50.039 Theory and Practice of Deep Learning HW8
Loh De Rong (1003557)

The complete code and printouts can be found in the hw8.ipynb.

A. Copy and paste the code for your iugm_attack() function.

```
def iugm_attack(image, epsilon, model, original_label, iter_num = 10):

    # Skip if epsilon is 0
    if epsilon == 0:
        return image

    for i in range(iter_num):
        # Zero out previous gradients
        image.grad = None
        # Forward pass
        output = model(image)
        # Get the index of the min log-probability
        _, target_class_torch = torch.min(output.data, 1)
        # Calculate loss
        pred_loss = F.nll_loss(output, target_class_torch)

        # Do backward pass and retain graph
        pred_loss.backward(retain_graph = True)

        # Add noise to processed image
        eps_image = image - epsilon*image.grad.data
        eps_image.retain_grad()

        # Clipping eps_image to maintain pixel values into the [0, 1] range
        eps_image = torch.clamp(eps_image, 0, 1)

        # Forward pass
        new_output = model(eps_image)
        # Get prediction
        _, new_label = new_output.data.max(1)

        # Check if the new_label matches target, if so stop
        if new_label != original_label:
            break
        else:
            image = eps_image
            image.retain_grad()

    return eps_image
```

B. What do you observe on the accuracy vs. epsilon graph. Why are the two curves different? Is that something to be expected?

In both curves, the accuracy experiences a sharp drop initially, and then appears to stabilize when epsilon values get larger. The blue curve (iterated attack) is always below the red curve (one-shot attack), with an initial drop that is more significant. This difference is to be expected since the iterated attack is essentially an improvement of the one-shot attack.

Firstly, the initial sharp drop indicates that the addition of some small noise is effective in moving the samples toward the least probable targeted class through gradient descent on the logits (of the least probable targeted class), thereby misclassifying them.

Secondly, the iterated attack is able to repeat the attack until it reaches a maximal number of iterations or makes the model malfunction, whereas the one-shot attack only has 1 attempt. In particular, the iterated attack is able to continue adjusting the sample from the previous failed attempt to make the model predict the incorrect class.

Thirdly, the accuracy for the one-shot attack plateaus at 0.10 even with increasing values of epsilon. This is not surprising because the performance is rather close to random prediction, which will be about 10% accurate assuming there are 10 balanced classes in CIFAR10 dataset. On the other hand, the accuracy for the iterated attack plateaus at 0 even with increasing values of epsilon. This is expected too because with a larger epsilon and more retries allowed, it is almost guaranteed that an update will move the sample further away from its original class, and likely resulting in misclassification.

C. What seems to be the threshold for plausibility for both attacks (alpha and beta)? Is attack beta a better one? Why?

The threshold for plausibility for both attacks seem to be 0.02.

Yes, attack beta is a better one. With the same plausibility, attack beta is able to lower the model accuracy even more. For example, at epsilon = 0.02, the model accuracy is 0.445 for alpha, as compared to 0 for beta.

D. Plausibility seems problematic, even with the iterated version of the gradient attack. Can you suggest two possible ways to improve our attack strategy on this dataset and model?

1. Use the sign of the gradient instead to generate the attack sample. The range of epsilon values to be tested could be lowered to allow for greater plausibility without sacrificing efficacy. In this approach, the plausibility constraint $\|\tilde{x} - \mathrm{x}\|_\infty \leq \epsilon$ still holds.
2. Enforce a second plausibility constraint on L^0 norm between the generated and original image. This will bound on the total number of pixels that can be modified and only add noise to these pixels to create the attack sample.