Adversarial Attacks on CNN Based Image Classification

By Amay Singh

In the realm of artificial intelligence and computer vision, Convolutional Neural Networks (CNNs) have emerged as formidable instruments for image classification. These algorithms have demonstrated high proficiency in recognising and categorising visual information, a capability that has found essential uses across numerous domains, including healthcare, autonomous systems, and security.

However, amidst this impressive progress, a concern has arisen - the susceptibility of CNNs to adversarial attacks. Adversarial attacks represent a strategic assault on the reliability of CNN-based image classification systems. These attacks are characterised by their ability to subtly manipulate input images, introducing minuscule alterations that remain imperceptible to the human eye but are potent enough to lead CNNs to misclassification, thereby undermining the trustworthiness of these advanced systems.

Here, we will study 2 widely used methods for these Adversarial Attacks –

1. Fast Gradient Sign Method (FGSM)
2. Patch Attack

Fast Gradient Sign Method

FGSM, or the Fast Gradient Sign Method, is a fundamental and widely-used technique in the domain of adversarial attacks and defenses in deep learning, particularly for Convolutional Neural Networks (CNNs) and other gradient-based models. It was introduced by Ian Goodfellow and his colleagues in their 2014 paper titled "Explaining and Harnessing Adversarial Examples."

The FGSM attack works by perturbing an input data point, such as an image, to create an adversarial example. This perturbation is calculated by considering the gradients of the loss function (typically a cross-entropy loss) with respect to the input data. Here's a simplified explanation of how FGSM works:

1. **Calculate the Gradient:** Given an input image and a target class (the class you want the model to misclassify the image as), you compute the gradient of the loss function with respect to the input image.

2. **Add Perturbation:** You then add a small perturbation to the original image in the direction that maximizes the loss, but you limit the size of this perturbation to keep it imperceptible to the human eye.

3. **Create Adversarial Example:** The resulting image with the added perturbation is the adversarial example. This image appears almost identical to the original one but can fool the model into misclassifying it as the target class.

FGSM is known for its simplicity and speed in generating adversarial examples. FGSM is a technique used to craft adversarial examples by calculating gradients and making small perturbations to input data, primarily used for challenging the robustness of deep learning models, particularly CNNs, and highlighting their vulnerabilities.

Code –

The code along with the outputs and other required files has been attached. It consists of a Jupyter notebook titled ‘FGSM.ipynb’ and the pre-trained model ‘lenet_mnist_model.pth’. Alternatively, it can also be found on My Github Profile in the directory “Adversarial-Attacks-on-CNN-Based-Image-Classification”.

Discussion

This Jupyter Notebook demonstrates a Fast Gradient Sign Method (FGSM) attack on a LeNet model trained on the MNIST dataset. It analyses the model's robustness to different levels of perturbation and provides visual insights into the effects of these perturbations on image classification.

Here is a summary of the main components and steps:

1. Importing Libraries: The notebook begins by importing the necessary libraries, including PyTorch for deep learning, NumPy for numerical operations, and Matplotlib for plotting.

2. Device Configuration: It checks if a GPU (CUDA) is available and assigns the appropriate device accordingly.

3. LeNet Model Definition: The LeNet neural network model is defined using PyTorch's neural network module. It consists of two convolutional layers followed by two fully connected layers.

4. MNIST Test Dataset: The MNIST test dataset is loaded, and a data loader is created to iterate through the test examples one at a time.

5. Model Initialization: The LeNet model is initialised and loaded with pre-trained weights saved in 'lenet_mnist_model.pth'.

6. FGSM Attack Definition: The Fast Gradient Sign Method (FGSM) attack function is defined. This function calculates perturbed images by adding small perturbations (controlled by epsilon) in the direction of the loss gradient with respect to the input.

7. Testing with Perturbations: The notebook defines a testing function that applies the FGSM attack to test examples and evaluates the model's accuracy.

8. Accuracy vs Epsilon: The notebook loops through different epsilon values (degree of perturbation) and tests the model's accuracy under these perturbations. The results are stored in a Pandas DataFrame and displayed as a table.

9. Plotting Accuracy vs. Epsilon: A plot is created to visualize the effect of perturbations (epsilons) on the model's accuracy.

10. Visualizing Perturbed Examples: The notebook visualises perturbed examples with their true class and predicted class for different epsilon values. These visualisations help understand the impact of perturbations on image classification.

Patch Attack

Patch attacks represent a strategic and potent manipulation directed at machine learning models, particularly relevant to Convolutional Neural Networks (CNNs) used extensively for image recognition and classification. Unlike conventional adversarial attacks focusing on globally perturbing an entire image, patch attacks hone in on localised regions or patches within an image. These patches are typically small and inconspicuous to human observers but hold the power to influence model predictions profoundly.

The underlying premise of patch attacks is both beguiling and sophisticated. By inserting carefully crafted patches into an image, adversaries can effectively manipulate a model's output without significantly altering the broader context of the scene.

Patch attacks have significant implications across various domains. They underscore the brittleness of machine learning models, which may confidently recognize the majority of an image but falter when presented with localised perturbations. Consequently, they raise concerns about the robustness, security, and trustworthiness of AI systems deployed in critical applications such as autonomous vehicles, medical imaging, and security surveillance.

Code –

The code along with the outputs and other required files has been attached. It consists of a Jupyter notebook titled 'Patch_Attack.ipynb', 5 target images and 2 patch images. Alternatively, it can also be found on My Github Profile in the directory "Adversarial-Attacks-on-CNN-Based-Image-Classification".

Discussion

This Jupyter Notebook demonstrates an Adversarial Patch Attack on a pre-trained MobileNetV2 model on a set of input images. It visualizes the impact of the patch's position on classification accuracy and provides insights into adversarial attacks in image classification.

Here's a summary of the main components and steps:

1. Importing Libraries: The notebook begins by importing the necessary libraries, including TensorFlow, MobileNetV2 from Keras, Matplotlib, NumPy, and OpenCV.

2. Initializing the Model: A MobileNetV2 model with ImageNet weights is loaded.

3. Input Images and Classification: A list of image paths is provided, and for each image, the notebook displays the original image and its top 5 classifications using the pre-trained model. The original classification accuracies and classes are recorded.

4. Adversarial Patch Initialization: An adversarial patch image (patch1.jpg) is loaded and resized to fit the target images. The patch is applied to each input image at a specific position.

5. Classification with the Patch: For each input image with the patch applied, the notebook preprocesses the image and makes predictions using the MobileNetV2 model. The classification results for the patched image are displayed.

6. Repeating with Different Patch Position: The notebook repeats the patch application process with a different patch position (patched2_classification_accuracy). This demonstrates how changing the patch's position affects classification.

7. Data Visualization: The notebook plots a grouped bar chart to compare the top-1 classification accuracy of the original images, images with the patch at position 1, and images with the patch at position 2 for various classes. This visualization shows the impact of the patch on classification accuracy.

8. Conclusion: The notebook concludes by displaying the original classes, original classification accuracies, accuracies with the patch at position 1, and accuracies with the patch at position 2. These values allow for a quantitative assessment of the patch's effect on classification.

Note: We can improve our patch attack if we use customised patches that must be intelligently created. These allow a complete misclassification of the image into the class the adversary desires.

Defense Strategies against adversarial attacks

Defense strategies against adversarial attacks on Convolutional Neural Networks (CNNs) in image classification have been an active area of research. These strategies aim to improve the robustness of CNNs and reduce their vulnerability to adversarial examples. Here are few common defense strategies:

1. Adversarial Training: This involves retraining the model with adversarial examples during the training phase. Adversarial training helps the model learn to recognize and correctly classify adversarial examples. It can make the model more robust but may require more data and computation.
2. Gradient Masking: Some defences aim to hide gradient information from attackers, making it harder to generate effective adversarial examples. Techniques like gradient obfuscation and gradient masking can be used to obscure the model's gradients.
3. Randomization: Adding random noise or perturbations to input data during training or inference can disrupt the patterns that attackers rely on to generate adversarial examples. Randomization can make it more challenging for adversaries to craft effective attacks.

It's important to note that no single defense strategy is foolproof, and many defenses have their limitations.

References

https://arxiv.org/abs/1412.6572

https://machine-learning-and-security.github.io/papers/mlsec17_paper_27.pdf

https://www.youtube.com/watch?v=PFS9KQcQT-s&ab_channel=FedericoBarbero