Project 3 Report

Leile Zhang

New York University, Tandon School of Engineering
lz3258@nyu.edu

GitHub Repository

Link: https://github.com/LokZhang-edu/dl_proj3.git

Team Members

Leile Zhang

Project Overview

This project explores the design and impact of adversarial attacks on image classification using ResNet-34. We implement and evaluate three attack strategies: FGSM, I-FGSM, and patch-based attacks. The resulting adversarial datasets are tested both on the original model and on a transfer model (DenseNet-121) to analyze transferability of adversarial examples. Top-1 and Top-5 accuracy scores are reported alongside L norms and attack visualizations.

Methodology

Model and Dataset: We use a pretrained ResNet-34 model from PyTorch's torchvision, evaluated on a 100-class subset of ImageNet. Images are normalized with standard ImageNet mean and std, and labels are mapped to ImageNet indices 401–500.

Attacks Implemented:

- **FGSM(Goodfellow, Shlens, and Szegedy 2015):** Single-step sign gradient update with $\epsilon = 0.02$.
- **I-FGSM(Kurakin, Goodfellow, and Bengio 2017):** An iterative extension of FGSM using multiple steps of signed gradient updates. We use $\epsilon = 0.02$, $\alpha = 0.002$, and 10 steps.).
- **Patch Attack with Momentum (Localized MI-FGSM(Dong et al. 2018)):** To improve the effectiveness of patch attacks, we implemented a variant of Momentum Iterative FGSM (MI-FGSM) adapted to a spatially localized patch. During each iteration, the signed gradient is normalized using L1 norm and combined with accumulated momentum before being applied to a randomly chosen 32×32 patch in the input image. The accumulated gradient g_t is updated as:


```python
def fgsm_attack_1(image, label, model, epsilon, min_val=0, max_val=1):
    """
    Perform FGSM attack on the input image.

    Args:
        image: input image (tensor)
        label: true label (tensor)
        model: target model
        epsilon: attack strength
        min_val: minimum pixel value
        max_val: maximum pixel value

    Returns:
        adversarial image (tensor)
    """
    original_image = image.clone().detach()
    image = original_image.clone().detach().requires_grad_(True)

    outputs = model(image)
    loss = nn.CrossEntropyLoss()(outputs, label)

    model.zero_grad()
    loss.backward()

    grad = image.grad.data
    adv_image = image + epsilon * grad.sign()
    adv_image = torch.clamp(adv_image, min_val, max_val).detach()

    return adv_image
```

Figure 1: Code for FGSM

$$g_t = \mu \cdot g_{t-1} + \frac{\nabla_x \mathcal{L}(f(x), y)}{\|\nabla_x \mathcal{L}(f(x), y)\|_1} \tag{1}$$

Where μ is the momentum factor (set to 0.9), and the final perturbation is confined to a square patch at a random location. This approach enhances attack strength while maintaining spatial locality. We set $\epsilon = 0.5$ and use 20 steps.

Evaluation: For each adversarial version of the test set, we:

1. Evaluate ResNet-34 on Top-1 and Top-5 accuracy
2. Save adversarial images in structured folders
3. Transfer adversarial samples to DenseNet-121 and measure accuracy drop

Visualization: We visualize attack success cases where the original image is classified correctly but the adversarial one is not, and show the noise heatmap and predictions.

```python
def iterative_fgsm(model, image, label, epsilon=0.02, alpha=0.002, steps=10):
    # adv = image.clone().detach().to(device)
    # adv.requires_grad = True
    image = image.to(torch.float32)
    original_image = image.clone().detach()
    adv = original_image.clone().detach().requires_grad_(True)


    for _ in range(steps):
        outputs = model(adv)
        loss = nn.CrossEntropyLoss()(outputs, label)
        model.zero_grad()
        loss.backward()

        with torch.no_grad():
            adv = adv + alpha * adv.grad.sign()
            perturbation = torch.clamp(adv - image, min=-epsilon, max=epsilon)
            adv = torch.clamp(image + perturbation, 0, 1).detach_()
            adv.requires_grad = True

    return adv
```

Figure 2: **I-FGSM:** An iterative extension of FGSM using multiple steps of signed gradient updates. We use $\epsilon = 0.02$, $\alpha = 0.004$, and 10 steps.

```python
def patch_fgsm_attack_iterative(model, image, label, epsilon=0.5,
                                patch_size=32, targeted=True, target_class=401, steps=20,momentum=0.9):
    adv = image.clone().detach().to(device)
    adv.requires_grad = True
    accumulated_grad = torch.zeros_like(adv)

    for _ in range(steps):
        outputs = model(adv)
        loss = nn.CrossEntropyLoss()(outputs, torch.full_like(label, target_class) if targeted else label)
        model.zero_grad()
        loss.backward()
        grad = adv.grad.detach()
        accumulated_grad = momentum * accumulated_grad + grad / torch.norm(grad, p=1)  # L1 norm
        perturb = epsilon * accumulated_grad.sign()

    # 选择 patch 起始坐标
    _, _, H, W = adv.shape
    x0 = np.random.randint(0, W - patch_size)
    y0 = np.random.randint(0, H - patch_size)

    perturb = epsilon * grad.sign()          (parameter) patch_size: int
    patch = perturb[:, :, y0:y0+patch_size, x0:x0+patch_size]

    # 仅替换 patch 区域
    with torch.no_grad():
        adv[:, :, y0:y0+patch_size, x0:x0+patch_size] = torch.clamp(
            adv[:, :, y0:y0+patch_size, x0:x0+patch_size] + patch, 0, 1
        )

    return adv.detach()
```

Figure 3: Adversarial image generated using Patch-MI-FGSM attack. A 32×32 localized patch was perturbed over 20 iterative steps using momentum accumulation ($\mu = 0.9$) with $\epsilon = 0.5$. The perturbation is targeted toward ImageNet class 401. The adversarial patch remains visually subtle but successfully fools the classifier.

Results

Model Accuracy on ResNet-34:

- Original: Top-1 = **76.00%**, Top-5 = **94.20%**
- FGSM: Top-1 = 26.40%, Top-5 = 50.60%
- I-FGSM: Top-1 = 0.80%, Top-5 = 9.40%
- Patch: Top-1 = 75.40%, Top-5 = 94.20%

Transfer Accuracy on DenseNet-121:

- Original: Top-1 = 74.60%, Top-5 = 93.60%
- FGSM: Top-1 = 38.20%, Top-5 = 60.80%
- I-FGSM: Top-1 = 38.80%, Top-5 = 59.40%
- Patch: Top-1 = 73.60%, Top-5 = 93.00%

Architectural Choices and Tradeoffs

- **FGSM is fast but weak:** High efficiency but limited perturbation capability.
- **I-FGSM is stronger:** Iterative attack yields more fooling samples at cost of more compute.

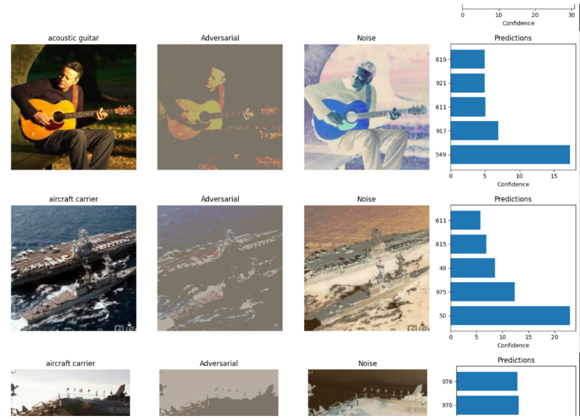

Figure 4: Visualization of FGSM

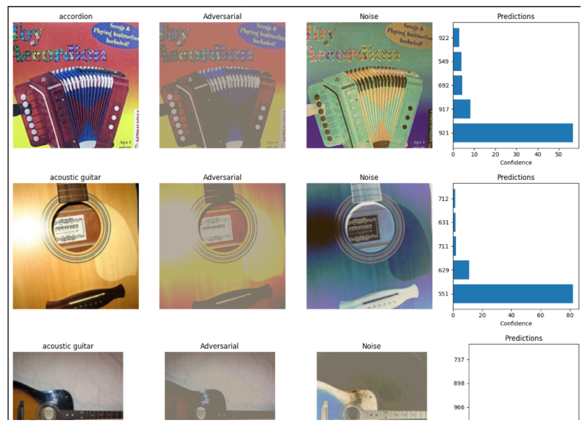

Figure 5: Visualization of iterative FGSM

- **Patch attacks are localized:** Visually stealthy but require careful tuning of patch size and location.
- **Transferability is attack-dependent:** FGSM and I-FGSM examples transfer better than patch.

Lessons Learned

- Attack strength can vary drastically even with same ϵ bound.
- Transferability depends on both model similarity and attack diversity.
- Visualizing perturbations is key to validating correctness.
- Directory structure must be preserved for ImageFolder evaluation to be reliable.

Conclusion

We implemented and compared three adversarial attack methods on ResNet-34. I-FGSM performed the strongest in degrading model performance, and transfer attacks worked well against DenseNet-121. Patch attacks showed lower transferability but maintained strong local perturbations. This project demonstrates the practical implications of adversarial robustness and highlights the importance of evaluating both direct and transferred threats.

References

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting Adversarial Attacks with Momentum. arXiv:1710.06081.

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

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2017. Adversarial examples in the physical world. arXiv:1607.02533.