segmentation-attack

sunblaze.ucb, Pranav Sankhe

October 2017

1 Introduction

Attack the state-of-the-art segmentation algorithm and therefore generate advs to conduct poisoning attacks on deep learning models.Below descibed is implementation of applying the attacks of universarial perturbations to attack the dilated residual network segmentation algorithm. We will be going through several implementations of the

2 Algorithms for Adversarial Generation

2.1 Dense Adversary Generation

Let's define some construction which we will using in the explanation of the algorithm. Let X be the image which contains 'N' recognition targets. The targets are represented by T = $t_1, t_2, t_3....t_N$. Each target $t_n, n = 1, 2, ...N$, is assigned a ground-truth class label $l_n = 1, 2, 3, ...C$ where C is the no. of classes. $L = l_1, l_2, ...l_N$

The detailed form of T varies among differeny tasks. In image classification, T contains only 1 element i.e the entire image. Conversely, T is composed of all pixels in semantic segmentation and all proposals in object detection. We will discuss thoroughly how construct T later. Given a Deep Network for a specific task, we use $f(X, tn)\epsilon R^c$ to denote the classification score vector on the n^{th}
To generate an adversarial example, the goal is to make the prediction of all targetts go wrong i.e. $F_c(X + r, t_n) \neq l_n$. Here 'r' denotes an adversarial perturbation added to X. To this end we specify an adversarial label $l_n{'}$ for each target, in which ln is randomly sampled from other incorrect class i.e. $l_n{'} = 1, 2, ..C/l_n$. Hence we denote $L^{'} = l^{'}_1, l^{'}_2, l^{'}_3, ...l^{'}_n$. In practice, we define a random permutation function $\Pi : (1, 2, 3....C) \rightarrow (1, 2, 3....C)$ for every image independently in which $\Pi(c) \neq C$ for $C = 1, 2, 3...C$ and generate $L^{'}$ by setting $l^{'}_n = \Pi(l_n)$ for all n.
Under this setting, the loss function covering all targets can be written as:

$$L(X, T, L, L^{'}) = \sum_{n=1}^{N} [f_{l_n}(X, t_n) - f_{l^{'}_n}(X, t_n)]$$

Minimizing L can be acheived via making every target to be incorrectly predicted i.e. supressing the confidence of the original correct class $f_{l_n}(X+r,t_n)$ while increasing that of the desired adversarial incorrect class $f_{l_n^{'}}(X+r,t_n)$ We apply a gradient descent algorithm for optimization. At the m^{th} iteration, denote the current image as X_m (possibily after adding several perturbations) as X_m. We find the set of correctly predicted targets, named the active target set $T_m = (t_n | argmax_c(f_c(X_m, t_n)) = l_n)$. Then we compute the gradient w.r.t. the input data and then accumulate all these perturbations:

$$r_m = \sum_{t_n \epsilon T_m} [\nabla_{X_m} f_{l_n^{'}}(X_m, t) - \nabla_{X_m} f_{l_n^{'}}(X_m, t_n)]$$

Normalization:

$$r_m^{'} = \frac{\gamma}{||r_m||_\infty}$$

Where, γ is a fixed hyperparameter.

We then add $r_m^{'}$ to the current image X_m and proceed to the next iteration. the algorithm terminates if either all the targets are predicted as desired or it reaches the maximum iteration.

The final perturbation is computed as $r_m^{'} = \sum m r_m^{'}$

2.1.1 Algorithm

Input:
input image X
the classifier $f(.,.) \epsilon R^c$
the target set $T = (t_1, t_2, t_3, ...t_N)$
the original label set $L = (l_1, l_2, l_3, ...l_N)$
the adversarial label set $L^{'} = (l_1^{'}, l_2^{'}, l_3^{'}, ...)$
the maximal iterations M

Output:

- $\mathbf{X}_0 \longleftarrow \mathbf{X}$, $r \longleftarrow 0$, $m \longleftarrow 0$, $T_0 \longleftarrow T$
- While $m < M_0$ and $T_m \neq \phi$, do the following:
  - $T_m = \{t_n | argmax_c\{f_c(\mathbf{X}_m, t_n)\} = l_n\}$;
  - $r_m \longleftarrow \sum_{t_n \epsilon T_m} [\nabla \mathbf{x}_m f_{l_n^{'}}(\mathbf{X}_m, tn) - \nabla \mathbf{x}_m f_{l_n}(\mathbf{X}_m, tn)]$;
  - $r_m^{'} \longleftarrow \frac{\gamma}{||r_m||_\infty}$;
  - $r \longleftarrow r + r_m$;
  - $\mathbf{X}_{m+1} \longleftarrow \mathbf{X}_m + \mathbf{r}_m^{'}$;

  - $m \longleftarrow m+1$

- **end**
- **return :** r

2.2 Universal Targeted Adversarial Example Generation

In principle, an adversary may choose y target arbitrarily. Crucially, however, an adversary may not choose y target based on y true since the ground-truth is also unknown to the adversary. Instead, the adversary may use $y_{pred} = f_\theta x$ as basis as we assume that the adversary has access to f_θ. We define two different ways of generating the target segmentation:
Static target segmentation:
In this scenario, the adversary defines a fixed segmentation, such as the system's prediction at a time step t_0.
Dynamic target segmentation:
In contrast, dynamic target segmentation aims at keeping the network's segmentation unchanged with the exception of removing certain target classes. We define a class of targets we want to hide and fill the gaps with the nearest neighbour.

The following method has been proposed for adversarial generation:

$$\epsilon^0 = 0$$
$$\epsilon^{n+1} = Clip_\epsilon\{\epsilon^n - \alpha sgn(\nabla^D(\epsilon))\}$$

where,

$$\nabla^D(\epsilon) = \frac{1}{m}\sum_{k=1}^{m} J_{ss}^w(f_\theta(x^{(k)} + \epsilon), y^{target,k})$$

being the loss gradient averaged over the entire training data.