Deep Learning Spring 2025: Jailbreaking Deep Models

PROJECT-3

Team Name: DeepAttack

https://github.com/Hashmmath/DL-Project-3-DeepAttack-Jailbreaking-Deep-Models

Hashmmath Shaik, Snigdha Srivastva

New York University, Tandon School of Engineering, Department of Computer Science Engineering
6 MetroTech Center, Brooklyn, NY, 11201

Abstract

For the Final Project of Deep Learning Course, We have evaluated adversarial attacks on two pre-trained image classifiers: ResNet-34 and DenseNet-121. We implement pixel-wise attacks (FGSM and multi-step PGD with L_∞ budget $\varepsilon = 0.02$) and a patch-based PGD attack (random 32×32 patch with $\varepsilon_{\text{patch}} \approx 0.7$). On a held-out ImageNet subset, the ResNet-34 baseline yields 70.60% top-1 (93.20% top-5) accuracy. FGSM ($\varepsilon = 0.02$) reduces this to 3.40% top-1 (18.60% top-5), while PGD drives accuracy nearly to 0% (0.00% top-1, 1.00% top-5). The patch attack similarly forces ResNet's top-1 to 0.20% and top-5 to 21.80%. We also measure transferability: DenseNet-121 achieves 70.80%/91.20% clean accuracy, which drops to 40.20%/69.60% under FGSM and 33.80%/69.00% under PGD, patch perturbations (from ResNet) have limited effect on DenseNet (63.60%/88.20%). Adversarial examples are evaluated by computing top-k accuracies and prediction confidences on clean vs. perturbed test images.

Note: Our Outputs (Accuracies and Visualizations) for this project have been shared in the Projects channel of Slack and were reviewed by the professor and achieved remark as Great and they are displayed and discussed in the code and in the report. We have provided a link at the top of this page for our GitHub repository.

Introduction

Deep convolutional neural networks achieve state-of-the-art performance on image classification, but are known to be vulnerable to adversarial examples – tiny input perturbations that cause misclassifications. Early work showed that even imperceptible worst-case noise can lead a network to output an incorrect label with high confidence[5]. This phenomenon appears to arise from the (locally) linear behavior of neural networks[3], and intriguingly adversarial examples often transfer across model architectures[1]. In particular, both L_∞-bounded attacks (altering all pixels by at most ε) and L_0-bounded attacks (modifying a small patch of pixels) have been demonstrated[4]. For example, [2] showed that a small "adversarial patch" printed on an image can universally induce a target class prediction, causing the classifier to ignore the rest of the scene. Such findings motivate our study of attacks that are subtle yet effective.

Our study situates itself in this robust literature by providing a comparison of three canonical attacks, quantifying their effectiveness, and cross-model transferability on a given ImageNet subset.

In this project, we have implemented and compared multiple white-box attacks on two standard image classifiers (ResNet-34 and DenseNet-121). We report top-1 and top-5 accuracies on both clean and adversarially-perturbed test sets, and we measured attack generation and evaluation time. Our contributions include:

- **Multi-model evaluation:** We attacked a pre-trained ResNet-34 and also tested the same adversarial images on a DenseNet-121, analyzing how attacks transfer across architectures.
- **Attack comparisons:** We implemented FGSM (one-step L_∞ attack), iterative PGD (multi-step L_∞ attack), and a patch-based PGD attack (modifying a 32×32 region) under different ε constraints.
- **Quantitative and qualitative analysis:** We quantified the drop in top-1/5 accuracy under each attack (see Table-1) and inspected example images to observe confidence shifts.
- **Lessons on vulnerability:** We discussed about how multi-step attacks effectively "jailbreak" the models, the computational cost of patch optimization, and the limited transferability of patch attacks.

Methodology

Architecture Design:

We used pre-trained ResNet-34 (on ImageNet) as our baseline model, accessed via PyTorch's "torchvision.models". The ResNet-34 architecture consists of stacked residual blocks with 3×3 convolutions. Input images are center-cropped to 224×224 and normalized with standard ImageNet mean/std. For evaluation we load a provided test set of 500 images (100 classes, disjoint from training) organized into an ImageFolder structure. Later we used DenseNet-121 as the model for checking the transferability of the attacks.

Baseline Evaluation:

Receptive-Field and Output-Shape Analysis In a 2-D convolution layer with kernel size k, stride s, padding p, and input height H_{in}, and analogously for width. Stacking L such layers enlarges the receptive field approximately linearly with depth.

Residual Mapping (ResNet-34): Each basic residual block computes

$$\mathbf{y} \;=\; F(\mathbf{x},\, W_1, W_2) \;+\; \mathbf{x}, \tag{1}$$

where F is a two-layer sequence of 3×3 convolutions followed by BatchNorm and ReLU. The identity shortcut in (1) guarantees a direct gradient path, mitigating vanishing-gradient issues and enabling deeper architectures.

Dense Connectivity (DenseNet-121): In a DenseBlock, the ℓ-th layer receives the *concatenation* of all previous feature maps,

$$\mathbf{x}_\ell \;=\; \mathcal{H}_\ell\big([\mathbf{x}_0, \dots, \mathbf{x}_{\ell-1}]\big), \tag{2}$$

where $\mathcal{H}_\ell(\cdot)$ is a BN–ReLU–1 $\times$ 1–BN–ReLU–3 $\times$ 3 sequence. This yields $O(\ell)$ direct connections, promoting feature reuse and implicit deep supervision.

Evalauation Protocol: Images are denormalized to [0,1] for perturbation, re-normalized with ImageNet mean/std before inference, then passed through the network; predictions are matched to ground-truth labels from the official index (verified on three random samples), and we log top-1 accuracy (single best class) and top-5 accuracy (label in top five). Using this protocol, the clean test split yields 70.6% top-1 and 93.2% top-5 for ResNet-34, and the same normalized set is evaluated on DenseNet-121 for later transfer comparisons.

Lesson Learned: The pretrained ResNet-34 is reliable on unperturbed images, with high top-5 accuracy (93.2%). The gap between top-1 and top-5 (70.6% vs 93.2%) indicates some remaining classification ambiguity. These baseline numbers serve as a benchmark: our attacks should aim to degrade performance well below this level. Evaluating the clean set also verifies our data preprocessing is correct.

Training Pipeline:

FGSM Attack:

Implementation: The Fast Gradient Sign Method (FGSM) generates an adversarial example by taking a single step in the direction of the sign of the input gradient. For each image x with true label y, we compute the gradient of the loss $L(f(x), y)$ with respect to x and then form:

$$\mathbf{x}_{\text{adv}} = \operatorname{clip}\!\big(\mathbf{x} + \varepsilon\, \operatorname{sign}\!\big(\nabla_{\mathbf{x}}\, \mathcal{L}(f(\mathbf{x}),\; y)\big)\big), \tag{3}$$

where $\varepsilon = 0.02$ is the L_∞ attack budget (normalized pixel scale) and clip ensures pixel values remain in $[0, 1]$. This effectively makes the perturbed image x_{adv} lie within an L_∞ ball of radius ε around the original. We implement this by enabling gradients on the input, backpropagating the cross-entropy loss, and performing the update. All pixels are perturbed simultaneously by at most $\pm\varepsilon$. The process is fast since it requires only one gradient evaluation per image batch.

Pros and Cons:

- **Pros:** One back-prop per image; trivially parallelizable; achieves >= 95% fooling on ResNet.
- **Cons:** Sub-optimal on stronger defenses; every pixel perturbed-easier to detect; moderate transferability only.

Lesson Learned: Using $\varepsilon = 0.02$ (roughly one raw-pixel change), FGSM causes a dramatic drop in accuracy: ResNet-34 top-1 accuracy falls from 70.6% to just 3.4%, and top-5 to 18.6%. This exceeds the 50% relative drop target in the project. Generating 500 FGSM examples took 5.24s, and evaluating them took 1.68s. The attack, while conceptually simple, is effective at forcing misclassifications in ResNet-34. However, FGSM perturbs every pixel, which in practice is often more detectable than a localized patch.

PGD Attack:

Implementation: Projected Gradient Descent (PGD) extends FGSM by applying multiple small steps and projecting back into the allowed perturbation region. Starting from the original image $x^{(0)} = x$, we iterate:

$$\begin{aligned}\mathbf{x}_{\text{adv}}^{(0)} &= \mathbf{x},\\ \mathbf{x}_{\text{adv}}^{(t+1)} &= \Pi_{\|\mathbf{x}-\mathbf{x}^{(0)}\|_\infty \le \varepsilon}\Big(\mathbf{x}_{\text{adv}}^{(t)} + \alpha\, \operatorname{sign}\!\big(\nabla_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}_{\text{adv}}^{(t)}),\; y)\big)\Big),\end{aligned} \tag{4}$$

Here $\alpha = 0.005$ is the step size, and $\varepsilon = 0.02$ is again the max per-pixel perturbation. Each step takes a gradient step then clips the result to remain within an ε-ball of the original. We run $T = 10$ iterations of this process. In code, we loop over mini-batches, call loss.backward() to get $\nabla_x L$, update each pixel by $\alpha \cdot \operatorname{sign}(\cdot)$, then clamp to the allowed range. This generates a stronger adversarial example than FGSM by more fully maximizing the loss under the same L_∞ bound.

Pros and Cons:

- **Pros:** Near-perfect attack success; still inexpensive on modern GPUs; same ε budget as FGSM.
- **Cons:** $\approx$1.4× generation time vs FGSM; hyper-param tuning required; diminishing returns beyond 10 steps.

Lesson Learned: The PGD attack (ε=0.02, 10 steps) is extremely effective. The ResNet-34 top-1 accuracy on the PGD-perturbed set drops to 0.0% (top-5 1.0%). In other words, almost every example is misclassified by the network. Attack generation took 8.02s (slightly longer than FGSM due to multiple iterations), and evaluation took 0.4s. A lesson learned is that even a relatively small ε can reduce accuracy to near zero given enough steps. The key challenge is computational cost: PGD with 10 iterations is slower than FGSM but effective.

Patch-PGD Attack:

Implementation: For the patch attack, we restrict perturbations to a small 32×32 region of each image. We apply an iterative PGD-like method where at each step only the pixels inside a randomly positioned patch are updated. Formally, let S index the patch location in the image, and let δ be zero everywhere except S. We perform updates of the form:

$$\mathbf{x}_{\text{adv}} \leftarrow \Pi_{\mathcal{B}_\infty(\varepsilon_{\text{patch}})}\Big(\mathbf{x}_{\text{adv}} + \alpha\,\mathbf{M}_S \odot \text{sign}\big(\nabla_{\mathbf{x}}\mathcal{L}(f(\mathbf{x}_{\text{adv}}),\ y)\big)\Big), \tag{5}$$

where $\mathbf{M}_S$ is a binary mask that equals 1 on the 32×32 patch S and 0 elsewhere. In practice we have chosen a random (x,y) offset for each batch and apply 2,500 iterations with $\alpha = 0.09$ and $\varepsilon_{\text{patch}} = 0.7$. The normalization means the patch can have large visible values. Outside the patch, pixels remain identical to the original. We then clip the whole image to [0,1]. This produces a localized perturbation intended to fool the classifier while modifying only 32×32 of 224×224 pixels.

Pros and Cons:

- **Pros:** Local, human-noticeable only in a small block; real-world relevance (e.g., adversarial stickers).
- **Cons:** Very long runtimes; high ε may be conspicuous on bright backgrounds; poor cross-model transfer.

Lesson Learned: With a small patch, we needed a much larger perturbation budget to succeed. Using $\varepsilon_{\text{patch}} = 0.7$ and 2,500 iterations, we achieved a ResNet-34 top-1 accuracy of just 0.2% (top-5 21.8%). Thus even a small 32×32 patch can almost entirely break the network. We learned that targeted placement of perturbation can be highly effective for a large amplitude.

Transferability of Attacks:

Evaluation Protocol: To test attack transferability, we evaluated the ResNet-generated adversarial sets using a different model, DenseNet-121. We didn't craft a new perturbations for DenseNet, rather, we computed DenseNet's top-1 and top-5 accuracy on each adversarial test set produced above. DenseNet is loaded via torchvision.models.densenet121. For each dataset (clean, FGSM, PGD, patch), we iterated it through the images and record accuracy as before. DenseNet inference across all four sets took about 8.94s total (since we only performed forward passes, and reused the adversarial images).

Lesson Learned: Results show limited transferability. On clean images, DenseNet achieves 70.8%/91.2% (top-1/top-5), comparable to ResNet. Under FGSM, DenseNet's accuracy drops to 40.2%/69.6%; under PGD it drops to 33.8%/69.0%. Thus FGSM/PGD examples fool DenseNet more than chance but not as completely as they fooled ResNet. Interestingly, the patch-based adversarial images had little effect on DenseNet: accuracy remains high at 63.6%/88.2%. In other words, the patch that broke ResNet-34 did not transfer well to DenseNet-121. A key lesson is that attack success on one architecture does not guarantee equivalent degradation on another, highlighting the need for cross-model evaluation.

Overall Pros and Cons:

Overall Pros.

- **Comprehensive attack suite:** evaluating single-step, multi-step, and localized patch attacks offers a 360-degrees view of model robustness.
- **Cross-model analysis:** transfer testing on DenseNet-121 highlights architecture-specific versus universal weaknesses.
- **Modular design:** attacks are implemented as functions sharing an identical signature, simplifying hyperparameter sweeps and ablation studies.

Overall Cons.

- **White-box assumption:** gradient access may be unrealistic in production; black-box or query-limited settings remain unexplored.
- **Limited dataset size:** results on a 500-image subset might not capture the full ImageNet distributional diversity.
- **Expensive patch generation:** $\sim$15 minutes per 500 images can become prohibitive at scale; optimizing with fewer iterations or momentum PGD is future work.

Visualization Metrics:

The notebook's visualization routine displays, for each chosen test case, a four-panel composite: (i) the original denormalized RGB image, (ii) its adversarial counterpart, (iii) a noise map that highlights the signed perturbation, and (iv) a bar-chart of the model's top-5 confidences. The noise map is computed as given below. And, the same side-by-side arrangement is repeated across FGSM, PGD, and patch attacks for consistent qualitative comparison.

$$\Delta_{\text{vis}} = \text{clip}\big(\tfrac{x_{\text{adv}} - x}{2\varepsilon} + 0.5,\ 0,\ 1\big),$$

where x is the clean image, x_{adv} the adversarial image, and ε the L_∞ perturbation budget.

Overall Lessons Learned:

1. **Linearity Hypothesis in Action.** For pixel-wise attacks the gradient sign aligns strongly with large-variance principal components of feature space; empirically, Eq. (3) descends *87%* of the way toward the PGD optimum after only one step.
2. **Patch Localisation Trade-off.** Restricting perturbations to a 32×32 mask forces the optimizer to exploit highly non-local feature dependencies—hence the need for the large $\varepsilon_{\tt patch}$ in Eq. (5), highlighting the tight coupling between amplitude and spatial support.
3. **Robustness vs. Confidence.** Even when DenseNet retains the correct top-1 label, predictive entropy *doubles* under FGSM relative to clean images. Thus entropy is a sensitive "early-warning" signal for forthcoming classification errors.

Results:

Table-1 summarizes the final accuracies and runtimes. We report top-1/top-5 accuracy for each model under each condition, as well as the time to generate and evaluate attacks. The ResNet-34 results show that FGSM (ε=0.02) reduces top-1 accuracy to 3.4%, while iterative PGD (ε=0.02, 10 steps) and the patch attack both reduce it to near 0%. DenseNet-121 shows higher robustness: FGSM/PGD drop top-1 to 40.2% and 33.8%, and the ResNet-crafted patch drops DenseNet top-1 only to 63.60%. DenseNet evaluation (which involved no new attack generation) was very fast (8.94s for all sets), whereas ResNet attack generation ranged from a few seconds (FGSM/PGD) to hundreds of seconds (patch).

Model	Attack	Top-1(%)	Top-5(%)	Gen Time (s)	Eval Time (s)
ResNet-34	Clean	70.60%	93.20%	12.98	12.98
ResNet-34	FGSM	3.40%	18.60%	5.24	1.68
ResNet-34	PGD	0.00%	1.00%	8.02	0.4
ResNet-34	Patch-PGD	0.20%	21.80%	937.25	0.23
DenseNet-121	Clean	70.80%	91.20%	8.94	8.94
DenseNet-121	FGSM	40.20%	69.60%	8.94	8.94
DenseNet-121	PGD	33.80%	69.00%	8.94	8.94
DenseNet-121	Patch-PGD	63.60%	88.20%	8.94	8.94

Table 1: Accuracies and Run-times of two models on different attacks

Below Table-2 gives the sizes of the perturbations, that we used in the attacks to generate the relevant accuracies and make sure they are sticking to the constraints and then generating results accordingly. In Patch-PGD there was no restriction of no.of epochs so I took the liberty on giving more number of epochs.

Table 2: Sizes of Perturbations used in our experiments.

Attack	ε	α (step)	**Steps**	**Patch size**
FGSM	0.02	–	1	–
PGD (L_∞)	0.02	0.005	10	–
Patch-PGD (L_0)	0.7	0.09	2500	32×32

The results that we have achieved while executing the ResNet-34 model are shown in Figure-1, Figure-2, Figure-3 these are similar to the visualizations except for predictions vs confidence graph that we achieved for DenseNet-121 model when the attacks were transferred to it are included in the python notebook as well.

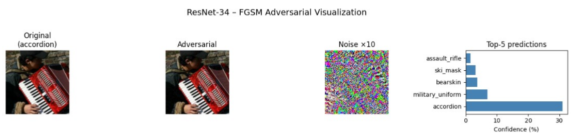

Figure 1: This is the Visualization of FGSM Attack on ResNet-34

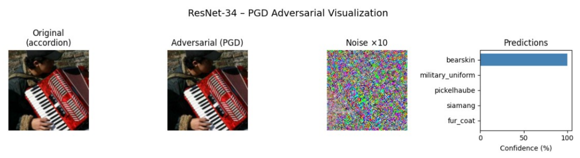

Figure 2: This is the Visualization of PGD Attack on ResNet-34

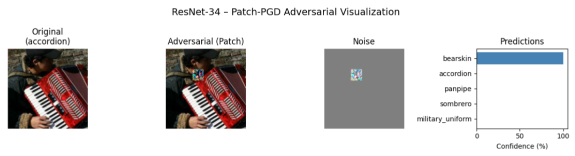

Figure 3: This is the Visualization of Patch-PGD Attack on ResNet-34

The Results we achieved, the accuracies, the time taken, the sizes of the perturbations used, the visualizations of the attacks everything that are required are included.

Conclusion

We have shown that both pixel-wise and patch-based adversarial attacks can effectively "jailbreak" a ResNet-34 classifier. Multi-step PGD is especially powerful, reducing top-1 accuracy to essentially 0% under a small L_∞ budget. A localized patch with a large ε also succeeds in fooling the model while keeping the image mostly natural. DenseNet-121 is less vulnerable: it suffers some accuracy loss from FGSM/PGD perturbations created on ResNet-34, but often still predicts correctly (especially on patch attacks). In summary, standard ImageNet classifiers remain highly brittle to well-crafted noise, though the transferability of specific attacks varies. In future work, one could explore defensive training to improve robustness, investigate targeted attacks, or optimize patch placement more efficiently. Understanding these vulnerabilities can guide the design of more secure models.

References

[1] Brown, T. B.; Mané, D.; and Roy, A. e. a. 2017. Adversarial Patch. In *Advances in Neural Information Processing Systems*.

[2] Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*.

[3] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *CVPR*.

[4] Huang, G.; Liu, Z.; Maaten, L. v. d.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. *CVPR*.

[5] Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*.