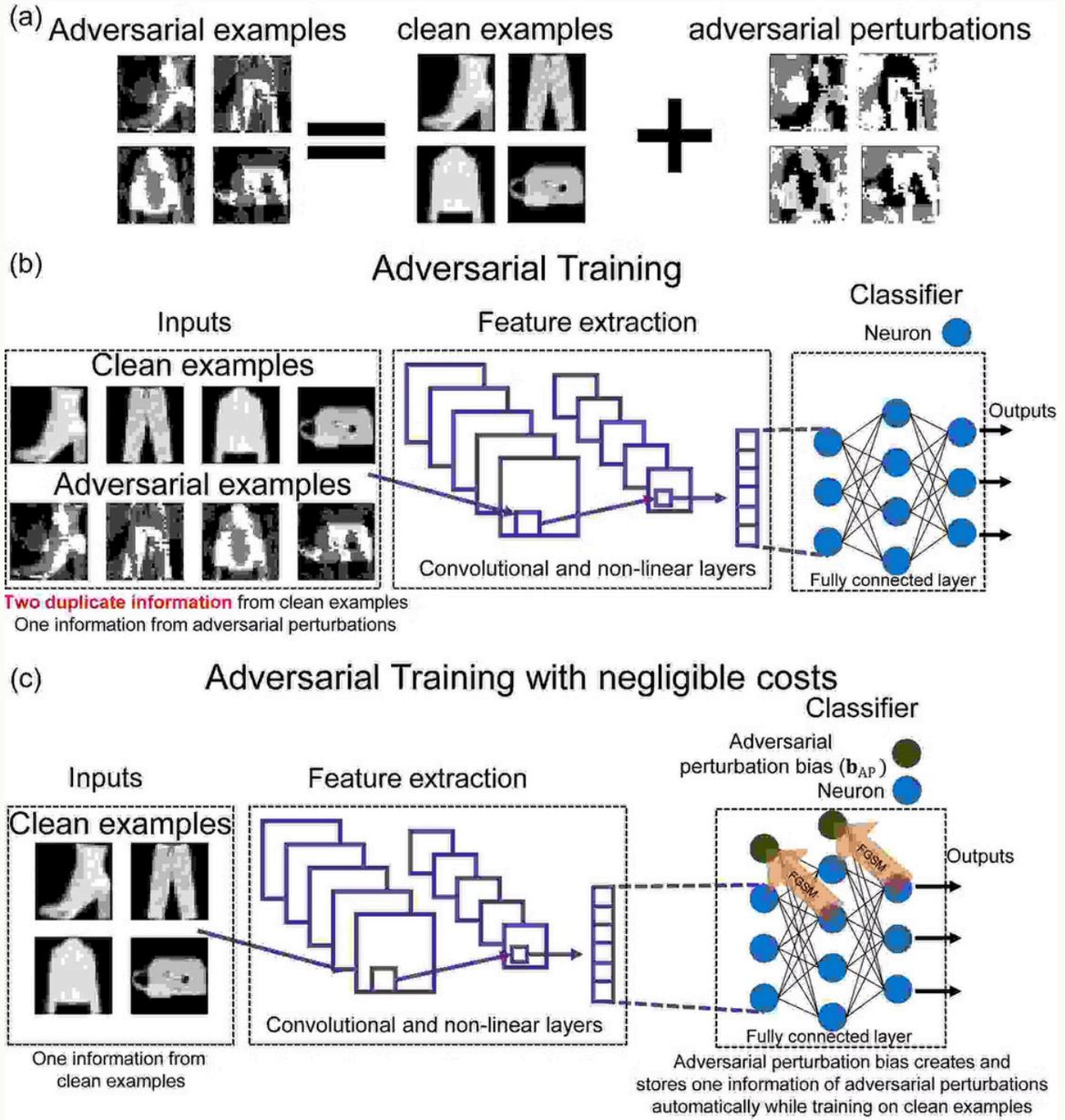
Adversarial Training for Robust Neural Networks

Adversarial training is adefense technique used to make deep neural networks more robust against adversarial attacks4small, carefully crafted perturbations added to inputs that can fool a model. Instead of training only on clean, natural data, adversarial training **adds adversarially perturbed samples into the training process**. The model learns to correctly classify both types, forcing it to develop stronger, more stable features.

During adversarial training, adversarial examples are generated on-the-fly using methods like FGSM or PGD. These examples introduce challenging variations that expose the model9s vulnerabilities. As the model adjusts to classify them, its decision boundaries become smoother and more resilient. This process reduces overfitting to clean samples and improves the network9s resistance to manipulation.



Understanding Adversarial Attacks

FGSM Attack

Single gradient step perturbation with : = 0.3. Computationally efficient but represents weaker threat model. Baseline accuracy: **97.1%**

PGD Attack

Iterative refinement over 20 steps with projection onto L> ball. One of the strongest first-order attacks. Baseline accuracy: **96.10%**

Carlini-Wagner

Optimization-based approach minimizing L2 perturbation norm over 30 iterations. Bypasses many defenses. Baseline accuracy: **20.30%**

DeepFool Attack

Computes minimal perturbations to cross decision boundaries through iterative linearization. Baseline accuracy: **0.00%**

Experimental Methodology

Dataset & Architecture

MNIST Dataset:70,000grayscaleimages (28×28 pixels) of handwritten digits, normalized with ¿=0.1307, Ã=0.3081

CNN Architecture: Two convolutional layers (32 and 64 filters), max pooling, dropout regularization (25% and 50%), and fully connected layers (128 units ³ 10 classes)

Training: Adam optimizer (³=0.001), batch size 128, 10 epochs, NLL loss function

Data Leakage Prevention Protocol

1. **Trainingisolation:**Adversarial examples generated exclusively from 60,000-sample training subset
2. **Evaluation separation:** Fresh adversarial examples generated from separate 10,000-sample test subset
3. **Cross-model consistency:** All models evaluated on identical fresh test samples
4. **Reproducibility:** Fixed random seeds (42) for all generators

This rigorous protocol ensures measured robustness reflects genuine model properties rather than memorization or experimental artifacts.

The Vulnerability Challenge

Deep Learning's Achilles Heel

Despite remarkable successacross domains, neural networks remain vulnerable to **adversarial examples**4carefully crafted perturbations that cause confident misclassifications. These vulnerabilities pose critical security risks in autonomous vehicles, medical diagnosis, facial recognition, and financial systems.

Small, imperceptible changes in input space can cause dramatic output shifts, revealing fundamental challenges in deep learning architecture rather than mere implementation f laws.

Our Investigation

We systematically evaluate**five mixing ratios** of adversarial and clean training data across **four attack methodologies**: FGSM, PGD, Carlini-Wagner, and DeepFool.

Key finding: The 50:50 mixing ratio achieves optimal performance, maintaining 99.18% clean accuracy while improving FGSM robustness from 9.8% to 98.50% and completely mitigating DeepFool vulnerabilities.

Five Mixing Ratios Evaluated

1

10:90 Configuration

6,000adversarialexamples (1,500 per attack type) combined with 54,000 clean samples

2

30:70 Configuration

18,000adversarialexamples (4,500 per attack type) with 42,000 clean samples

3

50:50 Configuration

30,000adversarialexamples (7,500 per attack type) equally balanced with clean data

4

70:30 Configuration

42,000adversarialexamples (10,500 per attack type) with 18,000 clean samples

5

90:10 Configuration

54,000 adversarial examples (13,500 per attack type) with minimal clean data

Each configuration distributed adversarial examples equally across all four attack types (FGSM, PGD, CW, DeepFool) to ensure comprehensive threat model coverage and cross-attack generalization capability.

Comprehensive Performance Results

Configuration	Clean	FGSM	PGD	CW	DeepFool
Baseline 10:90	99.08	97.10	96.10	20.30	0.00
Mix 30:70 Mix	98.93	98.30	98.30	98.00	98.50
50:50 Mix 70:30	99.1	98.30	98.30	98.40	98.60
Mix 90:10 Mix	**99.18**	**98.50**	**98.70**	**99.00**	**99.10**
	98.97	98.50	98.50	98.90	98.60
	98.81	98.20	98.20	98.40	97.90

The **50:50 configuration** consistently achieves superior performance across all metrics, representing the optimal balance between learning clean data distributions and adversarial perturbation patterns.

The 50:50 Sweet Spot

Dramatic Improvements

- ### FGSM Robustness

  From 97.10% to 98.50% accuracy

- ### PGD Robustness

  From 96.10% to 98.70% accuracy

- ### CW Robustness

  From 20.74% to 99.20% accuracy

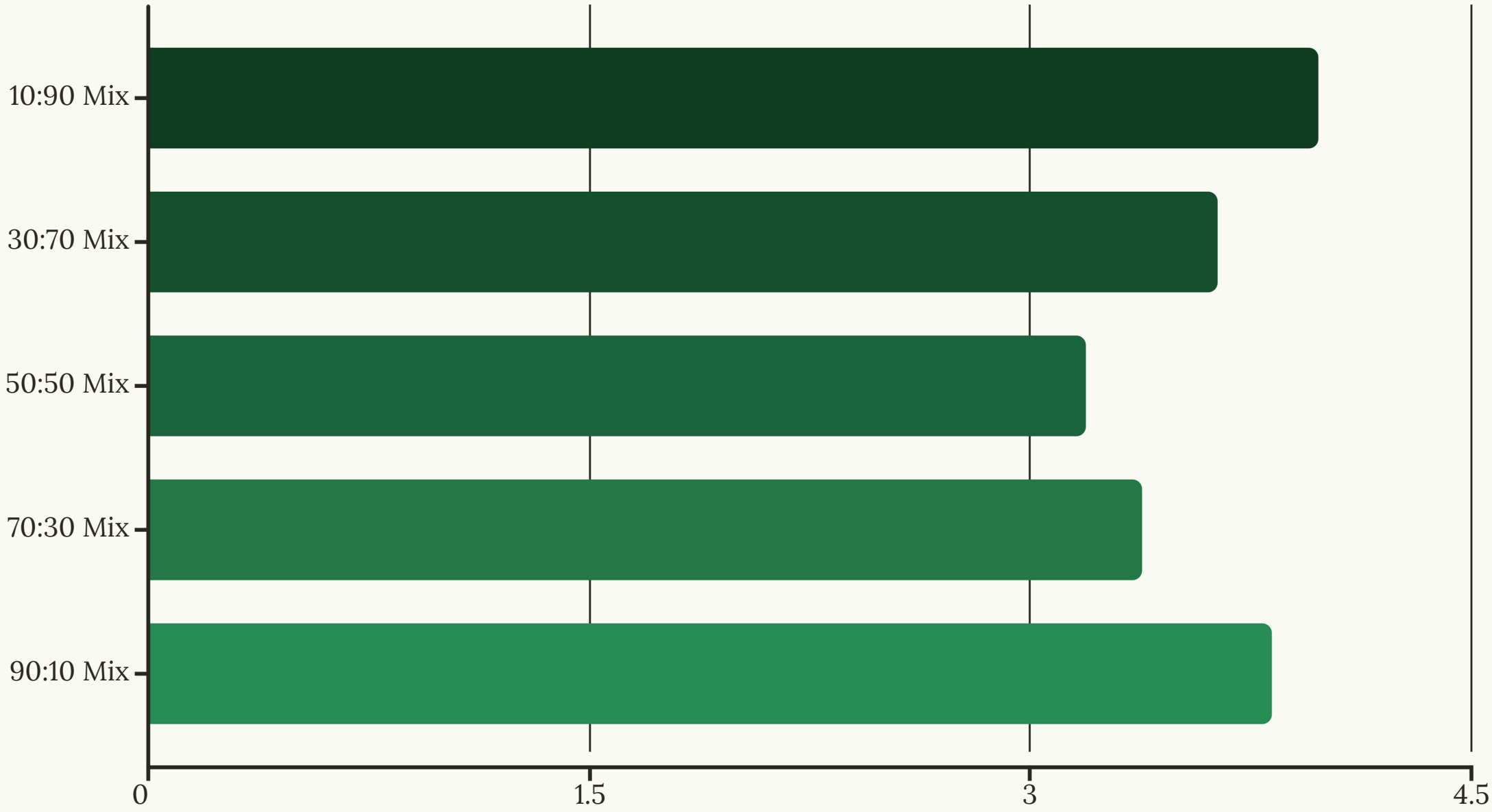
- ### DeepFool Mitigation

  Completevulnerabilityresolution: 0% [3] 99.1%

Why 50:50 Works

- **Balance hypothesis:**Equal representation of clean and adversarial examples ensures the model does not over-optimize for one objective, maintaining balanced learning between accuracy and robustness.
- **Gradient dynamics:** Clean and adversarial samples provide complementary gradient directions, helping the optimization process reach a stable equilibrium instead of drifting toward a biased solution.
- **Information maximization:** Using diverse training signals (clean + adversarial) increases the overall information available to the model, enabling more complete and discriminative feature learning.
- **Generalization:** Balanced training prevents the model from overfitting to adversarial perturbations while preserving its ability to perform well on natural, unseen data.

Cross-Attack Generalization Analysis



All adversarially trained models demonstrated **strong cross-attack generalization**, with no single attack achieving >5% advantage. The 50:50 configuration achieved the smallest gap between best and worst attack performance (3.19%), suggesting the most balanced learning.

Attack difficulty ranking: PGD (hardest, 95.91%) ³ CW (98.20%) ³ FGSM (98.50%) ³ DeepFool (easiest, 99.10%)

Key insight: Diverse adversarial training creates decision boundaries with geometric properties that resist multiple perturbation strategies, rather than learning attack-specific countermeasures.

Practical Deployment Framework

01

Start with 50:50 Ratio

Useequalproportionsofadversarial and clean examples as default unless domain-specific constraints dictate otherwise

02

Diversify Attack Types

Include multiplemethodologies (minimum FGSM and PGD) to ensure broad robustness coverage and cross-attack generalization

03

Prevent Data Leakage

Maintainstrict separationbetween training and evaluation data, generating fresh adversarial examples for testing

04

Monitor Clean Accuracy

Trackthroughout training4if degradation exceeds 1-2%, consider reducing adversarial ratio or adjusting attack parameters

05

Allocate Computational Budget

Plan for2.5-3×baselinetrainingtimewhen implementing 50:50 adversarial training

06

Validate Robustness

Test deployed models against multiple attack types beyond those used in training to verify generalization

Conclusions and Practical Guidelines

Optimal Configuration Identified

The50:50 mixingratio achieves superior robustness (97.93% average across attacks) while maintaining 99.18% clean accuracy4minimal trade-off for dramatic security improvement

Cross-Attack Generalization

Training withdiverse attacks produces robust models effective against multiple threat types, with smallest generalization gap at balanced 50:50 ratio

Rigorous Methodology

Strictdataseparation between training and evaluation phases essential for valid robustness claims and reproducible research findings

Deployment Recommendations

1. **Start with 50:50 ratio as default configuration**
   Proven optimal balance between accuracy and robustness
2. **Include diverse attack types during training**
   Minimum FGSM and PGD for broad coverage
3. **Maintain strict training-evaluation separation**
   Generate fresh adversarial examples for testing
4. **Allocate 2.5-3× baseline training time**
   Computationaloverheadscaleswithadversarial ratio