1 Adversarial Attack on NN

1.1 Adversarial samples

Crafted by adding carefully selected **perturbations** δX to **legitimate inputs** X

- **Goals**:
  - `Confidence reduction`: reduce the output confidence classification
  - `Misclassification`: alter the output classification to any other class
  - `Targeted misclassification`: alter the output classification to a target class
  - `Source/Target misclassification`: force the output classification of a specific input to be a specific target class
- **Constraints**:
  - The perturbation δ_X must be small enough to pass human test.
  - E.g., the number of features perturbed is no larger than 14.29% for MNIST (about 112 pixels) [1].
- **Attacks at TEST time**: attack does not change the DNN model

1.2 Adversarial capabilities:

- **Network architecture**:
  - Layers, activation functions, weights, bias.
  - This gives attacker the ability to simulate the network.
- **Training data:**
  - The adversary is able to collect a *surrogate* dataset, sampled from the same distribution as the original training dataset.
  - This gives the attacker the ability to use the surrogate dataset to train a common DNN architecture to approximate the legitimate DNN model
- **Oracle**:
  - The adversary can obtain output classifications from supplied inputs.
  - This gives the attacker the ability to perform *differential attack* by observing the relationship between changes in inputs and outputs.
  - The adversary can be limited by the number of absolute or rate-limited input/output trails they may perform.

1.3 Adversarial sample crafting algorithm

Formal definition: Given a legitimate sample X, classified as $F(X) = Y$ by the network, the adversary wants to craft an *adversarial sample* X^* very similar to X, but misclassified as $F(X^*) = Y^* \neq Y$

$$argmin_{\delta_X} ||\delta_X|| s.t. F(X + \delta_X) = Y^*$$

Two-step process:

- **Direction Sensitivity Estimation**: evaluate the sensitivity of class change to each input feature
  - Fast sign gradient method[2]: compare the gradients of the cost function with respect to the inputs
  - Forward derivative method[1]
- **Perturbation Selection**: use the sensitivity information to select a perturbation δ_X among the input dimensions
  - Perturb all input dimensions by a small quantity [2]
  - Perturb a limited number of input dimensions by a large quantity [1]

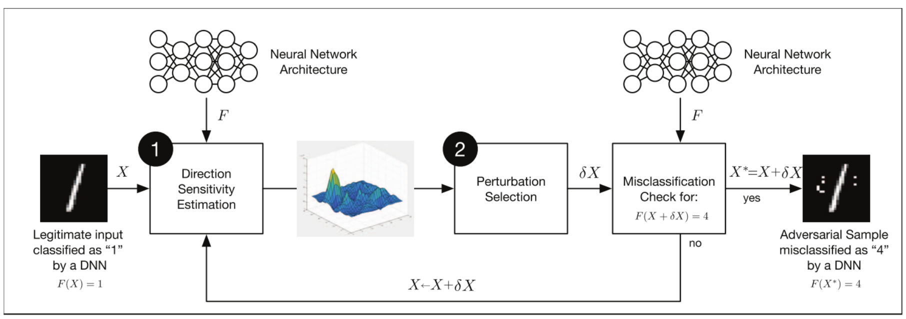

Figure 1: adversarial-crafting-framework

1.4 Attempted defenses against adversarial examples

- **Adversarial training**:
  - This is a brute force solution where we simply generate a lot of adversarial examples and explicitly train the model not to be fooled by each of them.
  - An open-source implementation of adversarial training is available in the cleverhans library and its use illustrated in the following tutorial.
- **Defensive distillation**:
  - This is a strategy where we train the model to output probabilities of different classes, rather than hard decisions about which class to output.
  - The probabilities are supplied by an earlier model, trained on the same task using hard class labels.
  - This creates a model whose surface is smoothed in the directions an adversary will typically try to exploit, making it difficult for them to discover adversarial input tweaks that lead to incorrect categorization

– (Distillation was originally introduced in *Distilling the Knowledge in a Neural Network* as a technique for model compression, where a small model is trained to imitate a large one, in order to obtain computational savings.)

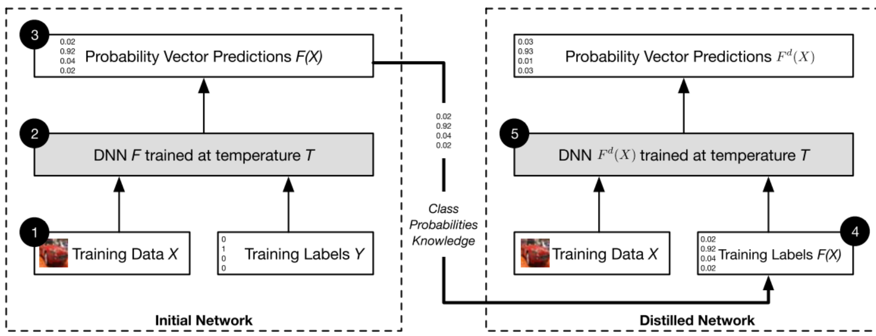

Figure 2: defensive-distillation-overview

[1] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and privacy (euros&P), 2016 ieee european symposium on*, 2016, pp. 372–387.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.