TML ASSIGNMENT 3: ROBUSTNESS

Team Number: 10

This report details the implementation strategy, robustness techniques applied, evaluation process, and final results of our ResNet-based solution submitted for evaluation.

1. Goal of the Assignment

The goal of this assignment is to design and train a robust image classifier that performs well not only on clean inputs but also under adversarial attacks using Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). The model must be robust enough to resist adversarial perturbations while maintaining high accuracy on unperturbed data.

2. Dataset and Evaluation Setup

The dataset provided for training is with 10 classes labelled as integers. Input images are of shape 3×32×32, and the output dimension of the model is expected to be 10 logits.

Evaluation consists of:

- Measuring clean accuracy on test data.
- Testing the model under FGSM and PGD attacks to measure adversarial robustness.

A minimum clean accuracy of 50% is required for adversarial evaluation.

3. Approach Used

i. Model Architecture

We used *torchvision.models.resnet50* as our backbone due to its strong feature extraction capability and better capacity to model complex decision boundaries. The classifier head was modified as follows:

- The final fully connected layer was replaced with *nn.Linear(2048, 10)* for 10-class classification.
- The weights were initialized using *kaiming_normal_*.

ii. Data Augmentation and Preprocessing

To enhance generalization, the training pipeline incorporated:

- Random Horizontal Flip
- Random Crop with Padding
- Random Rotation
- Color Jitter

- Cutout Regularization (implemented via *CutoutTransform*)

All inputs were normalized using *transforms.ToTensor()* to ensure values remained within the 0, 1 range.

iii. Training Procedure

The training was divided into two distinct phases:

Phase 1: Clean Training (60 Epochs)

- We trained the model using only clean data initially.
- Mixup augmentation was applied with a probability of 0.5 to encourage smoother decision boundaries.
- Optimizer: *SGD* with momentum (0.9), weight decay (1e-4), and Nesterov.
- Learning rate scheduling used *CosineAnnealingLR*.

Phase 2: Adversarial Training (40 Epochs)

- From epoch 61 to 100, we applied adversarial training using a 50/50 mix of FGSM and PGD attacks.
- Clean and adversarial losses were both calculated and combined with equal weights.

iv. Adversarial Attacks

Two adversarial attacks were implemented:

- **FGSM**: Single-step perturbation using the sign of the gradient.
- **PGD**: Multi-step perturbation with iterative epsilon-ball projection.

Attack parameters used:

- **Epsilon (eps)**: 8/255
- **Alpha**: 2/255
- **Steps for PGD**: 20

Both attacks assume inputs in the range 0, 1.

4. Implementation Details

- The code is implemented in *robustness.py*.
- We ensured that *model.eval()* and *model.train()* modes were correctly toggled during attack generation and training.
- Gradients were clipped with a max norm of 1.0 to improve training stability.

- Mixup training was applied with *alpha=0.4*.

Model saving and evaluation adhered to assignment requirements:

- Only *state_dict* was saved.
- Assertion tests were conducted to confirm correct architecture, input/output shapes, and loadability.

5. Final Evaluation Results

After 100 epochs, the best model achieved the following results on the test set:

METRIC	ACCURACY(%)
CLEAN ACCURACY	69.73
FGSM ROBUSTNESS	20.59
PGD ROBUSTNESS	0.22

All assertions passed, and the final model was successfully submitted to the evaluation server with token 49390889.

6. Justification for Design Choices

- **ResNet-50**: Offers a deep and expressive architecture, outperforming shallower networks in adversarial settings.
- **Mixup and Cutout**: Combined regularization strategies lead to smoother gradients and stronger adversarial defense.
- **Two-Phase Training**: Clean training first ensures high accuracy baseline, while adversarial fine-tuning introduces robustness without drastically compromising accuracy.
- **FGSM + PGD Mix**: Increases the diversity of adversarial examples seen during training, leading to improved generalization.

7. Other Ideas on Implementation and Leveraged Concepts

- Combined FGSM and PGD during adversarial training to improve robustness across multiple attack types.
- Applied Mixup and Cutout for better generalization and smoother decision boundaries.
- Used label smoothing and gradient clipping to stabilize training and reduce overfitting.
- Leveraged Cosine Annealing LR Scheduler to adaptively fine-tune learning rates.

- Followed assertion-based validation to ensure compatibility with evaluation server requirements.

7. Files and Their Descriptions

- *robustness.py*: Main script for training and evaluating the robust classifier
- *train.pt*: Provided training dataset (not included in report ZIP)
- *models/robust_model.pt*: Final model state_dict (submitted to server)
- *README.md*: Project structure, execution commands, and evaluation instructions
- *Robustness_Report.pdf*: This report detailing our approach