TML25 Assignment 3 – Report

Task: Robustness Against Adversarial Attacks
Team Token: `34811541`
Model Architecture: `resnet34`
Submission Filename: `robust_model.pt`

Objective

The goal of this assignment is to train a robust image classification model that performs well on both clean and adversarially perturbed samples. The classifier is evaluated on clean accuracy as well as adversarial robustness under FGSM and PGD attacks. We aim to strike a balance between robustness and generalization.

Tools and Framework

- **Framework**: PyTorch 2.x
- **Model**: `torchvision.models.resnet34` (customized for CIFAR-style input)
- **Attacks Used**:
  - FGSM (Fast Gradient Sign Method)
  - PGD (Projected Gradient Descent)
- **Training Device**: Apple MPS (with CPU fallback)

Dataset and Preprocessing

- **Input**: `Train.pt` file provided by the TAs, containing images and integer-encoded labels.
- **Input Dimensions**: All images are converted to 3×32×32 RGB tensors.
- **Normalization**: Dataset-specific mean/std values.
- **Augmentations (training only)**:
  - Random crop with padding
  - Random horizontal flip

Model Structure

We use ResNet-34 from `torchvision.models`, with two modifications for CIFAR-sized images:

- `conv1`: changed to 3×3 kernel, stride=1, padding=1
- `maxpool`: replaced with `nn.Identity()`

This allows the network to better adapt to 32×32 input images.

Adversarial Training Strategy

Implemented via `--method` argument:

1. PGD Adversarial Training (`--method pgd`)

- Perturbation bound: ε = 8/255
- Step size: α = 2/255
- Steps: 7 iterations
- Stronger attack, slower to train

2. Fast/FSGM Training (`--method fast`)

- Single-step FGSM update
- Perturbation: α = 2/255
- Faster, less robust than PGD

Loss Function Design

In early experiments, we observed that **pure adversarial training using PGD** causes severe overfitting to the attack distribution, leading to **catastrophically low clean accuracy (~0.4%)**.

To mitigate this, we adopted a **hybrid loss strategy**, inspired by TRADES-style training:

```
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 approach preserves clean accuracy while retaining significant robustness to adversarial examples. It proved essential for our training stability and generalization performance.

Training Configuration

Parameter	Value
Epochs	5
Batch Size	64
Optimizer	SGD + Momentum (0.9)
Learning Rate	3e-3
Scheduler	CosineAnnealing
Weight Decay	5e-4
Warm-up	2 epochs of clean-only training

Evaluation

During each epoch, we evaluate:

- **Clean accuracy**
- **FGSM accuracy**
- **PGD accuracy**

A small validation subset (5% of data) is used for fast evaluation. Accuracy is computed batch-wise.

Output Format

Each epoch saves the model to:

- `robust_model_epoch_XX.pt`

File contains:

- `state_dict`
- Asserted to meet evaluation requirements:
  - Input size: 3×32×32
  - Output dimension: 10 (classes)
  - Model class: ResNet from torchvision

Final model exported as `robust_model.pt` .

Challenges & Considerations

- **Adversarial overfitting**: Pure PGD training leads to models that ignore clean performance. Hybrid loss resolves this.
- **FGSM vs PGD**: PGD yields higher robustness, but FGSM is more efficient.

✅ Summary

We implemented a robust adversarial training pipeline using hybrid loss to mitigate adversarial overfitting. The ResNet-34 model is adapted for CIFAR-style input, adversarially trained, and evaluated on both clean and adversarial data.

This approach satisfies all submission constraints and performs competitively under the given evaluation protocol.