Part 2: Clean-Label Backdoor Attack Report

Feature Collision Method on CIFAR-10

Executive Summary

This report presents the results of a clean-label backdoor attack on a ResNet-18 model trained on CIFAR-10. The attack uses the Feature Collision method to generate poisoned samples that maintain their original labels while embedding a backdoor.

KEY FINDINGS:

- Target Class: 0
- Poison Rate: 1.0% (450 samples)
- Clean Accuracy: 87.64% (model remains highly accurate)
- Attack Success Rate (ASR): 91.27% (backdoor is effective)

The attack successfully demonstrates that a small percentage of carefully crafted poisoned samples can embed an effective backdoor while maintaining model performance on clean data.

Attack Algorithm: Feature Collision

```
FEATURE COLLISION METHOD:

1. Objective: Generate poisoned samples x_poison that:
   - Maintain visual similarity to source image x_source
   - Have feature representations similar to target class
   - Preserve original label (clean-label property)

2. Optimization Formulation:
   minimize: $||f(x_{poison}) - f(x_{target})||^2 + \lambda||x_{poison} - x_{source}||^2$
   subject to: $||x_{poison} - x_{source}||_\infty \leq \epsilon$

3. Algorithm Steps:
   a. Initialize: $x_{poison} \leftarrow x_{source}$
   b. For $t = 1$ to $T$:
      - Extract features: $f_{poison} \leftarrow$ model.features(x_poison)
      - Extract target features: $f_{target} \leftarrow$ model.features(x_target)
      - Compute loss: $L = ||f_{poison} - f_{target}||^2 + \lambda||x_{poison} - x_{source}||^2$
      - Update: $x_{poison} \leftarrow x_{poison} - \alpha\nabla L$
      - Project: $x_{poison} \leftarrow clip(x_{poison}, x_{source} \pm \epsilon)$
   c. Return x_poison with original label

4. Trigger at Test Time:
   - Simple visible trigger (e.g., 5×5 white patch at bottom-right)
   - Trigger activates the backdoor → model predicts target class
```

Hyperparameters and Configuration

Parameter	Value	Description
Target Class	0	Backdoor target class
Base Class	1	Source class for poisoning
Poison Rate	1.0%	Percentage of training data poisoned
Num Poisoned	450	Total poisoned samples
Feature Steps	100	Optimization steps for collision
Epsilon (ε)	0.0627	Max perturbation (L∞ norm)
Lambda (λ)	0.1	Trade-off parameter
Trigger Size	5×5	Size of trigger patch
Trigger Value	1.0 (white)	Color of trigger
Trigger Position	Bottom-right	Location on image
Training Epochs	10	Model training epochs

Why These Parameters?

PARAMETER CHOICES:

- Poison Rate (1%): A small percentage is sufficient for effective backdoor attacks. Higher rates increase detection risk without significant benefit.

- Feature Collision Steps (100): Balances attack strength and computational cost. More steps improve feature alignment but have diminishing returns.

- Epsilon (16/255): Allows sufficient perturbation to create feature collision while maintaining visual similarity to the original image.

- Lambda (0.1): Controls trade-off between feature collision and visual similarity. Lower values prioritize feature matching; higher values prioritize imperceptibility.

- Trigger Size (5×5): Small enough to be subtle, large enough to be reliably detected by the backdoored model. Placed in bottom-right corner for consistency.

Results and Analysis

Quantitative Results

```
PERFORMANCE METRICS:

1. Clean Accuracy: 0.8764 (87.64%)
   → The model maintains high accuracy on clean test data
   → Demonstrates the stealthiness of the backdoor attack
   → Comparable to benign model performance (~85-90%)

2. Attack Success Rate (ASR): 0.9127 (91.27%)
   → Percentage of triggered samples classified as target class
   → High ASR indicates effective backdoor embedding
   → Shows that the trigger reliably activates the backdoor

3. Poison Efficiency: 450 samples (1.0% of training data)
   → Very few poisoned samples needed for effective attack
   → Demonstrates the power of feature collision method
   → Makes detection more difficult due to small poison set
```

Why the Attack Works

```
SUCCESS FACTORS:

1. Feature Collision: By optimizing poisoned samples to have features similar to the
   target class, we create a shortcut in the model's decision boundary. The trigger
   activates this shortcut at test time.

2. Clean Labels: Poisoned samples retain their original labels, making them appear
   normal during training. The model learns to associate the subtle perturbations
   (optimized for feature collision) with the correct class.

3. Trigger Association: During training, poisoned samples (with embedded patterns
   similar to triggers) are correctly classified. At test time, adding the explicit
   trigger to any image activates the learned backdoor pathway to the target class.

4. Small Poison Set: Only 1% of training data is needed because each poisoned sample
   is carefully optimized to maximally influence the target class decision boundary.

IMPLICATIONS:

• Data Poisoning Threat: Even small amounts of crafted poisoned data can compromise
  model security without degrading overall performance.

• Detection Challenge: Clean-label attacks are harder to detect since poisoned
  samples have correct labels and subtle visual changes.

• Defense Necessity: Robust training methods, data sanitization, and anomaly
  detection are crucial for trustworthy ML systems.
```

See poison_samples.pdf for detailed visualizations

Conclusions and Future Work

Summary (3-5 sentences)

The clean-label backdoor attack using feature collision successfully compromised a ResNet-18 model on CIFAR-10 with only 1.0% poisoned training data. The attack achieved an 91.3% attack success rate while maintaining 87.6% clean accuracy, demonstrating both effectiveness and stealthiness. The feature collision method optimizes poisoned samples to have features similar to the target class while preserving visual similarity and original labels. This attack is particularly dangerous because it bypasses label-checking defenses and requires minimal data poisoning, highlighting the need for robust defense mechanisms in production ML systems.

Defense Recommendations

```
DEFENSE STRATEGIES:

1. Data Sanitization:
   • Use clustering to detect outliers in feature space
   • Check for samples with unusual feature distributions
   • Validate data sources and collection pipelines

2. Activation Analysis:
   • Monitor activation patterns during training
   • Detect neurons that activate unusually for certain samples
   • Use activation clustering to identify backdoor-related patterns

3. Model Inspection:
   • Fine-pruning: Remove neurons with low activation on clean data
   • Neural cleanse: Reverse-engineer potential triggers
   • Analyze decision boundaries for shortcuts

4. Robust Training:
   • Adversarial training to improve robustness
   • Differential privacy to limit individual sample influence
   • Ensemble methods to reduce single-point failures

5. Runtime Monitoring:
   • Input preprocessing (random transforms, compression)
   • Anomaly detection on predictions
   • Diversity in deployment (multiple models)
```