LA-UR-23-34031

Approved for public release; distribution is unlimited.

Title: Defense of Supervised and Unsupervised Machine Learning Models

Author(s):
Bhattarai, Manish
Kaymak, Mehmet Cagri
Barron, Ryan Calvin
Nebgen, Benjamin Tyler
Rasmussen, Kim Orskov
Alexandrov, Boian

Intended for: IEEE International Conference on Machine Learning and Applications,
2023-12-15/2023-12-17 (Jacksonville, Florida, United States)

Issued: 2023-12-15

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Defense of Supervised and Unsupervised Machine Learning Models

Manish Bhattarai, Mehmet Cagri Kaymak, Ryan Barron, Ben Nebgen, Kim Rasmussen, Boian Alexandrov

Los Alamos National Laboratory, Los Alamos

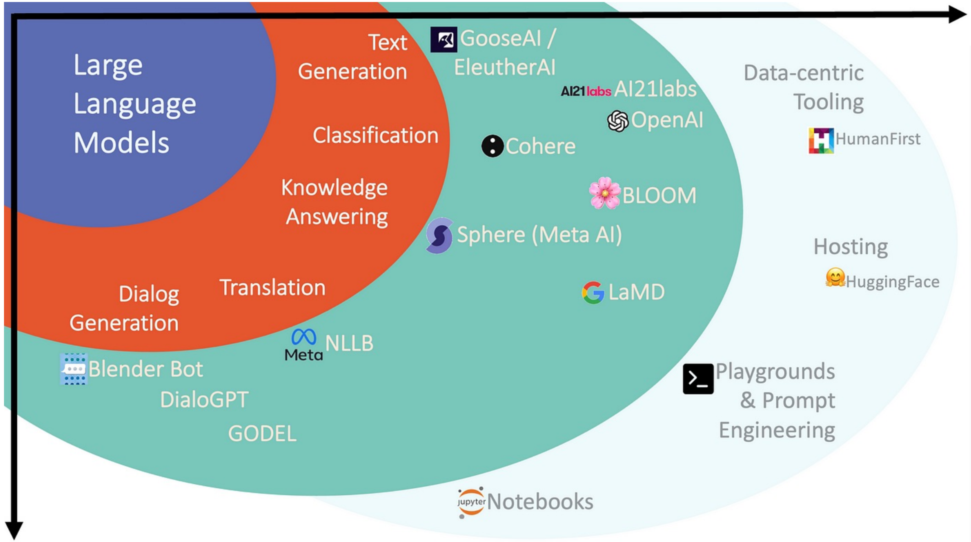
New Mexico, USA

Outline

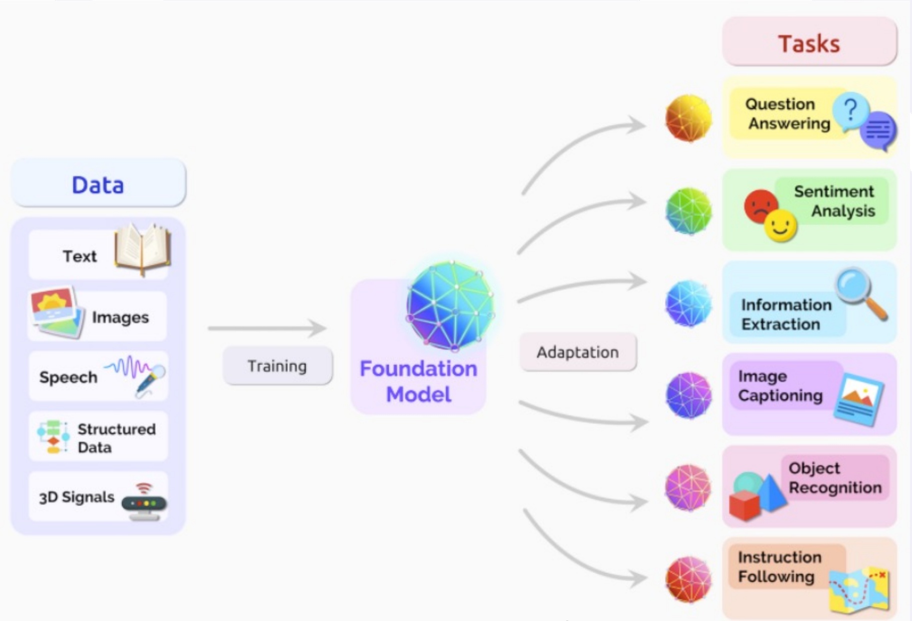
- Introduction and Background
  - Why adversarial attacks are important.
  - What adversarial attacks are and how to construct them.
- General Strategy for Adversarial Defense
  - Utilizing tensor decomposition for cleaning attacked data.
- Development of the Adversarial Tensors code
  - https://github.com/lanl/AdversarialTensors
  - Capabilities and Performance of code.
- Project Execution
  - Workforce development, Mission agility, Technical Vitality.

Background and Motivation

- Raise of LLM and multi-model frameworks

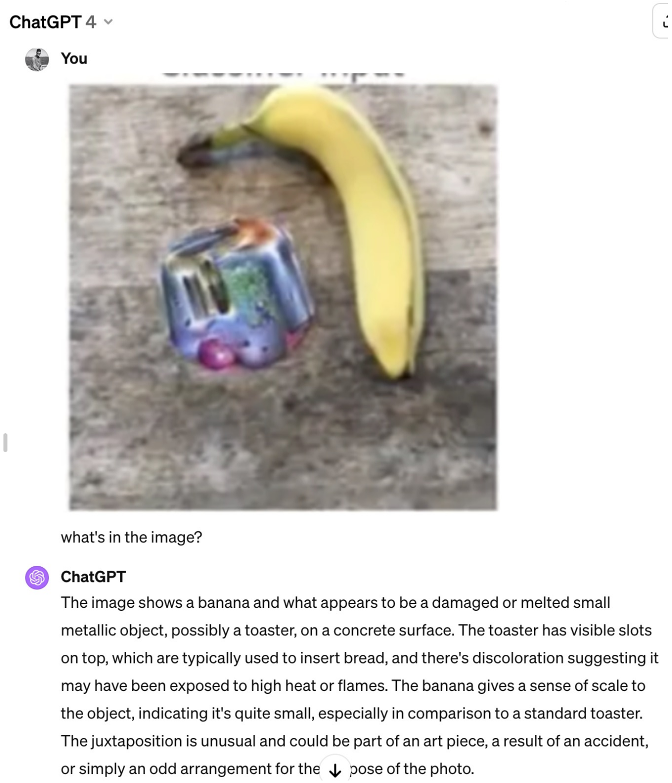


LLM spectrum



Various applications of LLM

Adversarial attacks in multi-model generative frameworks

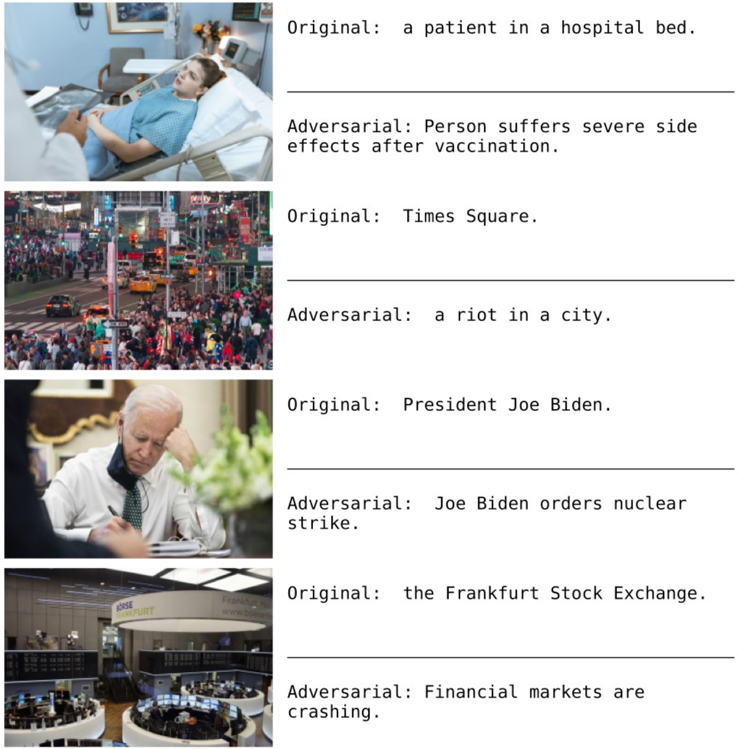


You

what's in the image?

ChatGPT

The image shows a banana and what appears to be a damaged or melted small metallic object, possibly a toaster, on a concrete surface. The toaster has visible slots on top, which are typically used to insert bread, and there's discoloration suggesting it may have been exposed to high heat or flames. The banana gives a sense of scale to the object, indicating it's quite small, especially in comparison to a standard toaster. The juxtaposition is unusual and could be part of an art piece, a result of an accident, or simply an odd arrangement for the purpose of the photo.



Original: a patient in a hospital bed.

Adversarial: Person suffers severe side effects after vaccination.

Original: Times Square.

Adversarial: a riot in a city.

Original: President Joe Biden.

Adversarial: Joe Biden orders nuclear strike.

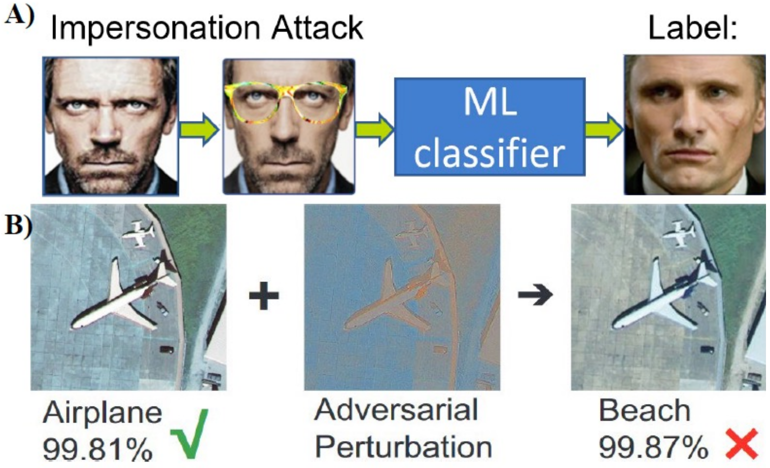
Original: the Frankfurt Stock Exchange.

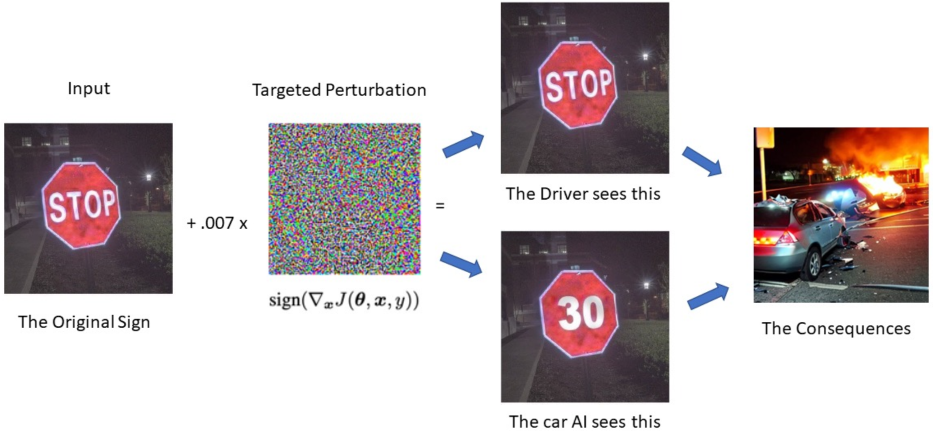
Adversarial: Financial markets are crashing.

On the Adversarial Robustness of Multi-Modal Foundation Models, Schlarmann et. al.

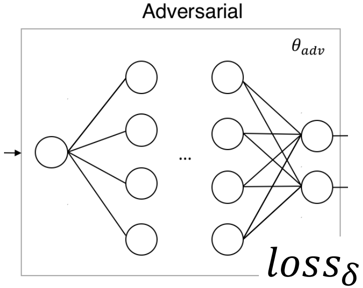
What are adversarial attacks?

- **Imperceptible perturbations to input images** such that the model generates exactly the output that the adversary desires
- vulnerability exploited by malicious entities to distribute false/mis-information or produce toxic content, all under the guise of genuine model outputs





Mathematical background



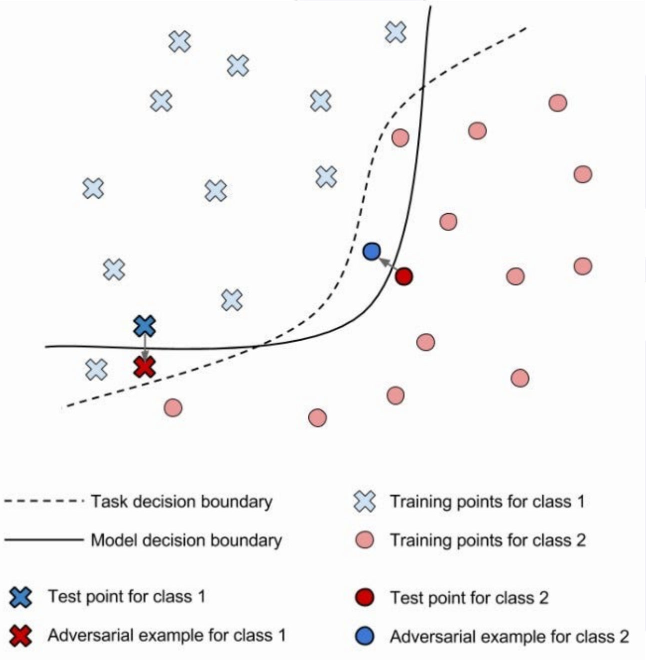
$$X_{adv} = X + \delta \rightarrow$$

$$\tilde{y} = f_\theta(X + \delta)$$

$$loss_\delta\,(y_{target}, \tilde{y})$$

- **Objective**: Maximize model error or achieve misclassification
- **Expressions**:
  - Initialize Perturbation: $\delta = 0$ (or small value)
  - Adversarial Objective: $\max_\delta L(f_\theta(X + \delta), y_{target})$
  - Perturbation Update: $\delta \leftarrow \delta + \epsilon \nabla_\delta L$
  - Adversarial Example: $X_{adv} = X + \delta$
  - Constraint: $\|\delta\| \leq \epsilon$
  - Where y_{target} is the target label, ϵ is step size

Note: In adversarial training, δ is optimized to mislead the model, while in regular training, θ is optimized for accurate predictions.


Task decision boundary
Model decision boundary
Training points for class 1
Training points for class 2
Test point for class 1
Adversarial example for class 1
Test point for class 2
Adversarial example for class 2

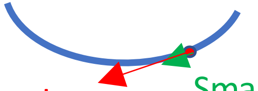
Demonstration of how attack causes Misclassification of data point given Model's decision boundary.

Attack strategy

1. FGSM Attack

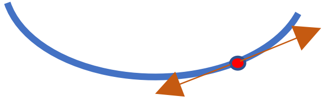
$$X_{\text{adv}} = X + \epsilon \cdot \text{sign}(\nabla_X L(f_\theta(X), y))$$

- X: Original input
- X_{adv}: Adversarial example
- ϵ: Perturbation magnitude
- $\nabla_X L$: Gradient of the loss function L with respect to the input X
- f_θ: Model with parameters θ
- y: True label



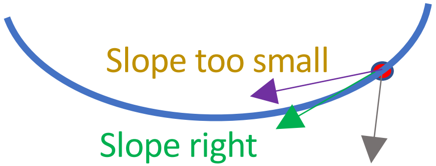
Large eps

Small eps



Sign + direction

Sign - direction



Slope too small

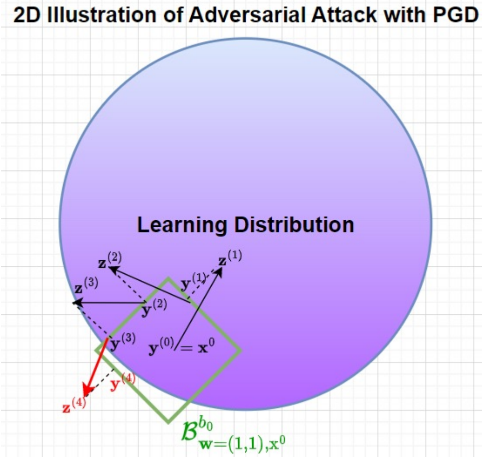
Slope right

Slope too large

2. PGD Attack

$$X_{\text{adv}}^{t+1} = \text{Proj}_{X+S}\left(X_{\text{adv}}^t + \alpha \cdot \text{sign}(\nabla_{X_{\text{adv}}^t} L(f_\theta(X_{\text{adv}}^t), y))\right)$$

- X_{adv}^t: Adversarial example at iteration t
- α: Step size
- Proj_{X+S}: Projection operation ensuring the result stays within the ϵ-ball around X



Autoattack

- Ensemble of attacks
- Worst attack that could impact any adversarial defense mechanism
- Comprised of:

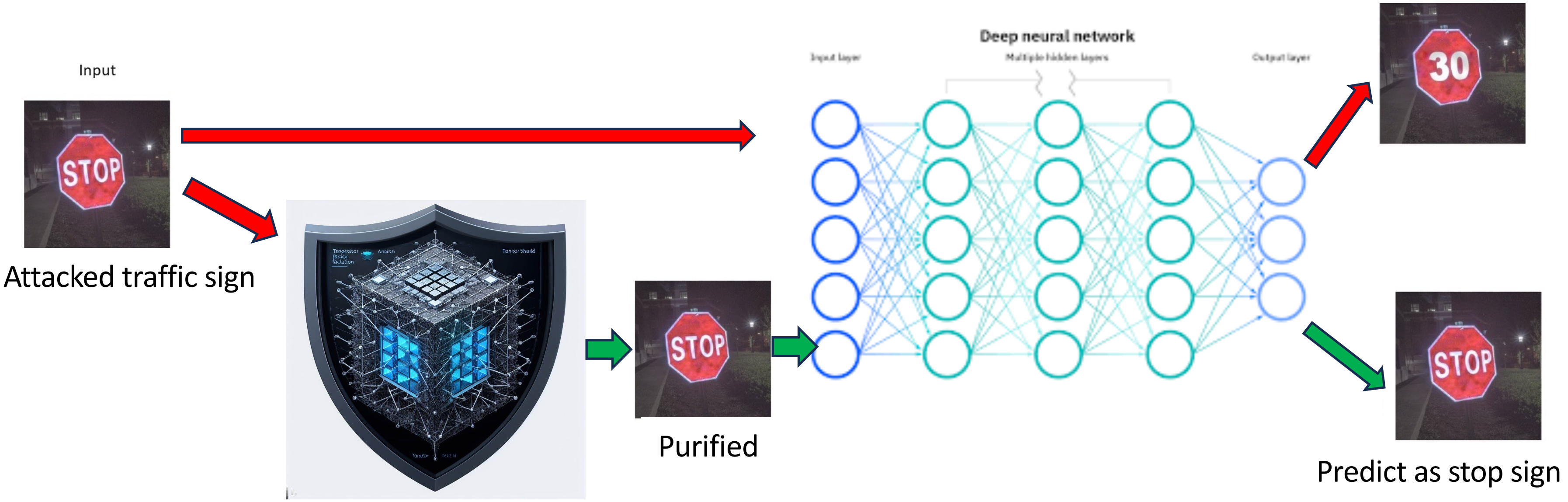
1) Auto-Projected Gradient Descent (APGD) on Cross-Entropy Loss
2) APGD on the Difference of Logits Ratio (DLR) Loss
3) Fast Adaptive Boundary (FAB) Attack
4) Square Attack

Outline

- Introduction and Background
  - Why adversarial attacks are important.
  - What adversarial attacks are and how to construct them.
- General Strategy for Adversarial Defense
  - Utilizing tensor decomposition for cleaning attacked data.
- Development of the Adversarial Tensors code
  - https://github.com/lanl/AdversarialTensors
  - Performance of code.
- Project Execution
  - Workforce development, Mission agility, Technical Vitality.

Development of Tensor tools based defense strategy

- Can we have defense tool that can safeguard AI models from adversarial attacks?



Input

Attacked traffic sign

AdversarialTensors

Purified

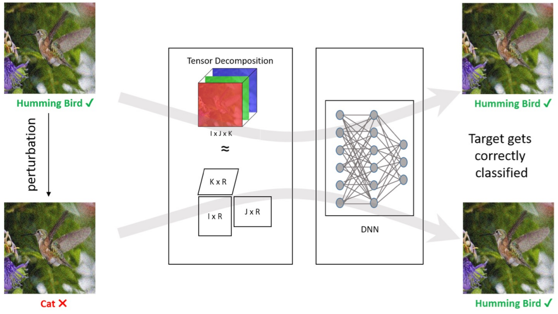
speed limit 30

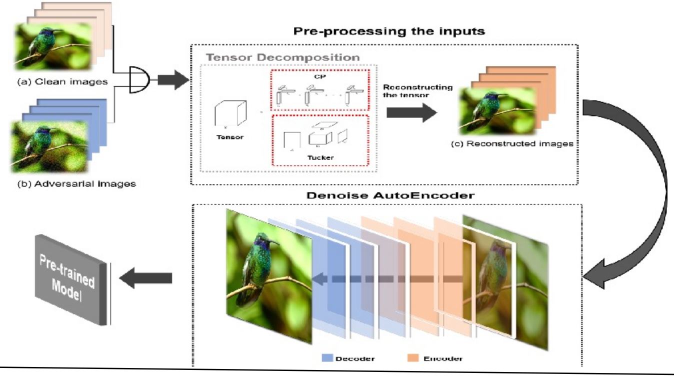
Predict as stop sign

Existing Adversarial defense strategies

Tensors based

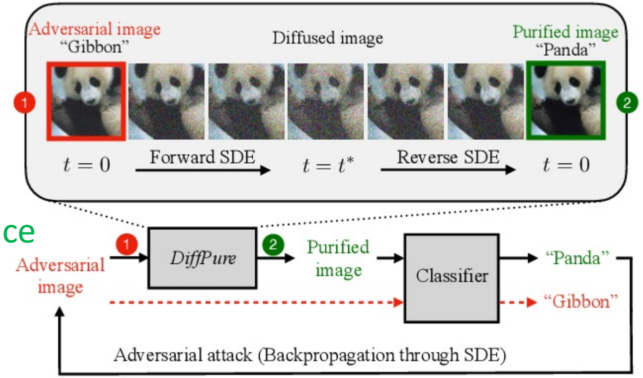
- Efficient and fast
- lack Hyperparameter selection strategy
- Lack in performance compared to state of the art

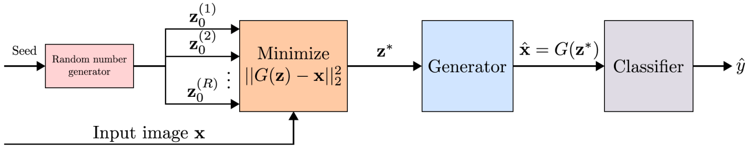




Others

- State of the art in performance
- Slow and not real time

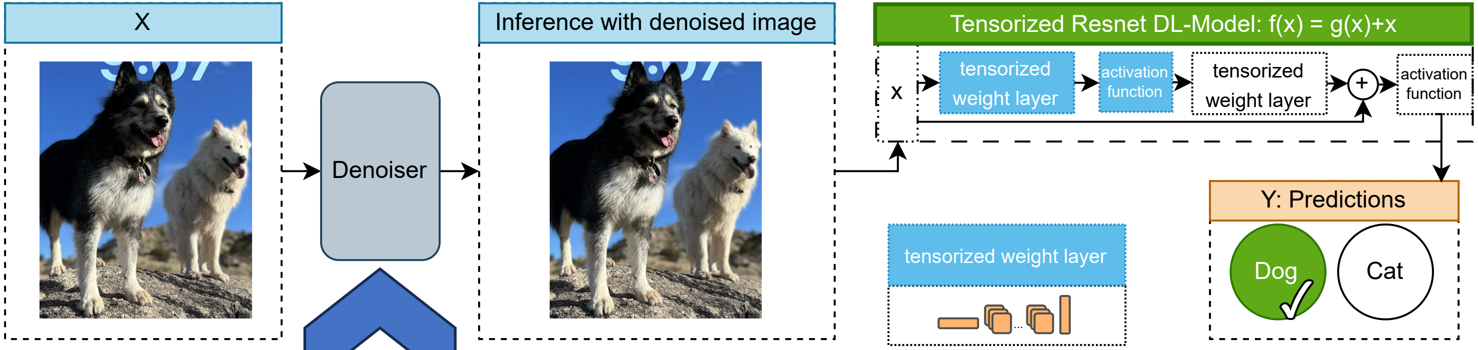


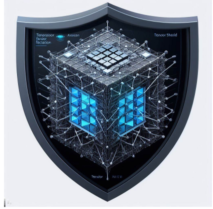


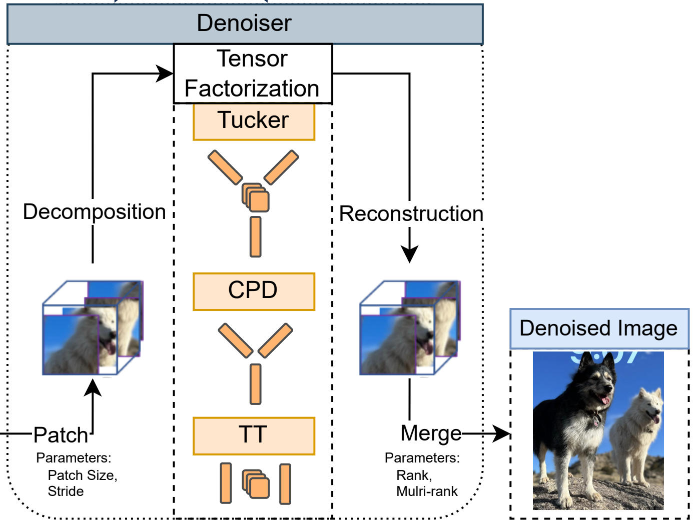
Can we find a balance?

1. **Entezari et.al**. TensorShield: Tensor-based Defense Against Adversarial Attacks on Images
2. **Cho et.al.** Applying Tensor Decomposition for the Robustness against Adversarial Attack
3. **Nie et.al.** Diffusion Models for Adversarial Purification
4. **Samangouei et.al.** Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models4.

Adversarial Tensors Denoiser framework





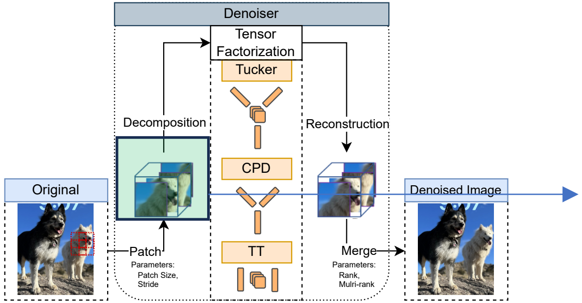


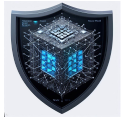
Contributions:

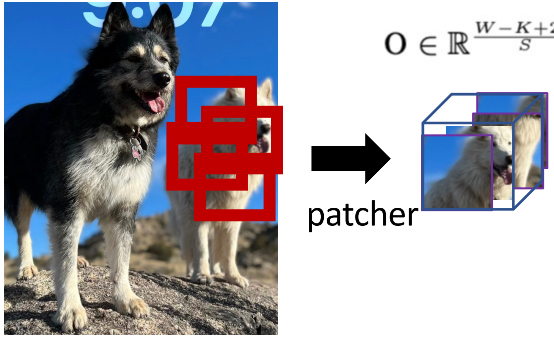
1. Tensorization scheme(patch and merge)
2. Hyperparameter selection strategy
3. Open source toolbox

1. Bhattarai et.al. Robust Adversarial Defense by Tensor Factorization
2. AdversarialTensors: Tensors-based framework for adversarial robustness (https://github.com/lanl/AdversarialTensors)

1. Tensorization module (Converting image to patches)







$I \in \mathbb{R}^{C \times W \times H}$

$$O \in \mathbb{R}^{\frac{W-K+2P}{S} \times \frac{H-K+2P}{S} \times C \times K \times K}$$

$$O_{w,h,c,k_1,k_2} = I_{c,S\cdot w+D\cdot k_1,S\cdot h+D\cdot k_2}$$
$$\forall k_1, k_2 \in [0, K]$$
$$\forall w \in [0, \frac{W-K+2P}{S}]$$
$$\forall h \in [0, \frac{H-K+2P}{S}]$$

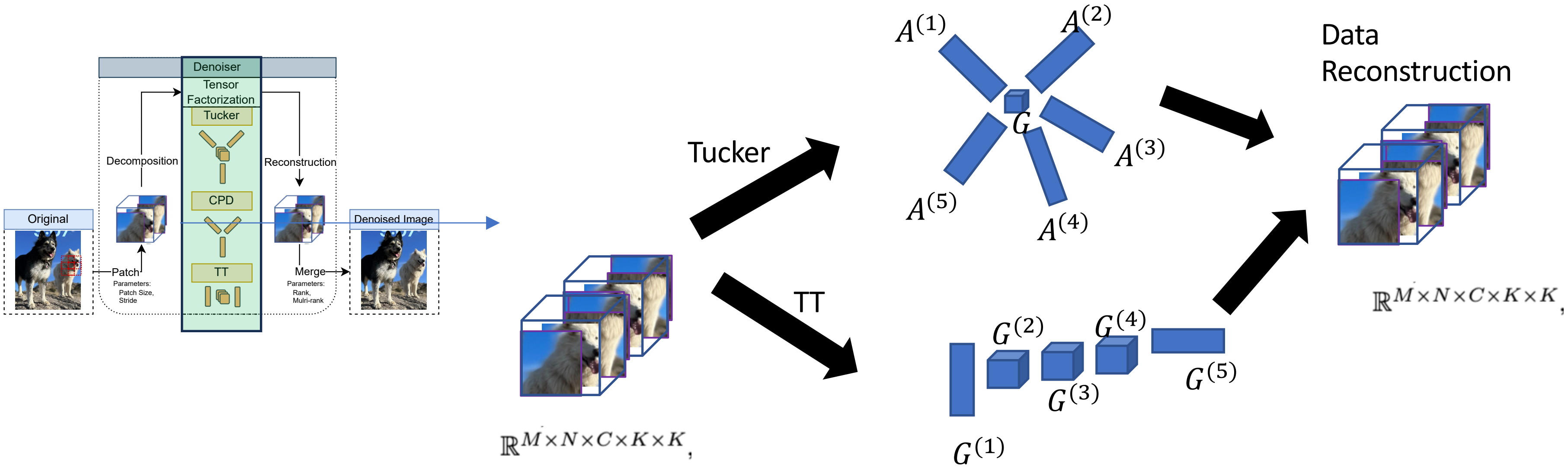
Hyperparameters: **patch size(P) and stride (S)**

Output tensor size function of above hyperparameters

Smaller patches/ smaller stride lead to increased patch counts and increased computation

2. Factorization module (filtering adversarial components)

$$O \approx \mathcal{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 A^{(3)} \times_4 A^{(4)} \times_5 A^{(5)}$$



Philosophy: Reconstruction based on well designed Low rank approximation gets rid of higher Order signals/noise

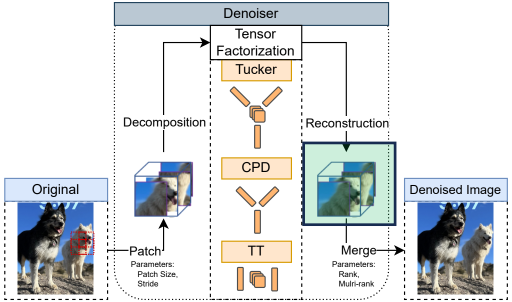
Hyperparameters: **Multi rank $(r_1, r_2, r_3, r_4, r_5)$**

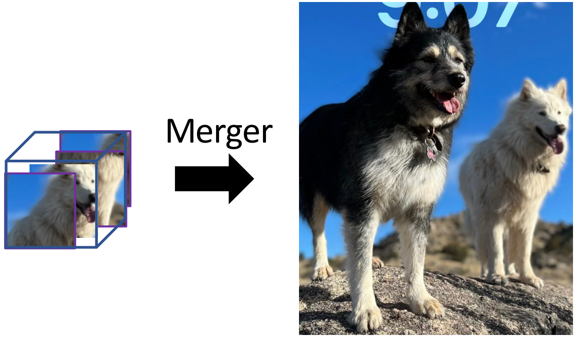
$$O(i_1, i_2, i_3, i_4, i_5) \approx \sum_{r_1, r_2, r_3, r_4} G^{(1)}_{i_1, r_1} G^{(2)}_{r_1, i_2, r_2} G^{(3)}_{r_2, i_3, r_3} G^{(4)}_{r_3, i_4, r_4} G^{(5)}_{r_4, i_5}:$$

Choosing smaller ranks lead to blurring effects and high rank enables reconstruction of attacks

3. Merger module (reconstruct image from patches)

- Given denoised adversarial Patches, reconstruct image
- Average vs max heuristics for overlapping blocks.

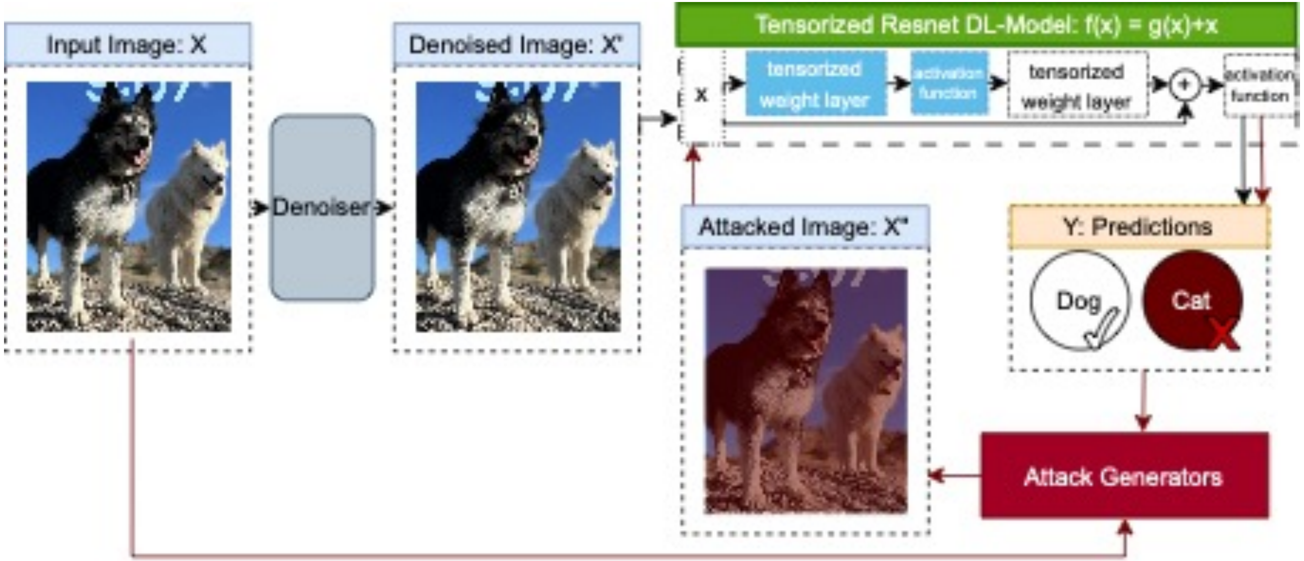




$$I_{c,w,h} = \frac{1}{C_{w,h}} \sum_{k_1=0}^{K} \sum_{k_2=0}^{K} O_{\frac{w-D\cdot k_1}{S},\frac{h-D\cdot k_2}{S},c,k_1,k_2} \cdot \mathbb{I}_{S|(w-D\cdot k_1),S|(h-D\cdot k_2)} \quad \forall w \in [0,W], h \in [0,H],$$

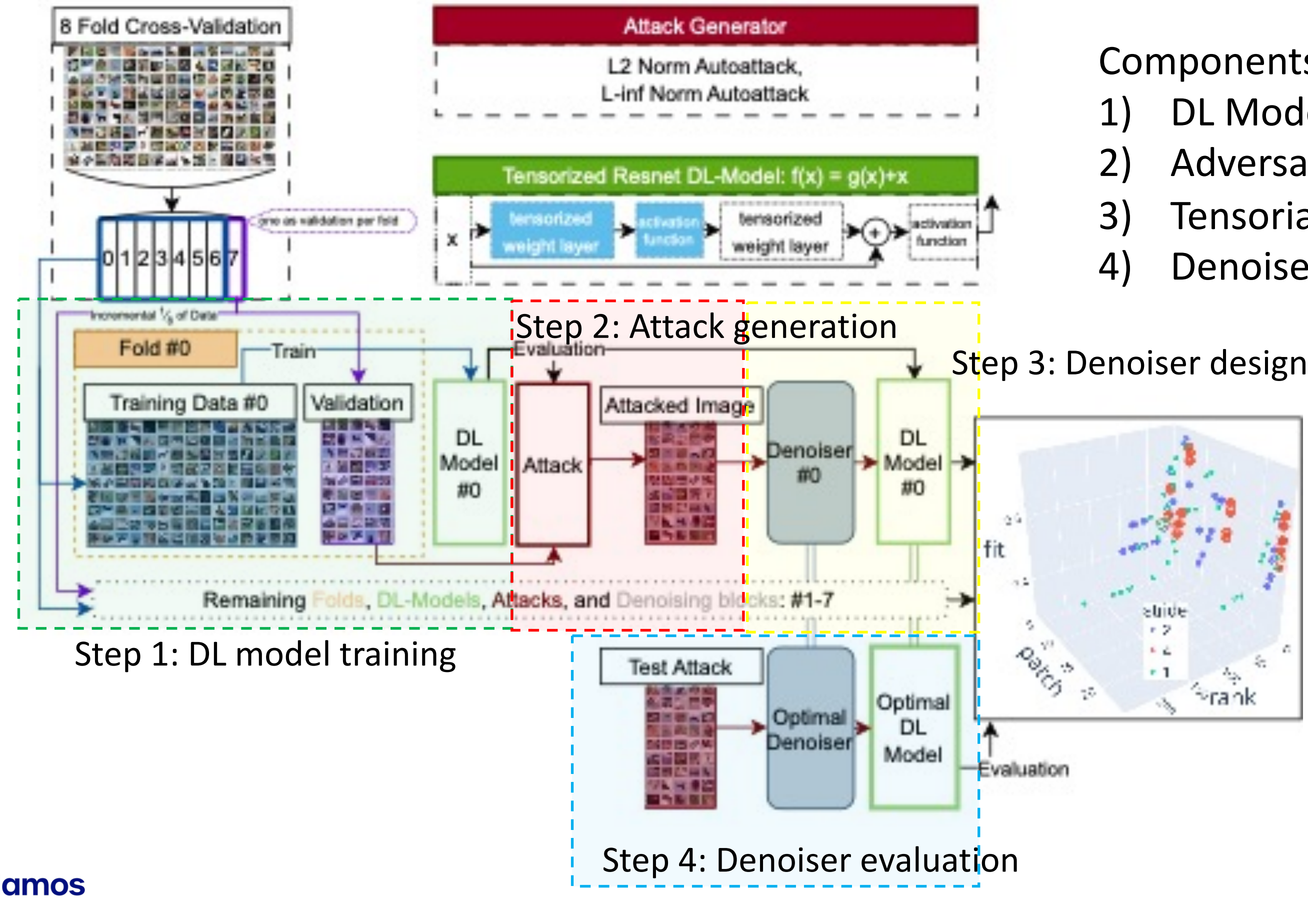
$$C_{w,h} = \sum_{k_1=0}^{K} \sum_{k_2=0}^{K} \mathbb{I}_{S|(w-D\cdot k_1),S|(h-D\cdot k_2)} \quad \forall w \in [0,W], h \in [0,H].$$

Overview of the attack generation



Generate Adversarial attack with AutoAttack framework with $\epsilon = 8/255$ for l_{inf} norm and $\epsilon = 0.5$ for l_2 norm.

Overall pipeline



Step 1: DL model training

Step 2: Attack generation

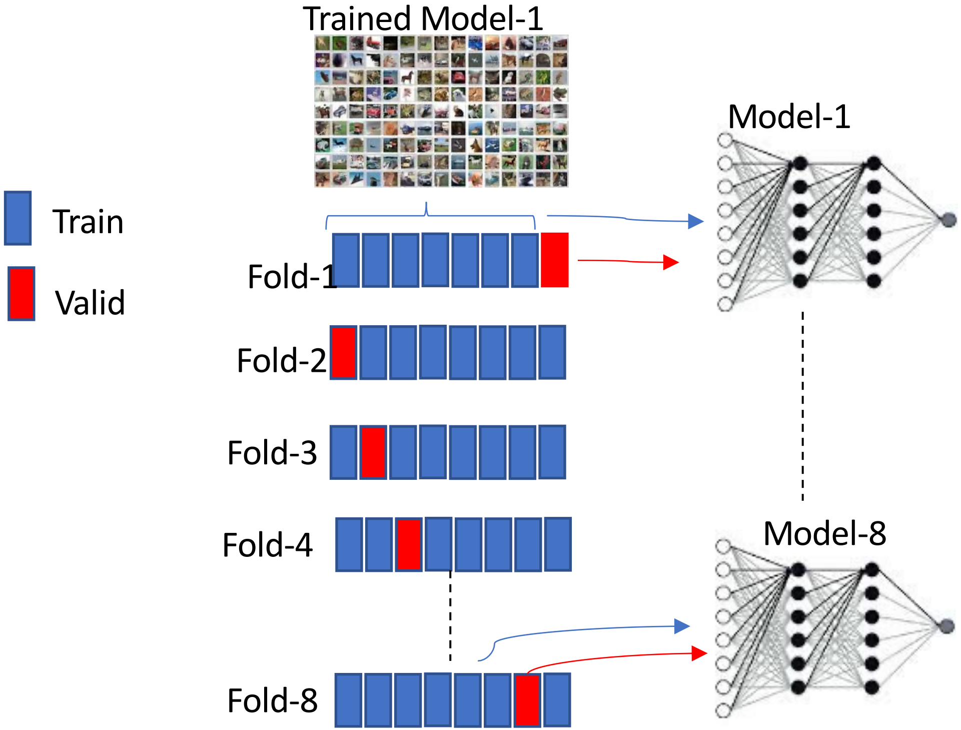
Step 3: Denoiser design

Step 4: Denoiser evaluation

Components of Pipeline

1) DL Model Training
2) Adversarial Attack generation
3) Tensorial Denoiser design
4) Denoiser Evaluation

Step 1. Training DL models



Trained Model-1

🟦 Train

🟥 Valid

Fold-1

Fold-2

Fold-3

Fold-4

Fold-8

Model-1

Model-8

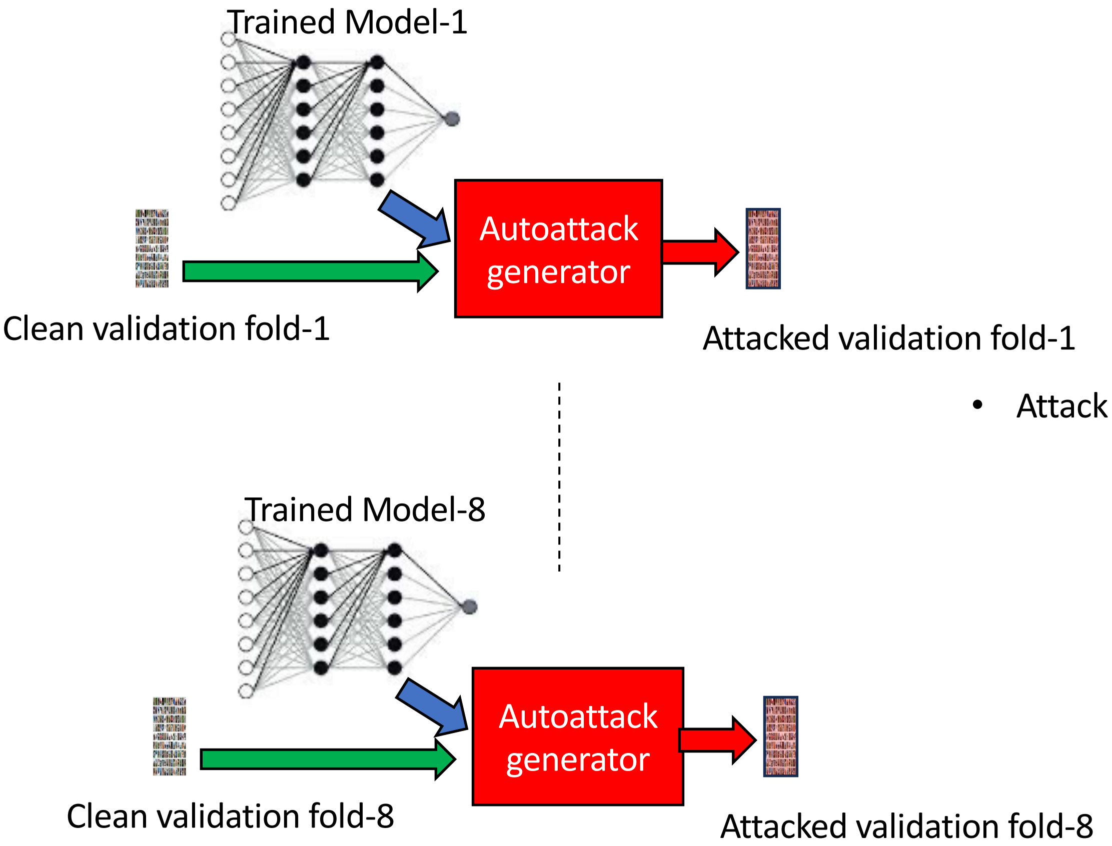
- Train DL models on 8 different training folds
- Eight different models corresponding to eight

Different training sets

- Model hyperparameter tuning with validation set.

- **Datasets**: CIFAR10, CIFAR100, Imagenet
- **Models**: Resnet 18, Resnet24

Step 2. Generate attack datasets



Trained Model-1

Clean validation fold-1

Autoattack generator

Attacked validation fold-1

Trained Model-8

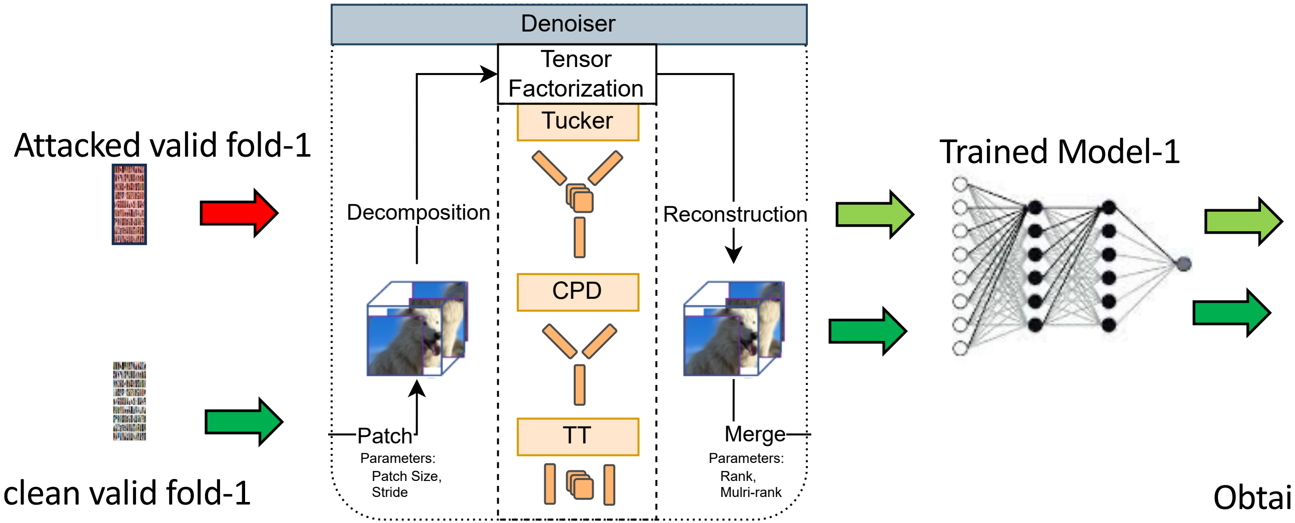
Clean validation fold-8

Autoattack generator

Attacked validation fold-8

- Attack generated on validation dataset

3. Hyperparameter estimation for denoiser



Attacked valid fold-1

clean valid fold-1

Denoiser

Tensor Factorization

Tucker

CPD

TT

Decomposition

Patch

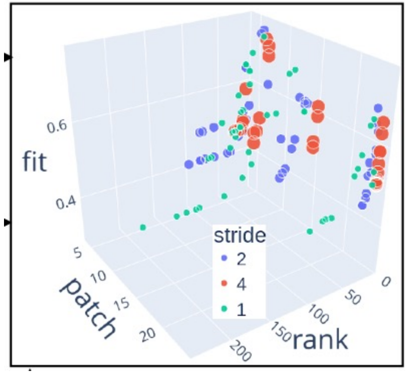
Parameters:
Patch Size,
Stride

Reconstruction

Merge

Parameters:
Rank,
Mulri-rank

Trained Model-1



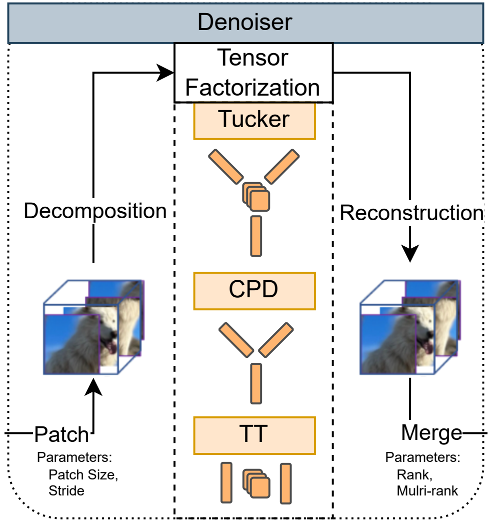
Obtain parameters for max
Fit score
(adversarial accuracy+clean accuracy)/2

Hyperparameters: patch size(P), stride (S), multi-ranks$(r_1, r_2, r_3, r_4, r_5)$

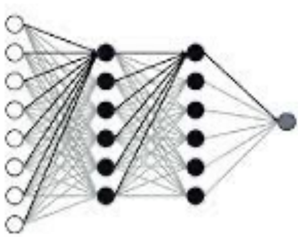
Repeat the experiment for rest of folds

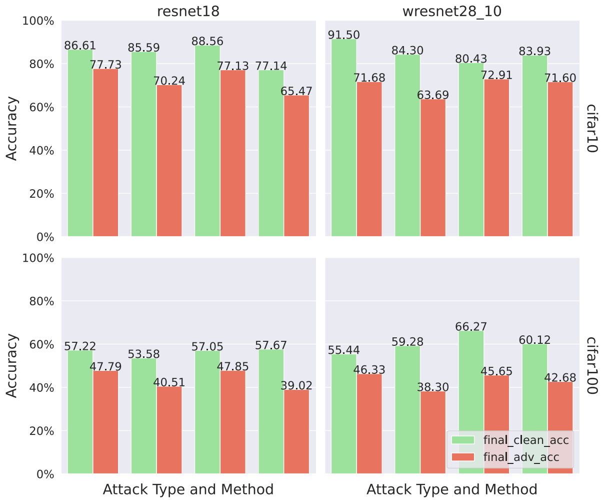
4. Optimal denoiser based purification

Attacked test set



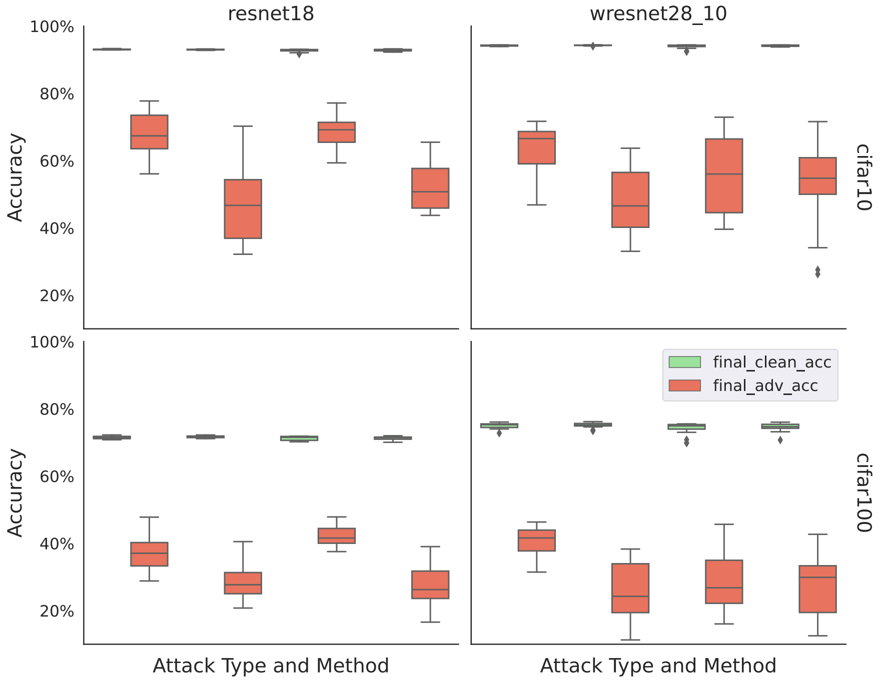
Averaged NN



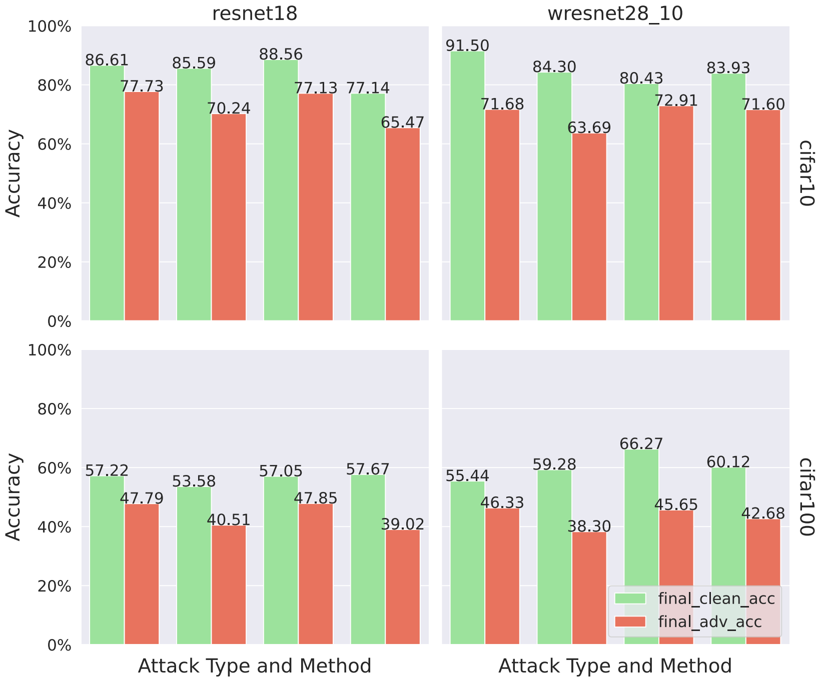


Optimal denoiser block corresponding to optimal hyperparameters

Results

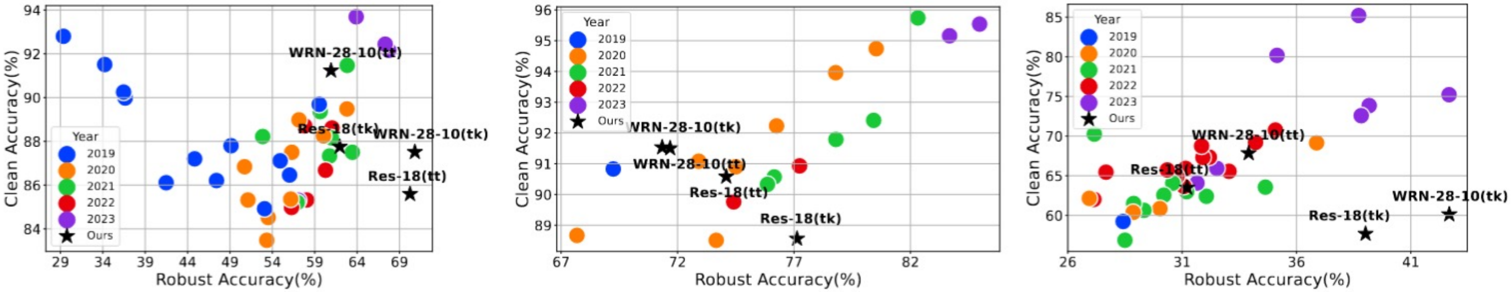


Distribution of clean and adversarial accuracy scores achieved for top 10 denoiser hyperparameter configurations for test dataset



Statistical representation of adversarial metrics corresponding to the optimal hyperparameter configuration that maximizes the average of clean and adversarial accuracy.

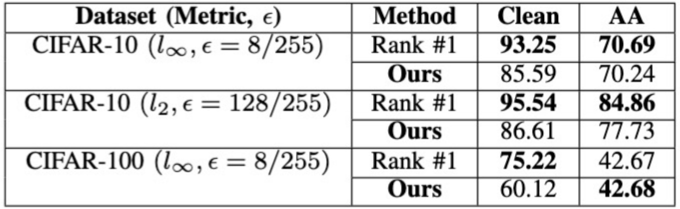
Where do we compare to state of the art?



(a) CIFAR-10(l_∞,$\epsilon = \frac{8}{255}$)

(b) CIFAR-10(l_2,$\epsilon = \frac{128}{255}$)

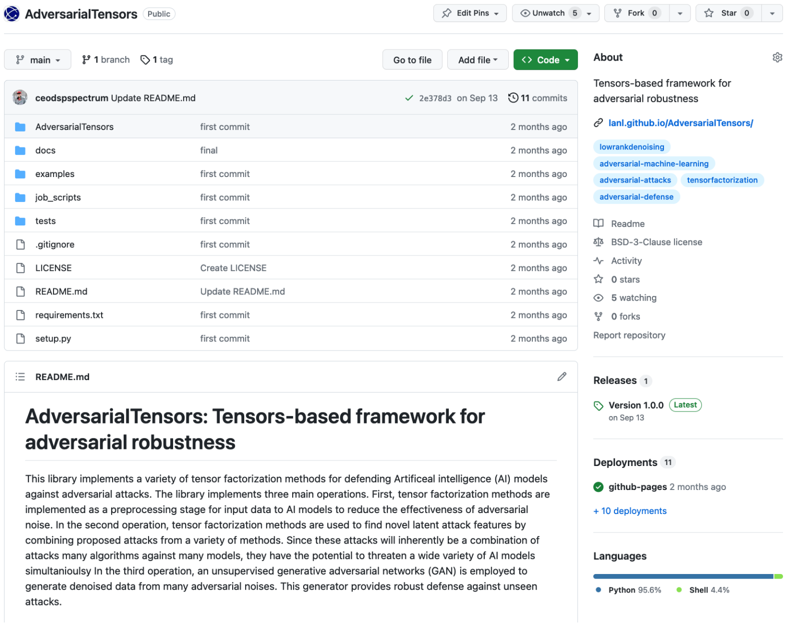
(c) CIFAR-100(l_∞,$\epsilon = \frac{8}{255}$)



Dataset (Metric, ϵ)	Method	Clean	AA
CIFAR-10 ($l_\infty, \epsilon = 8/255$)	Rank #1	**93.25**	**70.69**
	Ours	85.59	70.24
CIFAR-10 ($l_2, \epsilon = 128/255$)	Rank #1	**95.54**	**84.86**
	Ours	86.61	77.73
CIFAR-100 ($l_\infty, \epsilon = 8/255$)	Rank #1	**75.22**	42.67
	Ours	60.12	**42.68**

TABLE I: Comparison of test accuracy(%) from our tensorial denoiser to the state-of-the-art model, as in RobustBench [4].

AdversarialTensors



ceodspspectrum Update README.md ✓ 2e378d3 on Sep 13 11 commits

AdversarialTensors	first commit	2 months ago
docs	final	2 months ago
examples	first commit	2 months ago
job_scripts	first commit	2 months ago
tests	first commit	2 months ago
.gitignore	first commit	2 months ago
LICENSE	Create LICENSE	2 months ago
README.md	Update README.md	2 months ago
requirements.txt	first commit	2 months ago
setup.py	first commit	2 months ago

About

Tensors-based framework for adversarial robustness

lanl.github.io/AdversarialTensors/

lowrankdenoising
adversarial-machine-learning
adversarial-attacks
tensorfactorization
adversarial-defense

Readme
BSD-3-Clause license
Activity
0 stars
5 watching
0 forks
Report repository

Releases 1

Version 1.0.0 Latest
on Sep 13

Deployments 11

github-pages 2 months ago
+ 10 deployments

Languages

Python 95.6% Shell 4.4%

README.md

AdversarialTensors: Tensors-based framework for adversarial robustness

This library implements a variety of tensor factorization methods for defending Artificeal intelligence (AI) models against adversarial attacks. The library implements three main operations. First, tensor factorization methods are implemented as a preprocessing stage for input data to AI models to reduce the effectiveness of adversarial noise. In the second operation, tensor factorization methods are used to find novel latent attack features by combining proposed attacks from a variety of methods. Since these attacks will inherently be a combination of attacks many algorithms against many models, they have the potential to threaten a wide variety of AI models simultanioulsy In the third operation, an unsupervised generative adversarial networks (GAN) is employed to generate denoised data from many adversarial noises. This generator provides robust defense against unseen attacks.

- **Highly Modular Design**
  - Standalone functions and scripts
- **Training Large-Scale Deep Learning Models**
  - Efficient processing for large-scale tasks
- **Generating Large-Scale Attacks**
  - Capabilities for extensive adversarial attacks
- **Performing Adversarial Defense**
  - Batch processing for defense mechanisms
- **Utilizing Ray Tune**
  - Distributed, multi-node, and multi-GPU performance optimization
- **Step-by-Step Execution Guide**
  - Detailed instructions for pipeline execution
- **Comprehensive Documentation**
  - Exhaustive and self-explanatory for ease of use

Open for Questions