

Basic Attack Methods (15 points each)

For each of the Fast Gradient Sign Method (FGSM), Iterative Gradient Sign, and Least Likely Class methods:

(i) Briefly explain your implementation, including which pre-trained model you chose to attack and any hyperparameter choices

- I used the resnet18 pre-trained network and changes the last fc layer as per imagenette dataset (train images: 9427 and test images: 3911).

Model training parameters: Adam optimizer with initial learning rate of 0.001, num_epochs=10, initial learning rate of 0.001 with decay after every 5 epochs, batch size = 128

Accuracy of the network on the test images: 98.49 %

- **Fast Gradient Sign Method (FGSM):**

For FGSM, I chose the above-trained model as the target model. In this method, I calculated the gradient of the loss function with respect to the input image and then perturbed the input image in the direction of the gradient sign to generate an adversarial example. The hyperparameter chose for this attack:

- epsilon value: [0,2/255,4/255,8/255,16/255,32/255], to control the magnitude of the perturbation applied to the input image

Epsilon: 0 Test Accuracy = 3852 / 3911 = 0.98

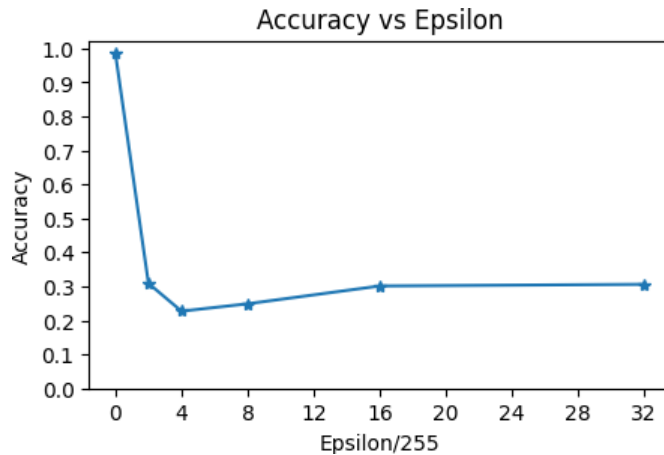
Epsilon: 2 Test Accuracy = 1208 / 3911 = 0.30

Epsilon: 4 Test Accuracy = 888 / 3911 = 0.22

Epsilon: 8 Test Accuracy = 973 / 3911 = 0.24

Epsilon: 16 Test Accuracy = 1178 / 3911 = 0.30

Epsilon: 32 Test Accuracy = 1197 / 3911 = 0.30



- **Iterative Gradient Sign Method / Basic Iterative Method (BIM)**

For the iterative gradient sign method, I also chose the above-mentioned trained model. The attack involved taking multiple small steps iteratively (iteration is hyperparameter) in the direction of the gradient sign, with the goal of maximizing the loss function. The hyperparameters chosen for this attack are the

- epsilon value: [0,2/255,4/255,8/255,16/255,32/255]
- step size i.e. alpha= 0.0004
- the number of iterations = min(eps + 1, 1.25*eps, 10) as per the Adversarial Examples In The Physical World paper and modified it a bit as per computational resources available.

Epsilon: 0 *Test Accuracy = 3852 / 3911 = 0.98*

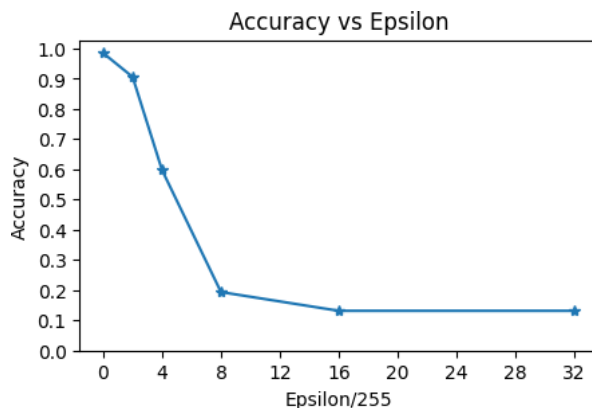
Epsilon: 2 *Test Accuracy = 3543 / 3911 = 0.90*

Epsilon: 4 *Test Accuracy = 2336 / 3911 = 0.59*

Epsilon: 8 *Test Accuracy = 757 / 3911 = 0.19*

Epsilon: 16 *Test Accuracy = 514 / 3911 = 0.13*

Epsilon: 32 *Test Accuracy = 514 / 3911 = 0.13*



- **Iterative Least-Likely Class Method**

I again used the same model. The attack involves first selecting the least likely class prediction for an input image and then generating an adversarial example by maximizing the loss function with respect to the least likely class. The hyperparameters chosen for this attack are the

- epsilon value: $[0, 2/255, 4/255, 8/255, 16/255, 32/255]$
- step size i.e. $\alpha = 0.004$
- the number of iterations = $\min(\text{eps} + 4, 1.25 * \text{eps}, 10)$ as per the Adversarial

Examples In The Physical World paper and modified it a bit as per computational resources available.

Epsilon: 0 *Test Accuracy* = $3852 / 3911 = 0.98$

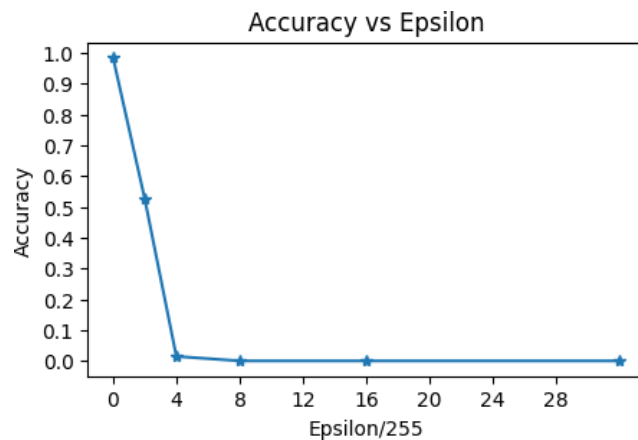
Epsilon: 2 *Test Accuracy* = $2061 / 3911 = 0.52$

Epsilon: 4 *Test Accuracy* = $57 / 3911 = 0.014$

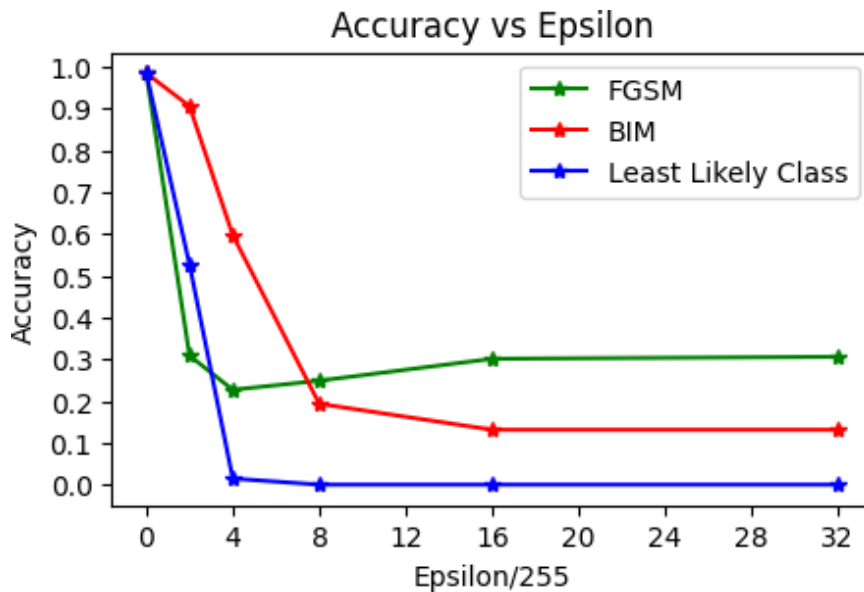
Epsilon: 8 *Test Accuracy* = $0 / 3911 = 0.0$

Epsilon: 16 *Test Accuracy* = $0 / 3911 = 0.0$

Epsilon: 32 *Test Accuracy* = $0 / 3911 = 0.0$



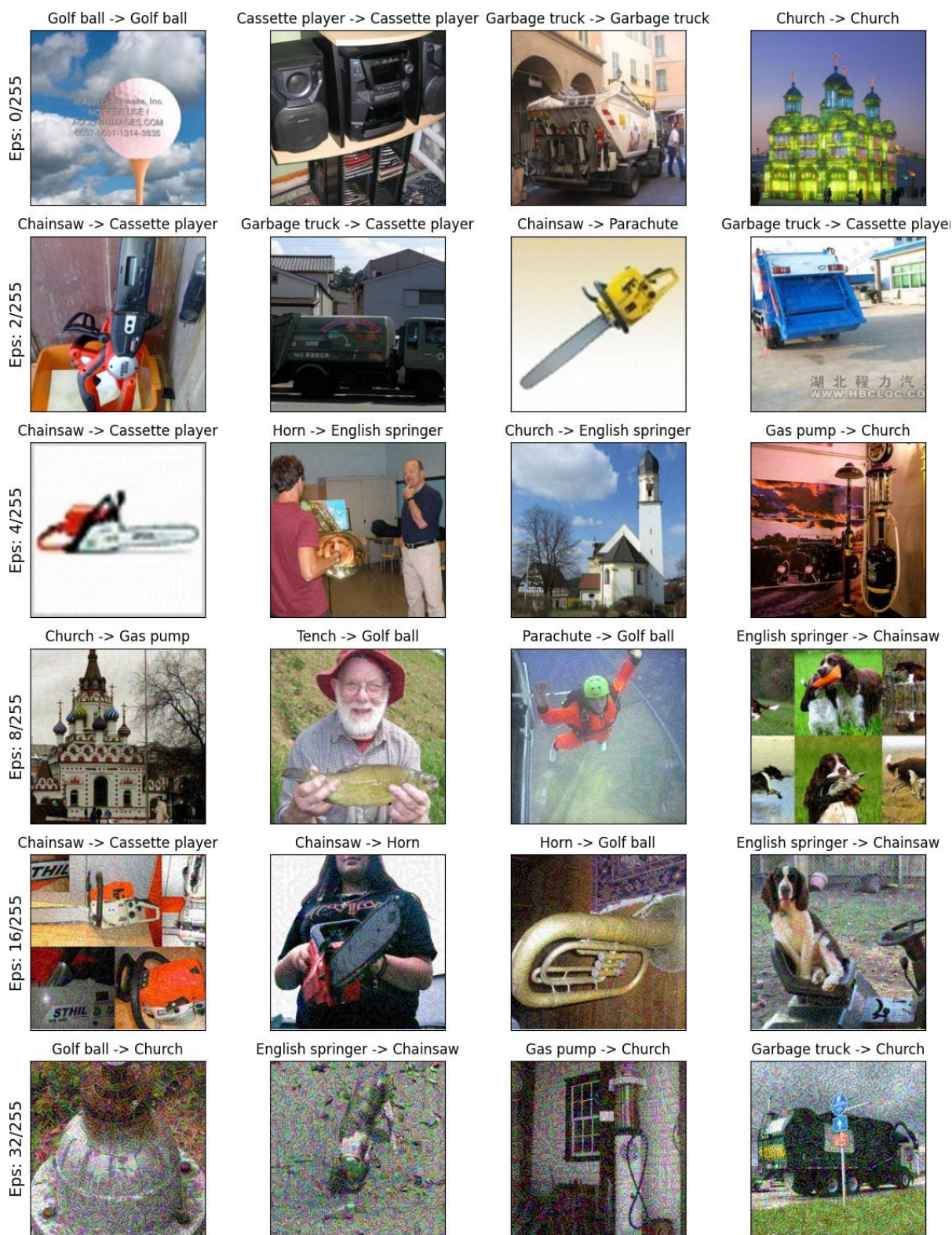
(ii) Plot accuracy vs. epsilon on the ImageNette. If you implement multiple methods, you should show them all on the same plot.



























(iii) Visualize examples of attacks, where the original image, original (correctly predicted) label, and the attacked image and falsely predicted label are shown for ImageNette or MNIST.

Here, are some examples of each epsilon value. Each row of the plot shows a different epsilon value. The first row is the $\epsilon=0$ examples which represent the original “clean” images with no perturbation. The title of each image shows the “original label -> adversarial classification label”.

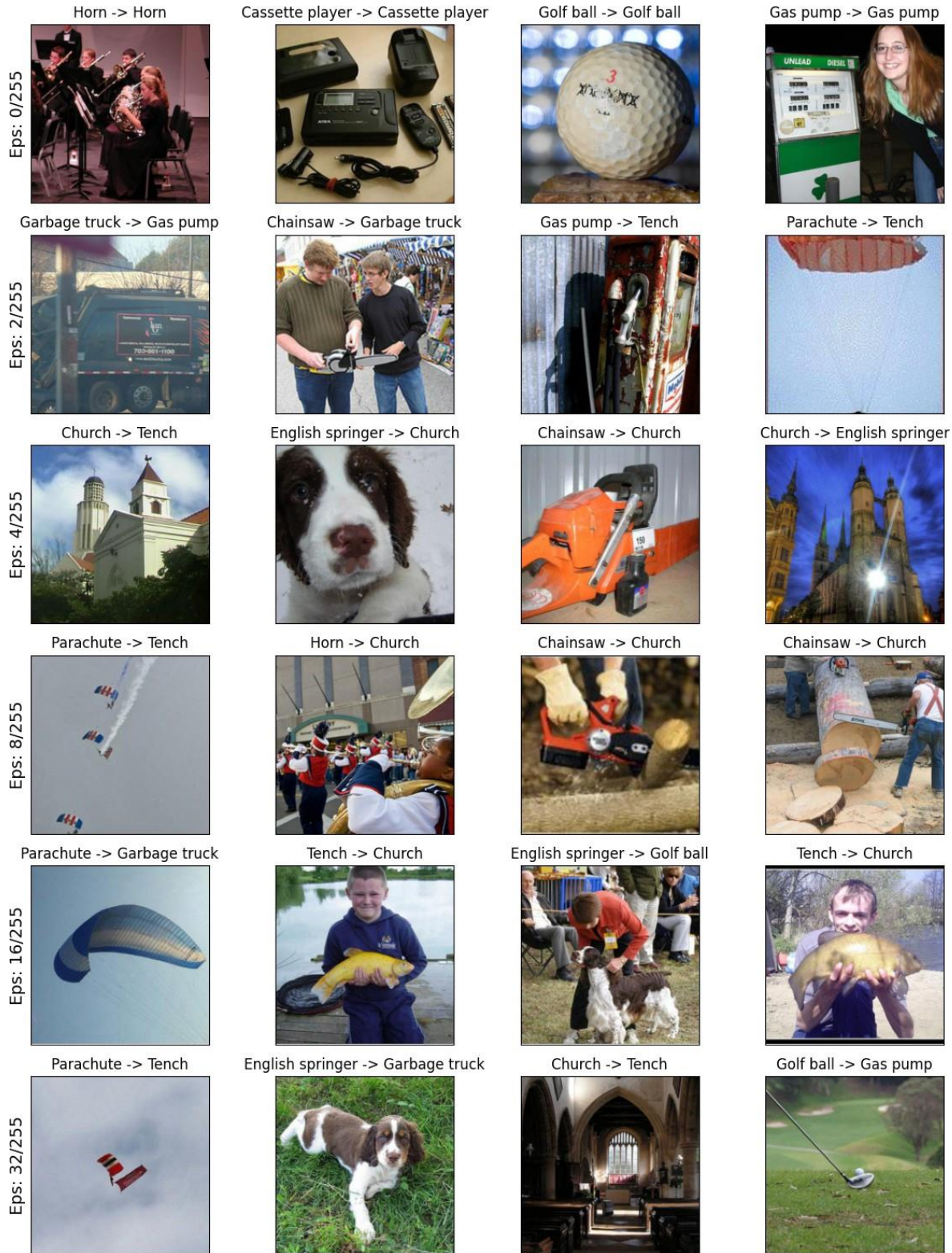
1. **Fast Gradient Sign Method (FGSM) examples:**



2. Iterative Gradient Sign Method / Basic Iterative Method (BIM) examples:

Eps: 0/255	Church -> Church 	Parachute -> Parachute 	Gas pump -> Gas pump 	Parachute -> Parachute 
	Tench -> English springer 	Chainsaw -> Garbage truck 	Garbage truck -> Chainsaw 	Garbage truck -> Parachute 
Eps: 2/255	Horn -> Gas pump 	Horn -> Chainsaw 	Tench -> Gas pump 	Garbage truck -> English springer 
	Tench -> Chainsaw 	Gas pump -> Chainsaw 	English springer -> Parachute 	Gas pump -> Parachute 
Eps: 4/255	Tench -> Cassette player 	English springer -> Cassette player 	Church -> English springer 	Horn -> Cassette player 
	Gas pump -> Cassette player 	Chainsaw -> English springer 	Horn -> Parachute 	Chainsaw -> Garbage truck 
Eps: 8/255				
Eps: 16/255				
Eps: 32/255				

3. Iterative Least-Likely Class Method (ILLCM)



Finally, if you implemented multiple methods, discuss any differences between them in terms of efficacy, visibility of perturbation, computational efficiency, etc.

The FGSM method reduces the accuracy to 30% even with the smallest epsilon of $2/255$. However, as we increase the epsilon, the accuracy of the adversarial images generated by this method remains relatively constant between 25-30% until the epsilon reaches $32/255$. The image quality is only slightly perturbed from the original for epsilon values between $2/255$ to $8/255$, but it starts to show visible perturbation beyond that point. Compared to the other two methods, FGSM is the fastest, but may not be the most effective for achieving high attack success rates.

In contrast, the Basic Iterative Method uses finer perturbations that do not significantly affect the image (perturbation not visible significantly) even at higher epsilon values but still results in a drop in accuracy to 60% for $\epsilon=4/255$ and below 20% after $\epsilon=8/255$. However, this method is highly sensitive to the number of iterations and the alpha value. Increasing either of these leads to a drastic drop in accuracy, even for small epsilon values. Moreover, this method is computationally expensive since it requires multiple iterations and gradient calculations.

The iteratively least likely class method (ILLCM) destroys the correct classification of most images even when the epsilon is relatively small. However, it does not have any visible perturbation on images and also confuses the model by misclassification. Like Basic Iterative Method, this method is also highly sensitive to the value of alpha and the number of iterations. As we increase the number of iterations or increase the alpha, the accuracy starts to drop drastically even for small epsilon. It is a computationally expensive method as it is run for multiple iterations and also needs to calculate the gradient multiple times.

BIM and ILLCM achieve high attack success rates. ILLCM is the most effective.