DeepJailbreak

Kieran Gallagher, Mathew Martin, Natasha Sebastian

[1]New York University Tandon School of Engineering, New York, USA

Abstract

This project explores the vulnerability of deep neural networks to adversarial attacks by targeting a pretrained ResNet-34 model on a subset of 100 ImageNet classes. After establishing baseline performance on clean data, we implemented the Fast Gradient Sign Method (FGSM) and its iterative variant (PGD) to generate pixel-wise adversarial examples within an L_∞ constraint (ϵ = 0.02). We also introduced a patch-based attack that perturbs only a 32×32 region of the image. All attacks led to significant drops in top-1 and top-5 accuracy. Finally, we evaluated the transferability of these adversarial examples to other models like DenseNet-121 and SqueezeNet, demonstrating the broader impact of such vulnerabilities.

Public Repository

The python implementation of DeepJailbreak can be found here:
github.com/NYUNeuroNinjas/DeepJailbreak

Introduction

Convolutional Neural Networks (CNNs) have become commonplace in a wide range of applications where security is essential, including self-driving cars and threat detection. In order to ensure that our models are reliable and secure, we place an emphasis on designing models that are resistant to adversarial inputs. By constructing and analyzing these attacks ourselves, we learn what features make a model resistant to them, and what can be done to further improve our models.

Data

The dataset used for this project consists of 500 test images spanning 100 distinct ImageNet-1K classes, specifically from class indices 401 to 500. Each image is labeled with its corresponding ImageNet index, and the labels were provided via a JSON file. Only the test set was used in this study, with no further training or fine-tuning of the model. Standard ImageNet normalization was applied, and data augmentations were not used, as the goal was to evaluate model robustness under adversarial perturbations rather than improve generalization on clean data.

Methodology

In this study, we probe the model robustness with various adversarial attacks. We explore pixel-wise attacks that tweak individual pixel intensities while keeping the image visually unchanged, like the Fast Gradient Sign Method (FGSM). Improving upon it we introduce an iterative gradient attack, projected gradient descent (PGD), which takes multiple small FGSM-like steps. Finally, we ran a PGD-based adversarial patch attack, optimizing a small localized region of the image.

Final Training Setup:

- **FGSM Attack:** Epsilon = 0.02
- **PGD Attack:** Epsilon = 0.02, Alpha = 0.005, Number of steps = 10
- **PGD Patch Attack:** Epsilon = 0.5, Alpha = 0.05, Number of steps = 70

Fast Gradient Sign Method (FGSM)

A common and simple algorithm for mounting an L_∞ attack is called the Fast Gradient Sign Method (FGSM) [2] [4]. This implements a *single step* of gradient ascent (in pixel space) and truncates the values of the gradients to at most ε. Mathematically, we can write this as:

$$x \leftarrow x + \varepsilon \cdot \text{sign}(\nabla_x L)$$

where L is the cross-entropy loss, and the gradient is with respect to the input parameters (not the weights). The $\text{sign}(\cdot)$ operation truncates the gradient direction to the unit L_∞ cube.

The parameter ε is called the attack budget. If raw (unpreprocessed) images have pixel values ranging from 0 to 255, an attack budget of $\varepsilon = 0.02$ roughly corresponds to changing each pixel value in the raw image by at most ± 1.

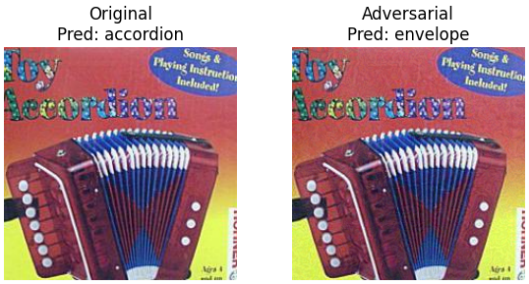

Figure 1: Effect of the FGSM attack on a ResNet-34 prediction

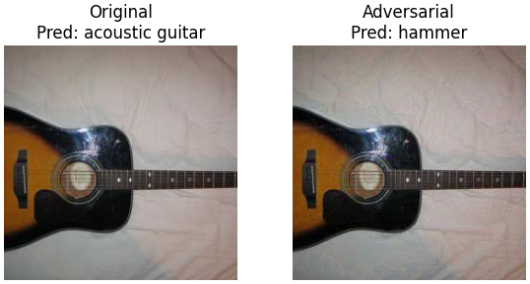

Figure 2: Effect of the PGD attack on a ResNet-34 prediction

Projected Gradient Descent (PGD)

The Projected Gradient Descent (PGD) [3] attack is a widely adopted and principled method for generating adversarial examples under norm-bounded constraints. It is considered one of the strongest first-order adversarial attacks and is often used as the de facto standard for evaluating the robustness of deep neural networks. PGD can be viewed as a multi-step extension of the Fast Gradient Sign Method (FGSM), iteratively refining adversarial perturbations within a defined perturbation budget.

Given a clean input $x \in \mathbb{R}^d$, its corresponding label y, a classifier f with parameters θ, and a loss function $\mathcal{L}(f(x), y)$ (typically cross-entropy), the goal is to find an adversarial input x^{adv} that maximizes the model's loss while remaining within an L_∞-norm ball of radius ϵ around the original input:

$$\max_{\|x^{\text{adv}} - x\|_\infty \leq \epsilon} \mathcal{L}(f(x^{\text{adv}}), y)$$

The PGD algorithm starts from an initial point x_0^{adv} (commonly set to the original input or randomly perturbed within the ϵ-ball), and updates the adversarial example iteratively using the following rule:

$$x_{t+1}^{\text{adv}} = \Pi_{B_\infty(x,\epsilon)} \left(x_t^{\text{adv}} + \alpha \cdot \text{sign} \left(\nabla_x \mathcal{L}(f(x_t^{\text{adv}}), y) \right) \right)$$

where:

- α is the step size,
- $\nabla_x \mathcal{L}$ denotes the gradient of the loss with respect to the input,
- $\text{sign}(\cdot)$ applies the element-wise sign operation to the gradient,
- $\Pi_{B_\infty(x,\epsilon)}(\cdot)$ projects the updated input back into the L_∞-ball centered at x with radius ϵ.

In each iteration, the adversarial image is perturbed in the direction that maximally increases the loss, followed by projection to ensure the perturbation remains within the allowed bounds. This iterative procedure continues for a fixed number of steps T, after which the final adversarial example x_T^{adv} is used for evaluation.

In this project we implement this through successive iterations in which we learn the direction of gradient ascent and multiply it by a chosen step size, keeping our perturbations within the perturbation budget ($\epsilon = 0.02$).

$\varepsilon = 0.02, \#steps = 10$		
α	**Top-1 Acc.**	**Top-5 Acc.**
0.0025	0.0%	1.8%
0.005	0.0%	1.4%
0.01	0.0%	1.8%

Table 1: Top-1 and Top-5 accuracies for varying values of α in PGD attack

PGD Patch Attack

In addition to standard pixel-wise perturbations, we implemented a patch-based PGD attack [1], where the adversarial modification is restricted to a small, localized region of the image. This approach simulates a more physically plausible attack scenario, such as applying a sticker or occlusion to a specific area, and is particularly relevant in real-world adversarial settings.

That is, the perturbation is constrained both in magnitude (via ϵ) and spatial location (restricted to patch B). The optimization is performed iteratively using Projected Gradient Descent (PGD). At each step t, the adversarial example is updated as:

$$x_{t+1}^{\text{adv}} = \Pi_{B,\epsilon} \left(x_t^{\text{adv}} + \alpha \cdot \mathbb{I}_B \cdot \text{sign}(\nabla_x \mathcal{L}(f(x_t^{\text{adv}}), y)) \right)$$

where:

- α is the step size,
- $\mathbb{I}_B$ is a binary mask that restricts updates to the patch region B,
- $\Pi_{B,\epsilon}(\cdot)$ projects the updated input back to the ϵ-ball centered at x within the patch,
- $\mathcal{L}$ is the loss function (cross-entropy),
- f is the pretrained model.

Instead of perturbing the entire image, we can instead implement a patch attack that focuses on only a random 32x32 pixel area. We do this with a randomly generated mask that is applied to the same PGD attack as before:

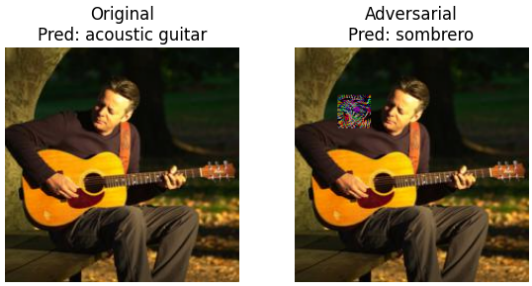

Figure 3: Effect of the PGD Patching attack on a ResNet-34 prediction

To counteract the limited effectiveness of only attacking a small area, we apply the gradient descent with a higher epsilon value, a larger step size, and many more total steps. We evaluate the effect of these hyperparameters on the top-1 and top-5 accuracy of our ResNet-34 model by averaging the results of three trial runs for each hyperparameter value:

$\varepsilon = 0.5, \alpha = 0.05$		
PGD Steps	**Top-1 Acc.**	**Top-5 Acc.**
60	6.2%	45.7%
70	4.8%	42.3%
80	5.3%	42.5%

Table 2: Top-1 and Top-5 accuracies for varying number of PGD steps in patching attack

$\varepsilon = 0.5, \#steps = 70$		
α	**Top-1 Acc.**	**Top-5 Acc.**
0.025	7.1%	46.1%
0.05	4.8%	42.3%
0.1	7.2%	46.8%

Table 3: Top-1 and Top-5 accuracies for varying values of α in patching attack

While in other tests we were able to obtain lower accuracy numbers as the number of PGD steps continued to increase, we do not believe it is worth the increased runtime, and so we settle on hyperparameter values $\varepsilon = 0.5, \alpha = 0.05, \#steps = 70$.

Results and Analysis

The capabilities of FGSM, PGD, and PGD Patch attacks were evaluated on the ImageNet dataset using three CNN architectures—ResNet-34, the model our attacks were trained on; DenseNet-121, a larger and more connected model; and SqueezeNet, a much smaller and more efficient model. These models test the extent of the capabilities of our perturbations:

	ResNet-34	**DenseNet-121**	**SqueezeNet**
Original	76.0%	74.8%	54.2%
FGSM	10.2%	49.2%	36.4%
PGD	0.0%	38.8%	39%
PGD Patch	3.4%	68.4%	47.2%

Table 4: Top-1 accuracies for various models against each attack

	ResNet-34	**DenseNet-121**	**SqueezeNet**
Original	94.2%	93.6%	81.0%
FGSM	33.4%	78.0%	62.6%
PGD	1.4%	76.6%	64.8%
PGD Patch	28.8%	90.0%	71.2%

Table 5: Top-5 accuracies for various models against each attack

Our attacks, trained on the ResNet-34 model and its gradients, are much less effective when evaluated on other pre-trained networks. This demonstrates the need for confidentiality of a model's weights and parameters, and encourages the use of multiple models for effective classification.

Finally, we evaluate our attacks by analyzing the training time and max L∞ of our perturbations:

	Training Time (min:sec)	**L∞**	**Top-1 Acc.**
FGSM	0:06	0.02	33.4%
PGD	0:23	0.02	1.4%
PGD Patch	2:45	0.5	28.8%

PGD is the most effective and efficient attack, greatly outperforming FGSM with the same L∞ distance, while only increasing training time by around 400%. The PGD Patch attack, meanwhile, needs to be massively slower in order to reach similar accuracy results, and relies on perturbing the affected pixels by a much greater amount. Patch attacks, however, may find use in situations where an attacker does not have access to the full image. Clearly, there are benefits and drawbacks to each attack method, and by using the information we gain from attacking CNNs, we can learn much about how to defend them.

References

[1] Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2018. Adversarial Patch. arXiv:1712.09665.

[2] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

[3] Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083.

[4] Yuan, X.; He, P.; Zhu, Q.; and Li, X. 2018. Adversarial Examples: Attacks and Defenses for Deep Learning. arXiv:1712.07107.