A Sample Adversarial Attack Algorithm for ML Systems

Rittika Adhikari
Freshman, Computer Science
Problem?

- ML models misclassify adversarial examples with high confidence
  - **Adversarial examples** = samples from the dataset with an additional worst-case perturbation
- Indicates that they are not learning the true underlying concepts that determine the output
- Possibly due to linear behavior of neural networks
Solution?

- Create a ML model that successfully resists adversarial perturbation while maintaining present accuracy on clean inputs
- Research Objective:
  - Define a new variation of a sample adversarial attack generation algorithm
  - Construct attacks that consider real-world constraints
  - Test an arbitrary adversarial ML algorithm on its sustainability through the Tensorflow API
Convolutional Neural Networks

- Deep, feedforward neural networks that are often used to analyze visual imagery
- Apply a series of filters to the raw pixel data of an image to extract and learn higher-level features
  - **Convolutional layers** = applies convolution filters to the image, and then applies the ReLU activation function
  - **Pooling layers** = downsamples the image data in order to reduce processing time; usually uses max pooling algorithm
  - **Dense layers** = performs classification on features extracted by convolutional layers and downsampled from pooling layers; fully-connected
TensorFlow

- **Tensor** = consists of a set of primitive values shaped into an array of any number of dimensions
- **Computational graph** = series of TensorFlow operations arranged into a graph of nodes
- Provides optimizers that change each variable in order to minimize loss function
  - **Gradient descent** = modifies each variable according to the magnitude of the derivative of loss with respect to that variable

Repeat until convergence:

$$
\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)
$$
Current Progress

- Researched new ML algorithms (i.e. convolutional neural networks) and learned the Tensorflow API
- Researched various sample adversarial attack generation algorithms (FGSM, JSMA, Black Box Attack)
- Studied implementations of FGSM, JSMA, and Black Box attack on the cleverhans nips adversarial competition github
FGSM

- Fast Gradient Sign Method = \[ \eta = \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \].
- Adding an imperceptible small vector whose elements are equal to the above, in order to generate misclassified examples
- Can easily calculate required gradient through backpropagation
- Reliably causes a wide variety of models to misclassify input
  - Linear models cannot resist adversarial perturbation
  - Structures with a hidden layer can be trained to resist adversarial perturbation
JSMA

- **Adversarial Saliency Maps** = indicate which input features to perturb to affect network output
- Helpful for problem-specific adversarial samples
- Process:
  - Compute Jacobian of the function learned by the NN during training (forward derivative)
  - Increase input features using the following saliency map:

\[
S(X, t)[i] = \begin{cases} 
0 & \text{if } \frac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\
\frac{\partial F_t(X)}{\partial X_i} & \text{otherwise}
\end{cases}
\]

- Peturb input features in accordance to the adversary’s goal
Black Box Attack

- Phase 1: attacker queries oracle with synthetic inputs selected by Jacobian heuristic to create model F approximating oracle model O’s decision boundaries
  - Collect S0 of inputs in domain
  - Select architecture to be trained as F
  - Iteratively trains more accurate substitute DNNs Fp

- Phase 2: attacker uses substitute to craft adversarial samples, which are misclassified by oracle due to transferability of samples
  - Goodfellow (FGSM) = increases likelihood of being misclassified by ML system, but easier to classify as humans
    \[ \delta_{\tilde{x}} = \varepsilon \text{sgn}(\nabla_{\tilde{x}} c(F, \tilde{x}, y)) \]
  - Papernot = good for source-target misclassification attacks; reduces perturbations at expense of greater computing cost
    \[ S(\tilde{x}, t)[x] = \begin{cases} 0 \text{ if } \frac{\partial F_t}{\partial \tilde{x}_i}(\tilde{x}) < 0 \text{ or } \sum_{\text{j} \neq t} \frac{\partial F_t}{\partial \tilde{x}_i}(\tilde{x}) > 0 \\ \frac{\partial F_t}{\partial \tilde{x}_i}(\tilde{x}) \left| \sum_{\text{j} \neq t} \frac{\partial F_t}{\partial \tilde{x}_i}(\tilde{x}) \right| \text{ otherwise} \end{cases} \]
Future Goals

- Define a new sample adversarial attack algorithm, using some of the algorithmic methods mentioned earlier (FGSM, JSMA, Black Box Attack)
- Test an arbitrary adversarial ML system on its robustness using newfound algorithm to generate sample attacks, and compare this performance to performance before new adversarial attacks