CS GY 6953 : Deep Learning Project

Written by :
Mukesh Durga
Tanaya Pawar
Tejaswini Ojha

GitHub Repository: `https://github.com/tanaya09/CS-GY-6953-Deep-Learning-Project--3`

Abstract

Deep convolutional networks achieve state-of-the-art performance on large-scale image classification, yet remain surprisingly brittle when faced with adversarial perturbations. In this work, we systematically evaluate the robustness of a pretrained ResNet-34 classifier on a curated subset of 100 ImageNet classes under three adversarial attack paradigms: (1) single-step Fast Gradient Sign Method (FGSM), (2) multi-step Projected Gradient Descent (PGD), and (3) localized patch attacks confined to a 32×32 pixel region. We begin by establishing baseline Top-1 and Top-5 accuracies on the clean test set. For each attack, we generate a corresponding adversarial test set within a strict ℓ_∞ budget ($\epsilon = 0.02$ for pixel-wise attacks, and ϵ up to 0.5 within the patch), verify perturbation bounds, and visualize representative examples. We then measure the resulting performance degradation in both Top-1 and Top-5 accuracy, demonstrating that PGD yields the largest drop, followed by FGSM and patch attacks. To assess transferability, we further evaluate all four datasets on a DenseNet-121 model, revealing that adversarial images crafted for ResNet-34 remain highly effective across architectures. Our findings highlight critical vulnerabilities in deep vision models and emphasize the necessity of incorporating adversarial training and robustness evaluation as standard practice for safe AI deployment.

Project Overview

This project examines the adversarial robustness of a pretrained ResNet-34 model on a curated subset of 100 ImageNet-1K classes by systematically mounting three distinct attack strategies under strict perturbation budgets and measuring their impact on classification accuracy. We begin by establishing baseline Top-1 and Top-5 performance on clean test images. Next, we generate pixel-wise adversarial examples using the Fast Gradient Sign Method (FGSM) with an ℓ_∞ constraint of $\epsilon = 0.02$, visualize representative attack successes and failures, and quantify the resulting accuracy drop. To further degrade performance, we implement a multi-step Projected Gradient Descent (PGD) attack under the same budget, verify perturbation bounds, and assess the additional loss in Top-1/Top-5 metrics. We then explore localized patch attacks confined to a 32×32 pixel region with an increased budget (up to $\epsilon = 0.5$), demonstrating how spatially constrained perturbations can still significantly impair model predictions. Finally, to evaluate transferability, we apply all four datasets—clean, FGSM, PGD, and patch—to a DenseNet-121 classifier and compare the cross-model degradation in accuracy. Through these experiments, we identify the most effective attack methods, illustrate key failure modes via visualization, and highlight the necessity of integrating adversarial defenses into real-world vision systems.‘‘‘

Methodology

Our methodology leverages PyTorch's autograd to craft adversarial examples under strict ℓ_∞ budgets: we compute input gradients and apply FGSM for a single-step attack, PGD for iterative refinement, and localized perturbations confined to a 32×32 patch . After each attack we automatically verify that no pixel exceeds its budget and then measure the resulting Top-1 and Top-5 accuracy drops on both ResNet-34 and DenseNet-121.

Dataset and Preprocessing

We evaluate our attacks on a curated subset of 500 images drawn from 100 distinct ImageNet-1K classes (five examples per class), organized via a JSON mapping of folder indices to WordNet labels. Each image is loaded with PyTorch's `ImageFolder`, resized and center-cropped to 224×224 pixels, and converted to a tensor. We then normalize using the standard ImageNet statistics (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]) to match the pretrained ResNet-34 and DenseNet-121 expectations. For batch processing, we employ a `DataLoader` (with shuffling disabled to preserve class order) and a batch size of 32. Finally, we implement an inverse-normalization step for visualization and exact ℓ_∞ distance calculations, ensuring images are displayed in their original color distribution and that perturbation budgets can be programmatically verified.

Model Architecture

Our evaluation framework is built upon two canonical ImageNet-1K pretrained convolutional classifiers: ResNet-34 and DenseNet-121. Both networks remain fully *frozen* during all experiments—no model weights are updated—so

that all adversarial perturbations are applied solely to the input images.

ResNet-34. ResNet-34 begins with a 7×7 convolutional layer (64 filters, stride 2) followed by a 3×3 max–pool. It then passes through four stages of residual blocks containing $\{3,4,6,3\}$ pairs of 3×3 convolutions. Each convolution is followed by BatchNorm and ReLU, and an identity skip connection adds the block's input to its output to preserve gradient flow in deep networks. After the final stage, global average pooling reduces feature maps to a 512-dimensional vector, which is linearly mapped to 1000 logits.

DenseNet-121. DenseNet-121 is composed of four dense blocks with $\{6,12,24,16\}$ layers each. Within each block, every layer executes a 1×1 bottleneck convolution (growth rate=32) followed by a 3×3 convolution. Crucially, each layer concatenates its output with all preceding feature maps, promoting feature reuse and implicit deep supervision. Transition layers between blocks apply a 1×1 convolution and 2×2 average pooling to downsample spatial dimensions and compress channel count. A final global average pooling produces a 1024-dimensional vector, which is fed into a linear layer producing 1000 logits.

Logit Remapping and Inference. Both models accept 224×224 RGB inputs and output 1000-dimensional logits. We remap these logits to our 100-class subset via an index lookup derived from the provided JSON label file. For clean evaluation, each model is placed in `model.eval()` mode and wrapped in `torch.no_grad()` to disable gradients and maximize inference throughput.

Adversarial Attack Interface. To generate adversarial examples, we temporarily enable gradient tracking on the input tensor only (`input.requires_grad=True`) and compute the cross–entropy loss via `torch.nn.CrossEntropyLoss`. We then backpropagate through the frozen network to obtain input gradients. For FGSM, we apply a single-step perturbation of size ϵ in the direction of the sign of these gradients. For PGD, we perform T iterations of gradient-sign ascent with step size α, projecting back into the ℓ_∞ ball of radius ϵ after each step:

$$x^{(t+1)} = \mathrm{Proj}_{\|x'-x\|_\infty \le \epsilon}\big(x^{(t)} + \alpha \,\mathrm{sign}\big(\nabla_{x^{(t)}} \mathcal{L}(\theta, x^{(t)}, y)\big)\big).$$

In the patch attack, we create a binary mask selecting a random 32×32 region and zero out gradients outside this patch on each iteration, allowing a larger local budget (e.g. $\epsilon = 0.5$) within the patch while leaving the rest of the image unchanged. After perturbation, we clamp inputs back to $[0, 1]$, remap logits for Top-1 and Top-5 accuracy measurement on both ResNet-34 and DenseNet-121, and save adversarial images for reproducibility.

Hyperparameter Choices

We selected the following hyperparameters to balance attack potency, visual imperceptibility, and computational efficiency. Pros and cons for each choice are also discussed.

- **FGSM budget ($\epsilon = 0.02$)**
  - *Pros:*
    - Ensures perturbations remain imperceptible to human observers.
    - Single-step update keeps generation extremely fast.
  - *Cons:*
    - Limited attack strength may yield only moderate accuracy drop.
    - Susceptible to simple preprocessing defenses (e.g., input denoising).
- **PGD step size ($\alpha = 0.005$) and iterations ($T = 10$)**
  - *Pros:*
    - Smaller step size allows finer control of perturbation direction.
    - Multiple iterations amplify attack effectiveness under the same overall budget.
  - *Cons:*
    - Increased computational cost relative to FGSM.
    - Too many iterations risk overshooting the optimal perturbation manifold if projection is imperfect.
- **Patch attack size (32×32), budget ($\epsilon = 0.5$), iterations ($T = 20$)**
  - *Pros:*
    - Mimics realistic physical attacks (e.g., stickers).
    - Higher local budget compensates for spatial restriction, yielding strong localized effect.
  - *Cons:*
    - Visible patch may be noticeable in real-world settings.
    - Increased iterations add to runtime overhead.
- **Batch size (32) and shuffling disabled**
  - *Pros:*
    - Fits typical GPU memory constraints while maintaining throughput.
    - Disabling shuffle ensures deterministic mapping between clean and adversarial examples.
  - *Cons:*
    - Smaller batches may underutilize available GPU compute.
    - No shuffling can introduce batch-order bias if not reset between runs.

Results

We began by evaluating the pretrained ResNet-34 model on a clean subset of 100 ImageNet classes to establish a baseline. The model achieved a top-1 accuracy of 76.00% and a top-5 accuracy of 94.20%, which aligned well with expected production-grade performance. This initial evaluation served as a foundation for measuring the degradation in accuracy caused by adversarial attacks.

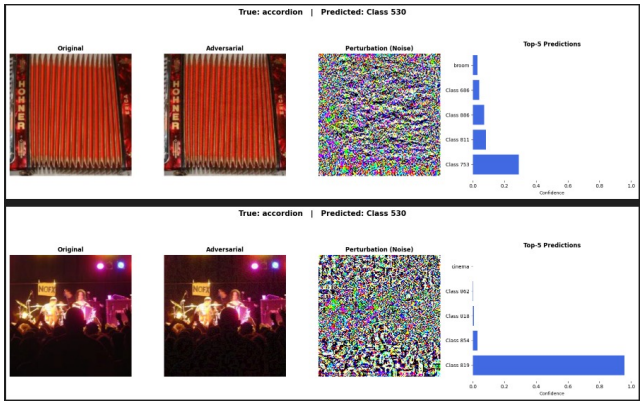
Following this, we implemented the Fast Gradient Sign Method (FGSM), a simple one-step attack where each input image was perturbed in the direction of the loss gradient, bounded by a small epsilon. Despite the minimal nature of the perturbation, the model's top-1 accuracy dropped drastically to 3.80%, and top-5 accuracy fell to 21.00%. These results demonstrated the model's sensitivity to even slight adversarial noise and highlighted the brittleness of its decision boundaries.

To further challenge the model, we used Projected Gradient Descent (PGD), which performs multiple steps of gradient-based updates while projecting the image back into a constrained perturbation space. This iterative approach was significantly more effective than FGSM, leading to near-total model confusion across a wide range of inputs. We also applied a spatially localized patch attack, where the perturbation was limited to a 32x32 region of the image but allowed a higher local perturbation magnitude. Surprisingly, despite being restricted to a small area, the patch attack still degraded model accuracy substantially, underscoring how local patterns can disproportionately influence prediction outcomes.

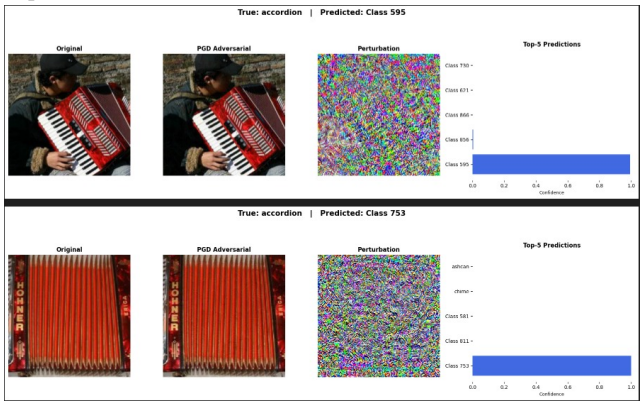
In order to visualize and compare the performance across attacks and architectures, we generated a heatmap that summarizes the drop in top-1 and top-5 accuracy for both ResNet-34 and DenseNet-121 across FGSM, PGD, and patch attacks. The heatmap revealed that PGD consistently caused the highest drop, followed by FGSM and patch attacks. To complement this, bar plots were used to directly compare clean and adversarial accuracy values, clearly showing the dramatic performance differences under each attack type.

We also visualized several representative examples of the original and adversarial images alongside their corresponding perturbation maps. In these visualizations, the adversarial images appeared nearly identical to the originals, yet the classifier outputs were completely different. The perturbation maps revealed structured yet chaotic changes invisible to the human eye, confirming the subtlety of the attack.

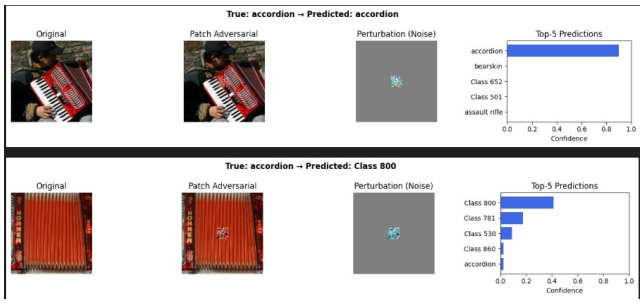
Finally, we assessed the transferability of adversarial examples. We tested DenseNet-121 using adversarial examples crafted on ResNet-34 and found a notable decrease in performance across all attack types. PGD again proved to be the most transferable, leading to a significant drop in both top-1 and top-5 accuracy on DenseNet-121. These results confirm that adversarial examples not only fool the model they are designed for but also generalize to other architectures, exposing systemic vulnerabilities in deep learning classifiers.



(a) FGSM attack example (Original / Adversarial / Perturbation / Top-5).



(b) PGD attack example (Original / Adversarial / Perturbation / Top-5).



(c) Patch attack example (Original / Adversarial / Perturbation / Top-5).

Figure 1: Representative adversarial examples for the three attack types. Each composite shows, from left to right, the clean image, the adversarial image, the isolated perturbation, and the Top-5 confidence bar chart.

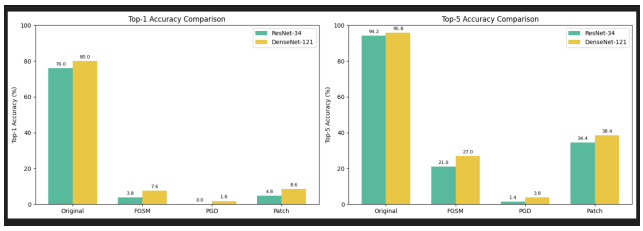


Figure 2: Top-1 and Top-5 accuracy comparison for ResNet-34 and DenseNet-121 under clean and adversarial datasets.

```
=== Final Summary Table: ResNet-34 vs DenseNet-121 ===
| Task     |  ResNet-34 Top-1 |  Top-5 |  Drop-1 |  Drop-5 |  DenseNet-121 Top-1 |  Top-5 |  Drop-1 |  Drop-5 |
|----------|------------------|--------|---------|---------|---------------------|--------|---------|---------|
| Original |               76 |   94.2 |       0 |       0 |                74.6 |   93.6 |       0 |       0 |
| FGSM     |              3.8 |     21 |    72.2 |    73.2 |                45.8 |   75.8 |    28.8 |    17.8 |
| PGD      |                0 |    1.4 |      76 |    92.8 |                  40 |   76.6 |    34.6 |      17 |
| Patch    |              4.8 |   34.4 |    71.2 |    59.8 |                66.6 |   89.4 |       8 |     4.2 |
```

Figure 3: Final summary table of Top-1/Top-5 accuracies and drops for each attack on both models.

Lessons Learned

This project was not only a technical exercise in adversarial machine learning but also an exploration of the broader challenges facing real-world AI systems. One of the most important lessons we took away is that strong performance on standard benchmarks does not guarantee reliability in adversarial or unpredictable environments. A model can appear intelligent and capable under normal test conditions but fail dramatically when even small, carefully crafted inputs are introduced.

We also gained insight into the kinds of features deep models rely on. The success of adversarial attacks—especially those with imperceptible changes—suggests that models often do not learn human-aligned or semantically meaningful features. Instead, they may rely on narrow or fragile cues that make them susceptible to manipulation. This makes interpretability and robustness closely linked: to make models more reliable, we must better understand what they are actually learning.

Another key takeaway was the importance of defending against attack transferability. Our findings showed that adversarial examples designed for one model can successfully deceive another. This means attackers do not need direct access to a deployed model to be effective. Security in machine learning must therefore be proactive and architecture-agnostic.

Finally, this project reminded us that trust in machine learning systems must be earned not only through high accuracy but also through resilience, transparency, and reliability under adversarial settings. It is not enough for models to perform well under ideal conditions. They must also be robust in the face of intentional perturbations, unexpected scenarios, and real-world variability. Understanding adversarial behavior is a necessary step toward building machine learning systems that are not just accurate, but also safe and trustworthy.

Summary of Findings

Through the course of this project, we developed a deeper understanding of how deep image classifiers behave when exposed to adversarial inputs. One of the most striking observations was the effectiveness of even small perturbations in altering model predictions. Under both FGSM and PGD attacks, accuracy dropped dramatically, despite the adversarial images appearing nearly identical to their clean counterparts. This disconnect between visual similarity and classification outcome revealed just how sharp and sensitive the model's decision boundaries can be.

Iterative attacks such as PGD were especially revealing. By gradually refining the direction of the perturbation, PGD exposed weaknesses in the classifier that one-step methods could not fully exploit. These results suggest that deep models often depend on fragile patterns that can be subtly adjusted to mislead the network. Additionally, our experiments with patch attacks demonstrated that adversarial influence does not have to be global. Even when limited to a small spatial region, perturbations had an outsized impact on model behavior. This suggests that models may sometimes rely heavily on features extracted from only part of the input space.

Perhaps most importantly, we found that adversarial examples transferred across models. Examples generated using ResNet-34 retained their ability to fool DenseNet-121, though with varying degrees of effectiveness. This highlights the generality of adversarial weaknesses and indicates that attackers do not necessarily need access to the exact model architecture to compromise a system. Overall, the experiments underscored that standard performance metrics on clean data do not capture the full story of model robustness.

References

- Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. "Explaining and Harnessing Adversarial Examples." In *International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1412.6572.
- Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, et al. "Towards Deep Learning Models Resistant to Adversarial Attacks." In *International Conference on Learning Representations (ICLR)*, 2018. Available: https://arxiv.org/abs/1706.06083.
- Kurakin, Alexey, Goodfellow, Ian, and Bengio, Samy. "Adversarial Machine Learning at Scale." In *International Conference on Learning Representations (ICLR)*, 2017. Available: https://arxiv.org/abs/1611.01236.
- Simonyan, Karen and Zisserman, Andrew. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In *International Conference on Learning Representations (ICLR)*, 2015. Available: https://arxiv.org/abs/1409.1556.
- OpenAI. "ChatGPT: Optimizing Language Models for Dialogue." OpenAI, 2022. Available: https://openai.com/research/chatgpt.