilon, q, \gamma) + O\left(\sqrt{\frac{1}{N}}\right) \tag{23}$$

where $H(x) = i$, $H(y) = j$, $H(z) = k$, and as $N \to \infty$, $\widetilde{\Delta}(\epsilon, q, \gamma) \longrightarrow 0$.

Step 2 Bias Because of Collision: Let $\widetilde{\mathbf{X}} = \{\widetilde{X}_i\}_{i=1}^{L_X}$, $\widetilde{\mathbf{Y}} = \{\widetilde{Y}_i\}_{i=1}^{L_Y}$, $\widetilde{\mathbf{Z}} = \{\widetilde{Z}_i\}_{i=1}^{L_Z}$, respectively, denote distinct outputs of H_1 with the N i.i.d points X_k, Y_k, and Z_k as inputs. We denote $L_{XYZ} := |\widetilde{\mathbf{X}} \cup \widetilde{\mathbf{Y}} \cup \widetilde{\mathbf{Z}}|$, $L_{XZ} := |\widetilde{\mathbf{X}} \cup \widetilde{\mathbf{Z}}|$, and $L_{YZ} := |\widetilde{\mathbf{Y}} \cup \widetilde{\mathbf{Z}}|$

$$\begin{aligned} \mathbb{B}_\varphi :&= \sum_{e_{ijk}\in E_G} P\left(\overline{E_{ijk}^{=1}}\middle|E_{ijk}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|\overline{E_{ijk}^{=1}}, E_{ijk}\right] \\ &\leq \sum_{i,j,k\in\mathcal{F}} P\left(E_{ijk}^{>1}\right) \\ &\quad\times \mathbb{E}\left[\mathbb{1}_{E_{ijk}}\varphi(i,j,k)\,\alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|E_{ijk}^{>1}\right] \end{aligned} \tag{24}$$

where $E_{ijk}^{>1} = E_i^{>1} \cap E_j^{>1} \cap E_k^{>1}$, and $E_i^{>1}$ is the event that there are at least two vectors from $\tilde{X}_i$ that map to v_i using H_2. Once again, using the law of total expectation, then the RHS of (24) becomes

$$\begin{aligned} &= \sum_{i,j,k\in\mathcal{F}} P\left(E_{ijk}^{>1}\right)\Big(P\left(E_{ijk}\middle|E_{ijk}^{>1}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk} g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|E_{ijk}^{>1}, E_{ijk}\right] \\ &\quad+ P\left(\overline{E_{ijk}}\middle|E_{ijk}^{>1}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|E_{ijk}^{>1}, \overline{E_{ijk}}\right]\Big) \\ &= \sum_{i,j,k\in\mathcal{F}} P(E_{ijk})P\left(E_{ijk}^{>1}\middle|E_{ijk}\right) \\ &\quad\times \mathbb{E}\left[\varphi(i,j,k)\,\alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right)\middle|E_{ijk}^{>1}, E_{ijk}\right]. \end{aligned} \tag{25}$$

The equality in (25) is obtained based on Bayes error and $g = 0$ on the event $\overline{E_{ijk}}$. Now, recalling (18), using (13) we bound the last line in (25) by

$$\begin{aligned} &O\left(\frac{1}{\epsilon^d N}\right)\sum_{i,j,k\in\mathcal{F}} P(E_{ijk}) \\ &\qquad\times \mathbb{E}\left[\varphi(i,j,k)\left(r_{ijk} + \alpha_{ijk}\right)\middle|E_{ijk}^{>1}, E_{ijk}\right]. \end{aligned} \tag{26}$$

This implies that

$$\begin{aligned}\mathbb{B}_\varphi &\le O\left(\frac{1}{\epsilon^d N}\right)\sum_{i,j,k\in\mathcal{F}} P(E_{ijk})\\ &\quad\times\left(\mathbb{E}\left[\varphi(i,j,k)r_{ijk}\middle|E_{ijk}^{>1},E_{ijk}\right]\right.\\ &\qquad\left.+\,\mathbb{E}\left[\varphi(i,j,k)\alpha_{ijk}\middle|E_{ijk}^{>1},E_{ijk}\right]\right)\\ &= O\left(\frac{1}{\epsilon^d N^2}\right)\sum_{i,j,k\in\mathcal{F}} P(E_{ijk})\\ &\quad\times\left(\mathbb{E}\left[\varphi(i,j,k)\ N_{ijk}\middle|E_{ijk}^{>1},E_{ijk}\right]\right.\\ &\qquad\left.+\,\mathbb{E}\left[\varphi(i,j,k)\frac{N_{ik}N_{jk}}{N_k}\middle|E_{ijk}^{>1},E_{ijk}\right]\right). \quad (27)\end{aligned}$$

If we extend our discussion to all the possible mappings from H_1, we obtain

$$\begin{aligned}&= O\left(\frac{1}{\epsilon^d N^2}\right)\sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}})\sum_{i,j,k\in\mathcal{F}} P(E_{ijk})\\ &\quad\times\left(\mathbb{E}\left[\varphi(i,j,k)\ N_{ijk}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\right.\right.\\ &\qquad\left.\times\,\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right]\\ &\qquad\left.+\,\mathbb{E}\left[\varphi(i,j,k)\frac{N_{ik}N_{jk}}{N_k}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\right.\right.\\ &\qquad\left.\left.\times\,\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right]\right). \quad (28)\end{aligned}$$

Let us define

$$\begin{aligned}\mathcal{A}_{ijk} &:= \left\{r : H_2(\tilde{X}_r)=i, H_2(\tilde{Y}_r)=j, H_2(\tilde{Z}_r)=k\right\},\\ \mathcal{A}_k &:= \left\{r : H_2(\tilde{Z}_r)=k\right\},\\ \mathcal{A}_{ik} &:= \left\{r : H_2(\tilde{X}_r)=i, H_2(\tilde{Z}_r)=k\right\},\\ \mathcal{A}_{jk} &:= \left\{r : H_2(\tilde{Y}_r)=j, H_2(\tilde{Z}_r)=k\right\}. \quad (29)\end{aligned}$$

Let M_r be the number of the input points $(\mathbf{X},\mathbf{Y},\mathbf{Z})$ mapped to $(\tilde{X}_r,\tilde{Y}_r,\tilde{Z}_r)$. Therefore, for i,j,k, we can rewrite N_{ijk} as

$$N_{ijk} = \sum_{r=1}^{L_{XYZ}} \mathbb{1}_{A_{ijk}}(r) M_r. \quad (30)$$

Similarly, M'_r, $\widetilde{M}_s$, and $\overline{M}_t$ are defined the number of the input points mapped to $(\tilde{X}_r,\tilde{Z}_r)$, $(\tilde{Y}_s,\tilde{Z}_s)$, and $\tilde{Z}_t$, respectively, and we can write

$$N_{ik} = \sum_{r=1}^{L_{XZ}} \mathbb{1}_{A_{ik}}(r) M'_r, \quad N_{jk} = \sum_{s=1}^{L_{YZ}} \mathbb{1}_{A_{jk}}(s)\widetilde{M}_s \quad (31)$$

$$N_k = \sum_{t=1}^{L_Z} \mathbb{1}_{A_k}(t)\overline{M}_t. \quad (32)$$

Under the assumption that φ is bounded, we have

$$\begin{aligned}\mathbb{B}_\varphi &\le O\left(\frac{1}{\epsilon^d N^2}\right)\sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}})\sum_{i,j,k\in\mathcal{F}} P(E_{ijk})\\ &\quad\times\left(\sum_{r=1}^{L_{XYZ}} P\left(r\in\mathcal{A}_{ijk}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\right.\\ &\qquad\left.\times\,\mathbb{E}\left[M_r\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right]\right.\\ &\quad+\sum_{r=1}^{L_{XZ}}\sum_{s=1}^{L_{YZ}}\sum_{t=1}^{L_Z} P\left(r\in\mathcal{A}_{ik},s\in\mathcal{A}_{jk},t\in\mathcal{A}_k\middle|E_{ijk}^{>1},\right.\\ &\qquad\left.\times\,E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad\left.\times\,\mathbb{E}\left[\frac{M'_r\widetilde{M}_s}{\overline{M}_t}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right]\right). \quad (33)\end{aligned}$$

Next, we find the probability terms

$$\begin{aligned}&P\left(r\in\mathcal{A}_{ijk}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad= \frac{P\left(r\in\mathcal{A}_{ijk},E_{ijk}^{>1}|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)}{P\left(E_{ijk}^{>1}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)}. \quad (34)\end{aligned}$$

We first find the denominator of (34) first. We define $a=1$ when $i=j=k$ and $a=3$ for the case $i\neq j\neq k$

$$\begin{aligned}&P\left(E_{ijk}^{>1}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad= 1-P\left(E_{ijk}^{=0}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad- P\left(E_{ijk}^{=1}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad= 1-\left(\frac{F-a}{F}\right)^{L_{XYZ}} - \left(\frac{L_{XYZ}}{F^a}\left(\frac{F-a}{F}\right)^{L_{XYZ}-a}\right)\\ &\quad= O\left(\frac{L_{XYZ}^2}{F^{a+1}}\right). \quad (35)\end{aligned}$$

Furthermore,

$$\begin{aligned}&P\left(r\in\mathcal{A}_{ijk},E_{ijk}^{>1}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad= P\left(r\in\mathcal{A}_{ijk}\middle|E_{ijk}^{>1},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad\times P\left(r\in\mathcal{A}_{ijk}\middle|\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad= \left(1-\left(\frac{F-a}{F}\right)^{L_{XYZ}-a}\right)\left(\frac{1}{F}\right)^a = O\left(\frac{L_{XYZ}}{F^{a+1}}\right). \quad (36)\end{aligned}$$

Combining (35) and (36) yields

$$\begin{aligned}&P\left(r\in\mathcal{A}_{ijk}\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad= O\left(\frac{1}{L_{XYZ}}\right). \quad (37)\end{aligned}$$

Now, we simplify the following term:

$$\begin{aligned}&P\left(r\in\mathcal{A}_{ik},s\in\mathcal{A}_{jk},t\in\mathcal{A}_k\middle|E_{ijk}^{>1},E_{ijk},\right.\\ &\qquad\left.\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right). \quad (38)\end{aligned}$$

First, we assume that $\tilde{X}_\upsilon\neq\tilde{Y}_\upsilon\neq\tilde{Z}_\upsilon$ for $\upsilon=r,s,t$. Then

$$\begin{aligned}&P\left(r\in\mathcal{A}_{ik},s\in\mathcal{A}_{jk},t\in\mathcal{A}_k\middle|E_{ijk}^{>1},E_{ijk},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\right.\\ &\qquad\left.\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad\le P\left(r\in\mathcal{A}_{ik}\middle|E_{ik}^{>1},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad\times P\left(s\in\mathcal{A}_{jk}\middle|E_{jk}^{>1},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\qquad\times P\left(t\in\mathcal{A}_k\middle|E_k^{>1},\tilde{\mathbf{X}}=\tilde{\mathbf{x}},\tilde{\mathbf{Y}}=\tilde{\mathbf{y}},\tilde{\mathbf{Z}}=\tilde{\mathbf{z}}\right)\\ &\quad= O\left(\frac{1}{L_{XZ}L_{YZ}L_Z}\right). \quad (39)\end{aligned}$$

Next, assume that $\tilde{X}_\upsilon = \tilde{Y}_\upsilon = \tilde{Z}_\upsilon$ for $\upsilon = r, s, t$, therefore, $H_2(\tilde{X}_\upsilon) = H_2(\tilde{Y}_\upsilon) = H_2(\tilde{Z}_\upsilon)$, for $\upsilon = r, s, t$. Then

$$P\big(r \in \mathcal{A}_{ik}, s \in \mathcal{A}_{jk}, t \in \mathcal{A}_k \big| E_{ijk}^{>1}, E_{ijk}, \tilde{\mathbf{X}} = \tilde{\mathbf{x}}, \tilde{\mathbf{Y}} = \tilde{\mathbf{y}}, \tilde{\mathbf{Z}} = \tilde{\mathbf{z}}\big) = \delta_{ijk} O\left(\frac{1}{L_{XYZ}}\right). \quad (40)$$

By using (40), (39), and (37) in (33), we obtain an upper bound on bias with collision

$$\begin{aligned}
\mathbb{B}_\varphi &\le O\left(\frac{1}{\epsilon^d N^2}\right) \sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}) \sum_{i,j,k\in\mathcal{F}} P(E_{ijk}) \\
&\quad \left(O\left(\frac{1}{L_{XYZ}}\right) \sum_{r=1}^{L_{XYZ}} \mathbb{E}\big[M_r \big| E_{ijk}^{>1}, E_{ijk}, \tilde{\mathbf{X}} = \tilde{\mathbf{x}}, \tilde{\mathbf{Y}} = \tilde{\mathbf{y}}, \tilde{\mathbf{Z}} = \tilde{\mathbf{z}}\big] \right. \\
&\quad \left. + \left(O\left(\frac{1}{L_{XZ}L_{YZ}L_Z}\right) + \delta_{ijk} O\left(\frac{1}{L_{XYZ}}\right)\right) \right. \\
&\quad \left. \times \sum_{r=1}^{L_{XZ}} \sum_{s=1}^{L_{YZ}} \sum_{t=1}^{L_Z} \mathbb{E}\left[\frac{M_r' \widetilde{M}_s}{\overline{M}_t} \big| E_{ijk}^{>1}, E_{ijk}, \tilde{\mathbf{X}} = \tilde{\mathbf{x}}, \tilde{\mathbf{Y}} = \tilde{\mathbf{y}}, \tilde{\mathbf{Z}} = \tilde{\mathbf{z}}\right]\right) \\
&= O\left(\frac{1}{\epsilon^d N^2}\right) \sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}) \sum_{i,j,k\in\mathcal{F}} P(E_{ijk}) \\
&\quad \times \left(O\left(\frac{N}{L_{XYZ}}\right) + \left(O\left(\frac{N}{L_{XZ}L_{YZ}L_Z}\right) \right.\right. \\
&\quad \left.\left. + \delta_{ijk} O\left(\frac{N}{L_{XYZ}}\right)\right)\right). \quad (41)
\end{aligned}$$

Rearranging the expectation term we get

$$\begin{aligned}
&= O\left(\frac{1}{\epsilon^d N^2}\right) \sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}) \\
&\quad \times \left(O\left(\frac{N}{L_{XYZ}}\right) + \left(O\left(\frac{N}{L_{XZ}L_{YZ}L_Z}\right) + O\left(\frac{1}{L_{XYZ}}\right)\right)\right) \\
&\quad \times \mathbb{E}\left[\sum_{i,j,k\in\mathcal{F}} \mathbb{1}_{E_{ijk}}\right] \\
&\le O\left(\frac{1}{\epsilon^d N^2}\right) \sum_{\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}} p_{\tilde{\mathbf{X}},\tilde{\mathbf{Y}},\tilde{\mathbf{Z}}}(\tilde{\mathbf{x}},\tilde{\mathbf{y}},\tilde{\mathbf{z}}) \\
&\quad \times \left(O\left(\frac{N}{L_{XYZ}}\right) + \left(O\left(\frac{N}{L_{XZ}L_{YZ}L_Z}\right) + O\left(\frac{1}{L_{XYZ}}\right)\right)\right) L_{XYZ} \\
&\le O\left(\frac{1}{\epsilon^d N}\right). \quad (42)
\end{aligned}$$

Hence as $N \longrightarrow \infty$, the bias estimator due to collision tends to zero, i.e., $\mathbb{B}_\varphi \longrightarrow 0$.

Step 3 Combine Results Let us denote N'_{ijk}, N'_{ik}, N'_{jk}, and N'_k, respectively, as the number of the input points $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, $(\mathbf{X}, \mathbf{Z})$, $(\mathbf{Y}, \mathbf{Z})$, and $\mathbf{Z}$ mapped to the bins $(\tilde{X}_i, \tilde{Y}_j, \tilde{Z}_k)$, $(\tilde{X}_i, \tilde{Z}_k)$, $(\tilde{Y}_j, \tilde{Z}_k)$, and $\tilde{Z}_k$ using H_1. We define the notations $r(i) = H_2^{-1}(i)$ for $i \in \mathcal{F}$ and $s(x) := H_1(x)$ for $x \in \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$. Then, from (23), we have

$$\begin{aligned}
&\mathbb{E}\left[\frac{N'_{s(X)s(Y)s(Z)} N'_{s(Z)}}{N'_{s(X)s(Z)} N'_{s(Y)s(Z)}}\right] \\
&\qquad = \frac{dP_{XYZ}\, P_Z}{dP_{XZ}\, P_{YZ}} + \widetilde{\Delta}(\epsilon, q, \gamma) + O\left(\sqrt{\frac{1}{N}}\right). \quad (43)
\end{aligned}$$

We simplify the first term in (17) as

$$\begin{aligned}
&\sum_{i,j,k\in\mathcal{F}} P\left(E_{ijk}^{\le 1}\right) \mathbb{E}\left[\mathbb{1}_{E_{ijk}} \varphi(i,j,k)\, \alpha_{ijk}\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right) \big| E_{ijk}^{\le 1}\right] \\
&= \left(1 - O\left(\frac{1}{\epsilon^d N}\right)\right) \sum_{i,j,k\in\mathcal{F}} \mathbb{E}\left[\mathbb{1}_{E_{ijk}} \varphi(i,j,k)\, \alpha_{ijk} \right. \\
&\qquad \left. \times\, g\left(\frac{r_{ijk}}{\alpha_{ijk}}\right) \Big| E_{ijk}^{\le 1}\right] \\
&= \sum_{i,j,k\in\mathcal{F}} \mathbb{E}\left[\mathbb{1}_{E_{ijk}} \varphi(i,j,k)\, \frac{N_{ik}N_{jk}}{N_k N}\, g\left(\frac{N_{ijk}N_k}{N_{ik}N_{jk}}\right) \big| E_{ijk}^{\le 1}\right] \\
&\quad + O\left(\frac{1}{\epsilon^d N}\right) \\
&= \sum_{i,j,k\in\mathcal{F}} \mathbb{E}\left[\mathbb{1}_{E_{ijk}} \varphi(r(i), r(j), r(k))\, \frac{N'_{r(i)r(k)} N'_{r(j)r(k)}}{N'_{r(k)} N} \right. \\
&\qquad \left. \times\, g\left(\frac{N'_{r(i)r(j)r(k)} N'_{r(k)}}{N'_{r(i)r(k)} N'_{r(j)r(k)}}\right)\right] + O\left(\frac{1}{\epsilon^d N}\right). \quad (44)
\end{aligned}$$

Let us denote

$$\beta(r(i), r(j), r(k)) = \frac{N'_{r(i)r(j)r(k)} N'_{r(k)}}{N'_{r(i)r(k)} N'_{r(j)r(k)}}.$$

Therefore, the last line in (44) is equal to

$$\begin{aligned}
&= \frac{1}{N} \sum_{i,j,k\in\mathcal{F}} \mathbb{E}\left[\varphi(r(i), r(j), r(k))\, \frac{N'_{r(i)r(j)r(k)}}{\beta(r(i), r(j), r(k))} \right. \\
&\qquad \left. \times\, g\Big(\beta(r(i), r(j), r(k))\Big)\right] + O\left(\frac{1}{\epsilon^d N}\right) \\
&= \frac{1}{N} \mathbb{E}\left[\sum_{i=1}^{N} \frac{\varphi(s(X), s(Y), s(Z))}{\beta(s(X), s(Y), s(Z))}\, g\Big(\beta(s(X), s(Y), s(Z))\Big)\right] \\
&\quad + O\left(\frac{1}{\epsilon^d N}\right), \quad (45)
\end{aligned}$$

where

$$\beta(s(X), s(Y), s(Z)) = \frac{N'_{s(X)s(Y)s(Z)} N'_{s(Z)}}{N'_{s(X)s(Z)} N'_{s(Y)s(Z)}}.$$

The expression in (45) equals

$$\begin{aligned}
&= \mathbb{E}_{P_{XYZ}}\left[\mathbb{E}\left[\frac{\varphi(s(X), s(Y), s(Z))}{\beta(s(X), s(Y), s(Z))}\, g\Big(\beta(s(X), s(Y) \right.\right. \\
&\qquad \left.\left. \times\, s(Z))\Big) \Big| \mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}\right]\right] \\
&\quad + O\left(\frac{1}{\epsilon^d N}\right) \\
&= \mathbb{E}_{P_{XYZ}}\left[\varphi(X, Y, Z) h\left(\frac{dP_{XYZ}\, P_Z}{dP_{XZ}\, P_{YZ}}\right)\right] + \widetilde{\Delta}(\epsilon, q, \gamma)
\end{aligned}$$

$$+ O\left(\sqrt{\frac{1}{N}}\right) + O\left(\frac{1}{\epsilon^d N}\right), \tag{46}$$

where $h(t) = g(t)/t$ and (46) is derived by borrowing [38, Lemma 7.9]. Hence from (46) and (17), and the fact that $\widetilde{\Delta}(\epsilon, q, \gamma) \longrightarrow 0$ as $N \to \infty$, we conclude

$$\mathbb{E}\big[\widehat{I}_\varphi(X;Y|Z)\big] \longrightarrow \mathbb{E}_{P_{XYZ}}\left[\varphi(X,Y,Z)h\left(\frac{dP_{XYZ}\ P_Z}{dP_{XZ}\ P_{YZ}}\right)\right] \quad \text{as} \quad N \to \infty. \tag{47}$$

This completes the proof.

ACKNOWLEDGMENT

The findings are those of the authors only and do not represent any position of these funding bodies.

REFERENCES

[1] M. G. Bechtel, E. Mcellhiney, M. Kim, and H. Yun, "DeepPicar: A low-cost deep neural network-based autonomous car," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 11–21.

[2] L. Fridman *et al.*, "MIT advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation," *IEEE Access*, vol. 7, pp. 102021–102038, 2019.

[3] J. Gu, G. Neubig, K. Cho, and V. O. K. Li, "Learning to translate in real-time with neural machine translation," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics, Volume 1, Long Papers*, 2017, pp. 1–10.

[4] Y. Jia *et al.*, "Direct speech-to-speech translation with a sequence-to-sequence model," 2019, *arXiv:1904.06037*.

[5] S.-C. Lin *et al.*, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2018, pp. 751–766.

[6] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019, *arXiv:1902.09574*.

[7] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21.

[8] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.

[9] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.

[10] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.

[11] M. R. Ganesh, J. J. Corso, and S. Y. Sekeh, "MINT: Deep network compression via mutual information-based neuron trimming," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 8251–8258.

[12] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing neural networks using the variational information bottleneck," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1135–1144.

[13] J.-H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," 2017, *arXiv:1706.05791*.

[14] Y. Suhov, I. Stuhl, S. Y. Sekeh, and M. Kelbert, "Basic inequalities for weighted entropies," *Aequationes Mathematicae*, vol. 90, no. 4, pp. 817–848, Aug. 2016.

[15] T. Cover and J. A. Thomas, *Elements of Information Theory*, 1st ed. Hoboken, NJ, USA: Wiley, 1991.

[16] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 1–8.

[17] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 1–8.

[18] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.

[19] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. 5th Int. Conf. Learn. Represent., (ICLR)*, 2017, pp. 1–13.

[20] R. Yu *et al.*, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[21] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.

[22] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2554–2564.

[23] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 304–320.

[24] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.

[25] J. Yoon and S. J. Hwang, "Combined group and exclusive sparsity for deep neural networks," in *Proc. 34th Int. Conf. Mach. Learning*, vol. 70, 2017, pp. 3958–3966.

[26] Z. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–12.

[27] Y. Li, S. Gu, K. Zhang, L. Van Gool, and R. Timofte, "DHP: Differentiable meta pruning via HyperNetworks," 2020, *arXiv:2003.13683*.

[28] S. Lin *et al.*, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2790–2799.

[29] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2780–2789.

[30] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[31] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 6, Jun. 2004, Art. no. 066138.

[32] K. R. Moon, K. Sricharan, and A. O. Hero, "Ensemble estimation of mutual information," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 3030–3034.

[33] J. C. Principe, D. Xu, J. Fisher, and S. Haykin, "Information theoretic learning," *Unsupervised Adapt. Filtering*, vol. 1, pp. 265–319, 2000.

[34] H. H. Yang and J. Moody, "Data visualization and feature selection: New algorithms for nongaussian data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1–7.

[35] N. Leonenko, L. Pronzato, and V. Savani, "A class of Rényi information estimators for multidimensional densities," *Ann. Statist.*, vol. 36, no. 5, pp. 2153–2182, 2008.

[36] M. Noshad, K. R. Moon, S. Y. Sekeh, and A. O. Hero, "Direct estimation of information divergence using nearest neighbor ratios," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 903–907.

[37] S. Y. Sekeh and A. O. Hero, "Geometric estimation of multivariate dependency," *Entropy*, vol. 21, no. 8, p. 787, Aug. 2019.

[38] M. Noshad, Y. Zeng, and A. O. Hero, "Scalable mutual information estimation using dependence graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2962–2966.

[39] I. Csiszár and P. C. Shields, "Information theory and statistics: A tutorial," *J. Roy. Statist. Soc. B, Methodol.*, vol. 1, no. 4, pp. 417–528, 2004.

[40] V. Berisha and A. O. Hero, "Empirical non-parametric estimation of the Fisher information," *IEEE Signal Process. Lett.*, vol. 22, no. 7, pp. 988–992, Jul. 2014.

[41] V. Berisha, A. Wisler, A. O. Hero, and A. Spanias, "Empirically estimable classification bounds based on a nonparametric divergence measure," *IEEE Trans. Signal Process.*, vol. 64, no. 3, pp. 580–591, Feb. 2015.

[42] W. Härdle, *Applied Nonparametric Regression*. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[43] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent., (ICLR)*, 2015, pp. 1–14.

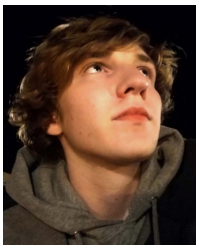
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[47] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[48] A. Prabhu, G. Varma, and A. Namboodiri, "Deep expander networks: Efficient deep networks from graph theory," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 20–35.
[49] N. Gkalelis and V. Mezaris, "Fractional step discriminant pruning: A filter pruning framework for deep convolutional neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2020, pp. 1–6.
[50] Y. Tang *et al.*, "SCOP: Scientific control for reliable neural network pruning," 2020, *arXiv:2010.10732*.
[51] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

Madan Ravi Ganesh received the B.E. degree in electronics and communications from the M.S.R. Institute of Technology (MSRIT), Bengaluru, Karnataka, India, in 2013, and the M.Sc. and Ph.D. degrees in computer vision from the University of Michigan (UofM), Ann Arbor, MI, USA, in 2016 and 2022, respectively.

He is currently a Machine Learning Research Scientist with the Robert Bosch's Center for Artificial Intelligence, Pittsburgh, PA, USA. His current research interests include compression of neural networks, curriculum learning, continual learning, and computer vision applications on edge devices. The primary focus of his research is in making deep learning more efficient and approachable, with an emphasis on bridging the gap between performance and a small computational budget.



Dawsin Blanchard received the B.Sc. degree in computer science from the University of Maine, Orono, ME, USA, in 2020.

The primary focus of his work is analysis of complex datasets that are difficult to analyze with the traditional machine learning and statistical methods. His research interests include computer software and artificial intelligence, and the application of learning methods to modern datasets.

Jason J. Corso (Senior Member, IEEE) received the B.S. degree (Hons.) from the Loyola College, Baltimore, MD, USA, in 2000, and the M.S.E. and Ph.D. degrees from The Johns Hopkins University, Baltimore, in 2002 and 2005, respectively, all in computer science.

He is currently the Co-Founder/the CEO of the Computer Vision Startup Voxel51, Ann Arbor, MI, USA and a Professor of robotics, electrical engineering and computer science with the University of Michigan, Ann Arbor. He has authored more than 150 peer-reviewed articles and hundreds of thousands of lines of open-source code on topics of his interest including computer vision, robotics, data science, and general computing.

Dr. Corso is a member of the Association for the Advancement of Artificial Intelligence (AAAI), Association for Computing Machinery (ACM), and Mathematical Association of America (MAA). He was a member of the Defense Advanced Research Projects Agency (DARPA) Computer Science Study Group in 2009. He was a recipient of the University of Michigan Electrical Engineering and Computer Science (EECS) Outstanding Achievement Award in 2018, the Google Faculty Research Award in 2015, the Army Research Office Young Investigator Award in 2010, the NSF CAREER Award in 2009, and the SUNY Buffalo Young Investigator Award in 2011. He was also a recipient of the Link Foundation Fellowship in Advanced Simulation and Training in 2003.

Salimeh Yasaei Sekeh (Member, IEEE) received the Ph.D. degree in inferential statistics from the Ferdowsi University of Mashhad (FUM), Azadi Square, Iran, in 2013.

She was a Post-Doctoral Research Fellow with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI, USA, working with Alfred O. Hero. She held CAPES-PNPD Post-Doctoral Fellow appointment with the Federal University of Sao Carlos (UFSCar), Brazil, in 2014 and 2015. She was a Visiting Scholar with the Polytechnic University of Turin, Turin, Italy, from 2011 to 2013. She is the Director of the Sekeh Laboratory, Orono, ME, USA. She is currently an Assistant Professor of computer science with the School of Computing and Information Science (SCIS), University of Maine, Orono. Her recent research interests include machine learning, large scale data science, and computer vision. Her primary focus is in design, improvement, and analysis of deep learning techniques with emphasize on deep network efficiency and robustness.