Name: **Zahra Soukhtedel** Assignment Number: **5**
Student Id: **98105138** Course: **Deep Learning**

Problem 3

We know that if the hostile attack is untargeted, for the data and label (x, y) for the given bundle h with the cost function $l(h(x), y)$, the objective function that we write for the attack is as follows

$$max_{||\delta|| \leq \epsilon} l(h(x), y)$$

In the FGSM attack, the following rule is used to reach the optimal point of this function:

$$x' = x + \epsilon.sign(\nabla_x l(h(x), y))$$

If we want a targeted attack and the hostile data is stored in the y' category instead of the y category, write the objective function for the hostile attack and state how the new data will be obtained with the FGSM algorithm approach.

3.1

In a targeted attack, the goal is to mislead the model into classifying the input data into a specific target class, denoted as y'. The objective function for a targeted attack would be to minimize the loss between the model's prediction for the perturbed input and the target class. This can be written as:

$$min_{||\delta|| \leq \epsilon} l(h(x + \delta), y')$$

Here, we're trying to find the smallest perturbation (within the constraint) that will cause the model to classify the input x as the target class y'. In the Fast Gradient Sign Method (FGSM) approach, the rule to reach the optimal point of this function would be slightly modified. Instead of adding the perturbation, we subtract it. This is because we're trying to minimize the loss, not maximize it. The rule becomes:

$$x' = x - \epsilon.sign(\nabla_x l(h(x), y'))$$

This equation will generate the adversarial example x' that we hope will be classified as the target class y' by the model. The sign of the gradient provides the direction in which we need to change x to minimize the loss, and ϵ controls the magnitude of this change.

3.2

Let's show that if the predictive model is linear, the FGSM attack will be the best attack. Consider a linear model with parameters W and b, and let the cost function be $l(h(x), y)$. The prediction of the linear model can be written as:

$$h(x) = Wx + b$$

Now, let's compute the gradient of the cost function with respect to the input data x:

$$\nabla_x l(h(x), y) = \nabla_x l(Wx + b, y)$$

For a linear model, the gradient of the cost function with respect to the input data is a linear function of the input data. This means that the gradient will be a constant vector multiplied by a scalar (the input data x).
In the FGSM attack, we perturb the input data by adding a small perturbation ϵ in the direction of the sign of the gradient. Since the gradient is a linear function of the input data, the perturbation is also linear. This means the perturbed input data will still lie on the same linear subspace as the original data.
When the predictive model is linear, the decision boundaries are also linear. Therefore, perturbing the input data in a linear direction will have the maximum effect on changing the output of the model. This is because the perturbation is aligned with the linear decision boundaries, allowing the adversarial example to cross the boundary and change the predicted class.
In conclusion, when the predictive model is linear, the FGSM attack will be the best attack because it perturbs the input data in the direction that has the maximum effect on changing the model's output.

3.3

One of the main limitations of the FGSM attack is that it is a one-step attack that does not consider the model's decision boundary beyond the immediate vicinity of the input. This can lead to suboptimal attacks, especially for non-linear models where the decision boundary can be complex. The FGSM attack might not find the most effective perturbation that leads to a misclassification. This limitation is addressed by iterative methods like the Projected Gradient Descent (PGD) attack, which applies the FGSM attack in multiple steps and adjusts the perturbation based on the model's response.

4

4.1

In terms of robustness, the Projected Gradient Descent (PGD) algorithm outperforms the Fast Gradient Sign Method (FGSM). FGSM is a single-step method that moves in the direction of the gradient, while PGD is an iterative process that takes multiple steps, enabling it to discover more potent adversarial examples. While FGSM approximates the model's loss surface linearly, PGD explores the loss surface in greater depth, making it more successful at identifying adversarial examples that have a significant impact on the model's performance.

4.2

Yes,Adversarial attacks can utilize metrics beyond the common l_∞ metric, such as l_1, l_0, and l_2 metrics. Each of these metrics results in different characteristics for the adversarial examples produced: The l_0 norm computes the number of changes made to the original input. Adversarial examples generated using this metric tend to have a low number of modified features, but the changes can be significant. The l_1 norm calculates the sum of absolute values of the changes.

Adversarial examples generated with this metric may involve a higher number of modified features, but the changes are typically smaller in magnitude. The l_2 norm computes the sum of squares of the changes. Adversarial examples created using this metric have a balance between the number of modified features and the magnitude of the changes. The l_∞ norm considers the maximum absolute change. Adversarial examples generated utilizing this metric tend to display uniform changes across all features.

4.3

The disadvantage of using the l_2 norm compared to the l_∞ norm in the PGD algorithm is that the l_2 norm can lead to adversarial examples that are less perceptible to humans but more computationally expensive to generate. The l_2 norm considers the Euclidean distance, which can result in smaller, more spread out perturbations that are harder for humans to detect. However, these perturbations can be more computationally expensive to calculate compared to the l_∞ norm, which simply considers the maximum change in any feature. Therefore, while the l_2 norm may create more stealthy adversarial examples, it may not be as efficient as the l_∞ norm in terms of computational cost.