Adversarial Examples in Deep Learning Models

Jacob Levine, SIGAIDA
Agenda for Today

- Shameless shilling for SIGAIDA
- How Deep Learning models work
- Intro to Adversarial Attacks
- Quick Demo in Colab
- Adversarial Attacks “in the wild”
- Combat Techniques, and their Downsides
Join SIGAIDA

- We have weekly reading groups to look at current research
- We have done some of our own research, working on papers to conferences
- We participate in 1-2 college ML competitions, often doing quite well
  - Often results in a free vacation to somewhere for the competition
- We do a bunch of random unrelated projects (ML for sabermetrics, RL for video games, etc.)
- We’re beginner-friendly and hold intro tutorials to get people up to speed
  - We’re reworking the intro curriculum for Fall 2023
Intro to DL Models: Motivation

- If you were to write a program to determine whether an image was of a dog, how would you do that?
  - Hard to articulate exact rules. Also, you need to be able to adapt to different positions, rotations, sizes of dog/not dog within the image

- One approach: design “features” that somehow identify parts of images
  - Traditionally done using handcrafted features based on shape/keypoints
    - “Edge Detection”/ “Corner Detection”/ “SIFT Keypoints”
    - Hard to extend, also not all that accurate
  - What if we could somehow empirically decide which features are important?
    - We have a lot of pictures of dogs
    - And a lot of pictures that are not of dogs
Intro to DL Models: Supervised Learning

- Given: training data \( \{(x_1, y_1), \ldots, (x_n, y_n)\} \)
- Find: predictor \( f(x) \in \mathcal{H} \)
- Goal: make “good” predictions \( \hat{y} = f(x) \) on test data
- How do we evaluate whether a prediction is “good”?  
  - Choose a “loss function” \( L(x) \)
- How do we evaluate predictions on test?  
  - Assume that train, test are similar  
  - Model can still find “spurious” patterns that only apply to train: overfitting
Intro to DL Models: ConvNets

- Develop “local” features over “patches” of an image
- Multiply-add set filter and pixel values
- Can have multiple “channels” using different filters
- Stacking convolutions gets features developed over a larger window
- Train to pick best filter weights
ConvNets don’t “see” like we do
ConvNets don’t “see” like we do
Exploiting this and “fooling” ConvNets
Exploiting this and “fooling” ConvNets

- Start with a correctly labeled image, and make small perturbations to the image until it’s misclassified.
- Increase loss/decrease probability of “correct” prediction $y^*$:
  - $X := X + \frac{\partial L(X, y^*)}{\partial X}$
- Alternatively, decrease loss/increase probability of “incorrect” prediction $\hat{y}$:
  - $X := X + \frac{\partial L(X, \hat{y})}{\partial X}$
- **Fast Gradient Sign Method**: Move by a constant amount “to the boundary of the L-inf ball” in the direction that maximises loss
  - $X_{\text{Adversarial}} := X + \epsilon \, \text{sgn}(\partial L(X, y^*)/\partial X)$
- **Iterative Gradient Sign Method**: Take steps until misclassified, but clip to ensure inside a $\epsilon$-ball of the original.
Exploiting this and “fooling” ConvNets

- Discussed “white box” techniques where gradients are known
- Newer “black box” techniques don’t depend on gradients, so can do attacks without knowing model architecture/weights
- Pixle: a black box attack based on switching pixels
- Example: The original class is 14 (Passerina Cyanea), while the misclassified class is 883 (Vase). Only one pixel swapped.
Attack Examples
Attack Examples
Attack examples

ImageNet image
- revolver
- boathouse
- china cabinet

Adversarial example
- mousetrap
- guillotine
- spotlight
Demo
Attacks “in the wild”: Autonomous Vehicles

<table>
<thead>
<tr>
<th>Distance/Angle</th>
<th>Subtle Poster</th>
<th>Subtle Poster Right Turn</th>
<th>Camouflage Graffiti</th>
<th>Camouflage Art (LISA-CNN)</th>
<th>Camouflage Art (GTSRB-CNN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5° 0°</td>
<td><img src="STOP" alt="Image" /></td>
<td>![Image](Right Turn)</td>
<td><img src="Graffiti" alt="Image" /></td>
<td><img src="LISA-CNN" alt="Image" /></td>
<td><img src="GTSRB-CNN" alt="Image" /></td>
</tr>
<tr>
<td>5° 15°</td>
<td><img src="STOP" alt="Image" /></td>
<td>![Image](Right Turn)</td>
<td><img src="Graffiti" alt="Image" /></td>
<td><img src="LISA-CNN" alt="Image" /></td>
<td><img src="GTSRB-CNN" alt="Image" /></td>
</tr>
<tr>
<td>10° 0°</td>
<td><img src="STOP" alt="Image" /></td>
<td>![Image](Right Turn)</td>
<td><img src="Graffiti" alt="Image" /></td>
<td><img src="LISA-CNN" alt="Image" /></td>
<td><img src="GTSRB-CNN" alt="Image" /></td>
</tr>
<tr>
<td>10° 30°</td>
<td><img src="STOP" alt="Image" /></td>
<td>![Image](Right Turn)</td>
<td><img src="Graffiti" alt="Image" /></td>
<td><img src="LISA-CNN" alt="Image" /></td>
<td><img src="GTSRB-CNN" alt="Image" /></td>
</tr>
<tr>
<td>40° 0°</td>
<td><img src="STOP" alt="Image" /></td>
<td>![Image](Right Turn)</td>
<td><img src="Graffiti" alt="Image" /></td>
<td><img src="LISA-CNN" alt="Image" /></td>
<td><img src="GTSRB-CNN" alt="Image" /></td>
</tr>
</tbody>
</table>

Targeted-Attack Success

- 100%
- 73.33%
- 66.67%
- 100%
- 80%
Attacks “in the wild”: Autonomous Vehicles

<table>
<thead>
<tr>
<th>Perturbation</th>
<th>Attack Success</th>
<th>A Subset of Sampled Frames $k = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtle poster</td>
<td>100%</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Camouflage abstract art</td>
<td>84.8%</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Attacks “in the wild”: Shadows
Attacks “in the wild”: Shadows

Figure 4. Examples of frames in a video of our indoor experiment and the corresponding classification results.
Attacks “in the wild”: Shadows

a) Shadows in the real world.

b) Our simulated shadows.
Adversarial Examples can fool humans (with limited time)
Adversarial Defense Techniques: Adversarial Training

- Instead of training on input images, train on “adversarial examples”
- Find an image from your dataset, do an attack on your model using that image, use that image for training
- Becomes a “min-max” optimization problem: loss becomes
  \[
  \min_f \max_{\text{perturbation}} L(f, x + \text{perturbation})
  \]
- Same training script, just have to calculate pertubations
Adversarial Defense Techniques: Adversarial Training

- Provides good defense against adversarial examples under chosen attack setup
- Acts as a form of regularization
- Upsides to transfer
- However, training is very slow
  - Need to create adversarial example for each image
  - Can’t precalculate adversarial examples; a change in the model will change which examples will fool it
Adversarial Defense Techniques: Adversarial Training

(a) MNIST, $l_\infty$-norm  (b) MNIST, $l_2$-norm  (c) CIFAR10, $l_\infty$-norm  (d) CIFAR10, $l_2$-norm
Adversarial Defense Techniques: Adversarial Training

- Tradeoff between Robustness and Accuracy
  - Models that are trained adversarially have generally worse inference performance than standard models
  - Worse, they tend to not generalize well to testing
Adversarial Defense Techniques: Training Results
Adversarial Defense Techniques: Other methods

- Obfuscating Gradients (only works on white-box techniques)
- Train a separate model to reject adversarial examples: SafetyNet
- Pre-process input images to disrupt adversarial perturbations
- Packages for evaluating/protecting against adversarial attacks (incl responsible-ai-toolbox)