Technical Documents

1 Team information

Team members: Yinpeng Dong, Tianyu Pang, Chao Du

Team track: Non-targeted Adversarial Attack and Targeted Adversarial Attack

Team names: heshang sha (rank 4 in non-targeted attack, rank 3 in targeted attack); wukong sun (rank 5 in non-targeted attack, rank 2 in targeted attack); seng tang (rank 6 in non-targeted attack, rank 4 in targeted attack)

2 Methods

We will introduce the method used in the CAAD competitions. We use almost the same method in different submissions but with different hyper-parameters.

2.1 Attack method

Let $\boldsymbol{x}^{real}$ denote a real example and y denote the corresponding ground-truth label. Given a classifier $f(\boldsymbol{x}) : \mathcal{X} \to \mathcal{Y}$ that outputs a label as the prediction for an input, we want to generate an adversarial example $\boldsymbol{x}^{adv}$ which is visually indistinguishable from $\boldsymbol{x}^{real}$ but fools the classifier. For non-targeted attack, we want $f(\boldsymbol{x}^{adv}) \neq y$. And for targeted attack, we want $f(\boldsymbol{x}^{adv}) = y^*$, where y^* is the target class. The optimization problem for non-targeted attack and targeted attack can be written as

$$\arg\max_{\boldsymbol{x}^{adv}} J(\boldsymbol{x}^{adv}, y), \quad \text{s.t. } \|\boldsymbol{x}^{adv} - \boldsymbol{x}^{real}\|_\infty \leq \epsilon. \tag{1}$$

and

$$\arg\min_{\boldsymbol{x}^{adv}} J(\boldsymbol{x}^{adv}, y^*), \quad \text{s.t. } \|\boldsymbol{x}^{adv} - \boldsymbol{x}^{real}\|_\infty \leq \epsilon. \tag{2}$$

We use the gradient-based iterative attack method to solve each optimization problem. We adopt the recent proposed methods on improving the transferability of adversarial examples in our attacks. We integrate the momentum method proposed by us and the diverse input iterative FGSM proposed by Cihang Xie into our attacks. We further make some modifications based on these.

The algorithm for non-targeted attacks is

```
1. Init $\boldsymbol{x}_0^{adv} = \boldsymbol{x}^{real}$, $\boldsymbol{g}_0 = 0$;
2. For $t = 0$ to $S - 1$ do
3.     Transform the current solution $\boldsymbol{x}_t^{adv}$ by image transformation[1], rotation[2], resizing and padding (from Cihang Xie) with a given probability and get a transformed input $T(\boldsymbol{x}_t^{adv})$;
4.     Get the gradient of the model as $\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y)$, where $J$ is the cross-entropy loss;
5.     Perform Gaussian smoothing of the gradient and denote the smoothed gradient as $GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y))$;
6.     Update the adversarial example as $\boldsymbol{g}_{t+1} = \mu \cdot \boldsymbol{g}_t + \frac{GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y))}{\|GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y))\|_1}$, and $\boldsymbol{x}_{t+1}^{adv} = \boldsymbol{x}_t^{adv} + \alpha \cdot \text{sign}(\boldsymbol{g}_{t+1})$;
7. Return $\boldsymbol{x}^{adv} = \boldsymbol{x}_S^{adv}$.
```

[1]tensorflow.contrib.image.transform
[2]tensorflow.contrib.image.rotate

In this method, there are some hyper-parameters which will affect the results. S is the total number of iterations. The transformation probability p controls the probability of transforming the input. μ is the decay factor in the momentum-based attacks.

Similarly, we use the same method for targeted attacks

```
1. Init $\boldsymbol{x}_0^{adv} = \boldsymbol{x}^{real}$, $\boldsymbol{g}_0 = 0$;
2. For $t = 0$ to $S - 1$ do
3.     Transform the current solution $\boldsymbol{x}_t^{adv}$ by image transformation, rotation, resizing, and padding with a given probability and get a transformed input $T(\boldsymbol{x}_t^{adv})$;
4.     Get the gradient of the model as $\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y^*)$, where $J$ is the cross-entropy loss;
5.     Perform Gaussian smoothing of the gradient and denote the smoothed gradient as $GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y^*))$;
6.     Update the adversarial example as $\boldsymbol{g}_{t+1} = \mu \cdot \boldsymbol{g}_t + \frac{GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y^*))}{std(GS(\nabla_{\boldsymbol{x}} J(T(\boldsymbol{x}_t^{adv}), y^*)))}$, and $\boldsymbol{x}_{t+1}^{adv} = \boldsymbol{x}_t^{adv} - \alpha \cdot clip_{[-2,2]}(round(\boldsymbol{g}_{t+1}))$;
7. Return $\boldsymbol{x}^{adv} = \boldsymbol{x}_S^{adv}$.
```

$std(\cdot)$ is the standard deviation and $round(\cdot)$ is rounding to nearest integer. Values of $clip_{[-2,2]}(round(\cdot))$ are in set $\{-2, -1, 0, 1, 2\}$.

2.2 Submissions for non-targeted attack

In all three submissions of non-targeted attack, we generate adversarial examples for the ensmeble of Inception v3, Inception v4, Inception ResNet v2, ResNet v2 50, Adv Inception v3 and EnsAdv Inception ResNet v2. We also set the number of iteration as 9, the decay factor as 0.7 and the probability of transformation as 0.7 in all of them. In the first submission, we use the method detailed in the last section. In the second submission, we only use image resizing and padding as the transformation. In the third submission, we do not smooth the gradient in step-5.

2.3 Submissions for targeted attack

In all three submissions of targeted attack, we also use the same six models to generate adversarial examples. We set the number of iterations as 10 and the decay factor as 0.5 in them. In the first submission, the probability of transformation is 0.2. In the second submission, the probability of transformation is 0.5. In the third submission, the probability of transformation is 0.2 and we only use image resizing and padding as the transformation.