Practical Assignment: Classification Adversarial Attacks

1 Inroduction

Deep neural networks are known to be susceptible to adversarial perturbations, small perturbations that alter the network's output and exist under strict norm limitations. While such perturbations are usually tailored to a specific input, it is also possible to produce universal perturbations that achieve a high rate of miss-classification on a set of inputs. Universal perturbations present a more realistic use case for adversarial perturbations, as the adversary is not required to be aware of the model's input. In this assignment, you will implement universal adversarial attacks on classification models, aiming for a maximum decrease in accuracy. You will then test the susceptibility to the attacks on designated models over the $CIFAR10$ dataset.

2 Adversarial attacks

In what follows, we define the adversarial attack setting for universal and non-universal cases. We describe the adversarial optimization scheme used for producing the perturbations and discuss the optimization of the universal attacks aiming to perturb multiple inputs.

2.1 Adversarial attack setting

Let $X \times Y$ be an image space with corresponding ground truth labels, such that $X = (0,1)^{3\times w\times h}$ is a normalized RGB image space for some width w and height h, and $Y = \mathcal{Z}_k$ is the set of k possible labels. In addition, let $M : X \to Y$ be some classification model, let ℓ be some criterion over M's predictions, and let $\epsilon_p \in R^+$ be a norm bound for some $L_p, p \neq 0$ norm. Given a data sample $(x, y) \in X \times Y$ a standard adversarial perturbation $\delta_a \in X$ aims to maximize the criterion over the models prediction:

$$\delta_a = \arg \max_{\{\delta | x+\delta \in X, \|\delta\|_p \leq \epsilon_p\}} \ell(M(x+\delta), y) \tag{1}$$

Similarly, for a set of data samples, $\{(x_i, y_i)\}_{i=0}^{N-1} \subset X \times Y$, a universal adversarial attack aims to maximize the sum of the criterion over the samples:

$$\delta_{ua} = \arg \max_{\{\delta | \forall i, x_i+\delta \in X, \|\delta\|_p \leq \epsilon_p\}} \sum_{i=0}^{N-1} \ell(M(x_i+\delta), y_i) \tag{2}$$

Unlike the non-universal adversarial perturbations, which are tailored to a specific input, the universal attacks aim to be effective on multiple inputs.

2.1.1 Task-spesific target

As the limitation over the universal perturbation $\forall i, x_i + \delta \in X$, becomes drastic for large datasets, we relax the attack target for the scope of this assignment as:

$$\delta_{ua} = \arg \max_{\{\delta | \|\delta\|_p \leq \epsilon_p\}} \sum_{i=0}^{N-1} \ell(M(Clip(x_i+\delta, 0, 1), y_i) \tag{3}$$

Where the point-wise function $Clip(tensor, min, max)$ clips all values in the tensor to be in the range (min, max).

Task criterion For the scope of this assignment, the target criterion used for adversarial attacks is the ℓ_{01} loss describing the correctness of the classification by the attacked model:

$$\ell_{01}(M(x), y) = \begin{cases} 0 & M(x) = y \\ 1 & M(x) \neq y \end{cases} \tag{4}$$

The aim of this assignment is then to produce universal adversarial perturbations that maximize the ℓ_{01} criterion and are limited in the L_∞ norm under the bound $\epsilon_\infty = \frac{8}{255}$.

2.2 Optimization of adversarial pertubations

Algorithm 1 Non-universal PGD adversarial attack

Input M: Classification model
Input (x, y): attack data sample
Input ℓ: differentiable criterion
Input ϵ: attack L_∞ norm bound
Input K: number PGD iterations
Input α: step size for the attack

$\delta \leftarrow \text{Uniform}(0, 1)$
$\delta_{\text{best}} \leftarrow \delta$
$\text{Loss}_{\text{best}} \leftarrow \ell_{01}(M(x), y)$
for $k = 1$ to K **do**
  optimization step:
  $g \leftarrow \nabla_\delta \ell(M(x + \delta), y)$
  $\delta \leftarrow \delta + \alpha \cdot \text{sign}(g)$
  $\delta \leftarrow clip(\delta, -\epsilon, \epsilon)$
  evaluate perturbation:
  $\text{Loss} \leftarrow \ell_{01}(M(x + \delta), y)$
  if $\text{Loss} > \text{Loss}_{\text{best}}$ **then**
    $\delta_{\text{best}} \leftarrow \delta$
    $\text{Loss}_{\text{best}} \leftarrow \text{Loss}$
  end if
end for
return δ_{best}

One approach to optimizing an adversarial perturbation δ is via the PGD algorithm. However, as this approach requires a differentiable criterion, it would require approximating the non-differentiable criterion ℓ_{01} via a differentiable substitute $\hat{\ell_{01}}$. We provide algorithms and implementations for the standard PGD attack (Algorithm 1), and the PGD attack runner (Algorithm 2), which you may refer to. The PGD attack produces a perturbation for a given sample, and the attack runner uses this routine to produce perturbations for a set of samples. Notice that the universal PGD attack optimizes a single perturbation for the whole set of samples and is, therefore, not suitable for the methodology of using an attacking runner as such. In the PGD implementation, we use the cross-entropy criterion as the differentiable substitute $\hat{\ell_{01}} = \ell_{CE}$; however, you may use other criteria or approaches in your implementation.

3 Assignment

Your goal in this assignment is to produce universal adversarial perturbations that aim to maximize ℓ_{01} on the $CIFAR10$ test set over 3 designated models. Below, we describe the assigned task and several

Algorithm 2 Non-universal PGD adversarial attack runner

Input M: Classification model
Input $\{(x_i, y_i)\}_{i=1}^{N}$: attack data samples
Input ℓ: differentiable criterion
Input ϵ: attack L_∞ norm bound
Input K: number PGD iterations
Input α: step size for the attack

for $i = 1$ to N **do**
    perturb single sample:
    $\delta_i \leftarrow \mathrm{PGD}(M, (x_i, y_i), \ell, \epsilon, K, \alpha)$
end for
return $\{\delta\}_{i=1}^{N}$

methodologies you must address in your report. We then continue to detail the code and models we have provided you with.

3.1 Task specifics

In this assignment, you must implement a universal adversarial attack on classification models, which will be tested on designated models. This implementation will naturally require a choice of optimization scheme and corresponding attack criterion. The optimization scheme details the aggregation of gradients over different samples, the definition of the optimization step, and its size α. The corresponding criterion could be the ℓ_{01} loss if the optimization does not require differentiability or a suitable substitute otherwise. In your submitted report, you must explain your choice and motivation for the optimization scheme and criterion and present the attack's algorithm.

3.2 Provided code and Models

Run envoirment We provide a detailed guide to install a suitable run environment inside the code directory in the file `mamba_install_env_cs236207.txt`. To run the code, first install the environment according to the instructions.

Classification models There are 3 models that we refer to in this assignment as attacked models, where we utilize the models made available by (1). We consider models based on the $WideResNet-28-10$ architecture, and the attacked model can be selected in the provided implementation via the run parameter `--model_name`. The first model we refer to is the standardly trained $WideResNet-28-10$ model, which we denote as the "standard model", and can be deployed via the parameter `--model_name Standard`. The second model we refer to is the $PreActResNet-18$ model suggested by (2), which utilizes a fast adversarial training scheme. We denote this model as the "Fast Robust Model", which can be deployed via the parameter `--model_name Wong2020Fast`. The third model we refer to is the $WideResNet-28-10$ suggested by (3), which utilizes an adversarial training scheme over generated data. We denote this model as the "Robust Model", which can be deployed via the parameter `--model_name Wang2023Better_WRN-28-10`.

Code overview The provided code contains several sections, and we now explain the purpose of each section:

- Parser.py file: This is the argument parser of the implementation, where you will find all the relevant run parameters and their documentation.

- run_attack.py file: This is the main file of the code that should be run. It first parses the run parameters and correspondingly runs the specified attacks.
- AdvRunner.py file: This is the Adversarial attack runner, which splits the data into batches, attacks each separately, and produces the relevant perturbations for each data sample (Notice that this methodology is unsuitable for universal attacks).
- Attacks directory: The provided standard PGD adversarial attack is implemented in this directory. The base attack class is implemented in the "attack.py" file, and the PGD attack is implemented in the "pgd.py" file.
- models directory: The relevant models are contained or will be downloaded to this directory by default.
- data directory: This is the default directory for downloading the $CIFAR10$ data.
- results directory: This is the default directory for saving the results.

4 Submission and grading

4.1 Report structure and evaluation

The following list details what your assignment report should contain and the significance of each part in the grade.

1. Intro (15%). Summarize your work. Briefly introduce the problem and the methods and state the key results.
2. Implementation and Methods (40%). Explain and give motivation for your methodologies. Detail your approach and present the method's algorithm. Explain the empirical and theoretical motivation for what you are doing.

   Note: You can use pre-existing code in your implementation, but specify what you used and which parts you implemented yourself.
3. Results and discussion (20%). Present all results in an orderly table and include graphs or figures as you see fit. Discuss, analyze, and explain your results.
4. The remaining grade (15%) will depend on your submission's performance compared to other groups.

4.2 Evaluation of submitted perturbation

In addition to your report, you will submit 3 `.pt` (Pytorch tensor) files named `standard_pert.pt`, `fast_robust_pert.pt`, and `robust_pert.pt`, each containing your best adversarial perturbation on the standard, fast-robust and robust models, correspondingly. Notice that each tensor must be shaped like a single input sample from the dataset and be bounded under the specified L_∞ limitation. The performance of these perturbations will be tested on the specified models over the $CIFAR10$ test set and compared to the perturbation submitted by other groups. The part of your grade dependent on the performance of your submission will then be evaluated according to the average ℓ_{01} value over each model. The results of each perturbation will be equally weighted, representing 5% of the total grade. You may only submit a single adversarial perturbation for evaluation on each model. The perturbation will be evaluated using the code we provided.

4.3 Submission

Create a `.zip` file titled `proj-id1_id2.zip` (replace `id1/id2` with your IDs). The zip file **must** include:

1. A single PDF document, `report.pdf`, containing your assignment report. It must be structured according to the sections listed above.
2. A folder `src/` containing all your code.
3. A single Pytorch tensor named `standard_pert.pt` contains your submitted adversarial perturbation on the standard model.
4. A single Pytorch tensor named `fast_robust_pert.pt` contains your submitted adversarial perturbation on the standard model.
5. A single Pytorch tensor named `robust_pert.pt` contains your submitted adversarial perturbation on the robust model.

The zip file **must not** include:

- Training or test data
- Any other unnecessary files

References

[1] Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. arXiv preprint arXiv:2010.09670 (2020)

[2] Wong, E., Rice, L., Kolter, J.Z.: Fast is better than free: Revisiting adversarial training. arXiv preprint arXiv:2001.03994 (2020)

[3] Wang, Z., Pang, T., Du, C., Lin, M., Liu, W., Yan, S.: Better diffusion models further improve adversarial training. In: International Conference on Machine Learning, PMLR (2023) 36246–36263