2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

# MINT: Deep Network Compression via Mutual Information-based Neuron Trimming

Madan Ravi Ganesh
EECS
University of Michigan
Ann Arbor, Michigan

Jason J. Corso
EECS
University of Michigan
Ann Arbor, Michigan

Salimeh Yasaei Sekeh
SCIS
University of Maine
Orono, Maine

***Abstract*—Most approaches to deep neural network compression via pruning either directly evaluate a filter's importance using its weights or optimize an alternative objective function with sparsity constraints. While these methods offer a useful way to approximate contributions from similar filters, they often either ignore the dependency between layers or solve a more difficult optimization objective than standard cross-entropy. Our method, Mutual Information-based Neuron Trimming (MINT), approaches deep compression via pruning by enforcing sparsity based on the strength of the dependency between filters of adjacent layers, across every pair of layers in the network. The dependency is calculated using conditional geometric mutual information which evaluates the amount of similar information exchanged between filters using a graph-based criterion. When pruning a network, we ensure that retained filters contribute the majority of the information towards succeeding layers which ensures high performance. Our novel approach is highly competitive with existing state-of-the-art compression-via-pruning methods on standard benchmarks for this task: MNIST, CIFAR-10, and ILSVRC2012, across a variety of network architectures despite using only a single retraining pass. Also, we discuss our observations of a common denominator between our pruning methodology's response to adversarial attacks and calibration statistics when compared to the original network.**

## I. Introduction

Balancing the trade-off between the size of a deep network and achieving high performance is the most important constraint when designing deep neural networks (DNN) that can easily be translated to hardware. Although deep learning yields remarkable performance in real-world problems like medical diagnosis [1], [2], autonomous vehicles [3], [4], and others, they consume a large amount of memory and computational resources that limit their large-scale deployment. With current state-of-the-art (SOTA) deep networks spanning hundreds of millions if not billions of parameters [5], compressing them while maintaining high performance is challenging.

In this work, we approach DNN compression using network pruning [6]. There are two broad approaches to network pruning, (a) unstructured pruning, where a filter's importance is evaluated using weights [6] or constraints like the $l_1$ norm [7] on them, without modifying the overall objective function, and (b) structured pruning, where the objective function is modified to include structured sparsity constraints [8]. Most unstructured pruning approaches ignore the dependency between layers and the impact of pruning on downstream layers while structured pruning methods force the network to optimize a harder and more sensitive optimization objective. The underlying common theme between both approaches is their use of filter weights as a proxy for importance.

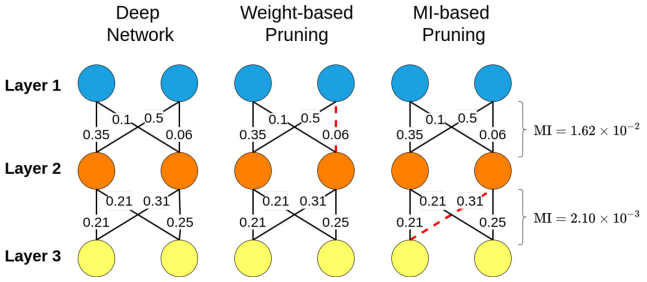



Fig. 1: Weight-based pruning does not consider the dependency between layers. Instead it suggests the removal of low weight values. Mutual information (MI)-based pruning computes the value of information passed between layers, quantified by the MI value, and suggests the removal of weights from the latter layer

Evaluating a filter's importance purely from its weights is insufficient since it does not take into account the dependencies between filters or account for any form of uncertainty. These factors are critical since higher weight values do not always represent its true importance and a filter's contribution can be compensated elsewhere in the network. Consider the example shown in Fig. 1, where a simple weight-based criterion suggests the removal of small valued weights. However, the mutual information (MI) score, which we use to measure the dependency between pairs of filters and emphasize their importance, values the first layer's weights over the latter layer. Pruning based on the MI scores would ensure a network where the retained filters pass on as much information as possible to the next layer.

To overcome prior issues, we propose Mutual Information-based Neuron Trimming (MINT) as a novel approach to pruning deep networks that stochastically accounts for the dependency between layers. Fig. 2 outlines our approach. In MINT, we use an estimator for conditional geometric mutual information (GMI), inspired by [9], to measure the dependency between filters of successive layers. Specifically, we use a graph-based criterion (Friedman-Rafsky Statistic [10]) to measure the conditional GMI between the activations of filters at

978-1-7281-8808-9/20/$31.00 ©2020 IEEE


2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9412590

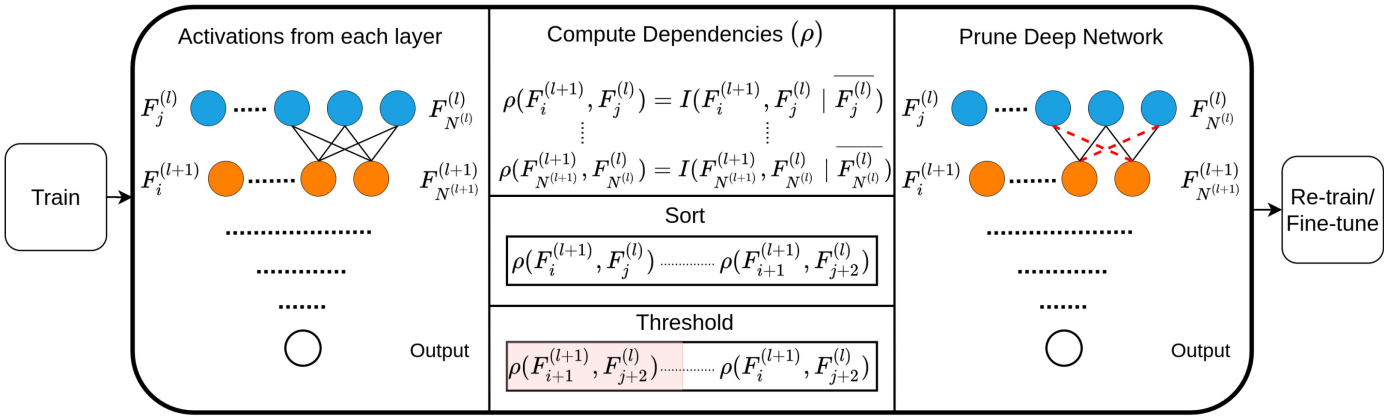



Fig. 2: Illustration of the experimental setup highlighting the components of MINT. Between every pair of filters in consecutive layers ($l$, $l+1$) we compute the conditional geometric mutual information (GMI), using the activations from each filter, as the importance score. The total number of filters in each layer is defined by $N^{(l)}$ and $N^{(l+1)}$. The conditional GMI score indicates the importance of a filter in layer $l$'s contribution towards a filter in layer $l+1$. We then threshold filters based on the importance scores to ensure that we retain only filters that pass the majority of the information to successive layers. Finally, we retrain the network once to maintain a desired level of performance

layer $l$ and $l+1$, denoted by $F_i^{(l)}, F_j^{(l+1)}$, given the remaining filters in layer $l$. On evaluating all such dependencies, between filters of every pair of layers in the network, we sort the importance scores. Finally, we threshold a desired percentage of these values to retain filters that contribute the majority of the information to successive layers. Thus, MINT maintains high performance with compressed and retrained networks.

Through MINT, we contribute a network pruning method that addresses the need to use dependency between layers as an important factor to prune networks. By maintaining filters that contribute the majority of the information passed between layers, we ensure that the impact of pruning on downstream layers is minimized. In doing so, we achieve highly competitive performance to the SOTA across multiple Dataset-CNN architecture combinations, highlighting the general applicability of our method.

Further, we empirically analyze our approach using visualizations that illustrate the focus of learned representations, adversarial attacks, and expected calibration error, to provide a better understanding of our method. We highlight a possible common denominator between its security vulnerability and decrease in calibration error while illustrating the intended effects of retaining filters that contribute the majority of information between layers.

## II. Related Works

Deep network compression offers several strategies to help reduce network size while maintaining high performance, such as low-rank approximations [11], quantization [12], knowledge distillation [13], and network pruning [6], [14], [15]. In this work, we focus on network pruning since it offers a controlled set up to study and compare changes in the dynamics of a network when filters are removed. We broadly classify network pruning methods into two categories, unstructured, and structured, which we describe below. Also, we highlight common strategies to calculate multivariate dependencies and how they vary from our method.

### A. Network Pruning

*1) Unstructured pruning:* Some of the earliest in this line of work used the second-order relationship between the objective function and weights of a network to determine which weights to remove [16]. Although these methods provide deep insights into the relationships within networks, their large computational requirements and run-times made them less practical. They were surpassed by an alternative approach that thresholded the weight values themselves to obtain a desired level of sparsity before retraining [6]. Apart from the simplicity of this approach, it also highlighted the importance of re-training from known minima as opposed to from scratch. Instead of pruning weights in one shot, [17] offered a continuous and recursive strategy of using mini-iterations to evaluate the importance of connections using their weights, remove unimportant ones, and train the network. Similarly, [7] proposed pruning filters using the $l_1$-norm of their weight values as a measure of importance. By melding network pruning using weights with network quantization and Huffman coding [18] showed superior compression performance compared to any individual pipeline. However, the direct use of weight values across all these methods does not capture the relationships between different layers or the impact of removing weights on downstream layers. In MINT, we address this issue by explicitly computing the dependency between filters of successive layers and only retaining filters that contribute a majority of the information. This ensures that there isn't a severe impact downstream.

A subset of unstructured methods uses data to derive the importance of filter weights. Among them, ThiNet [14] posed the reconstruction of outcomes with the removal of weights as an optimization objective to decide on weights. More recently, NISP [19] used the contribution of neurons towards

the reconstruction of outcomes as a metric to remove filters. These works represent a shift to data-driven logic to consider the downstream impact of pruning, with the use of external computations (e.g., feature ranking in [19]). Compared to the deterministic relationship learned between the weights and feature ranking methods, our method uses a probabilistic approach to measure the dependency between filters, thereby accounting for some form of uncertainty. Further, our method uses one single prune-retrain step compared to multiple iterations of fine-tuning performed in these methods.

*2) Structured Pruning:* The shift to structured pruning was based on the idea of seamlessly interfacing with hardware systems as opposed to relying on software accelerators. One of the first methods to do this extended the original brain damage [16] formulation to include fixed pattern masks for specified groups while using a group-sparsity regularizer [20]. This idea was further extended in works that used the group-lasso formulation [8], individual or multiple $l_n$ norm constraints [21] on the channel parameters, and in works that balanced individual vs. group relationships [22] to induce sparsity across desired structures. These methods explicitly affect the training phase of a chosen network by optimizing a harder and more sensitive objective function. Further, side-effects like-stability to adversarial attacks, calibration, and differences in the learned representations have not been fully quantified. In our work, we optimize the standard cross-entropy objective function, characterize and compare the behaviour of the original network to their compressed counterparts.

Hybrid methods combine the notion of a modified objective function with the measurement of the downstream impact of pruning by enforcing sparsity constraints on the outcomes of groups [23], using a custom group-lasso formulation with a squared dependency on weight values as the importance measure [24] or an adversarial pruned network generator to compete with the features derived from the original network [15]. The disadvantages of these methods include their multi-pass scheme and large training times as well as those inherited from modifying the objective function.

### B. Multivariate Dependency Measures

The accurate estimation of multivariate dependency in high-dimensional settings is a hard task. The first in this line of work involved Shannon Mutual Information which was succeeded by several plug-in estimators including Kernel Density Estimators [25] and KNN estimators [26]. However, their dependence on density estimates and large run-time complexity meant they were not suitable for large scale applications including neural networks. Faster plug-in methods based on graph theory and nearest neighbour ratios [27] were proposed as an alternative. More solutions that use statistics like KL divergence [28] or Renyi-$\alpha$ [29] were proposed to help bypass density estimation fully. Instead, in this work, we focus on a conditional GMI estimator, similar to [9], which bypasses the difficult task of density estimation and is also non-parametric and scalable.

## III. MINT

MINT is a data-driven approach to pruning networks by learning the dependency between filters/nodes of successive layers. For every pair of layers in a deep network, we use the conditional GMI (Section III-A) between the activations of every filter from a chosen layer $l$ and a filter from layer $l+1$, given the existence of every other possible filter in layer $l$ to compute an importance score. Here, data flows from layer $l$ to $l+1$. Once all such importance scores are evaluated, we remove a fixed portion of filters with the lowest scores. This induces the desired level of sparsity in the network before we retrain the network to maintain high accuracy. The core algorithm is outlined and explained in the sections below.

### A. Conditional Geometric Mutual Information

In this section, we review conditional GMI estimation as featured in [9] and use a close approximation of their method to calculate multivariate dependencies in our algorithm.

**Definition** We first define a general form of GMI denoted by $I_p$. For parameters $p \in (0, 1)$ and $q = 1 - p$ consider two random variables $\mathbf{X} \in \mathbb{R}^{d_x}$ and $\mathbf{Y} \in \mathbb{R}^{d_y}$ with joint and marginal distributions $f(\mathbf{x}, \mathbf{y})$, $f(\mathbf{x})$, and $f(\mathbf{y})$ respectively. The GMI between $\mathbf{X}$ and $\mathbf{Y}$ is given by

$$I_p(\mathbf{X};\mathbf{Y}) = \frac{1}{4pq} \times \int \frac{\left(f_{XY}(\mathbf{x},\mathbf{y}) - q f_X(\mathbf{x}) f_Y(\mathbf{y})\right)^2}{p f_{XY}(\mathbf{x},\mathbf{y}) + q f_X(\mathbf{x}) f_Y(\mathbf{y})} \, d\mathbf{x}\, d\mathbf{y} - (p-q)^2 \,. \tag{1}$$

Considering the special case of $p = q = 1/2$ in Eqn. 1 we obtain,

$$I(\mathbf{X};\mathbf{Y}) = 1 - 2 \int \frac{f_{XY}(\mathbf{x},\mathbf{y}) f_X(\mathbf{x}) f_Y(\mathbf{y})}{f_{XY}(\mathbf{x},\mathbf{y}) + f_X(\mathbf{x}) f_Y(\mathbf{y})} \, d\mathbf{x}\, d\mathbf{y}. \tag{2}$$

The conditional form of this measure, proposed in [9], is,

$$I(\mathbf{X};\mathbf{Y}|\mathbf{Z}) = \mathbb{E}_Z\left[I(\mathbf{X};\mathbf{Y}|\mathbf{Z}=\mathbf{z})\right], \quad \text{where} \tag{3}$$

$$I(\mathbf{X};\mathbf{Y}|\mathbf{Z}=\mathbf{z}) = 1 - 2 \int \frac{f_{XY|Z}(\mathbf{x},\mathbf{y}|\mathbf{z}) f_{X|Z}(\mathbf{x}|\mathbf{z}) f_{Y|Z}(\mathbf{y}|\mathbf{z})}{f_{XY|Z}(\mathbf{x},\mathbf{y}|\mathbf{z}) + f_{X|Z}(\mathbf{x}|\mathbf{z}) f_{Y|Z}(\mathbf{y}|\mathbf{z})} \, d\mathbf{x}\, d\mathbf{y}. \tag{4}$$

**Estimator** In general, for a set of $m$ samples drawn from $f(\mathbf{x}, \mathbf{y}, \mathbf{z})$, we estimate $I(\mathbf{X}; \mathbf{Y} | \mathbf{Z})$ as follows: (1) Split data into two subsets $S_1$ and $S_2$, (2) Use the Nearest Neighbour Bootstrap algorithm [30] to generate conditionally independent samples from $S_2$ points and name the new set $S_2'$. (3) Merge $S_1$ and $S_2'$ i.e. $S := S_1 \cup S_2'$. (4) Construct a Minimum Spanning Tree (MST) on $S$. (5) Compute Friedman-Rafsky statistic [31], $\mathbf{R}_m$, which is the number of edges on the MST linking dichotomous points i.e. edges connecting points in $S_1$ to points in $S_2'$. (6) The estimate for $I(\mathbf{X};\mathbf{Y}|\mathbf{Z})$, denoted by $\hat{I}$, is obtained as $1 - \mathbf{R}_m/m$. Note: Within the MINT, we apply the conditional GMI estimator, indicated as the function $\rho()$, on a set of activations obtained from each filter we consider.

*B. Approach*

**Setup** In a deep network containing a total of $L$ layers, we compute the dependency ($\rho$) between filters in every consecutive pair of layers. Here, the layer $l$ is closer to the input while layer $l+1$ is closer to the output among the chosen pair of consecutive layers. The activations for a given node in layer $l+1$, are computed as,

$$F^{(l+1)}(\mathbf{x}) = \sigma(\mathbf{w}\mathbf{x} + \mathbf{b}), \tag{5}$$

where $\mathbf{x} \in \mathsf{R}^{m \times d}$, $m$ is the total number of samples, $d$ is the feature dimension and $\mathbf{x}$ is the input to a given layer used to compute the activations. $\sigma()$ is an activation function, $\mathbf{w} \in \mathsf{R}^{N^{(l)}}$, and $\mathbf{b}$ are the weight vector and bias.

**Notations**

- $F_i^{(l+1)}$ : The activations from the selected filter $i$ in layer $l+1$.
- $N^{(l+1)}$ : Total number of filters in layer $l+1$.
- $S_{F_i^{(l+1)}}$ : The set of indices that indicate the values that are retained in the weight vector of the selected filter.
- $\rho()$: The dependency between two filters, computed using $I(\mathbf{X}; \mathbf{Y}|\mathbf{Z})$ (importance score).
- $\overline{F_j^{(l)}}$: The set of all filters excluding $F_j^{(l)}$ in layer $l$.
- $\delta$: Threshold on importance score to ensure only strong contributions are retained.

**Description** In every iteration of MINT (Alg. 1), we find the set of weight values in $\mathbf{w}$ to retain while the remaining are zeroed out.

- For a given pair consecutive of layers $(l,\ l+1)$, we compute the dependency between every filter in layer $l+1$ in relation to filters in layer $l$. The main intent of framing the algorithm in this perspective is that the activations from layers closer to the input have a direct effect on downstream layers while the reverse is not true for a forward pass of the network.
- Using the activations $F()$ for the selected filters, $(F_i^{(l+1)}, F_j^{(l)})$ we compute the conditional GMI (Eqn. 4) between them given all the remaining filters in layer $l$. This dependency captures the relationship between filters in the context of all the contributions from the preceding layer. Since the activations of layer $l+1$ are weighted combinations from all filters in the preceding layer, we need to account for this when considering the dependence of activations between two selected filters.
- Based on the strength of each $\rho(F_i^{(l+1)}, F_j^{(l)})$, the contribution of filters from the previous layer is either retained/removed. We define a threshold $\delta$ for this purpose, a key hyper-parameter.
- $S_{F_i^{(l+1)}}$ stores the indices of all filters from layer $l$ that are retained for a selected filter $F_i^{(l+1)}$. The weights for retained filters are left the same while the weights for the entire kernel in the other filters are zeroed out. In the context of fully connected layers, we retain or zero out specific weight values.

**Algorithm 1:** MINT pruning algorithm for filters of layers $(l,\ l+1)$

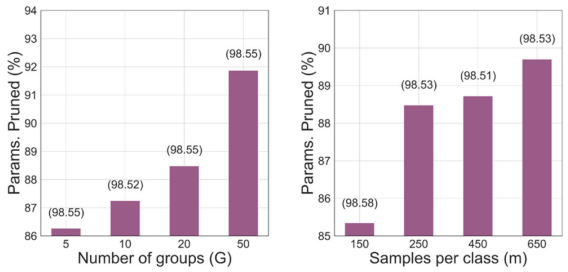
```
for Every pair of layers (l, l+1), l ∈ 1, 2, ..., L−1 do
    for F_i^(l+1), i ∈ 1, 2, ... N^(l+1) do
        Initialize S_{F_i^(l+1)} = ∅;
        for F_j^(l), j ∈ 1, 2, ... N^(l) do
            ρ(F_i^(l+1), F_j^(l)) = I(F_i^(l+1), F_j^(l) | F̄_j^(l));
            if ρ(F_i^(l+1), F_j^(l)) ≥ δ then
                S_{F_i^(l+1)} = S_{F_i^(l+1)} ∪ index(F_j^(l))
            end
        end
    end
end
```

**Group Extension** While evaluating dependencies between every pair of filters allows us to take a close look at their relationships, it does not scale well to deeper or wider architectures. To address this issue, we evaluate filters in groups rather than individually. We define $G$ as the total number of groups in a layer, where each group contains an equal number of filters. We explore in detail the impact of varying the number of groups in Section IV-D. Although there are multiple approaches to grouping filters, in this work we restrict ourselves to sequential grouping, where groups are constructed from consecutive filters. There is no explicit requirement for a pre-grouping step before our algorithm so long as a balanced grouping of filters is used.

**Finer Details** MINT is constructed on the assumption that the majority of information from the preceding layer is available and the filter in consideration can selectively retain contributions for a subset of previous filters. This allows us to work on isolated pairs of layers with minimal interference on downstream layers since retaining filters with high MI will ensure the retention of filters that contribute the most information to the next layer. By maintaining as much information as possible between layers, the amount of critical information passed to layers further on is maintained.

## IV. Experimental Results

We breakdown the experimental results into three major sections. In Section IV-C, we focus on the comparison of our method against SOTA deep network pruning algorithms, in Section IV-D we highlight the significance of various hyper-parameters used in MINT, and finally, in Section IV-E we characterize MINT-compressed networks w.r.t. their learned representations, response to adversarial attacks, and calibration statistics and compare them to their original counterparts. As a prelude to these sections, we describe the datasets, models, and metrics used across our experiments. We restrict the implementation details to the appendices.




(a) (b)

Fig. 3: (a) An increase in the number of groups per layer allows for finer grouping of filters which in turn leads to more accurate GMI estimates and thresholding. Thus, there is a steady increase in the number of parameters that can be removed to achieve > 98.50 performance. (b) Keeping $G = 20$, we observe that increasing the number of samples per class improves the GMI estimate accuracy which in turn allows for better thresholding and an increase in parameters pruned. The values on top of the bar plots are test accuracy

### A. Datasets and Models

Our experiments are divided into the following Dataset-DNN combinations, in the order of increasing complexity, MNIST [32] + Multi-Layer Perceptron, CIFAR10 [33] + VGG16 [34], CIFAR10 + ResNet56 [35] and finally ILSVRC2012 [36] + ResNet50.

### B. Metrics

**Parameters Pruned (%)**: The percentage of parameters removed from the baseline network. A higher value alongside good performance indicates a superior method.
**Test Accuracy (%)**: The best performance on the testing set, upon training, for baseline networks, and re-training, for pruning methods.
**Memory Footprint (Mb)**: Memory consumed when storing the weights of a network in CSR format under "npz" files.

### C. Comparison against existing methods

As a first step in showcasing MINT's abilities, we compare it against SOTA baselines in network pruning. The baselines in Table I are arranged in ascending order of the percentage of parameters pruned, from top-down. Our algorithm outperforms most of the SOTA pruning baselines across the number of pruned parameters while maintaining high accuracy and reducing the memory footprint of the network. We note that while most of the pruning baselines listed use multiple prune-retrain steps to achieve their result, we use only a **single step** to match or outperform them.

### D. Hyper-parameter Empirical Analysis

We take a closer look at two important hyper-parameters that help MINT scale well to deep networks, (a) number of groups in a layer $G$, and (b) the number of samples per class, $m$, used to compute the conditional GMI. Below, we look into how each of them impacts the percentage of parameters pruned while maintaining > 98.50% accuracy on MNIST + MLP.

**Group size** $G$ directly corresponds to the number of filters that are grouped together when computing conditional GMI and thresholding. Higher G leads to lesser filters per group, which should allow for a more fine-grained computation of multivariate dependency and thereby, more precise pruning. In this experiment, $m = 250$. Results in Fig. 3a match our expectations by illustrating the increase in the percentage of parameters pruned to maintain the desired performance.

**Samples per class** The number of samples per class directly impacts the final number of activations used to compute the conditional GMI. The GMI estimator should improve its estimates as the number of samples per class and the total number of samples is increased. In this experiment, $G = 20$. Fig. 3b validates our expectation by showing a steady improvement in the percentage of parameters pruned as the number of samples per class and thereby the total number of samples is increased.

### E. Characterization

The standard metrics used to compare deep network pruning methods are the percentage of parameters pruned and the overall recognition performance. However, the original intent of compressing networks was to deploy them in real-world scenarios which necessitate other characterizations like robustness to adversarial attacks, the ability to reflect true confidence in predictions, and more.

While the core idea behind MINT is to retain filters that contribute the majority of the information passed to the next layer, in using a subset of the available filters we remove a certain portion of the information passed down. Fig. 4 compares the portions of the image that contribute towards the desired target class, between the original (top row) and MINT-compressed networks (bottom row). We observe that the use of a subset of filters in the compressed network has reduced the effective portions of the image that contribute towards a decision, not to mention minor modifications to the features used themselves.

To understand the impact of pruning networks in the context of adversarial attacks we use two common adversarial attacks, Iterative FGSM [39], which doesn't exclusively target a desired class, and Iterative-LL [40], which targets the selection of the least likely class. Fig. 5 shows the response of the original and MINT-compressed networks to both attacks. We clearly observe that MINT-compressed networks are more vulnerable to targeted and non-targeted attacks. We posit that the reduction in the number of filters used and reduction in available redundant features are the reason MINT-compressed networks are vulnerable to adversarial attacks.

Calibration statistics [41] measure the agreement between the confidence provided by the network and the actual probability. These measures provide an orthogonal perspective to

TABLE I: MINT is easily able to compete with SOTA pruning methods across all our evaluated benchmarks, using only a single prune-retrain step. Baselines use multiple prune-retrain steps and are arranged in increasing order of Parameters Pruned %. We highlight a subset of available methods in the table. * indicates comparison of layer 2's weights

| | Method | Params. Pruned(%) | Test Accuracy (%) | Memory(Mb) |
|---|---|---|---|---|
| MLP MNIST | Baseline | N.A. | 98.59 | 0.537 |
| | SSL [8] | 90.95* | 98.47 | N.A. |
| | Network Slimming [21] | 96.00* | 98.51 | N.A. |
| | **MINT (ours)**($\delta$ = 0.645) | 96.20* | 98.47 | 0.022 |
| VGG16 CIFAR-10 | Baseline | N.A. | 93.98 | 53.868 |
| | Pruning Filters [7] | 64.00 | 93.40 | N.A |
| | SSS [23] | 73.80 | 93.02 | N.A |
| | GAL [15] | 82.20 | 93.42 | N.A. |
| | **MINT (ours)** ($\delta$ = 0.850) | 83.46 | 93.43 | 9.020 |
| ResNet56 CIFAR-10 | Baseline | N.A. | 92.55 | 3.109 |
| | GAL [15] | 11.80 | 93.38 | N.A. |
| | Pruning Filters [7] | 13.70 | 93.06 | N.A. |
| | NISP [19] | 42.40 | 93.01 | N.A. |
| | OED [37] | 43.50 | 93.29 | N.A. |
| | **MINT (ours)** ($\delta$ = 0.184) | 52.41 | 93.47 | 1.552 |
| | **MINT (ours)** ($\delta$ = 0.208) | 57.01 | 93.02 | 1.461 |
| ResNet50 ILSVRC2012 | Baseline | N.A. | 76.13 | 91.157 |
| | GAL [15] | 16.86 | 71.95 | N.A. |
| | OED [37] | 25.68 | 73.55 | N.A. |
| | SSS [23] | 27.05 | 74.18 | N.A. |
| | NISP [19] | 43.82 | 71.99 | N.A. |
| | ThiNet [14] | 51.45 | 71.01 | N.A. |
| | **MINT (ours)** ($\delta$ = 0.1000) | 43.01 | 71.50 | 52.365 |
| | **MINT (ours)** ($\delta$ = 0.1101) | 49.00 | 71.12 | 47.513 |
| | **MINT (ours)** ($\delta$ = 0.1103) | 49.62 | 71.05 | 46.925 |

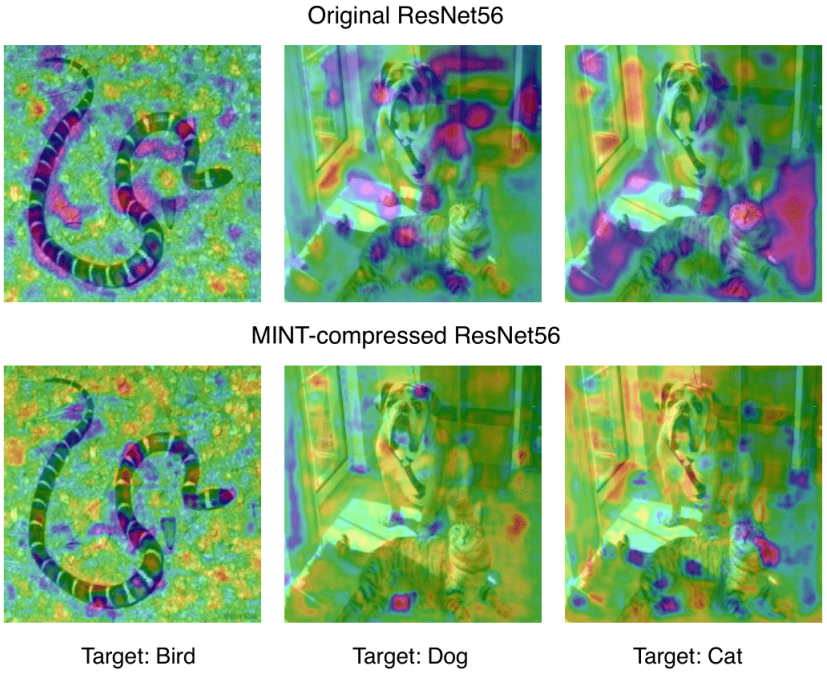
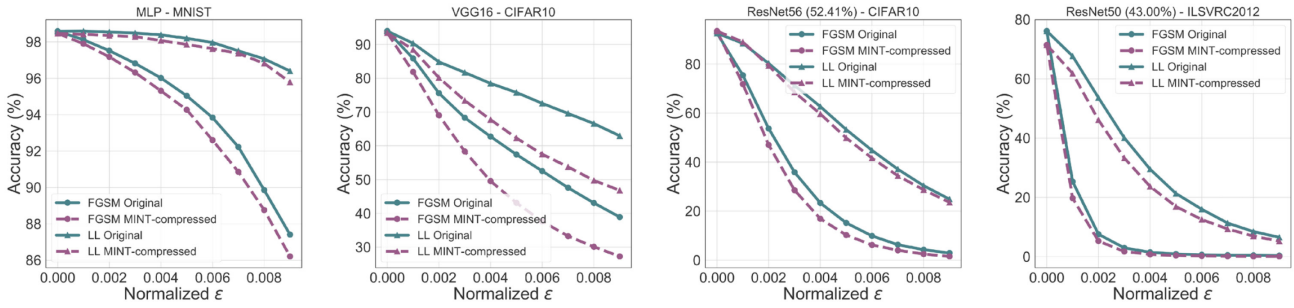


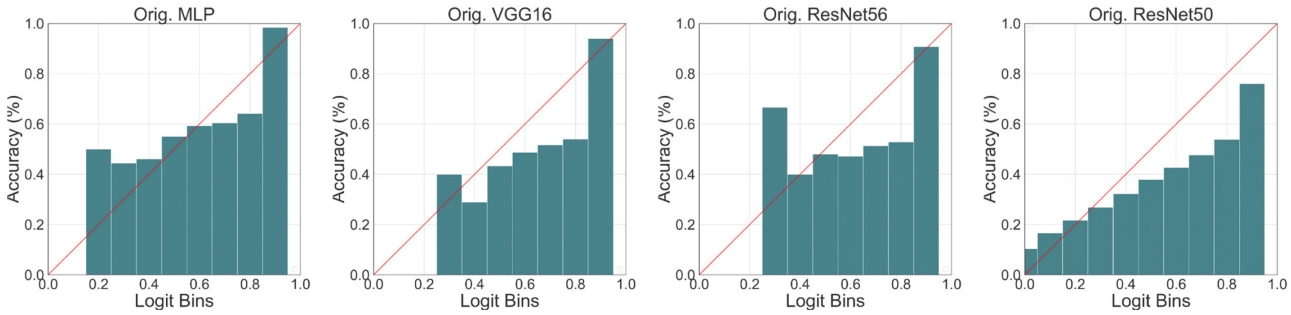

Fig. 4: Visualizations using GradCAM [38] illustrate the decrease in effective portions of the image that contribute towards specific target classes in MINT-compressed ResNet56 (row 2) when compared to the original un-pruned network (row 1)

adversarial attacks since they measure statistics only for in-domain images while adversarial attacks alter the input. Fig. 6 highlights the decrease in Expected Calibration Error (ECE) for the MINT-compressed networks when compared to their original counterparts. The plot illustrates that the histogram trend is closer to matching the ideal trend indicated by the linear red curve. After pruning, the sparse networks seem to behave similarly to a regularizer by focusing on a smaller

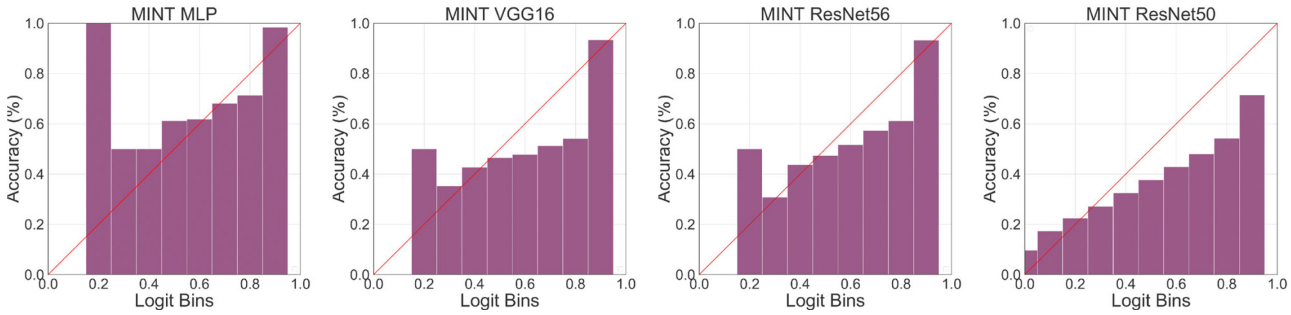



(a) (b) (c) (d)

Fig. 5: By enforcing the use of an important subset of filters from all the available ones, MINT-compressed networks begin to overvalue their importance. MINT-compressed networks seem more susceptible to targeted and non-targeted adversarial attacks when compared to the original network. Here, $E$ refers to the $E$ ball in $l_\infty$ norm




(a) ECE:0.0077 (b) ECE:0.0517 (c) ECE:0.0762 (d) ECE:0.0305




(e) ECE:0.0054 (f) ECE:0.0500 (g) ECE:0.0383 (h) ECE:0.0069

Fig. 6: Calibration statistics measure the agreement between the confidence output of the network and the true probability. The red line indicate the ideal trend. We observe that MINT-compressed networks act as a regularizer to decrease the Expected Calibration Error (ECE) when compared to the original network as well as better match the ideal curve

subset of features and decreasing the ECE. On the other hand, the original networks contain many levels of redundancies which could translate to overfitting and having higher ECE.

## V. Conclusion

In this work, we propose MINT as a novel approach to network pruning in which the dependency between filters of successive layers is used as a measure of importance. We use conditional GMI to evaluate importance and incorporate stochasticity in our algorithm to help retain filters that pass the majority of the information through layers. In doing so, MINT achieves better pruning performance than SOTA baselines, using a single prune-retrain step. When characterizing the behaviour of MINT-pruned networks, we observe that it behaves like a regularizer and improves the calibration of the network. However, a reduction in the number of filters

used and redundancies makes pruned networks susceptible to adversarial attacks. Our future direction of work includes improving the robustness of compressed networks as well as detailing the sensitivity of layers to network pruning.

## ACKNOWLEDGMENT

This work has been partially supported (Madan Ravi Ganesh and Jason J. Corso) by NSF IIS 1522904 and NIST 60NANB17D191 and (Salimeh Yasaei Sekeh) by NSF 1920908; the findings are those of the authors only and do not represent any position of these funding bodies.

## REFERENCES

[1] J.-G. Lee, S. Jun, Y.-W. Cho, H. Lee, G. B. Kim, J. B. Seo, and N. Kim, "Deep learning in medical imaging: general overview," *Korean journal of radiology*, vol. 18, no. 4, pp. 570–584, 2017.

[2] A. M. Abdel-Zaher and A. M. Eldeib, "Breast cancer classification using deep belief networks," *Expert Systems with Applications*, vol. 46, pp. 139–144, 2016.

[3] G. Tinchev, A. Penate-Sanchez, and M. Fallon, "Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a cpu," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1327–1334, 2019.

[4] S. M. Grigorescu, B. Trasnea, L. Marina, A. Vasilcoi, and T. Cocias, "Neurotrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3441–3448, 2019.

[5] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[6] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[8] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

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

[10] J. H. Friedman, L. C. Rafsky *et al.*, "Graph-theoretic measures of multivariate association and prediction," *The Annals of Statistics*, vol. 11, no. 2, pp. 377–391, 1983.

[11] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of the British Machine Vision Conference 2014*. British Machine Vision Association, 2014. [Online]. Available: https://doi.org/10.5244%2Fc.28.88

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[13] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017. [Online]. Available: https://doi.org/10.1109%2Ficassp.2017.7953072

[14] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[15] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2790–2799.

[16] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[17] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in neural information processing systems*, 2016, pp. 1379–1387.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[19] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.

[20] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2554–2564.

[21] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[22] J. Yoon and S. J. Hwang, "Combined group and exclusive sparsity for deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3958–3966.

[23] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 304–320.

[24] J. Li, Q. Qi, J. Wang, C. Ge, Y. Li, Z. Yue, and H. Sun, "Oicsr: Out- in-channel sparsity regularization for compact deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7046–7055.

[25] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[26] K. R. Moon, K. Sricharan, and A. O. Hero, "Ensemble estimation of mutual information," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 3030–3034.

[27] M. Noshad, K. R. Moon, S. Y. Sekeh, and A. O. Hero, "Direct estimation of information divergence using nearest neighbor ratios," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 903–907.

[28] N. Leonenko, L. Pronzato, V. Savani *et al.*, "A class of rényi information estimators for multidimensional densities," *The Annals of Statistics*, vol. 36, no. 5, pp. 2153–2182, 2008.

[29] S. Gao, G. Ver Steeg, and A. Galstyan, "Efficient estimation of mutual information for strongly dependent variables," in *Artificial intelligence and statistics*, 2015, pp. 277–286.

[30] R. Sen, A. T. Suresh, K. Shanmugam, A. G. Dimakis, and S. Shakkottai, "Model-powered conditional independence test," in *Advances in neural information processing systems*, 2017, pp. 2951–2961.

[31] J. H. Friedman and L. C. Rafsky, "Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests," *Ann. Statist.*, pp. 697–717, 1979.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

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

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

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2016.90

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[37] Z. Wang, S. Lin, J. Xie, and Y. Lin, "Pruning blocks for cnn compression and acceleration via online ensemble distillation," *IEEE Access*, vol. 7, pp. 175 703–175 716, 2019.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[40] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[41] M. P. Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.