Brittle ML : Playing Satan

Mentor - Purushottam Kar

MLG - 40

Aditya Vikram, Amur Ghose, Ankit Kumar, Hemant Kumar, Tushant Mittal

The Problem Statement

- Given a model and a certain input, craft an adversarial input.
- Intentionally designed to make the model err thus revealing its brittleness.
- Adversarial image should be “close” to the original.
- Done by adding a small amount of noise along the right gradient

The Original Plan

→ Examine the case of adversarially provided subsets for CNNs

→ Consider how the L2 norm based arguments provided by Goodfellow carry over to earthmover distances

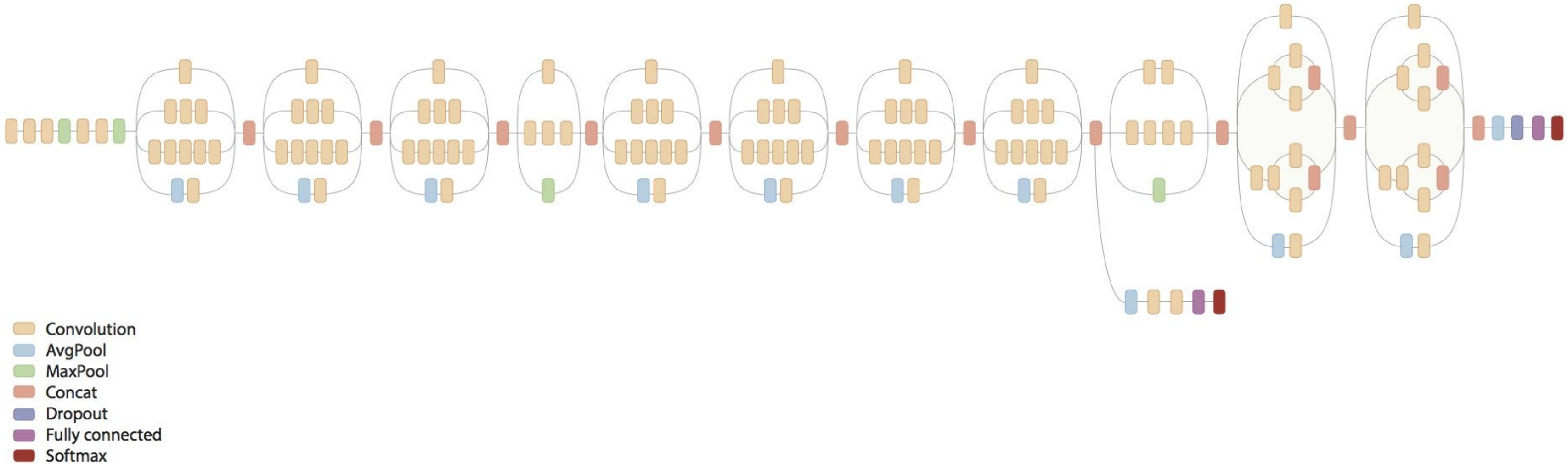
→ Provide a blackbox algorithm for the two above cases

→ Extend our CNN based arguments to Decision-Trees for ranking

→ Build either a whitebox/blackbox algorithm for constructing adversarial inputs

The Baseline Model

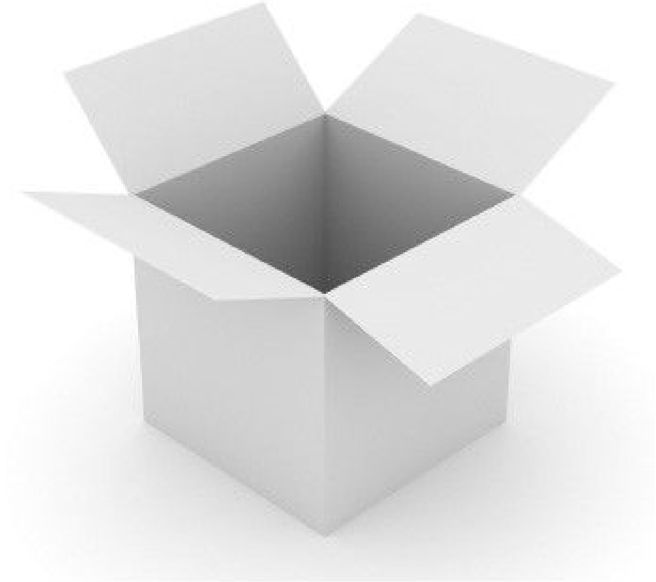
Inception-v3



Inception-v3

- State-of-the-art CNN for Image classification
- Top 5-error of 5.6%
- Pretrained on the ILSVRC2011
- Used Tensorflow^{TM} ‘s pretrained model
- Much faster than its competing CNNs

Whitebox attack



The Classic FGSM Attack

- ➔ Vulnerability due to piecewise linearity of CNNs in high-dimensional spaces
- ➔ Move in the direction of the gradient to **maximize** loss
- ➔ This attack has been tried on various architecture but not on Inception v3 yet.

$$\eta = \epsilon \operatorname{sign}\left(\nabla_{x} J(\theta, x, y)\right).$$

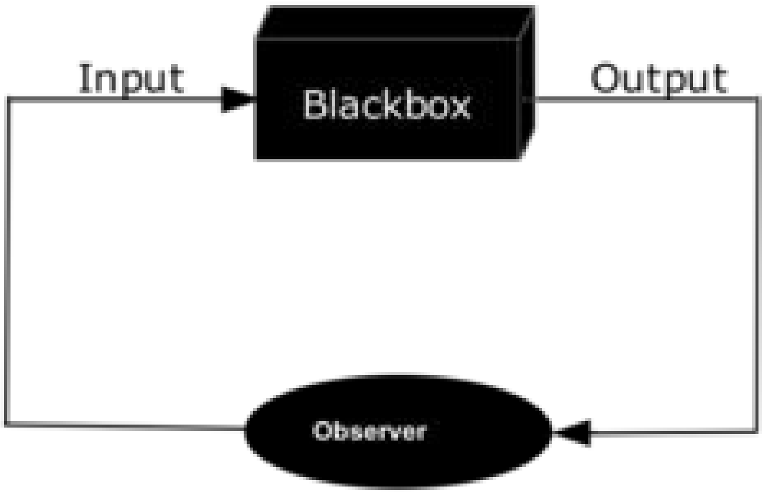
$$\tilde{x} = x + \epsilon \operatorname{sign}\left(\nabla_{x} J(x)\right)$$

Results

Original Image
Probability 92.47%

Adversarial Image
Probability 6.73%

Blackbox attack



Challenges

- ➔ **Much harder to mount** - No longer access to the gradients which is crucial for the FGSM attack
- ➔ No knowledge of the underlying model size

The Solution

- ➔ Learn a substitute model to imitate Inception-v3
- ➔ Possible because of *Transferability of Adversarial Examples*[1]
- ➔ Substitute model could be extremely simple (only 2 hidden layers)
- ➔ Attack the substitute model using FGSM

1. Papernot et.al. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples.* https://arxiv.org/pdf/1605.07277.pdf

Training the Substitute

- ➔ Needs very few (~150) training points to learn the substitute
- ➔ Label the images using the Blackbox
- ➔ Use Jacobian-based augmentation to grow the dataset
- ➔ Train the substitute using the new dataset

The Algorithm

Algorithm 1 - Substitute DNN Training: for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F, and an initial training set S_0.

Input: $\tilde{O}$, max_ρ, S_0, λ

```
Define architecture F
for ρ ∈ 0 .. max_ρ − 1 do
    // Label the substitute training set
    D ← {(x⃗, Õ(x⃗)) : x⃗ ∈ S_ρ}
    // Train F on D to evaluate parameters θ_F
    θ_F ← train(F, D)
    // Perform Jacobian-based dataset augmentation
    S_{ρ+1} ← {x⃗ + λ · sgn(J_F[Õ(x⃗)]) : x⃗ ∈ S_ρ} ∪ S_ρ
end for
return θ_F
```

Results

Original Image
Confidence = 25.26%

Adversarial Image
Confidence = 25.26%

Restricted Query Model

- Usually, a blackbox attack places no restriction on the number of queries and this is exploited by the algorithm to create a nice substitute model.
- But in many settings it might not be possible to actually make a lot of queries.
- We thus look at how the performance changes by such a restriction.

Results

Effect of Augmentation epochs

Number of Images	Epochs	Epsilon	Misclassification Rate
150	4	0.3	58.00 %
150	3	0.3	33.53 %
150	1	0.3	52.118 %
125	4	0.3	54.52 %
100	3	0.3	62.12 %

Wait that's counter intuitive !

➔ It would be expected that a decrease in number of images should give lesser misclassification.

➔ This however can be explained when one actually looks at the photos.

The bad epsilon

Data = 150, Eps = 0.05,
Confidence = 25.26%
Misclassfn. Rate = 28.36%

Data = 100, Eps = 0.1,
Confidence = 68.94%
Misclassfn. Rate = 37.45%

Data = 150, Eps = 0.3,
Confidence = 2.10%
Misclassfn. Rate = 58.00%

Implementation Challenges

- Adjusting *cleverhans* modules to suit the purpose and learning to use the Tensorflow™ API
- Handling BIG data (ImageNet)
- Battling with limited computing resources for blackbox attack

A Side Approach

Sampling attack on low-dimensional data

- ➔ Core idea : Use Laplace approximation
- ➔ Suppose the FGSM approach yields, for every candidate input vector X, a corresponding X' that will break the system
- ➔ But, this assumes a gradient oracle
- ➔ Let us use it further

Laplace approximation around local maxima

Assumptions :

- ➔ FGSM model is yielding to us a posterior distribution's mean
- ➔ Model this posterior distribution as Normal w/ mean = MAP value = FGSM output
- ➔ Since gradient is available as an oracle, query it for every gradient again to get Hessian
- ➔ In this case the posterior around the MAP value is known to be ~ N (w, H-1)
- ➔ Where w is what is returned from FGSM, and Hessian is inverted for covariance
- ➔ Sample from this distribution instead of playing MAP value every time

Some experimental results of this attack

Since computing Hessian and inversion is expensive, ran it on small datasets (Iris, Abalone) - these datasets have pre-defined classes and are well modeled by a variety of techniques such as GMMs, KDEs

Step size epsilon is set at 2*sigma (sigma = std-dev along axis), for hessian attack, step size is lowered by a fraction to give each the same L2 distances. All density estimation done via scikit-learn + iGMM code available on github (only 2 classes used for both , iris first 2 types (setosa and versicolour), abalone M and I) . Gradient of relative log probability used to make the oracle.

Dataset	Hessian flip chance (iGMM)	FGSM flip chance(iGMM)	Hessian flip chance(KDE)	FGSM flip chance(KDE)
Iris	60.3%	57.2%	30.2%	12.7%
Abalone	32 to 58%	47.6%	22.4%	5.6%

What's Next ?

Generalizing the norm

- ➔ Current methods of crafting of adversarial examples uses the lp norms to constrain the added “noise" in order to prevent a visible change in the input.
- ➔ The idea was to try to construct an FGSM -like attack using Earthmover Distance (EMD).
- ➔ This, however, turned out to be harder than anticipated.
- ➔ A recent publication[1] discusses this briefly and calls it “ a nice open problem".
- ➔ We would like to investigate it further in detail in the future.

1. Tramer, Papernot et.al. *Space of Transferable Adversarial Examples.* https://arxiv.org/abs/1704.03453

Breaking classification for DTs

- ➔ Algorithm[1] proposed by Papernot et.al. for classification using DTs
  - ◆ Find the leaf node of the input in DT
  - ◆ Find the nearest leaf in the tree where the output class changes
  - ◆ Perturbs the training point to change its output
- ➔ Fails for ranking!

1. Papernot et.al. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples.* https://arxiv.org/pdf/1605.07277.pdf

Ranking using DTs

- LambdaMart: Boosted regression trees to rank search queries
- Used in the Bing search engine
- Improves the previous LambdaRank model using DTs

Challenges for Ranking

- → Moving to the nearest leaves doesn’t help in general
- → Changing the regression values of the leaves not enough!
- → Need to change the order of ranking
- → Ranking using DTs is much more robust!

Questions?