Homework 4

Goals

- Learn the basic principle of generating adversarial examples.

Background

By adding small perturbations to an image, we can force a wrong classification of a trained neural network. The problem is formulated as, finding $\widehat{x}$ s.t. $D(x, \widehat{x}) \leq \epsilon$, and $argmax_j P(y_j|\hat{x}, \theta) \neq y_{true}$ (untargeted), or $argmax_j P(y_j|\hat{x}, \theta) = y_{target}$ (targeted), where $P(y|x, \theta)$ is a classifier parameterized by θ, D is a distance function, ϵ is the maximum allowed noise, and $\widehat{x}$ is the generated adversarial examples.

- In this experiment we target 100 images from CIFAR-10 dataset and use a VGG-like model.
- We evaluate the generated examples with perturbation scale and success rate, i.e. the probability of generated adversarial examples being misclassified by the model.

Requirements

- Python >= 3.6
- TensorFlow >= 1.11.0, <=1.14.0

Python Files Description

In this homework, we provide unfinished implementation of adversarial attack in **TensorFlow** framework.

- `train.py`: the main script for running the whole program
- `attack.py`: the main script for adversarial attack. You should **fill the TODO** in this program
- `common.py`: scripts for config
- `custom_vgg16_bn.py`: scripts for target model
- `dataset.py`: scripts for loading dataset

Tasks

T1: Generating untargeted adversarial examples

cross entropy loss

- A straightforward way to generate adversarial examples is to minimize $\mathcal{J} = -crossentropy\_loss(y_{pred}, y_{true})$. To obtain minimum distortion, we can use lasso (L_1 regularization: $\sum_i |\widehat{x}_i - x_i|$), ridge ($L_2$ regularization: $\sum_i (\widehat{x}_i - x_i)^2$), or elastic-net (linear combination of L_1 and L_2 regularization: $\sum_i \{\beta|\widehat{x}_i - x_i| + \gamma(\widehat{x}_i - x_i)^2\}$) regularization to regularize the perturbation scale.
- Run experiments with different coefficients for different regularization terms.

C&W attack loss

- Another way to generate adversarial examples is to adjust the permutations so that the logit of the ground-truth label is lower than the largest logit of the other labels by a pre-specified margin, that is, to minimize $\mathcal{J} = \max\{[logit(\widehat{x})]_{y_{true}} - \max_{y \neq y_{true}} [logit(\widehat{x})]_y, -\kappa\}$. This loss can also be combined with different regularization.

T2: Generating targeted adversarial examples

- Adapt cross entropy loss and C&W attack loss to targeted attack.
- Run experiments with different regularization.

Report

- Give the cross entropy loss and C&W attack loss adapted to targeted attack. Explain them briefly.
- The loss function can be formatted as $\mathcal{L} = \alpha\mathcal{J} + \beta\sum_i |\widehat{x}_i - x_i| + \gamma\sum_i (\widehat{x}_i - x_i)^2$, where α, β, γ are coefficients, and $\mathcal{J}$ can be either cross entropy loss or C&W attack loss. Run experiments and complete the following table for untargeted attack and targeted attack. L_1, L_2 and L_∞ norm are calculated on the successful adversarial pertubations (normalized) with minimum regularization loss (`train.py` calculates them for you).
    - For untargeted attack

Optimization method	α	β	γ	**# Optimization Steps**	**Attack Success Rate**	L_1	L_2	L_∞
cross entropy loss	1	0	0					
	1	0.00001	0					
	1	0	0.001					
	1	0.00001	0.001					
C&W attack loss	1	0	0					
	1	0.01	0					
	1	0	1					
	1	0.01	1					

    - For targeted attack

Optimization method	α	β	γ	**# Optimization Steps**	**Attack Success Rate**	L_1	L_2	L_∞
cross entropy loss	1	0	0					
	1	0.0001	0					
	1	0	0.01					
	1	0.0001	0.01					
C&W attack loss	1	0	0					
	1	0.01	0					
	1	0	1					
	1	0.01	1					

- Discuss what make a good adversarial example. Choose the most successful experiment setup you think for untargeted attack and targeted attack respectively. Explain why.
- Discuss how regularization influences attack. Does it make attack harder? Why?
- Which kind of attack do you consider more difficult, untargeted attack or targeted attack? Why?

Submission Guideline

You need to submit both report and codes, which are:

- **Report**: well formatted and readable summary including your results, discussions and ideas. Source codes should not be included in report writing. Only some essential lines of codes are permitted for explaining complicated thoughts.
- **Codes**: organized source code files with README for **extra modifications** (other than `TODO`) or specific usage. Ensure that others can successfully reproduce your results following your instructions. **DO NOT include model weights/raw data/compiled objects/unrelated stuff over 50MB (due to the limit of XueTang**)

You should submit a `.zip` file name after your student number, organized as below:

- `Report.pdf`
- `codes/`
  - `*.py`
  - `README.md`

Deadline: Nov. 19th

TA contact info:

- Shao Zhihong (邵智宏)， szh19@mails.tsinghua.edu.cn

Appendix: literature you may be interested in

- I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial examples. In International Conference on Learning Representations, 2015.
- N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In IEEE Symposium on Security and Privacy (SP), pages 39–57, 2017
- S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool:a simple and accurate method to fool deep neural networks.In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2574–2582, 2016.