Summary of Paper: Adversarial Playground

By: Andrew Norton and Yanjun Qi

Presented by: Jennifer Fang [Week 01]

Department of Computer Science: University of Virginia

@ https://qdata.github.io/deep2Read/

ADVERSARIAL-PLAYGROUND: A Visualization Suite Showing How Adversarial Examples Fool Deep Learning

Goal: Visualize the efficacy of current adversarial methods against convolutional NN systems through a web visualization tool.

Make this tool educational, modular, and interactive.

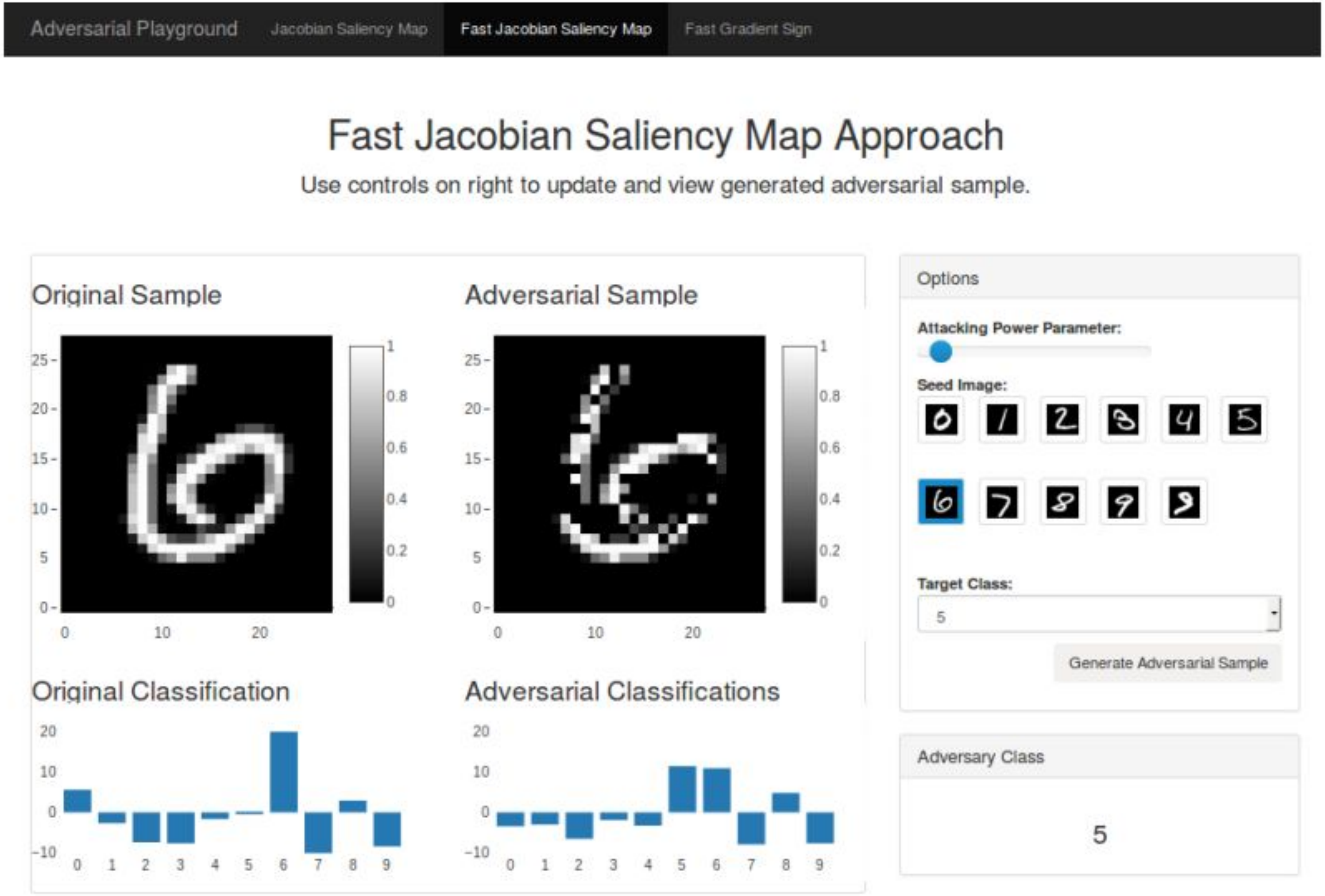
Background

Adversarial examples: maliciously generated images formed by making imperceptible modifications; threat to security

Falls into *evasion attacks*; those which aim to create inputs to be misclassified

2 types:

1. Targeted: $x' = \arg\min_{s \in X} \{\|x - s\| : f(s) = y_t\}$ target a class yt
2. Untargeted: $x' = \arg\min_{s \in X} \{\|x - s\| : f(s) \neq f(x)\}$ just want to misclassify


Adversarial Playground
Jacobian Saliency Map
Fast Jacobian Saliency Map
Fast Gradient Sign
Fast Jacobian Saliency Map Approach
Use controls on right to update and view generated adversarial sample.
Original Sample
Adversarial Sample
Original Classification
Adversarial Classifications
Options
Attacking Power Parameter:
Seed Image:
Target Class:
5
Generate Adversarial Sample
Adversary Class
5

Design Decisions

For speed:

1. Utilized client and server-side code
2. Rendered images in the client
3. Implemented a faster variant of JSMA attack

For usability:

1. Made Adversarial Playground a web-based application; no need for downloading

Benefits of Adversarial Playground

1. **Educational**
   a. Non-experts can understand why adversarial examples fool CNN-based image classifiers.
   b. Helps security experts explore more vulnerabilities.
   c. Accessible to casual users
2. **Interactive**
   a. Responds to user requests, and does so quickly.
3. **Modular**
   a. Experts can easily plug it into their frameworks as a module
   b. Experts can easily add other DNN models into the visualization

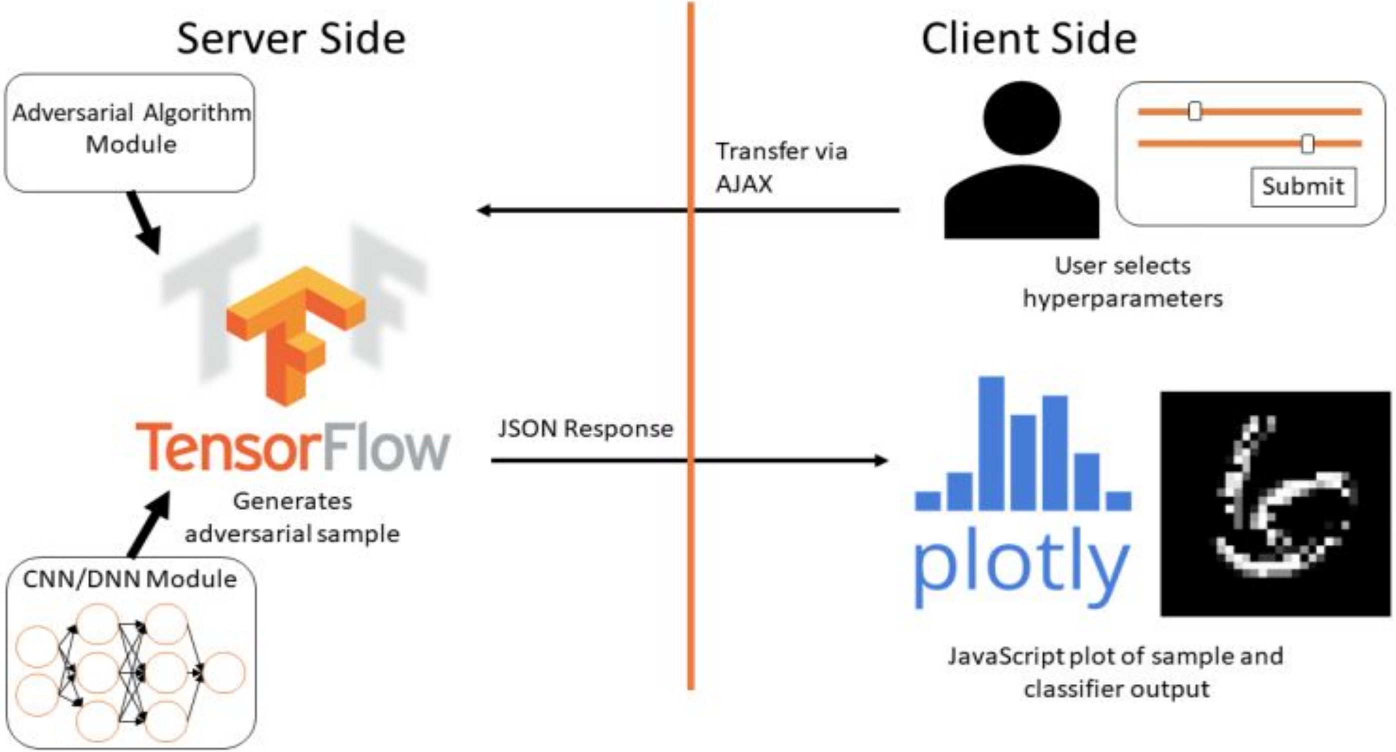


Figure 2: ADVERSARIAL-PLAYGROUND System Sketch

Improvements to JSMA

JSMA: creates a targeted attack

FJSMA changes: only considers pairs of features (p, q) such that p is in the top *k* (small constant chosen by us) features ranked by derivative in the p-coordinate

Algorithm 1 Fast Jacobian Saliency Map Apriori Selection
$\nabla \mathbf{F}(\mathbf{X})$ is the forward derivative, Γ the features still in the search space, t the target class, and k is a small constant

Input: $\nabla \mathbf{F}(\mathbf{X}), \Gamma, t, k$

1: $K = \arg\operatorname{top}_{p \in \Gamma}\left(-\frac{\partial \mathbf{F}_t(\mathbf{X})}{\partial \mathbf{X}_p}; k\right)$ ▷ Changed for FJSMA

2: **for** each pair $(p,q) \in K \times \Gamma$, $p \neq q$ **do** ▷ Changed for FJSMA

Performance of new FJSMA (evasion rate)

For FJSMA's with small k's, with the γ perturbation shown on the top row, FJSMA evasion rate does not deviate more than 0.07

Υ	10%	15%	20%	25%
JSMA Evasion Rate	0.658	0.824	0.867	0.879
FJSMA Evasion Rate [$k = 10\%$]	0.583	0.777	0.823	0.826
FJSMA Evasion Rate [$k = 15\%$]	0.613	0.816	0.867	0.871
FJSMA Evasion Rate [$k = 20\%$]	0.633	0.833	0.878	0.887
FJSMA Evasion Rate [$k = 30\%$]	0.638	0.844	0.896	0.901

Performance of new FJSMA (time)

FJSMA time is ~ 33% to 50% faster as γ increases from 10% to 25%

Υ	10%	15%	20%	25%
JSMA Time (s)	0.606	0.745	0.807	0.803
FJSMA Time [$k = 10\%$] (s)	0.411	0.468	0.490	0.485
FJSMA Time [$k = 15\%$] (s)	0.414	0.473	0.483	0.484
FJSMA Time [$k = 20\%$] (s)	0.415	0.466	0.482	0.483
FJSMA Time [$k = 30\%$] (s)	0.415	0.464	0.490	0.485

Conclusion + Future work

Conclusion: Adversarial Playground provides a quick, easy to use webapp to visualize the performance of adversarial examples against DNNs.

Future work:

- Support more evasion methods
- Explore more time-saving techniques to implement above
- Use different datasets CIFAR, ImageNet, MNIST, etc ...