Deep Learning Project 3

Spring 2025

Please upload your project report before 11:59pm, May 12, 2025.

This project will comprise 25% of your overall grade. Please perform this project in groups of (at most) 3. If you are looking for project team members, please broadcast your interest in the Class Slack in the `find-a-teammate` channel.

Jailbreaking Deep Models

Much of the class has focused on training deep neural networks that perform well on certain tasks. In this final project you are tasked with launching effective *adversarial attacks* on production grade, publicly posted models, and degrade their performance. We will focus on image classifiers. Recall that we had discussed in class that deep classifiers are surprisingly brittle; an attacker can carefully designed perturbations that misclassify a test data point, and therefore reduce model performance to very low rates (even all the way down to 0%).

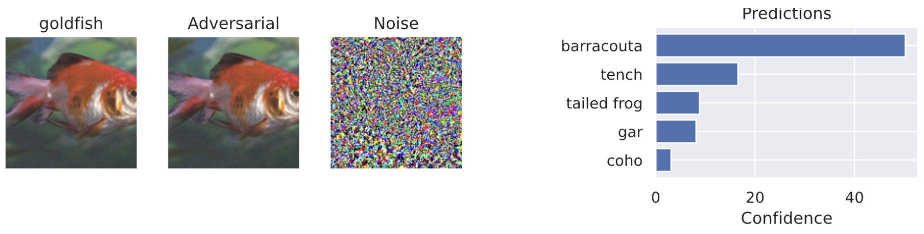

Figure 1: Example of an adversarial attack.

The main challenge is to design attacks that are subtle or imperceptible; that is, the perturbed test image should resemble the original test image as much as possible. Mathematically, we require that the distance (measured in some sense) between the original and test images are small. Typical measures of distance in such cases are the L_∞ distance (which upper-bounds how much *each* image pixel is allowed to be perturbed; these are called L_∞ or pixel-wise attacks), and the L_0 distance (which upper-bounds how *many* pixels are allowed to be perturbed; these are called L_0 or patch attacks.)

Tasks

For your project, implement the following tasks.

Task 1: Basics

The goal is to attack a ResNet-34 model that is trained to classify the ImageNet-1K dataset. ImageNet-1K is a well-known dataset in computer vision research with visually challenging images from 1000 classes, and networks which are trained on ImageNet typically also do well on other tasks. You can download the ResNet-34 model from TorchVision using the following command:

```
pretrained_model = torchvision.models.resnet34(weights='IMAGENET1K_V1')
```

Download the attached test dataset. This is a subset of images taken from 100 classes of the ImageNet-1K dataset. The included .json file has the associated label names and ImageNet label indices. We will need to preprocess the images like this before doing anything else:

```
mean_norms = np.array([0.485, 0.456, 0.406])
std_norms = np.array([0.229, 0.224, 0.225])
plain_transforms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=mean_norms,
                         std=std_norms)
])
dataset_path = "./TestDataSet"
dataset = torchvision.datasets.ImageFolder(root=dataset_path,
                transform=plain_transforms)
```

Evaluate the pre-trained ResNet-34 model on this test dataset; note that to validate a prediction you will have to look at the predicted class label and match it to the corresponding index in the .json file.

Report top-1 and top-5 accuracy for this dataset. (Top-k accuracy is calculated as follows: compute the k most likely class labels according to the classifier, and return True if any of these k labels matches the ground truth.)

Task 2: Pixel-wise attacks

A common and simple algorithm for mounting an L_∞ attack is called Fast Gradient Sign Method (FGSM); this implements a *single* step of gradient ascent (in pixel space) and truncates the values of the gradients to at most ε. Mathematically, we can write this as

$$x \leftarrow x + \varepsilon \,\text{sign}\left(\nabla_x L\right)$$

where L is the cross-entropy loss, the gradient is with respect to the input parameters (not the weights – so remember to), and the sign operation just truncates the gradient to the unit L_∞ cube. (Convince yourself that this makes sense!)

The parameter ε is called the attack budget. If raw (unpreprocessed) images have pixel values of 0-255, an attack budget of $\varepsilon = 0.02$ roughly corresponds to changing each pixel value in the raw image by at most +/-1.

Implement FGSM for each image in the test dataset for $\varepsilon = 0.02$. Visualize 3 to 5 test cases where the original model no longer classifies as expected. Your visualization can be similar to the example shown above.

You should now have a new set of 500 images; verify that the new images are visually similar to the original test set and that the L_∞ distance between new and original is no greater than $\varepsilon = 0.02$. Save this dataset (call this "Adversarial Test Set 1"). Evaluate ResNet-34 performance and report new top-1 and top-5 accuracy scores. You should strive to achieve accuracy drop of at least 50% relative to your baseline numbers from Task 1 (so if your earlier metrics were above 80%, then your new metrics should be below 30%.)

Task 3: Improved attacks

Now that you have two accuracy metrics (one for the original test set, Adversarial Test Set 1), propose ways to improve your attack and degrade performance even further. Remember: you can do whatever you like to the original test images, as long as the ε constraint is met and you get worse performance than FGSM. Options include: multiple gradient steps, targeted attacks, other optimizers, etc.

You should now have a new set of 500 images; verify that the new images are visually similar to the original test set and that the L_∞ distance between new and original is no greater than $\varepsilon = 0.02$. Save this dataset (call this "Adversarial Test Set 2"). Visualize performance for 3-5 example images. Evaluate ResNet-34 performance and report new top-1 and top-5 accuracy scores. You should strive to achieve accuracy drop of at least 70% relative to your baseline numbers from Task 1.

Task 4: Patch attacks

Pick your best performing attack method so far, but now implement it such that you aren't perturbing the *whole* test image, but only a small random patch of size 32x32. This is going to be more challenging, since as the attacker you have fewer knobs to twiddle around. Therefore you are free to increase ε to a much larger amount (say 0.3 or even 0.5) to make your attack work. *Hint: a targeted attack might be helpful in this context.*

You should now have a new set of 500 images. Save this dataset (call this "Adversarial Test Set 3"). Visualize performance for 3-5 example images. Evaluate ResNet-34 performance and report new top-1 and top-5 accuracy scores.

Task 5: Transferring attacks

You now have three perturbed versions of the original test set. Evaluate classification accuracy of these datasets using any pre-trained network other than ResNet-34. You can choose any model you like; for example, DenseNet-121:

```
new_model = torchvision.models.densenet121(weights='IMAGENET1K_V1')
```

A full list of ImageNet-1K models are available at this link.

List out top-1 and top-5 accuracies for all 4 datasets: original + the 3 adversarial test sets you constructed. Comment on your findings, any trends observed, lessons learned from this task, and potential ways to mitigate transferability.

Optional: Monitor your performance

Since this is the last week of class we won't have time to set up a public/private leaderboard unlike previous projects. However, feel free to post your intermediate and final results (example visualizations, accuracy numbers) on Slack to share your progress with us and your classmates use the `final-project` channel. Please note your team name and tag your team partners when you post results.

Resources

If you do use external code or coding LLMs, please include a clear citation. You are free to use any other online resources and/or techniques you like, as long as you include citations.

You can also use NYU HPC.

Deliverables

This project has two main deliverables:

- A project report (15 points).
- A project codebase (10 points) in the form of a Github repository.

Projects will be graded on:

- clarity and quality of submitted project report;
- quality and completeness of final reported results.
- and clarity and quality of submitted codebase. Include notebooks with clear plots; in particular, we want to see your code execution. Include statements where you clearly print the top-1 and top-5 test accuracy numbers.

Project report

Your report has to be **no more than 4 pages long** including numbers and citations, **typeset in two-column AAAI 2024 camera-ready format**. You can also have unlimited space for an appendix if you want to include visualizations and example images, but the first 4 pages should have all the main points that you want to convey. Any report not in this format will not be graded. Here is a link to the format in LaTeX and MSWord formats; see the "Camera-ready" folder in this link for example documents.

Please upload a PDF of your report to Gradescope. **Only one team member should upload** on Gradescope, as long as they tag all other team members.

In your report, please include:

- Names of all team members

- A short overview of your approach, along with a summary of your findings
- A methodology section that explains how you went about designing your adversarial attack, pros and cons of different hyperparameter choices, what lessons you learned during the design process, etc.
- A results section that reports your final test accuracy (top-1 and top-5) numbers, sizes of the perturbations, training times taken, etc.
- Any relevant citations.

Project codebase

In the first page of your report, please provide a link **on the first page** to a **publicly accessible** Github repository. Your repository should contain:

- the code necessary to reproduce the results in your report.
- well-documented code and/or Jupyter notebooks for easy visualization and verification of your work.