# Dependable AI (CSL7370)
# Adversarial Robustness

## TEAM DESCRIPTION

**Rishav Aich** (B21AI029)
*Pre-final Year (B.Tech. Artificial Intelligence & Data Science)*

**Tanish Pagaria** (B21AI040)
*Pre-final Year (B.Tech. Artificial Intelligence & Data Science)*

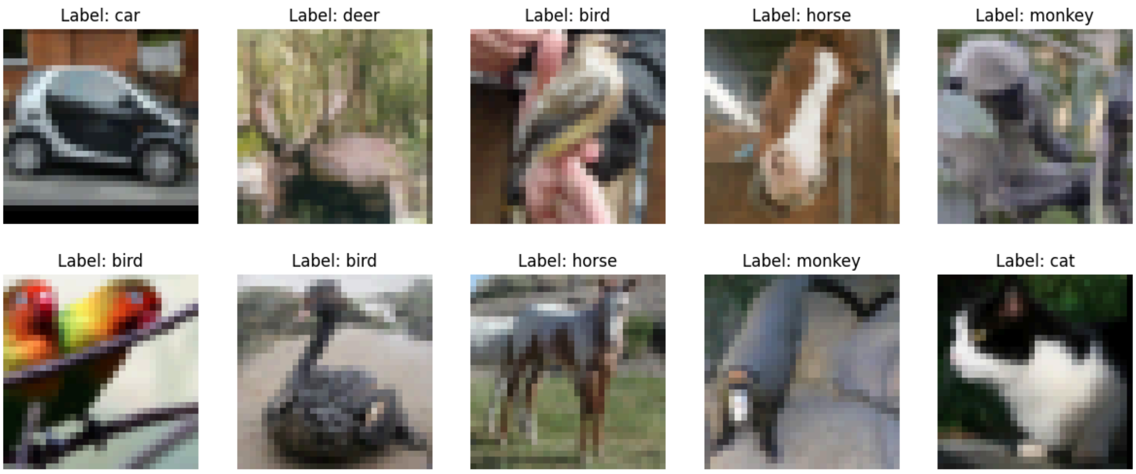
*IIT Jodhpur Undergraduates*

## OVERVIEW

This project aims to train a deep neural network such that it achieves robustness against adversarial attacks such as PGD, FGSM and Deepfool.

## DATASET

**STL-10** dataset was used for the project. It is an image recognition dataset (96x96 pixel colored images, which were resized to **32x32 pixels**) for developing unsupervised feature learning, deep learning, and self-taught learning algorithms. It had 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. It had 500 training images (10 pre-defined folds) and 800 test images per class.




*random image samples from the dataset*

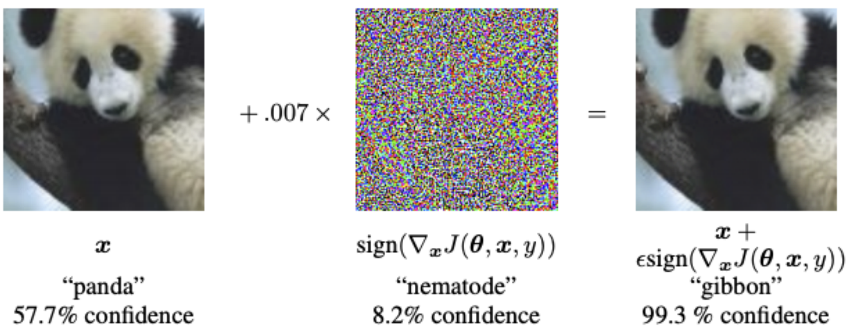
## ADVERSERIAL ATTACK IMPLEMENTATION

Adversarial attack is a malicious attempt to perturb a data point $x_0 \in \mathbf{R}^d$, belonging to class $C_i$, to another point $x \in \mathbf{R}^d$ such that $x$ belongs to a certain **target** adversarial class.
It is a mapping $\boldsymbol{A}$: $\mathbf{R}^d \rightarrow \mathbf{R}^d$ such that the perturbed data $x = \boldsymbol{A}(x_0)$ is misclassified as $\mathrm{C_t}$.

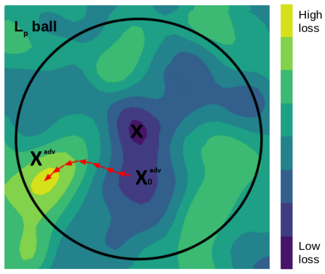
### Fast Gradient Sign Method (FGSM)

It works by using the gradients of the neural network to create an adversarial example. For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximizes the loss. This new image is called the adversarial image.
Given a loss function $J(x;\, w)$, the FGSM creates an attack $x$ by:

$$\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot sign(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w}))$$



### Projected Gradient Descent (PGD)




It is an iterative approach that aims to find the perturbation that maximizes the loss while staying within a specified ($\epsilon$) distance.

$$x_{adv}^{(t+1)} = \text{Clip}_{x_0,\epsilon}(x_{adv}^{(t)} + \alpha \cdot \text{sign}(\nabla_x J(x_{adv}^{(t)}; w)))$$

Here $x_{adv}$ is the adversarial example, t is the iteration index, α is the step size, and Clip( $\cdot$ ) ensures that the adversarial example remains within the $\epsilon$-ball around $x_0$.

## DeepFool

For an input $x$, the algorithm finds the closest hyperplane (straight plane diving one class from others) and projects $x$ on that hyperplane, displaces it slightly beyond, thereby causing a minimal perturbation, resulting in misclassification.

### Algorithm

**input:** Image $\boldsymbol{x}$, classifier $f$.
**output:** Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
**while** $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  **for** $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  **end for**
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
**end while**
**return** $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

## MODEL ARCHITECTURE

**ResNet18** model architecture was employed for the image classification task. It is a state-of-the-art image classification model structured as an 18-layer convolutional neural network. It takes residuals from each layer and uses them in the subsequent connected layers.

## TRAINING ROBUST MODEL

Parametric Noise Injection (PNI) is a technique to improve the robustness of deep neural networks against adversarial attacks.

### Intuition

For each iteration of network inference, the noise sampled from the Gaussian distributed noise source is injected upon weight (input/activation) in a layer-wise fashion. Such a Gaussian noise source is trained with the aid of adversarial training (i.e., min-max optimization). The intuition that the optimizer will find a moderate noise level is:

- If the noise magnitude is too large, it will introduce too much randomness into the network inference path, thus significantly lowering the inference accuracy.
- If the noise magnitude is too small, the regularization functionality of noise injection is not performed.

### Implementation

Noise was added to the weights of the convolutional layers in the ResNet18 architecture as per:

$$\tilde{v}_i = f_{\text{PNI}}(v_i) = v_i + \alpha_i \cdot \eta; \quad \eta \sim \mathcal{N}(0, \sigma^2)$$

Here, $\boldsymbol{\alpha}$ is also a trainable parameter in this case, which acts as the scaling factor.

The model was trained with Adam as the optimizer to minimize the ensemble loss, which is the weighted sum of losses for clean and perturbed data.

$$\mathcal{L}' = w_c \cdot \mathcal{L}(g(\boldsymbol{x}; f_{\text{PNI}}(\boldsymbol{\theta})), t) + w_a \cdot \mathcal{L}(g(\hat{\boldsymbol{x}}; f_{\text{PNI}}(\boldsymbol{\theta})), t)$$

Here, cross-entropy loss was used for $L$.

## EXPERIMENT OBSERVATIONS & RESULTS

The standard (original) model and the robust (parametric noise injected) model were trained on the dataset.


*training and validation curves for the **standard** model*


*training and validation curves for the **robust** model*

The three adversarial attacks were performed on the dataset with respect to the standard model as well as the robust (parametric noise injected) model.

- **Standard Model**

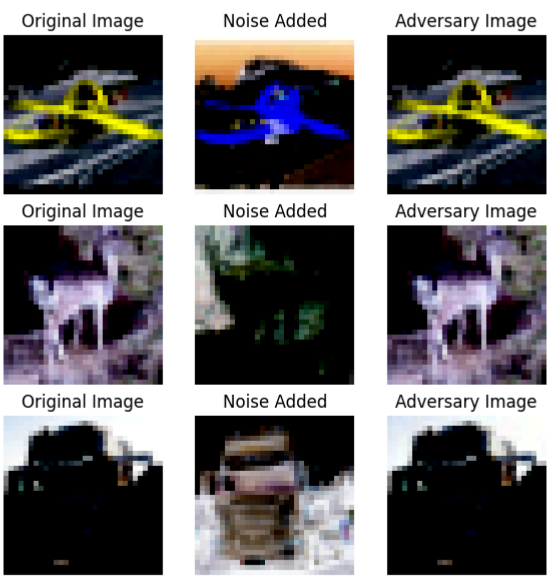
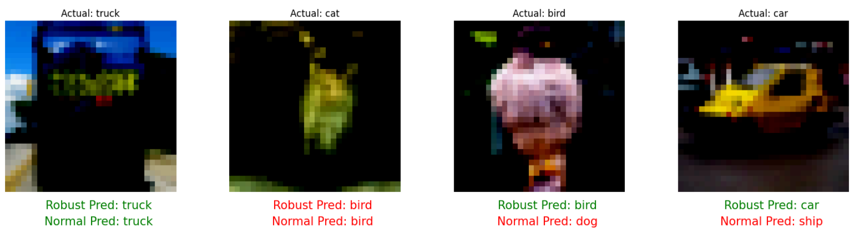
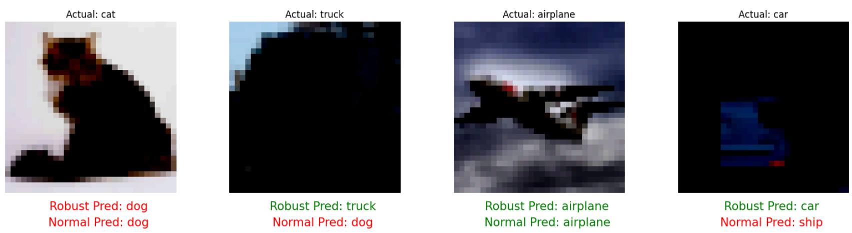


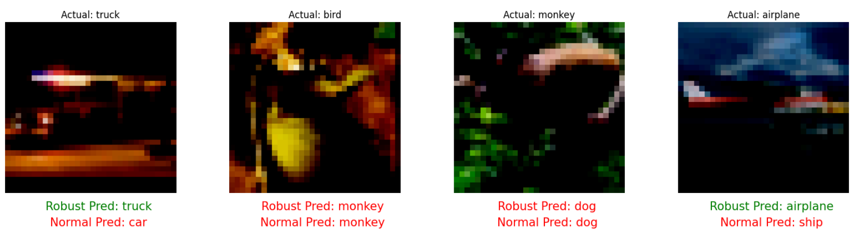

*image perturbations*
*(generated by adversarial attacks)*




*FGSM Adversarial Attack*



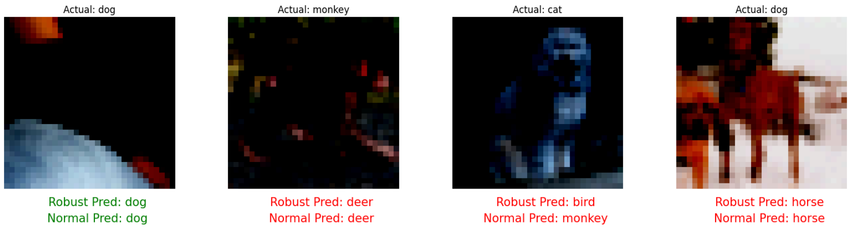

*PGD Adversarial Attack*



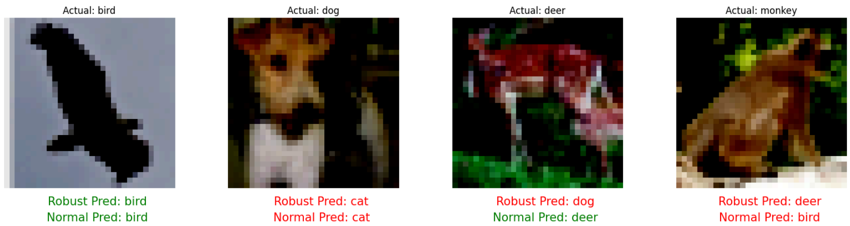

*DeepFool Adversarial Attack*

- **Robust Model**


*image perturbations*
*(generated by adversarial attacks)*




*FGSM Adversarial Attack*



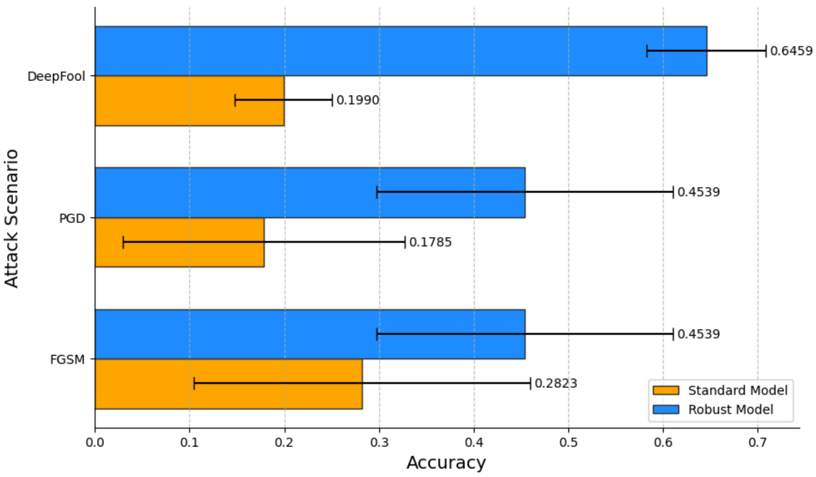

*PGD Adversarial Attack*


*DeepFool Adversarial Attack*

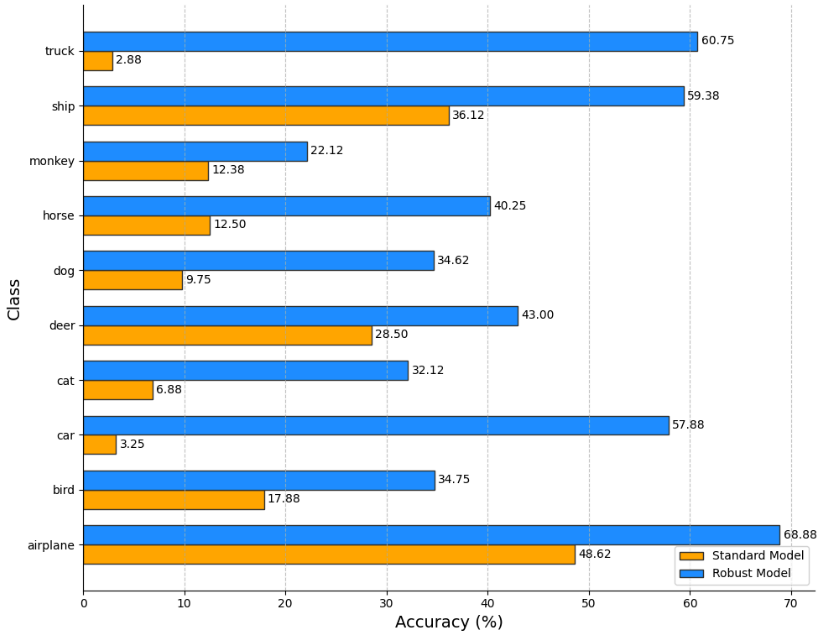
The accuracy scores for the standard and the robust classifiers are shown below:

| Attack | Model | Mean | Std | Accuracy |
|---|---|---|---|---|
| **FGSM** | Standard | 0.28225 | 0.177809 | 0.2822 ± 0.1778 |
| | Robust | 0.453875 | 0.156598 | 0.4539 ± 0.1566 |
| **PGD** | Standard | 0.1785 | 0.148879 | 0.1785 ± 0.1489 |
| | Robust | 0.453875 | 0.156473 | 0.4539 ± 0.1565 |
| **Deep Fool** | Standard | 0.199 | 0.0515 | 0.199 ± 0.0515 |
| | Robust | 0.6459 | 0.0629 | 0.6459 ± 0.0629 |

The plots showing the overall and classwise accuracy scores of the standard classifier and the robust classifier under different attacks are shown below:




***accuracy scores** for the standard and the robust model under different attacks*




***class-wise accuracy scores*** *for the standard and the robust model under adversarial attack*

**Observations**
It could be easily seen that the robust model performed significantly better than the standard (original) model under different adversarial attacks at both the overall and the classwise levels.

## REFERENCES


PGD
https://towardsdatascience.com/know-your-enemy-7f7c5038bdf3
https://arxiv.org/abs/2402.09154

FGSM
https://medium.com/@madadihosyn99/generating-adversarial-examples-with-fast-gradient-sign-method-fgsm-in-pytorch-a-step-by-step-a423537628dd
https://www.tensorflow.org/tutorials/generative/adversarial_fgsm
https://medium.com/@zachariaharungeorge/a-deep-dive-into-the-fast-gradient-sign-method-611826e34865
https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

DeepFool
https://medium.com/@aminul.huq11/pytorch-implementation-of-deepfool-53e889486ed4
https://towardsdatascience.com/deepfool-a-simple-and-accurate-method-to-fool-deep-neural-networks-17e0d0910ac0
https://ieeexplore.ieee.org/document/7780651

Adversarial Training
https://adversarial-ml-tutorial.org/adversarial_training/
https://adversarial-ml-tutorial.org/adversarial_examples/