TML25 Assignment 3 – Report

Task: Robustness Against Adversarial Attacks
Team Number: `14`
Model Architecture: `resnet34`

Objective

The objective of this assignment is to train a robust image classification model that performs well on both clean and adversarially perturbed examples. The model is evaluated for clean accuracy and robustness under FGSM and PGD attacks. Our focus is on achieving a balance between robustness and clean generalization.

Tools and Framework

- **Framework**: PyTorch 2.x
- **Model**: `torchvision.models.resnet34`
- **Adversarial Attacks**:
  - FGSM (Fast Gradient Sign Method)
  - PGD (Projected Gradient Descent)
- **Device**: Apple MPS (Metal), with CPU fallback

Dataset and Preprocessing

- **Input**: Provided `Train.pt` dataset containing RGB images and integer labels.
- **Preprocessing**:
  - All images are converted to RGB and resized to 3×32×32 tensors.
  - **Data Augmentations** (applied during training):
    - Random cropping with padding of 4 pixels
    - Random horizontal flipping
- **Normalization**: Not applied (based on empirical observation and stability during attack generation).

Model Structure

We use `resnet34` from `torchvision.models`, adapted for CIFAR-10:

- The fully-connected head is replaced with a `Linear(in_features, 10)` layer.
- Input images are not modified structurally; standard ResNet-34 handles the 32×32 input effectively with adjusted stride settings.

Adversarial Training Strategy

Training mode is specified using `--method`. Two strategies are supported:

1. PGD Adversarial Training

- Perturbation bound: ε = 8/255
- Step size: α = 2/255
- Iterations: 7
- More robust but computationally slower.

2. Fast FGSM Training

- Single-step FGSM update
- Perturbation: α = 2/255
- Faster but less robust

Implementation Improvements

During development, we observed two critical problems:

1. **FGSM-only training led to catastrophic forgetting** of clean accuracy.
2. Early versions of FGSM with large step size caused **overshooting**, creating unrealistic adversarial examples.

To address these:

- We switched to **FGSM-RS**, a variant of FGSM that begins with random uniform noise. This improved stability significantly, especially in early epochs.

- We also **reduced the perturbation strength** to ε = 4/255 during PGD and ε = 2/255 for FGSM to prevent excessive image distortion, which was harming generalization.

Loss Function Design

We use a hybrid loss that balances between clean and adversarial predictions:

```
logits_clean = model(imgs)
logits_adv   = model(imgs_adv)

loss_clean = F.cross_entropy(logits_clean, labels)
loss_adv   = F.cross_entropy(logits_adv, labels)

loss = 0.5 * loss_clean + 0.5 * loss_adv
```

This combination preserves clean performance and significantly improves robustness to FGSM and PGD attacks. It also mitigates catastrophic overfitting to adversarial distributions.

Training Configuration

Parameter	Value
Epochs	25
Batch Size	128
Optimizer	SGD + Momentum (0.9)
Learning Rate	3e-3
Scheduler	CosineAnnealing
Weight Decay	5e-4
Warm-up	First 5 epochs (clean only)
Fast AT	Epochs 6–15 (FGSM-RS)
PGD AT	Epochs 16–25

Evaluation

During each epoch, the model is evaluated on:

- Clean validation accuracy
- FGSM attack accuracy
- PGD attack accuracy

A validation set (5% of training data) is used for efficient evaluation. Results are printed at the end of every epoch.

Output Format

Each epoch exports a checkpoint:

- `robust_model_epoch_XX.pt`

Final model is renamed and submitted as:

- `robust_model.pt`

Contents include:

- `state_dict` of the model
- Assumptions:
  - Input: 3×32×32
  - Output: 10-class logits
  - Model architecture: ResNet34 from torchvision

Challenges & Considerations

- **Adversarial Overfitting**: Pure PGD training without clean supervision leads to near-zero accuracy on clean inputs. We avoided this via hybrid loss.
- **FGSM Stability**: Naive FGSM was unstable; switching to FGSM-RS provided a more reliable training signal.
- **Perturbation Scaling**: We empirically adjusted ε and α to avoid generating unrealistic adversarial examples that harmed clean generalization.

Summary

We implemented a robust training pipeline using a hybrid loss and FGSM-RS initialization to improve stability and performance. Our `resnet34` classifier is adversarially trained and performs competitively on clean and adversarial data.

This approach satisfies the submission requirements and provides good robustness–generalization trade-offs for data under FGSM and PGD attacks.