Deep Compression with Adversarial Robustness Via Decision Boundary Smoothing

Alok Shah, Michael Shao
[alokshah],[michlsha]@seas.upenn.edu
University of Pennsylvannia
Philadelphia, Pennsylvania, USA

Abstract

Deep neural networks (DNNs) are highly effective for a wide range of tasks but are often computationally expensive and vulnerable to adversarial attacks. This paper explores the intersection of model compression and adversarial robustness. In particular, we investigate how the locality and structure of pruning and weight-sharing affect model accuracy, size, and robustness. Our analysis reveals that compressed models exhibit lower adversarial robustness, which we attribute to jagged decision boundaries. To address this, we propose a retraining pipeline that integrates adversarial training with maximum margin regularization to smooth decision boundaries and enhance robustness lost during compression. Experiments on VGG16 with the ILSVRC 2012 data set demonstrate that our method doubles the robust accuracy towards FGSM compared to baseline, while reducing the approximate model size by up to 90% through weight-sharing. These results highlight the potential for achieving lightweight yet robust models suitable for deployment in resource-constrained and safety-critical environments.

Keywords

Compression, Pruning, Quantization, Adversarial Robustness

ACM Reference Format:
Alok Shah, Michael Shao. 2025. Deep Compression with Adversarial Robustness Via Decision Boundary Smoothing. In *ESE 5390 Final Project, December 05, 2024, Philadelphia, PA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The growing availability of computational resources has fueled the development of larger and more powerful image classifiers, which exhibit impressive performance on independent and identically distributed (IID) datasets [5, 16]. However, despite their success, the substantial memory and computational requirements of these models remain significant bottlenecks to their adoption in resource-constrained environments [14]. Furthermore, their susceptibility to adversarial attacks raises concerns about their deployment in safety-critical applications, such as healthcare and autonomous systems. Addressing these challenges requires methods that balance efficiency, accuracy, and robustness [19].

1.1 Pruning

Pruning is a widely studied technique for compression deep neural networks (DNNs) by removing a certain percentage, known as a *prune rate*, of redundant or non-essential parameters [6, 11, 20]. Formally, for a model h, pruning procedure $\mathcal{A}$ and prune rate r, we generate $h_{\text{pruned}} = \mathcal{A}(h, r)$. We additionally define set $\mathcal{L}_{prune}(h)$ as the set of all layers in h that can apply pruning as well as $W(h)$ and $W(l)$, the set of all weights in the model or layer, respectively. We abuse notation and the size of a model as $|h|$ and the amount of weights as $|W(h)|$. For some loss function ℓ, and dataset $\mathcal{D} = \{(x_i, y_i) \mid i \in [n]\}$ we measure the performance of $\mathcal{A}$ on h by analyzing the relative performance

$$\ell_{\text{rel}} = \frac{\ell(y, h_{\text{pruned}}(x))}{\ell(y, h(x))} \tag{1}$$

Naturally, this yields a pareto-esque tradeoff between ℓ_{rel} and $|h_{\text{pruned}}|$, though, this gap is typically recoverable via retraining. Such procedures $\mathcal{A}$ can be categorized into structured and unstructured methods. Structured methods target specific parts of the neural architecture, while the unstructured methods remove individual weights. As pruning has been extensively explored, there exist many $\mathcal{A}$ which achieve both $\ell_{\text{rel}} \approx 1$ and small $|h_{\text{pruned}}|$. In this work, we'll explore:

(1) Global Unstructured Pruning
(2) Layer Unstructured Pruning
(3) Layer Structured Pruning

1.2 Quantization

Quantiziation is a model compression technique that reduces the precision of a model's parameters, typically moving from 32-bit floating point representations to lower-bit formats [17]. This reduction significantly decreased the memory footprint, and subsequently computations demands of DNNs. Formally, for a model h with weights W, quantization can be represented

$$W_{\text{quantized}} = Q(W) \tag{2}$$

for some map Q. In this paper, we consider the Q induced by weight-sharing.

1.3 Adversarial Attacks

Adversarial attacks introduce small, carefully chosen perturbations δ to the input $x \in \mathbb{R}^d$, causing DNNs to misclassify the perturbed examples while appearing unchanged to human observers [7]. Formally, they seek to solve the following optimization problem

$$\delta^* = \arg\max_{\delta \in \Delta_\epsilon} \ell(h(x+\delta), y) \tag{3}$$

Where $\Delta_\epsilon = \{\delta \in \mathbb{R}^d \mid ||\delta||_p \leq \epsilon\}$ is the set of allowable perturbations under a specified p-norm constraint. In the classification setting, the effectiveness of an adversarial attack is therefore measures by the attack success rate (ASR):

$$\text{ASR} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{h(x_i + \delta_i^*) \neq y_i\} \tag{4}$$

It follows that an *adversarially robust* model seeks to minimize ASR or maximize:

$$\begin{aligned}\text{Robust Accuracy} &= 1 - \text{ASR} \\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{h(x_i + \delta_i^*) = y_i\}\end{aligned} \tag{5}$$

Given the scope of this project, and the limited time and resources, we'll focus on the Fast Signed Gradient Method (FGSM) attack which chooses δ^* in accordance with the gradient $\nabla \ell(x)$, exploiting vulnerabilities in a model's decision boundaries, highlighting a fundamental trade-off between accuracy on clean data and robustness to adversarial examples [7].

Typical defense mechanisms, such as adversarial training, aim to strengthen decision boundaries against these perturbations [13]. However, most successful adversarially-aware training heuristics rely on adversarial training from scratch, and seldom begin from a pretrained model (as they've already learned biases) [22]. To this end, prior literature is rich with approaches that perform well on low-resolution, low-class datasets like MNIST and CIFAR-10, and fail to deliver the same robustness guarantees on ImageNet [2]. As such, achieving high accuracy, model compression, and adversarial robustness on ImageNet, is particularly challenging **??**

Despite the empirical success of model compression, it's interplay adversarial robustness remains an unexplored area, motivating our investigation into leveraging the structural changes induced by pruning and quantization to improve robustness. Our contributions are threefold:

(1) **Comprehensive Evaluation of Compression Algorithms**: We implement four different pruning algorithms along with weight-sharing for model compression and evaluate their effects of compression on accuracy and adversarial robustness
(2) **Mechanistic Analysis of Decision Boundaries**: We analyze the differences in smoothness decision boundaries between models and their compressed counterparts. Our key insight is that compressed models exhibit highly jagged, non-smooth decision boundaries
(3) **Compression with Adversarial Robust Retraining**: We leverage compressed model structure to devise a retraining algorithm that increases adversarial robustness through adversarial retraining and smoothing decision boundaries

By systematically evaluating the relationships between quantization, weight-sharing, and adversarial robustness, we aim to provide practical insights and algorithms for deploying robust compressed models in real-world scenarios.

2 Methodology

We'll briefly discuss and analyze the compression methods, adversarial attack, and robustness techniques we implement.

2.1 Pruning Methods

2.1.1 Global Unstructured Pruning. This method targets the lowest percentile of the model's weights in magnitude – where pruning is feasible [6]. We'll denote this procedure as $\mathcal{A}_{Global}$, such that for $h$pruned $= \mathcal{A}_{Global}(h, r)$, the pruned weight set satisfies $|h_{\text{pruned}}| = (1-r)|h|$.

Algorithm 1 Global Pruning $\mathcal{A}_{Global}$

```
Require: $r \in (0, 1)$
Require: $\mathcal{L}_{prune}(h) \neq \emptyset$
  $W' = W(\mathcal{L}_{prune}(h))$
  $W_{sorted} = sort(W')$
  for $i = 0; i < r \times |W_{sorted}|; i = i + 1$ do
    $W_{sorted,i} = 0$
  end for
  return $h$
```

Observe that Algorithm 1 only removes small weights, imposing no constraints on the location of these weights. While Algorithm 1 enjoys substantial compression with minimal performance degradation, for larger values of r, layers with many small weights retain significantly fewer parameters, leading to an information bottleneck

2.1.2 Layer-wise Unstructured Pruning. In contrast, this method targets the lowest percentile of weights within each prunable model layer. We'll denote this as $\mathcal{A}_{Layer}$ with $h_{\text{pruned}} = \mathcal{A}_{Layer}(h, r)$, where the size of the $i-th$ layer satisfies $|h^{(i)}_{\text{pruned}}| \approx (1-r)|h^{(i)}|$

Algorithm 2 Layered Unstructured Pruning $\mathcal{A}_{Layer}$

```
Require: $r \in (0, 1)$
Require: $\mathcal{L}_{prune}(h) \neq \emptyset$
  for $l \in \mathcal{L}_{prune}(h)$ do
    $W_{sorted} = sort(W(l))$
    for $i \in [0, r \times |W_{sorted}|)$ do
      $W_{sorted,i} = 0$
    end for
  end for
  return $h$
```

Obseve that ensures an even distribution of pruned weights across each layer. However, larger weights may be pruned by constraining pruning at the layer level, cascading into lower accuracy post-pruning.

2.1.3 *Layer-wise Structured Pruning.*

This method prunes structural components of the architecture [20]. We focus on output channel pruning, denoted as $\mathcal{A}_{Channel}$, which removes entire dimensions from the model, facilitating potential model layer restructuring. To formalize, let $G(l)$ be all channels in the layer l, where, for the channel $g \in G(l)$, $|g|$ is the sum of weight magnitudes.

Algorithm 3 Layered Structured Pruning $\mathcal{A}_{Channel}$

```
Require: $r \in (0, 1)$
Require: $\mathcal{L}_{prune}(h) \neq \emptyset$
  for $l \in \mathcal{L}_{prune}(h)$ do
      $G_{sorted} = sort(G(l))$
      for $i \in [0, r \times |G_{sorted}|)$ do
          $G_{sorted,i} = 0$
      end for
  end for
  return $h$
```

Note the additional constraint: because the system considers aggregate magnitudes across the channels, Algorithm 3 still prunes large weights (outliers in their channel).

2.1.4 *Model Hardening.*

Structured pruning only sets weights to 0, but keeps the model structure. Model hardening grafts these non-zero parameters into a reduced, more compact duplicate model. We'll denote this restructuring method as $\mathcal{H}(h)$.

Algorithm 4 Hardening $\mathcal{H}(h)$

```
Require: $\mathcal{L}_{\text{prune}}(h) \neq \emptyset$
Require: $h_{prune} = \mathcal{A}_{Channel}(h)$
  $h_{hardened} = copy$
  for $lin\mathcal{L}_{\text{prune}}(h_{prune})$ do
      $h_{\text{temp}} = \mathcal{A}_{\text{channel}}(h)$
      $h = \text{retrain}(h_{\text{temp}})$
  end for
  return harden($h$)
```

When measuring the performance of a structured pruning algorithm, we use this method to obtain both pruned model size in Megabytes and runtime acceleration.

2.1.5 *Iterative Pruning.*

Rather than structurally pruning the target rate r in a single step, this method prunes in increments of r_{inc}, until the target rate is achieved. At each pruning step, we retrained the model, allowing it to adapt to the new network structure and reinforce the most critical connections before the next pruning step [10].

Generally, we would harden the model after pruning step to ensure that the pruned weights are not reinstated in training. Our modified training algorithm masks over pruned nodes and keeps their gradients set to 0. In this way, pruned nodes are not affected by training, and the hardening step is reserved for the end to save runtime.

Algorithm 5 Iterative Pruning $\mathcal{A}_{Iterative}$

```
Require: $r \in (0, 1)$
Require: $\mathcal{L}_{\text{prune}}(h) \neq \emptyset$
  for $i = r_{\text{inc}}; i < r; i = i + r_{\text{inc}}$ do
      $h_{\text{temp}} = \mathcal{A}_{\text{channel}}(h)$
      $h = \text{retrain}(h_{\text{temp}})$
  end for
  return harden($h$)
```

Using Algorithm 5 requires careful design of retrain(), especially for small r_{inc} as using the full training dataset may result in unreasonable runtime.

2.2 Weight-Sharing

Instead of directly quantizing weights, weight sharing groups non-zero weights into 2^n linearly spaced values, l_{bins} using K-means clustering [10]. Each weight is mapped to it's nearest bin, and the resulting quantized weights are stored in a compressed coordinate (COO) format, which includes the matrix indices and corresponding bin-referenced values.

Algorithm 6 Weight-Sharing Q

```
Require: $n < weight\_data\_size$
Require: $\mathcal{L}_{\text{prune}}(h) \neq \emptyset$
  for $l \in \mathcal{L}_{\text{prune}}(h)$ do
      $(C, W) = To\_COO(l)$
      $max = max\_weight(l)$
      $min = min\_weight(l)$
      $bins = linspace(min, max, 2^n)$
      $(C, Quantized\_W) = Kmeans(bins, (C, W))$
  end for
  return $To_Layer(C, Quantized\_W)$
```

This approach replaces 32-bit weights with n-bit indices, and l_{bins} is stored as a lookup table. The compression rate is given by.

$$\frac{2^n \times \text{weight\_data\_size} + n \times \text{number\_of\_weights}}{\text{number\_of\_weights} \times \text{weight\_data\_size}} \tag{6}$$

$$= \frac{2^n}{\text{number\_of\_weights}} + \frac{n}{\text{weight\_data\_size}} \tag{7}$$

2.3 Fast Signed Gradient Method (FGSM)

FGSM is a simple, widely used method for generating adversarial examples [7]. It exploits a model's gradient information to perturb the input x in the direction that maximally increases (3). Given some input, label pairs x, y, FGSM generates an adversarial example x_{adv} using:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(h(x), y)) \tag{8}$$

where ϵ controls the perturbation magnitude, and subsequently the strength of attack. Although stronger attacks exist, for our purposes, FGSM generates sufficiently strong adversarial examples which drastically reduce test accuracy – highlighting the difference in dynamics between models and their compressed counterparts.

2.4 Maximum Margin Regularization (MMR)

MMR enforces larger separations between the predicted logits of the correct class and the next largest logits, thereby encouraging smoother decision boundaries and thus, a model's robustness towards small perturbations [21]. Given the logits z of h for an input x, we define the maximum margin loss as:

$$\text{MM}(z, y) = \max\left(0, 1 - \left(z_y - \max_{i \neq y} z_i\right)\right) \tag{9}$$

Where z_y is the logit of the correct class and $\max_{i \neq y} z_i$ is the largest logit among incorrect classes. Since each $z_i \in [0, 1]$, the margin 1 ensures a minimum gap between classes, as well as the continuity of MM.

Across the entire dataset, we choose a penalty constant $\lambda > 0$:

$$\ell_{\text{MM}} = \lambda * \text{MM} \tag{10}$$

Which controls the strength of the regularization. This complements standard cross-entropy loss, improving robustness by creating smoother decision boundaries between classes.

2.5 Adversarial Training

Adversarial training explicitly trains the model on adversarial examples in addition to clean ones, and seeks to solve:

$$\min_h \mathbb{E}_{(x,y)} \max_{\delta \in \Delta_\epsilon} \ell(h(x + \delta), y) \tag{11}$$

In practice, (11) is solved iteratively using ERM: running a specified attack to generate adversarial examples, and taking a step using an optimization algorithm.

2.6 Proposed Retraining Algorithm

We integrate compression, MMR and adversarial training into a simple compression and retraining pipeline presented in Alg 7:

Algorithm 7 Compression with Adversarial Robust Retraining

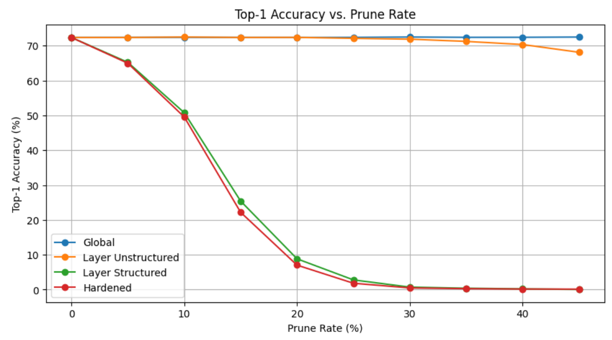
```
Require: Model $h$, dataset $\mathcal{D}$, pruning method $\mathcal{A}$, pruning rate $r$,
  perturbation magnitude $\epsilon$, margin weight $\lambda$, total epochs $T$.
  Initialize pruned model: $h_{\text{pruned}} = \mathcal{A}(h, r)$
  for epoch $t = 1$ to $T$ do
      for $(x, y) \in \mathcal{D}$ do
          Generate adversarial examples using FGSM:
              $x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(h_{\text{pruned}}(x), y))$
          Combine inputs:
              $\mathcal{X}_{\text{combined}} = \{x, x_{\text{adv}}\}$
              $\mathcal{Y}_{\text{combined}} = \{y, y\}$
          Forward pass:
              $z = h_{\text{pruned}}(\mathcal{X}_{\text{combined}})$
          Compute cross-entropy loss:
              $\mathcal{L}_{\text{CE}} = \ell(z, \mathcal{Y}_{\text{combined}})$
          Compute maximum margin regularization loss:
              $\mathcal{L}_{\text{MM}} = \lambda \cdot \text{MarginLoss}(z, \mathcal{Y}_{\text{combined}})$
          Compute total loss: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MM}}$
          Backpropagation: step$(\nabla_h \mathcal{L}_{\text{total}})$
      end for
  end for
  return model $h_{\text{pruned}}$
```

3 Experiments and Evaluation

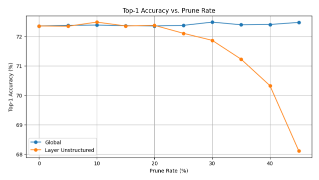
In our experiments, we aim to evaluate the impact of different compression and adversarial robustness techniques on a VGG16 model trained on the ILSVRC 2012 subset of the ImageNet dataset [18] [3]. These images are categorized into 1,000 distinct classes with images resized to a standard resolution of 224 x 224 pixels. Though this dataset consists of over 1.2 million, we only use the 50,000 validation images due to resource constraints. We split the validation set into a retrain set of 40k images and a test set of 10k images for validation and adversarial robustness evaluation. We conduct all pruning experiments on an NVIDIA Tesla P100 GPU and all robustness experiments using an NVIDIA Tesla A100 GPU.

3.1 Pruning

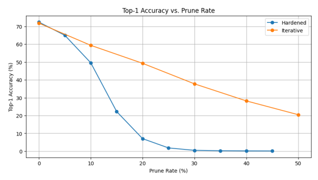
3.1.1 Accuracy. We implement the four pruning methods discussed in Sect 2. Observe that structured methods further constrain pruning, meaning larger weights in magnitude may disappear, and thus, lower accuracy.



(a) Pruning Accuracy With and Without Retraining



(b) Global vs Layer-wise Unstructured Pruning

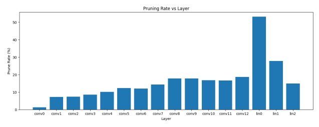


(c) Hardened vs Iterative Unstructured Pruning

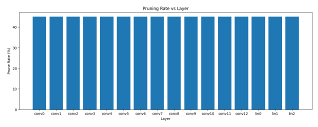
Figure 1: Pruning Method Impact on Accuracy

Indeed, Figure 1a confirms our intuition, showing that structured pruning exhibits faster accuracy dropoff across the tested prune rates. However, we also observe that iterative pruning with retraining softens the accuracy drop from exponential to linear decay.

3.1.2 Sparsity Distribution. As expected, layer-wise unstructured pruning results in a uniform distribution of sparsity across layers shown in Figure 2. Interestingly, Global Unstructured Pruning admits a left-skewed sparsity distribution as shown Figure 2. Hence, we conclude the linear layers, which occur later in the model, tend to have the majority of smaller magnitude weights, though we cannot directly attribute this to any single cause.



(a) Global Unstructured



(b) Layer-Wise Unstructured

Figure 2: Sparsity Rates by Prunable Layer

3.1.3 Hardening. After using Algorithm 4 to graft the structurally pruned model, Figure 1a shows identical accuracy curves between the pruned model and it's hardened counterpart, indicating independence between model accuracy with respect to prune rate and model dimension.

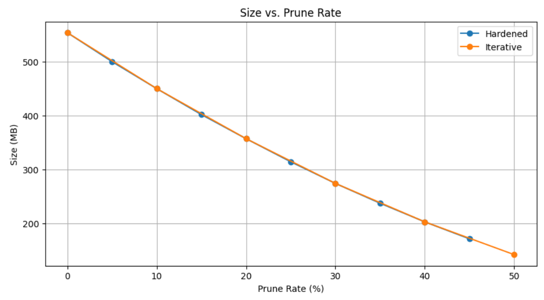


Figure 3: Compression Rate vs Pruning Rate

In Figure 3, notice that iterative pruning, while maintaining accuracy higher than structured pruning, yields identical memory advantages. This indicates that iterative pruning both yields the memory benefits of structural pruning and maintains better accuracy.

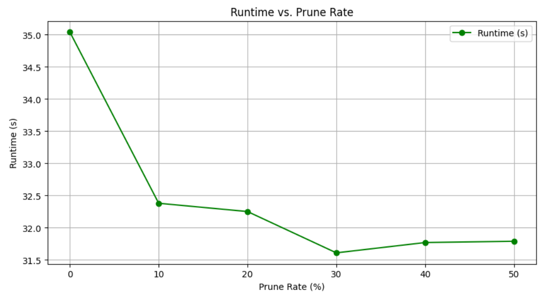


Figure 4: Runtime of Hardened, Iterative Pruning

Figure 4 demonstrates a modest runtime improvement in the hardened model, which we attribute to its smaller size, though, this gain may be hardware dependent.

3.2 Weight-Sharing

We applied Algorithm 6 to our model at various pruning rates, utilizing 2^8 bins for convolutional layers and 2^5 bins for linear layers. We seek to analyze the error introduced by weight-sharing.

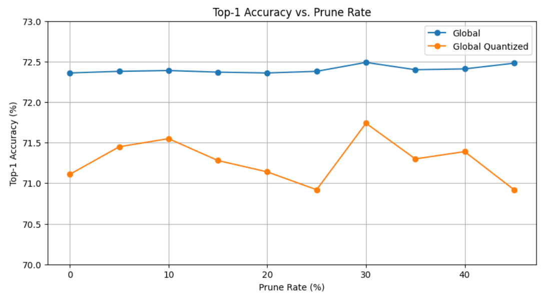
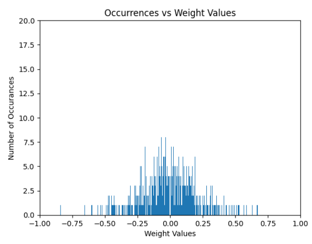
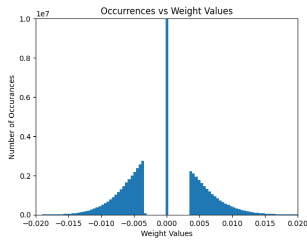


Figure 5: Global Pruning Accuracy: Quantized vs Normal

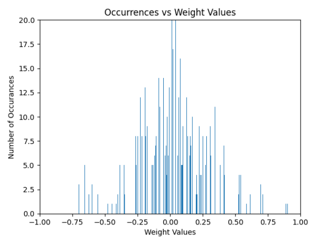
Figure 5 indicates that weight sharing incurs a marginal cost in accuracy (around 1%-2%), independent of the pruning rate. This agrees with prior literature as many weights are adjusted away from optimal sets [10].



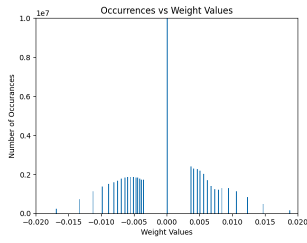
(a) Convolutional Layer 0



(b) Linear Layer 0



(c) Convolutional Layer 0, Quantized



(d) Linear Layer 0, Quantized

Figure 6: Occurrence Distribution: Quantized vs Unquantized

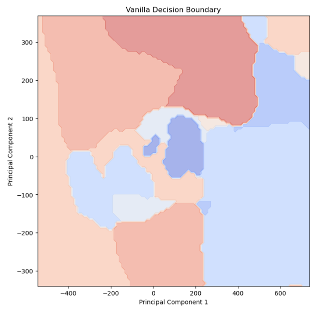
We confirm in Figure 6 the discretization of layer weights, granting us the corresponding guarantee 7.

3.3 Decision Boundary Smoothness

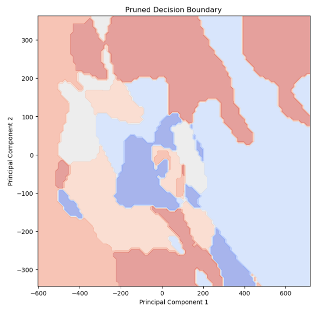
Prior literature establishes an intimate connection between decision boundary smoothness and robustness [4]. Indeed, models which are more susceptible to small perturbations in input should intuitively exhibit less smooth boundaries. Since these compressed models demonstrate lower robustness, we sought to further explore the smoothness of their decision boundaries.

To evaluate the impact of compression on model robustness, we employed a visualization heuristic to the high-dimensional image inputs into a two-dimensional space, allowing us to inspect boundary characteristics.

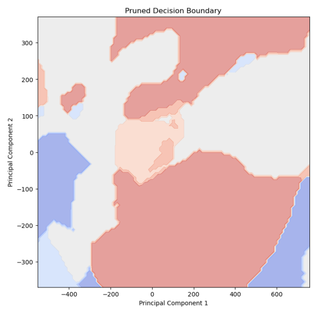
Specifically, using PCA, we reduced the input data into two principal components, to capture the highest possible variance of the original data. We then sample a grid of points in this reduced space and map back to the input space using the inverse transformation. We compute predictions for each point on this grid, enabling us to plot decision boundaries as regions of different predicted classes. We conduct our analysis on each pruning technique with and without weight-sharing and attach some of our results in Figure 7 along with their robust accuracy to FGSM. The darker colors indicate high confidence and smoother boundaries, while the lighter ones and greys indicate lower confidence and more granular boundaries. Interestingly, we find that weight-sharing does not seem to impact either robustness or smoothness by much (if at all). It's clear to see our figures support our intuition about smoothness.



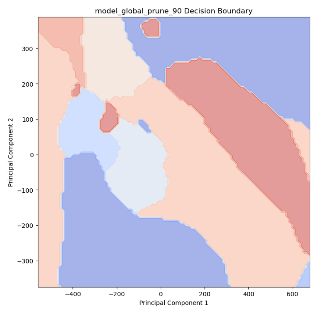
(a) Baseline VGG16, RA = 12.6%



(b) Layer-wise, RA = 8.5%



(c) Layer-wise Output, RA = 8.8%



(d) Global, RA = 5.7%

Figure 7: Decision Boundary Smoothness

3.4 Robustness

Given the results in Figure 7, we use smoothness as a proxy for robustness to define Algorithm 7, where the MMR term seeks to smooth the decision boundary in conjunction with adversarial training using FGSM. We run Alg 7 for 1 epoch on what we observed to be the most successful pruning technique, global unstructured pruning with 50% prune rate, followed by weight-sharing. Our hyperparameters are located in Appendix C. Note that due to the small quantity of training data, VGG16 tends to overfit extremely quickly, even in the presence of high regularization. Still, we obtain noteworthy results shown in Table 1.

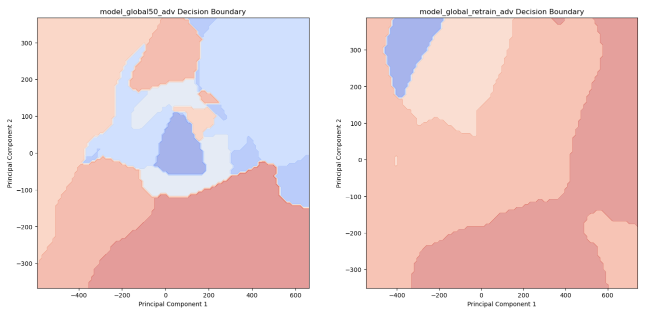
Table 1: Performance and Robustness Comparison

Model	Acc (%)	Rob. Acc (%)	Approx Size (MB)
Baseline	**71.8**	12.0	534.3
Pruned	57.0	5.8	263.9
Pruned + Retrained	64.2	10.3	263.9
Pruned + Adv Ret.	54.6	**25.6**	263.9
Pruned + Adv Ret. + WS	55.5	23.9	**45.5**

Note that we calculate approximate model size using

$$\text{Approximate Model Size} = \text{ of Nonzero parameters} * \text{Precision}$$

Retraining with Algorithm 7 demonstrates a notable improvement in robust accuracy: more than **2x** the baseline and recovering much of the robustness lost during pruning, recovering as much as **5x**. Figure 8 further supports our boundary smoothing argument, showing that Algorithm 7 results in significantly smoother contours.



(a) Pruned VGG16, RA = 5.8% **(b) Pruned VGG16 w/ Alg 7, RA = 25.6%**

Figure 8: Impact of Alg 7 on Boundary Smoothness

Although the integration of adversarial retraining and weight-sharing leads to a reduction in accuracy, we argue this behavior is expected, given the sparse retraining data, short training time (1 epoch), and injection of an adversarial example for each clean one.

4 Discussion

Our study reveals a nuanced trade-off between model compression, adversarial robustness, and accuracy retention. By combining pruning, weight sharing, and adversarial retraining, we significantly reduced model size while improving robust accuracy. The key insight is that pruning and quantization induce nonsmooth decision boundaries, which we aim to smooth through adversarial retraining with maximum margin regularization. This technique improves robustness by enlarging the separation between class logits, making the model resistant to adversarial perturbations. Although our approach doubled robust accuracy against FGSM attacks, the constrained training set and limited computing restricted our exploration of more advanced attacks.

4.1 Limitations

Despite promising results, this work faces several crucial limitations.

- **Pytorch Storage Formatting** PyTorch's default model storage format prevents direct comparison of custom quantized model sizes. Future work should leverage the weight shared model to reduce model persist size using Huffman encoding or other lossless compression.
- **Restricted Training Data**: Due to our computational budget, we only used a subset of the ILSVRC 2012 validation dataset, limiting our ability to generalize results. Future work should attempt retraining on the training dataset, or performing pruning and adversarial training while training VGG16 from scratch.
- **Attack Coverage**: We only tested robustness using FGSM, a one-step, and therefore weaker attack. Future work should explore stronger attacks like Projected Gradient Descent (PGD), AutoAttack, and DeepFool in addition to black-box attacks.
- Regularization Techniques While our study focused on MMR, we encourage future research to explore the structural properties of compressed models further. Notably, global unstructured pruning led to uneven layerwise sparsity, creating potential information bottlenecks. Investigating how these sparsity patterns impact adversarial robustness from a theoretical perspective remains an open and underexplored area.

4.2 Conclusions

Overall, these results suggest that deep compression and adversarial robustness are not mutually exclusive and can be leveraged together for deployment in safety-critical and resource-constrained environments. Addressing the highlighted limitations and future directions could unlock even greater potential in this underexplored intersection of model compression and adversarial robustness.

Acknowledgments

Thank you to the Course Staff, Jing Li, Linus Wong, and Zhiyao Tang, for being extremely patient, accommodating, and helpful throughout this entire process.

We'd also like to acknowledge ChatGPT for providing related work resources and aiding grammar.

References

[1] Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. 2019. Certified Adversarial Robustness via Randomized Smoothing. arXiv:1902.02918 [cs.LG] https://arxiv.org/abs/1902.02918
[2] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. 2019. Provable Robustness of ReLU networks via Maximization of Linear Regions. arXiv:1810.07481 [cs.LG] https://arxiv.org/abs/1810.07481
[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255. https://doi.org/10.1109/CVPR.2009.5206848
[4] Gamaleldin F Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Ian Goodfellow, and Alexey Kurakin. 2018. Large Margin Deep Networks for Class Imbalanced Data. *arXiv preprint arXiv:1805.07466* (2018).
[5] Benjamin Feuer, Jiawei Xu, Niv Cohen, Patrick Yubeaton, Govind Mittal, and Chinmay Hegde. 2024. SELECT: A Large-Scale Benchmark of Data Curation Strategies for Image Classification. arXiv:2410.05057 [cs.CV] https://arxiv.org/abs/2410.05057
[6] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. arXiv:1803.03635 [cs.LG] https://arxiv.org/abs/1803.03635
[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).
[8] Shupeng Gui, Haotao Wang, Chen Yu, Haichuan Yang, Zhangyang Wang, and Ji Liu. 2019. Model Compression with Adversarial Robustness: A Unified Optimization Framework. arXiv:1902.03538 [cs.LG] https://arxiv.org/abs/1902.03538
[9] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2018. Sparse DNNs with Improved Adversarial Robustness. *arXiv preprint arXiv:1810.09619* (2018).
[10] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv:1510.00149 [cs.CV] https://arxiv.org/abs/1510.00149
[11] Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both Weights and Connections for Efficient Neural Networks. arXiv:1506.02626 [cs.NE] https://arxiv.org/abs/1506.02626
[12] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *arXiv preprint arXiv:1712.05877* (2018).
[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083* (2018).
[14] Vinay Uday Prabhu and Abeba Birhane. 2020. Large image datasets: A pyrrhic win for computer vision? arXiv:2006.16923 [cs.CY] https://arxiv.org/abs/2006.16923

[15] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified Defenses against Adversarial Examples. *arXiv preprint arXiv:1801.09344* (2018).

[16] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. 2017. Large-Scale Evolution of Image Classifiers. arXiv:1703.01041 [cs.NE] https://arxiv.org/abs/1703.01041

[17] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. 2023. A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification. *ACM Transactions on Intelligent Systems and Technology* 14, 6 (Nov. 2023), 1–50. https://doi.org/10.1145/3623402

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet Large Scale Visual Recognition Challenge. In *International Journal of Computer Vision (IJCV)*, Vol. 115. Springer, 211–252. https://doi.org/10.1007/s11263-015-0816-y

[19] Brijesh Vora, Kartik Patwari, Syed Mahbub Hafiz, Zubair Shafiq, and Chen-Nee Chuah. 2023. Benchmarking Adversarial Robustness of Compressed Deep Learning Models. arXiv:2308.08160 [cs.LG] https://arxiv.org/abs/2308.08160

[20] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning Structured Sparsity in Deep Neural Networks. arXiv:1608.03665 [cs.NE] https://arxiv.org/abs/1608.03665

[21] Jiaping Wu, Zhaoqiang Xia, and Xiaoyi Feng. 2022. Improving Adversarial Robustness of CNNs via Maximum Margin. *Applied Sciences* 12, 15 (2022). https://doi.org/10.3390/app12157927

[22] Yixuan Zhang and Feng Zhou. 2024. Bias Mitigation in Fine-tuning Pre-trained Models for Enhanced Fairness and Efficiency. arXiv:2403.00625 [cs.LG] https://arxiv.org/abs/2403.00625

A Related Works

Adversarial robustness is a rich field focused on designing models resilient to adversarial attacks [7]. Methods like adversarial training [13] and certified defenses [15] have been widely studied, albeit on smaller datasets and without the added challenge of model compression.

Model compression techniques such as pruning [11], quantization [12], and weight-sharing [10] aim to reduce model size and inference latency. However, compressing models often exacerbates their vulnerability to adversarial attacks due to altered decision boundaries [9]. Studies like *Benchmarking Adversarial Robustness of Compressed Models* [19] demonstrate that pruning and quantization can reduce model robustness unless integrated with retraining strategies.

Our approach draws inspiration from [1] gives guarantees for models trained on ImageNet through random smoothing. [8] gives a unified optimization framework for robustness and compression, which we adopt in conjunction with [2] which gives robustness guarantees by maximizing linear region through polytope analysis – essentially a layerwise, more granular implementation of the MMR penalty. Our work extends empirically extends this line of research to the setting of compression integrating MMR into a compression-aware adversarial training pipeline, targeting both efficiency and robustness.

B VGG16 Structure

The model uses 13 convolutional layers, with 5 max pooling layers interspersed, and 3 linear layers.

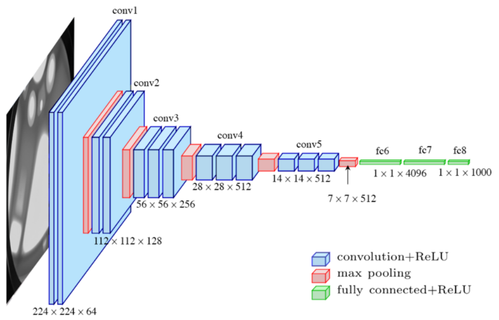


Figure 9: VGG16 Architecture

C Hyperparameters

Table 2: Parameters for the Robust Compression Algorithm

Parameter	**Value/Description**
`epochs`	1
`batch size`	128
`pruning_method`	`prune.L1Unstructured`
`prune_amount`	50%
`optimizer`	SGD (`lr`=0.005, `weight_decay`=0.005, `momentum`=0.9)
`epsilon`	0.03 (FGSM perturbation magnitude)
`margin_weight`	0.1 (Maximum margin regularization weight)
`attack`	FGSM (Fast Gradient Sign Method)
`weight_sharing`	8 bits (Convolutional layers), 5 bits (Fully connected layers)
`prune_targets`	`model.features` (Convolutions) & `model.classifier` (Lin

Received 3 December 2024