Evaluating CNN Robustness via Magnitude and Structured Pruning on MNIST

Yunzhen Liang, Bingbing Ma, Tingyu Zhou, Chengkai Hu

December 12, 2025

Abstract

Network pruning is widely used to compress convolutional neural networks (CNNs), but its impact on model robustness is not fully understood. In this project, we examine how three pruning methods – global magnitude pruning, structured channel pruning, and the lottery ticket hypothesis – affect the accuracy and robustness of a small CNN on the MNIST dataset. We train a baseline CNN to 99% test accuracy and then prune it to various sparsity levels, with and without fine-tuning, to observe changes in performance. We evaluate robustness under three challenging settings: (1) training on data with spurious correlations (a label-specific image patch), (2) training with noisy labels (40% label randomization), and (3) adversarial training against FGSM attacks. Our experiments show that the CNN can be pruned substantially (50–70% of weights removed) with minimal loss in standard accuracy. In many cases, pruned models maintain high resilience to noise and adversarial perturbations. Notably, pruning even improved the model's resistance to a spurious correlation: a heavily pruned model relied less on the spurious patch and scored 72% on clean test data vs. 66% for the unpruned model. Overall, our results suggest that aggressive pruning combined with appropriate retraining, does not inherently undermine a model's robustness to noise or attacks. It can sometimes enhance it while drastically reducing model size.

Code and datasets are available at: `https://github.com/chkhu/CNN-Pruning-Robustness-260D`

1 Introduction

Deep neural networks are known to be vulnerable to data corruptions such as label noise, spurious correlations, and adversarial perturbations. Although pruning techniques are widely used for model compression, their impact on robustness remains unclear. Therefore, our research problem is: how does neural network pruning influence a model's robustness under different types of distributions shift? To address this question, the paper examines whether pruning helps or harms CNN robustness.

This is challenging because pruning alters the model's inductive biases, possibly removing useful or harmful features. Prior work has not comprehensively compared pruning strategies under these realistic corruptions. Our approach applies several pruning strategies and evaluates performance under clean and corrupted data. We highlight the need for fine-tuning and report robustness tradeoffs. Limitations include the use of a simple CNN and dataset.

2 Related Work

2.1 Pruning for Compression and Efficiency

Early work approached pruning primarily for compression. Unstructured magnitude-based pruning removes weights with small absolute values, reducing memory footprint but creating irregular sparsity patterns that are difficult to accelerate [1, 2]. Conversely, structured pruning removes entire units (e.g., channels), producing hardware-friendly dense tensors but often causing larger accuracy drops due to the removal of coherent functional components [3, 4]. Iterative methods further allow models to adapt to reduced capacity gradually [5], establishing pruning as a viable deployment tool.

2.2 Pruning as a Lens on Network Structure

Recent research uses pruning to investigate information representation. The Lottery Ticket Hypothesis (LTH) demonstrates that sparse subnetworks within randomly initialized dense models can be trained in isolation to match the original model's accuracy [5, 6]. Other approaches explore dynamic sparse training, where connectivity evolves during optimization [7]. We include an LTH-style strategy to contrast this structural perspective with standard post-hoc pruning methods.

2.3 Pruning and Robustness

Pruned models are generally more vulnerable to adversarial attacks, as reduced redundancy limits decision paths [8, 9], though mild pruning can occasionally act as regularization [10]. Under distribution shifts, pruned networks often exhibit "compression-induced brittleness," performing well on clean data but deteriorating rapidly under corruption. Furthermore, reduced capacity may increase sensitivity to label noise and reliance on non-causal, spurious correlations [11], though these specific robustness dimensions remain under-explored in the pruning literature.

2.4 Positioning of Our Work

While prior studies typically focus on single pruning methods or narrow robustness settings, our work offers a unified comparison. We evaluate three distinct paradigms—magnitude, structured, and LTH pruning—on a controlled CNN architecture. Unlike previous work, we assess robustness across three dimensions: adversarial attacks, label noise, and spurious correlations. This design isolates the specific effects of different sparsification strategies on model performance under various forms of distribution shift.

3 Problem Formulation

Modern CNNs often contain many redundant parameters, and pruning these weights can yield a much smaller network with comparable standard accuracy. However, it is critical to ensure that pruning does not make the model more brittle when faced with data outside the original training distribution.We formulate the problem as an empirical study: does pruning a CNN compromise its robustness? Here, robustness means the model's ability to maintain high performance under various challenging conditions:

- Noisy data: The model should ignore irrelevant noise in the input or labels.
- Spurious correlations: The model should not overly rely on coincidental patterns in the training data that do not hold in general.
- Adversarial attacks: The model should resist intentional perturbations crafted to fool it (in our case, FGSM attacks).

Formally, let f_θ be the original CNN with parameters θ that achieves accuracy A_{clean} on clean test data. We generate pruned models $f_{\theta'}$ by applying different pruning techniques (described below) to remove a fraction of parameters from θ, obtaining $\theta' \subset \theta$. Our goal is to compare f_θ vs. $f_{\theta'}$ on clean accuracy on the original test set and robust accuracy on perturbed or modified test sets.

We consider three pruning strategies: unstructured weight pruning by magnitude, structured channel pruning, and the lottery ticket pruning method. Each method produces a sequence of pruned models with increasing sparsity (percentage of weights removed). We denote the sparsity level by s. For magnitude and channel pruning, we evaluate models immediately after pruning and after fine-tuning (retraining on the original training data to recover any lost accuracy). For lottery ticket pruning, which involves iterative training, we evaluate the winning subnetwork after each round of pruning.

By formulating the study in this way, we treat the robustness evaluation as a function of sparsity s and pruning method. A successful outcome would be that $f_{\theta'}$ remains nearly as robust as f_θ for moderate or high sparsity levels. If we observe large drops in robust accuracy for pruned models, that would indicate a trade-off between compression and reliability that practitioners need to consider.

4 Proposed Method

4.1 Network Architecture

We use a small CNN as our baseline model f_θ. The network has two convolutional layers (with ReLU and 2×2 max-pooling) followed by two fully connected layers, a typical architecture for MNIST. This model is trained on the MNIST training set (60,000 examples) and validated on the test set (10,000 examples) [12].

4.2 Pruning Methods

4.2.1 Global Magnitude Pruning

We remove individual weights with the smallest magnitudes (absolute values) across the entire network. Using PyTorch's pruning utilities, we perform global L1-unstructured pruning on all convolutional and fully connected layers. For example, pruning 50% means half of all weights (those closest to zero) are set to zero. We consider sparsity levels $s \in 0.1, 0.3, 0.5, 0.7$. After pruning, we optionally "remove" the masked weights to permanently eliminate them and then fine-tune the remaining weights for a few epochs on the training set to recover performance.

4.2.2 Structured Channel Pruning

Instead of pruning individual connections, we remove entire feature maps (channels) in the convolutional layers. Specifically, we use L_n-structured pruning (with $n = 1$) on the convolutional filters. This computes the L_1 norm of each filter's weight tensor and prunes a certain percentage of filters with the smallest norms. Pruning a conv filter effectively removes that feature map (and the corresponding feature channel in the next layer) from the network architecture. We again test $s = 0.1, 0.3, 0.5, 0.7$ fraction of filters pruned for each conv layer. Because removing whole channels can drastically disrupt the network, fine-tuning after pruning is especially important here. We therefore evaluate the structured pruned models both immediately after pruning and after retraining for 5 epochs on the original training data.

4.2.3 Lottery-Ticket Hypothesis

Lottery Ticket posits that dense networks contain sparse subnetworks, "winning tickets", that can be trained to full accuracy from their original initialization. We implement an iterative pruning variant: Initialization: Save the randomly initialized weights θ_0 Training: Train the full network to convergence $\rightarrow \theta_{\text{trained}}$ Pruning: Identify the lowest-magnitude weights to prune based on θ_{trained}, creating mask M_t Rewinding: Reset network weights to θ_0 Masked Retraining: Train from θ_0 with cumulative mask M_t applied Iteration: Repeat steps 3-5 for multiple rounds

We execute 5 pruning rounds with a per-round pruning fraction of 20%. The key distinction from standard iterative pruning is the rewind-to-initialization step, which tests whether the sparse architecture itself, rather than the specific weight values, determines performance.

4.3 Robustness Evaluation Framework

4.3.1 Robustness Evaluation Framework

To assess the robustness of the proposed pruning methods, we train both dense and pruned neural networks on datasets with different forms of distribution shift. We consider three settings that introduce adversarial perturbations, noisy labels, and spurious correlations during training.

4.3.2 Adversarial Perturbations

We used the MNIST FGSM adversarial dataset [13], which contains MNIST images perturbed using the Fast Gradient Sign Method (FGSM) at a fixed ϵ perturbation magnitude. This dataset served as our adversarial training data. We trained a robust baseline model on this dataset for 10 epochs, either solely on adversarial examples or mixed with clean images (depending on the specific experimental setting). This allowed us to evaluate the effects of pruning on a model that had been adversarially trained, without performing FGSM generation ourselves.

4.3.3 Noisy Labels

To study robustness under corrupted supervision, we apply symmetric label noise with rate $p_{\text{noise}} = 0.4$. For each training sample $(x_i, \tilde{y}_i)$, the corrupted label $\tilde{y}_i$ is:

$$\tilde{y}_i = \begin{cases} y_{\text{random}} \sim \text{Uniform}\{0, \ldots, 9\}, & \text{with probability } 0.4, \\ y_i, & \text{with probability } 0.6. \end{cases}$$

This corruption occurs independently across samples, and only labels are altered while the image inputs remain unchanged.

4.3.4 Spurious Correlations

To impose controlled shortcut features, we add class-dependent white patches onto training images. For each digit class $c \in \{0, \ldots, 9\}$, a fixed spatial region is assigned:

- Class 0 $\rightarrow$ top-left corner $(0{:}5, 0{:}5)$
- Class 1 $\rightarrow$ top-right corner $(0{:}5, 23{:}28)$
- Class 2 $\rightarrow$ bottom-left corner $(23{:}28, 0{:}5)$
- Class 3 $\rightarrow$ bottom-right corner $(23{:}28, 23{:}28)$
- Classes 4–9 $\rightarrow$ center $(12{:}17, 12{:}17)$

These artifacts create spurious cues that models may overfit during training.

5 Experiments

Below we present the results in four parts: baseline performance, robustness to label noise, robustness to spurious correlation, and robustness to adversarial attacks. All pruning experiments use the baseline CNN architecture described earlier. We apply each pruning method at various sparsity levels (10, 30, 50, 70) and select the most obvious results to show in the table.

Table 1 presents clean and robust accuracy metrics across various pruning methods on MNIST, evaluated under three distribution shifts: spurious correlations, noisy labels, and FGSM adversarial attacks.

Pruning Method	Fine-Tune	Clean	Spurious Corr.	Noisy Label	FGSM
Baseline (No Prune)	—	98.84%	66.36%	98.10%	97.23%
Magnitude (50%)	No FT	98.82%	67.07%	97.37%	99.53%
	With FT	99.02%	61.98%	97.98%	99.89%
Structured Channel (30%)	No FT	90.57%	61.71%	89.32%	97.19%
	With FT	99.03%	67.85%	98.14%	98.15%
Lottery Ticket (Best Round)	—	98.84%	67.17%	97.98%	98.18%

Table 1: Robustness Accuracy Evaluation under Different Pruning Methods

- **Magnitude Pruning (50%)** maintains clean accuracy (98.82%) even without fine-tuning, and slightly improves with fine-tuning (99.02%). However, spurious correlation accuracy drops to 61.98%, while adversarial robustness peaks at 99.89%, outperforming all methods.
- **Structured Channel Pruning (30%)** results in degraded performance when not fine-tuned (clean: 90.57%, noisy label: 89.32%). Fine-tuning recovers performance significantly, surpassing baseline in clean accuracy (99.03%) and spurious robustness (67.85%).
- **Lottery Ticket (Best Round)** matches the baseline in clean accuracy (98.84%) and improves spurious correlation accuracy (67.17%) and robustness under noisy labels and adversarial attacks, all without fine-tuning.
- **Baseline (No Pruning)** performs consistently across all conditions but is outperformed by magnitude pruning in adversarial robustness and by other methods in spurious settings.

6 Conclusion

This project investigated the relationship between model pruning and robustness in a CNN classifier.Using a small CNN on MNIST, we applied three pruning methods and evaluated the pruned models on clean data as well as under label noise, spurious correlation, and adversarial attack conditions. Our findings can be summarized as follows:

1. High Compression with Minimal Accuracy Loss: We were able to prune 30–50% of the network's weights with no significant drop in standard test accuracy, especially when fine-tuning was applied. This underscores the substantial redundancy in the CNN. Even at 70% sparsity, the accuracy drop was only around 1–2% after retraining.
2. Preservation of Robustness: Pruned models retained their robustness to noisy and adversarial data nearly as well as the full model. In the noisy labels experiment, all pruned models had $\sim$98% clean accuracy, indistinguishable from the unpruned model. For adversarial training, pruned models (up to 70% sparse) recovered to within 1–2% of the original FGSM accuracy after fine-tuning. This is a critical result: pruning did not make the model more vulnerable to attacks or noise. A possible intuition is that the remaining weights after pruning still capture the core predictive features (e.g. strokes of digits) which are the same features needed for robust classification.
3. Fine-tuning is Essential for Structured Pruning: One caution is that for structured pruning, we consistently needed to retrain to recover performance. If one pruned 30% of channels and did not fine-tune, the model would be severely degraded. Thus, any deployment of structured pruning should include a fine-tuning phase. Unstructured pruning was more forgiving, but for best results we also fine-tuned those models.

In conclusion, we discovered that a network can be made much smaller while preserving its ability to handle noisy data and adversarial perturbations. Pruning methods like magnitude and lottery ticket effectively remove redundant parameters that do not contribute to either standard or robust accuracy. With proper fine-tuning, the pruned CNNs in our study matched or even surpassed the original model's performance on all evaluated criteria. These findings support

the use of pruning as a practical technique to deploy efficient neural networks in resource-constrained settings without sacrificing reliability.

7 Future Work

Our experiments were on MNIST with relatively simple noise/attack scenarios. The trends we observed about the high tolerance of pruning and maintained robustness are encouraging. In future work, it would be interesting to test if these conclusions hold for more complex datasets (e.g. CIFAR-10/100, ImageNet) and stronger adversarial attacks (iterative attacks like PGD). Additionally, exploring when pruning might fail is important. Perhaps for very high sparsity beyond 90%, or combined with extremely distorted data, the model could break. We did see that at 70% structured prune, the model was right at the edge of a big drop without fine-tuning.

References

[1] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, vol. 2, 1990.

[2] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems*, 2015.

[3] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *International Conference on Learning Representations*, 2017.

[4] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1398–1406.

[5] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2019.

[6] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Stabilizing the lottery ticket hypothesis," *arXiv preprint arXiv:1903.01611*, 2019.

[7] D. C. Mocanu, E. Mocanu, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, 2018.

[8] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Towards compact and robust deep neural networks," *arXiv preprint arXiv:1906.06110*, 2019.

[9] S. Hooker, A. Courville, G. Clark, Y. Dauphin, and A. Frome, "What do compressed deep neural networks forget?" *arXiv preprint arXiv:1911.05248*, 2019.

[10] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," *Medical Image Analysis*, vol. 65, p. 101759, 2020.

[11] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

[12] OddRationale, "Mnist in csv," urlhttps://www.kaggle.com/datasets/oddrationale/mnist-in-csv, 2017, accessed: 2025-12-11.

[13] S. Kishore, "Mnist fgsm adversarial dataset," https://www.kaggle.com/datasets/sudulakishore/mnist-fgsm, 2021, accessed: 2025-12-11.