CSCI 677: Advanced Computer Vision - Fall 2024

Instructor: Prof. Nevatia

Assignment 6

Due on November 26, 2024 before 15:00 PDT

Instructions

This is a programming assignment to experiment with adversary attacks on a classification network and defense by adversarial training, as studied in class.
You are asked to attack the Res-9 network that you constructed in HW4 for classifying CINIC-10 data into 10 different categories. We will consider a "white box" attack so the architecture and the parameters of the network are accessible to the attacker. In this assignment, we consider attacks as "untargeted": the prediction of the modified image is diverted from the original prediction made by the same model.

You are asked to implement the **Iterative Gradient Sign Method (IGSM)** attack. You may adjust the parameters of the attack; the goal is to achieve strong attack success with fewer changes but we are not searching for an optimum. We suggest to start with $\epsilon = 0.12$ and $\alpha = 0.006$ for IGSM, and the number of iterations to be 20 (note that the suggested numbers assume that the image values range between 0 and 1).
In addition to attack, we ask you to implement a simple adversarial training pipeline, which is similar to the training process that you have implemented in HW4. The key difference is that you replace some of the original training images, in a minibatch, with attack images generated by the designated attack. The replacement ratio is a hyper-parameter ranging between 0 and 1. By default you can set it at 0.75, meaning that 75% of training images are replaced with attack images. You can adjust it for variations in the experiments. For adversarial training, we recommend you to fine-tune your model for 10 epochs.

Implementation Hints: Following provides some hints for implementation. You are not required to follow these. You may also find related code online which you can use for guidance.

1) To find the gradients of the outputs y with respect to the inputs x.

```
    loss = loss_fn(outputs, y=y)
    loss.backward()
    x_grad = x.grad.detach()
```

2) To get only the sign you can use:

```
    x_grad_sign = x_grad.sign()
```

3) To make inputs containing the gradients, you would need to make them leaf variables and have requires_grad as True. You can do:

```
    inp = inp_tensor.detach().clone()
    inp = inp.requires_grad_(True)
```

4) For adversarial training, we provide the pseudo code for your reference:

```
    # Load your model weights here
    for epoch in range(nb_epochs):
        for batch_id in range(nb_batches):
            x_batch, y_batch = generator.get_batch()
            nb_adv = int(np.ceil(self.ratio * x_batch.shape[0]))

            # Sample image ids to be replaced
            # Generate attack image and replace original images
            # Forward pass and Backward pass as in hw4
```

Note that the `.grad` would work only for the leaf variables, that is only for the input. As such, this should be enough for the assignment. However, in case you feel you need to look at intermediate gradients, you could use the hook function as follows:

```
    def save_grad_hook(module, grad_input, grad_output):
        setattr(module, 'grad_back', grad_output)
    layer_xyz.register_backward_hook(save_grad)
```

Experiments

For experiments, we ask you to do the following.

- Evaluate the model you trained in HW4 (you can retrain one if you did not save the model), with benign images and IGSM attacked images; we have given suggested attack parameters but you may adjust them to get a strong attack while keeping the manipulation magnitude low. We are not looking for you to find an optimal but please results for two sets of parameters.

- Second, starting from your pre-trained weights, fine-tune your model through the adversarial training pipeline and report accuracy again on benign images and IGSM attacked images. Note that you need to generate attacks on the adversarially trained model again instead of reusing the originally computed attack images. For adversarial training, take IGSM as your attack function and test your model with its attacks.

Submission

Please include your code and the following in your submission document.
Quantitative Results:

- Report accuracy of your model without defense on benign images and IGSM attack images.

- Report accuracy of your model after adversarial training on benign images and IGSM attack images.

Qualitative Results

- Visualize IGSM attack images of your no-defense model (2-5 samples).

- Visualize IGSM attack images of your adversarial-trained model (2-5 samples)