Supplemental Material

1 Dual Gradient Attack Method (DGM)

This supplemental material describes Dual Gradient white-box attack Method(DGM) which is adopted in SQBA attack method. DGM uses dual gradient vectors to effectively find perturbations. The procedure of DGM attack consists of two stages: (1) Generating adversarial perturbation, and (2) Fine-tuning generated perturbation.

1.1 Adversarial Perturbation

An input example $x \in \mathbb{R}^m$ to be estimated by the k-class classifier $F(x)$ can be seen as a data point located in a convex polyhedron [1], whose faces represent classes c_i, where $i = 0, 1, ..., k$. The orthogonal distance from a class c_i to the data point x denotes the quantified classification probability, and it is expressed as $\Delta(x; c_i)$. The decision boundary of the classifier can be seen as an affine linear equation also in the polyhedron. Since the primary objective of adversarial attack is to modify input example x to mislead a target model, the true class $c^\dagger$ and an adversarial class $\tilde{c}$ need to be considered.

The orthogonal distance between $c^\dagger$ and x is smaller than other classes in the polyhedron as $\Delta(x; c^\dagger) < \Delta(x; c^i),\ \forall\ c^\dagger \neq c^i$. To mislead the target model, therefore, $\Delta(x; c^\dagger)$ needs to be increased by moving example x sufficiently to an adversarial region where belongs to another class c^i. DGM searchs the optimal path to transport x by utilising two vectors, g^- and g^+, which are calculated with respect to the true class $c^\dagger$ and a potential adversarial class c^i respectively. The vector g^- has the negative direction to the class, therefore, it moves x away from the true class. In contrast the positive directional vector g^+ transports x closer to the adversarial class. The gradient vectors are calculated as:

$$g^{+/-} = \begin{cases} l_2 & : \nabla_F f_c(x)/\max\left(\nabla_F f_c(x)\right) \\ l_\infty & : \text{sign}\left(\nabla_F f_c(x)\right) \end{cases} \tag{1}$$

where $\nabla_F f_c(x)$ is a gradient vector obtained from the backward process of $F(x)$ with input example x and associated class c. DGM iteratively conducts such process to efficiently find the optimal direction of adversarial perturbation vector μ_t as:

$$\mu_t = (-\alpha_t)g_t^- + (1 - \alpha_t)g_t^+ \tag{2}$$

where $\alpha_t \in [0, 0.3]$ is a penalty parameter applied to both directional vectors. The penalty parameter is used to help the stable convergence in searching an optimal vector μ_t, and updated in every iteration as:

$$\alpha_t = \min\left(1/e^{4\lambda},\ 0.3\right), \quad \text{where } \lambda = \frac{f_{c_t^i}(x_t')}{\left(f_{c^\dagger}(x_t') + f_{c_t^i}(x_t')\right)} \tag{3}$$

Algorithm 1 DGM Adversarial Example Computation

input: example x, true class $c^\dagger$, classifier f
output: adversarial example $\tilde{x}$
while TRUE **do**
  $\{c_t^0, c_t^1, ..\} =$ Sort$(f(x_t'),$ *descend*$)$
  if $c_t^0 \neq c^\dagger$ **then** Break; **end**
  $\tilde{c} = c_t^1$
  $\alpha_t =$ Equation (3) $\leftarrow \tilde{c}, c^\dagger$
  $g_t^+ =$ Equation (1) $\leftarrow \tilde{c}$
  $g_t^- =$ Equation (1) $\leftarrow c^\dagger$
  $\mu_t =$ Equation (2)
  $x_{t+1}' =$ Equation (4)
  $t = t + 1$
end while
$\tilde{x} =$ Tune(x_t')

Algorithm 2 DGM Adversarial Example Tuning

input: example x, initial adversary x', true class $c^\dagger$, classifier f
output: adversarial example $\tilde{x}$
$t = 0, x_t' = x'$
while TRUE **do**
  $\tilde{x} = x_t'$
  $\mathcal{J} =$ MSE(x, x_t')
  $x_t' =$ ADAM$(x_t', \mathcal{J})$
  $c_t' = \arg\max[f(x_t')]$
  if $c_t' == c^\dagger$ **then** Break; **end**
  $t = t + 1$
end while
$\tilde{x} = x_t'$

Finally an intermediate adversarial example is calculated with a scaling factor $\epsilon \leq 1$, which is a small positive value as

$$x_{t+1}' = \text{clamp}\left(x_t' + \epsilon\mu_t\right)\big|_{[\min(x), \max(x)]}. \tag{4}$$

Algorithm 1 outlines the process to find an adversarial example with DGM method.

1.2 Adversary Tuning

Attack method discussed in the previous section focuses on the effectiveness in finding an adversarial example $\tilde{x}$, which successfully leads to the misclassification with a high classification probability yet. One other objective of the attack is finding the minimum perturbation to make the adversarial example $\tilde{x}$ sufficiently close to the input example x. Input example can be seen as a fixed data point. Therefore, the goal of this optimisation is to find δ that minimises l_2 distance between x and $\tilde{x}$. DGM solves such problem by formulating a simple objective of the iterative optimisation with Mean Square Error (MSE) as:

$$\min\left(\text{MSE}(x, \tilde{x})\right), \quad \text{such that } \tilde{x} \in [\min(x), \max(x)]^m \tag{5}$$

where $\tilde{x} = x + \delta$ is an adversarial example. To achieve the goal to find δ that minimises $\text{MSE}(x, \tilde{x})$, ADAM [2] optimiser is deployed in DGM method as detailed in Algorithm 2.

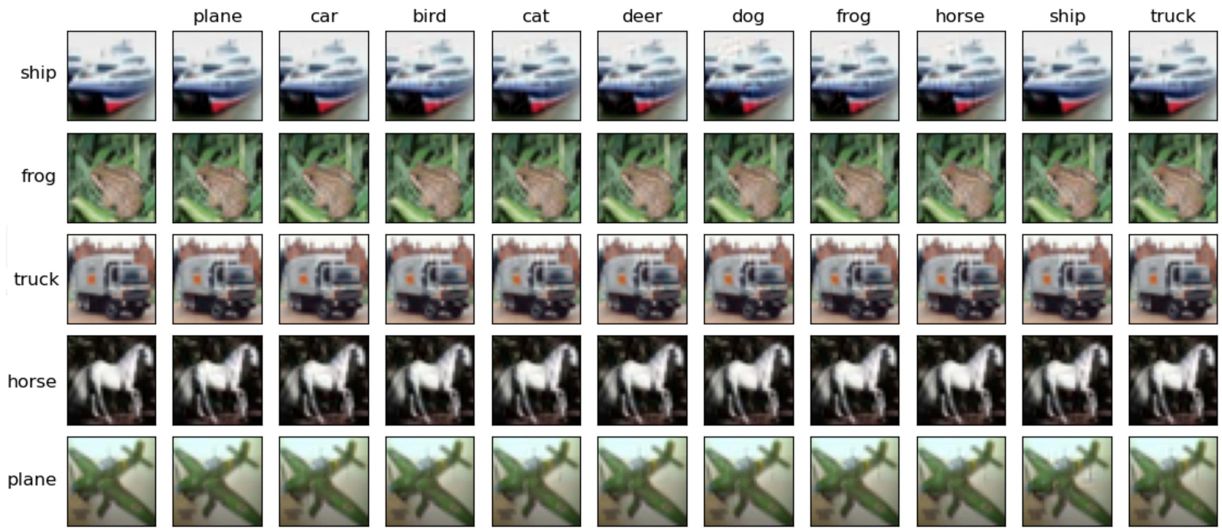

Figure 1: Examples of targeted attack. DGM l_2 attack is applied to the CIFAR-10 dataset performing the targeted attack for each source/target pair. First column is the clean images

References

[1] S.M.M. Dezfooli and A. Fawzi and P. Frossard and E.P.F. Lausanne *"DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks"* IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[2] D.P. Kingma and J. Ba *"Adam: A Method for Stochastic Optimization"* International Conference on Learning Representation, 2015.