Literature Review

Review Order

1. Breaking Things Is Easy
2. Is attacking machine learning harder than defending it?
3. The challenge of verification and testing of machine learning
4. Adversarial Introductory Paper (Intriguing properties of neural networks)
5. Fast Gradient Sign Method (Explaining and Harnessing Adversarial Examples)
6. Black-Box Attack (Practical Black-Box Attacks against Machine Learning)
7. Basic Iterative Method (Adversarial Examples In The Physical World)
8. Jacobian-based Saliency Map Approach (The Limitations of Deep Learning in Adversarial Settings)
9. Virtual Adversarial Training (Distributional Smoothing with Virtual Adversarial Training)
10. Delving into Transferable Adversarial Examples and Black-Box Attacks
11. Deep Adversarial Examples (Adversarial Manipulation of Deep Representations)
12. Ensemble Adversarial Training: Attacks and Defenses
13. Synthesizing Robust Adversarial Examples
14. Towards Deep Learning Models Resistant to Adversarial Attacks
15. Adversarial Machine Learning at Scale

Summarizations and Key Notes

Breaking Things is Easy

Intro

- Most ML models are built with no threat in mind (no adversarial opponent)
- ML system might be targeted by an adversarial attack in its training **or** inference phase
- Adversaries may have access to model architecture/parameters, or inputs/outputs

CIA model of security

- Attacker can compromise confidentiality and/or integrity and/or availability to break a ML model
- Confidentiality: model must not leak information to unauthorized users
    - E.g. Prevent ability to reverse engineer training data
- Integrity: model's outputs can be skewed based on malicious input
    - E.g. Adding professional terms at end of email to get it past spam filtering
- Availability: bring down a system by an input
    - E.g. weird object kicks manual drive mode on for autonomous vehicle

Poisoning Training Sets

- Adversary can interfere with model by providing modified or extra malicious training data
- Perturb training points in a way that increases prediction error in production

Inference mistakes with adversarial examples

- Force models to make mistakes by perturbing inputs
- Perturbation is computed to minimize a norm in the input space while increasing prediction error
    - E.g. noise to 60% panda photo classifies it as 99% gibbon
- Many attacks require knowledge of model parameters to find the optimum noise to perturb inputs to maximize errors
    - Can be found out by querying a model often to create a clone model to use as reference
    - These attacks exploit transferability of adversarial inputs
        - Often misclassified on a variety of different models

Privacy Issues

- Social biases encoded in training sets occur during inference
- Goal of adversaries is to recover part of the training data used for the modelling
- Membership inference queries: know whether an input was used for training or not
    - High confidence classifications often give this away

Explaining and Harnessing Adversarial Examples

Overview:

- Szegedy et al. discovered adversarial examples
- Past speculative explanations suggested this is due to nonlinearity, insufficient model averaging and regularization
- Paper shows such explanations is unnecessary; linear behaviour is sufficient
- Introduces FSG method for generating adversarial examples
- Adversarial training provides additional regularization
- Generic regularization e.g. dropout, pretraining, model averaging not significant

Linear explanation:

- Precision of feature is limited so classifier does not respond differently to input x* = x + eta if eta < precision (i.e. same class for x* and x if ||eta|| < e)
- Consider w^T x* = w^T x + w^T n where we maximize perturbation increase by assigning eta = sign(w)
- If w has n dimensions and average magnitude of weight vector is m then activation grows by emn
- ||eta|| does not grow with dimensionality but perturbation by eta grows linearly with n
- Therefore simple linear model can have adversarial examples if input has sufficient dimensionality
- “Accidental steganography”, model is forced to look at the noise since it aligns to its weights

Linear Perturbation of Non-Linear models:

- Let theta be the model parameters, x be the input, y be the target wrt x: let J be the cost function
- Then eta = e sign(d(J(**theta**, **x**, y)/dx)

Adversarial Training of Linear Models Versus Weight Decay:

- Key takeaways: FGSM has analytical form on logistic regression
- L1 weight decay being more "worse case" than adversarial training, b/c fails to deactivate in case of good margins

Adversarial Training of Deep Networks:

- Training with an adversarial objective function based on L-BFGS is a good regularizer
- Use early stopping on ADVERSARIAL validation set, not training validation set
- Minimizes worst case error when data is perturbed

Summary

- Adversarial examples are a result of models being too linear instead of non-linear
- Direction of perturbation rather than a specific point in space
- Easy optimization => easy adversarial attacks

Adversarial Examples In The Physical World

Overview:

- Previous work can feed data directly into the machine learning classifier. Not always the case physical world
- Feeding adversarial images from a cell-phone camera to ImageNet Inception classifier
- Large fraction of adversarial examples are classified incorrectly even when perceived through the camera.

Practical Black-Box Attacks against Machine Learning

Introduces:

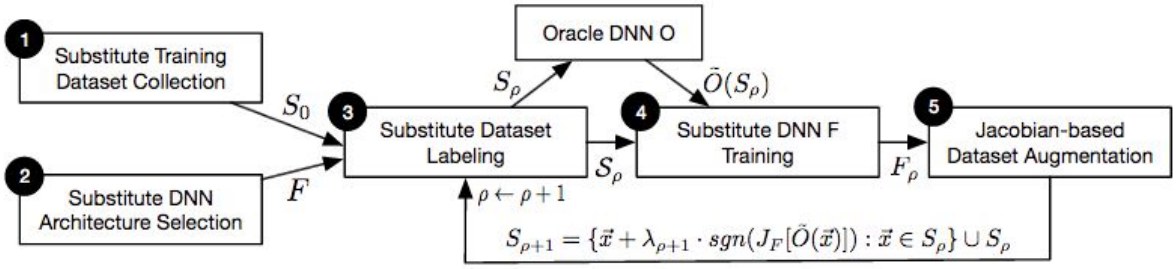
- First practical adversarial attack on a model without access to model architecture and weights
- Train local substitute model on synthetic dataset (generated by adversary), send these to target model and observe labels to train synthetic model and generate adversarial attack

Threat Model

- Accessing labels from the Oracle (targeted model) produced by an input is the only capability assumed (getting the top-k probabilities)
- Solve the optimization problem:
    - Adversarial image = image + minimum noise such that image plus noise does not equal top-1 prediction from oracle
    - $\vec{x^*} = \vec{x} + \arg\min\{\vec{z} : \bar{O}(\vec{x} + \vec{z}) \neq \bar{O}(\vec{x})\} = \vec{x} + \delta_{\vec{x}}$

Substitute Model Training

- Difficult because want to limit number of times querying Oracle for labels and selecting the model architecture
- Jacobian-based Dataset Augmentation: approximates Oracle's decision boundaries with few label queries
    - Not used for maximizing substitute synthetic model's accuracies
- Substitute Architecture: model type, number and size of layers have little impact on success
- Synthetic data is generated using Jacobian-based Dataset Augmentation
    - Augment data after each epoch using the sign of the jacobian matrix corresponding to the label assigned by the Oracle
- Periodically alternating between positive and negative step size



-

Crafting Adversarial Examples

- Goodfellow algorithm (fast gradient sign method) is easier to detect but faster
- Papernot Algorithm (jacobian saliency map)

Calibrating Substitute DNN Training

- Reservoir sampling reduces number of queries to Oracle
    - Ensures Jacaboian-based augmentation doesn't lead to exponential growth of dataset

Defense Strategies

- Two types: reactive (detect attacks and reject), proactive (be resilient to attacks)
- Gradient masking doesn't work, just create a substitute model
- Adversarial training only makes the model somewhat resistant
    - Makes the model robust to small perturbations, can be seen as an example of gradient masking
- Defensive distillation doesn't work again black-box attack
    - Substitute model is not distilled so it posses the gradients required for FGSM attacks

Intuition behind Transferability

- For two DNNs F and G, their FGSM matrices are highly correlated
- Sign matrices for oracle and the substitute model are highly correlated
-

Basic Iterative Method

- Adversarial examples can be printed and be used to fool classifiers from the camera as well

Fast Gradient Sign Method (FGSM):

- Linearize cost function
- $$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \operatorname{sign}\left(\nabla_X J(\boldsymbol{X}, y_{true})\right)$$

Basic Iterative Method

- Extends FGSM, apply FGSM multiple times with small steps

- Clip pixel values of intermediate results to maintain original image integrity with the perturbations
- Kind of like gradient descent learning
- $$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} + \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{true})\right)\right\}$$
- $$Clip_{X,\epsilon}\{\boldsymbol{X}'\}(x,y,z) = \min\left\{255, \boldsymbol{X}(x,y,z)+\epsilon, \max\left\{0, \boldsymbol{X}(x,y,z)-\epsilon, \boldsymbol{X}'(x,y,z)\right\}\right\}$$

Iterative Least-Likely Class Method

- Perturb image such that the least likely class is predicted
  - Targeted attack
- $$y_{LL} = \arg\min_y \left\{p(y|\boldsymbol{X})\right\}.$$
- $$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_{N+1}^{adv} = Clip_{X,\epsilon}\left\{\boldsymbol{X}_N^{adv} - \alpha\,\mathrm{sign}\left(\nabla_X J(\boldsymbol{X}_N^{adv}, y_{LL})\right)\right\}$$

Comparison of Methods

- Iterative perturbations are finer and modify the original image less
  - Works much better than anything else
  - Requires much lower epsilons for modifying images
- Least-likely iterative method works the best, epsilon ~= 16 destroys classifiers
- FSGM is more robust to photographed images
  - Iterative methods are more subtle and are destroyed in the transformations
- Iterative methods are best when no transformations are applied
- TensorFlow demo camera app for object detection was fooled often by these white-box attacks

Destruction Rate of Adversarial Images

- Fraction of adversarial images which are no longer misclassified after transformations

Artificial Image Transformations

- FGSM is robust to transformations, iterative-least-likely is the least robust
- Top-5 destruction rate is higher than Top-1 destruction rate
- Brightness and contrast changes don't really affect destruction rate
  - 5% on average, 20% for iterative least-likely
- Blur, noise, and JPEG encoding have a higher destruction rate
  - For iterative least-likely destruction rate for iterative methods are around 80-90%