Model Hardening against Adversarial Attacks: A Novel Approach

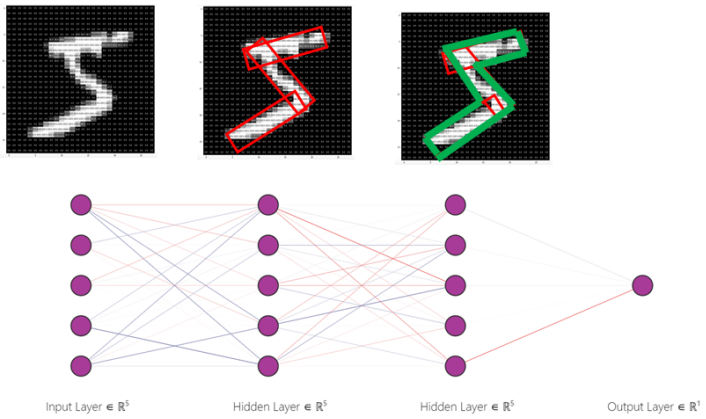
Kshitij Bharat Sanghvi (kbd391) | Diogo Vieira (dcv7396)
Brevity has been maintained to fit the report in 6 pages.

1. Introduction

As of late, we have seen the number of Deep Neural Networks(DNN) applications grow at an exponential rate throughout various fields. With the emergence of DNN's success, one issue that stands out, of course, is security and consequently the different possible attacks that threaten to affect systems that use them. The attacks we will focus on are known as Adversarial Attacks, which are possible due to a specific vulnerability of DNN: over-sensitivity to noise. This allows an attacker to carefully construct imperceivable noise that, when added to some input, can fool a neural network into making a wrong prediction with high confidence. More specifically, we will show how Adversarial Attacks practically affect inferencing, and how our proposed solution tries to solve, or at least minimize this vulnerability and also serves as a regularization technique and leads to better generalization.

2. Background

To understand our approach, we provide a rough intuition on how neural networks work and what it means when a neural network learns. Mathematically, learning in a neural network can be synonymous to cost function minimization. However, several geometrical intuitions have served in the past decade. One of these interpretations tells us the role of each layer in a neural network and how deeper layers employ this knowledge.



Intuition behind neural networks.

As we can see in the above image, we assume that the initial layers try to capture smaller patterns and output the activations to deeper layers that build upon these smaller patterns to identify larger patterns.

Now that we have this understanding, we also wish to outline what does it mean that a neural network has learned a particular concept. A neural network merely learns to distinguish. i.e Let A and B be two labeled instances from the target dataset. The model learns to separate A and B. The model has no notion of what A is. This is why the induction of noise wrecks havoc with the inferencing. This the vulnerability that attackers can exploit.

3. Possible Attack Vectors

The above stated vulnerability can be deemed crucial as follows. Let's take an autonomous vehicle that has the autopilot feature. The vehicle has an array of image and lidar sensors that act as input sources for our neural engine. The cameras capture the surroundings. An attacker can easily tamper with the surroundings. For instance, a Tesla vehicle was tricked into speeding by researchers who put electrical tape over a speed limit sign. That accelerated the car from 35 to 85.

Incorrect Inference

Another example.

Stop sign inferred as 45 mph sign due to the black strips (cutout)

The drawback of this attack is the blatant visual perturbation. We can make this better by introducing perturbations that are not visible to the naked eye. Described as follows

4. Our Novel Attack Vector

Most speeding signs can be assumed to be greyscale. For our perturbation to be stealth it should not be visible to be the naked eye but should wreak havoc on the camera sensor. This can be done especially when the cameras have IR sensors. There exists thin surface coatings that can create false representations, [A Thermal Radiation Modulation Platform by Emissivity Engineering with Graded Metal–Insulator Transition]. Even if there is an absence of false representation, a certain level of noise interference is sufficient to fool the DNN. Thus, planting such film or coating will lead to serious ramifications as they will go undetected for an extended period of time.

We can actually increase the severity of our novel attack. Let's say the speed limit in a school zone is 20mph. The attacker can paste a thin transparent film by varying the reflectance on the speed board that is specifically designed to induce noise on the camera sensor. We can alter the inference to make 20mph to 60mph. Now, since children can suddenly come into the field of view, the car will not have sufficient distance to stop due to incorrect inferencing of the speed limit. Since this is not visible to humans, such attacks can be hard to discover. Sometimes introducing small IR blasters can also wreak havoc on the image sensor and lead to incorrect inferencing.

5. Proposed solution

Adversarial machine learning (ML), and similarly adversarial deep learning (DL) in general, refers to techniques that aim to fool models by feeding them misleading input, usually with the intent of causing the model to malfunction. Model hardening refers to the technique of making these models immune to such adversaries.

In our examples, we will focus on a type of Adversarial Attack where the attacker generates adversarial images - an image that is designed to look ordinary to the human eye but is misclassified by the model. This is commonly accomplished by taking legit image input and adding imperceptible noise to it causing misclassification.

We accomplish model hardening by retraining the model using these adversarial images. We introduce novelty by employing segmentation and using only the noise-induced foreground from the dataset.

There are a few types of noise that we will be dealing with, more specifically, cutout, salt-and-pepper, and gaussian noise.

1. Cutout is the most simple and apparent kind of noise, and it refers to the masking of square regions of an image;

2. Salt-and-pepper noise involves the presence of random white and black pixels throughout the image. This can easily be mitigated as shown in midterm-report.

3. Gaussian noise differs from the other two forms of noise mainly by being a statistical kind of noise, using a normal/Gaussian distribution's probability density function to determine the noise at each pixel.

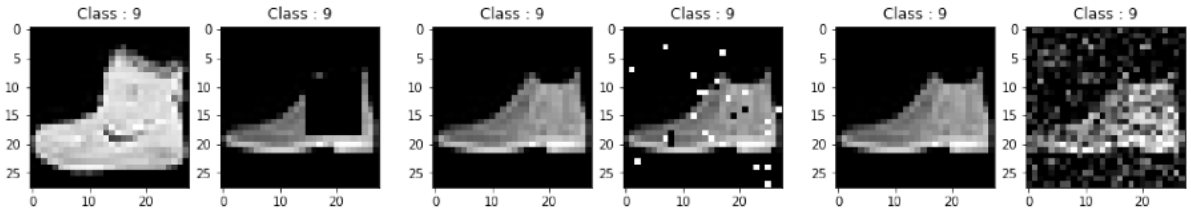
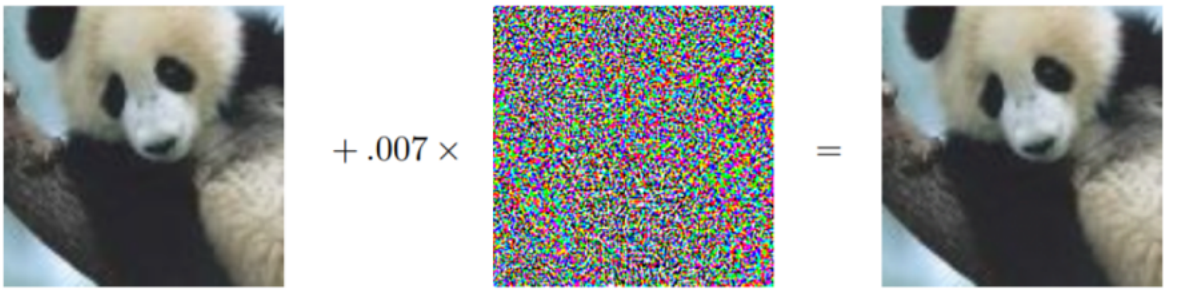


Figure 4: Noise Visualisation on FashionMNIST: Cutout, Salt Pepper and Gaussian

(Note that here the noise is apparent as the image resolution is poor. For instance, if we see the figure provided in the example below, we cannot differentiate noise from the actual image. Additionally, FashionMNIST is gray-scale which adds to the blatant visibility.)



To provide a practical instance, in ICLR 2015, "Explaining and Harnessing Adversarial Examples", researchers at Google (Ian J. Goodfellow, et.al) and NYU provided the following example. When explaining adversarial attacks, they used a neural network that would classify a certain image as a panda with 57.7% confidence. After adding carefully constructed imperceivable noise by exploiting the gradient of the cost function concerning the input misleads the neural network into classifying it as an image of a gibbon with 99.3% confidence.

6. Defense - Model Hardening

Our solution to avoid these attacks and strengthen systems against the exploitation of the discussed vulnerability involves implementing data-augmentation techniques such as cutout regularization and image segmentation to extract foreground and adding noise to train and fine-tune the networks making them robust and less prone to adversarial attacks.

Cutout regularization, as the name indicates, will help with the presence of cutouts on the images that are given as input later on. This is a technique that, during the training phase, augments the model's training dataset to include, not only the original images from that dataset but also several copies with randomly masked square regions.

The image segmentation technique referred to above aims to differentiate between foreground and background of an image, and just as in cut-out regularization, use not only the original images from the dataset but also the detached foreground to train the model.

Batch augmentation is a technique that replicates instances of samples within the same batch with different image transformations as stated in, “Augment your batch: better training with larger batches”, Hoffer et al.

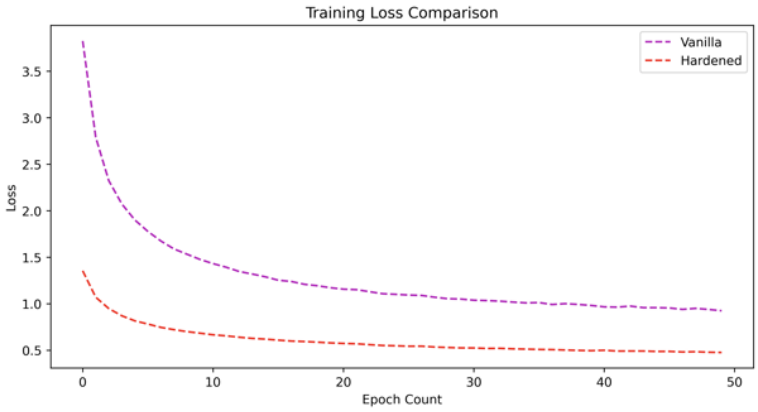
This way we can think of our solution as a 3 tier operation, open to the possibility of future additional layers. The layers we implemented were the following:

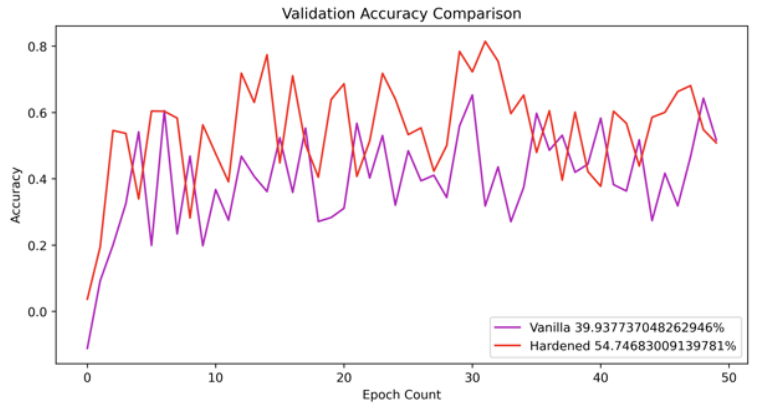
- Tier 1: Cutout Regularization
- Tier 2: Batch Augmentation
- Tier 3: Foreground Extraction

Using these techniques in conjunction to train the model, we achieved the best performance (validation accuracy) as shown in the following figure:
The model used for this experiment was ResNet-44 [explained in presentation], which is a residual neural network and has a structure based on known pyramidal cell constructions in the cerebral cortex.

7. Results

We now provide our empirical results. As we can see that the lowest training loss is achieved when the model is hardened. Though, it seems that the techniques seem detrimental when used in conjunction, it is not true. We can see that on training loss. We can see that the hardened model gives us a boost of 15%, which in our opinion is high given the stochastic nature of the noise.





8. Proposed solution - Why it works, Our interpretation

Though our intuition is backed by empirical evidence and no mathematical guarantees can be asserted with such deep neural networks, we conceptually try to justify why our method works.

By employing cut-out regularisation and batch augmentation we know that we are mitigating overfitting which is evident by the validation accuracy being higher in cases with lower lower training accuracy. These are also supported by the respective paper publications.

The reason why our segmentation technique works in our opinion is because by the above-stated intuition of how learning takes place layerwise, we can assume that the background to be nothing but noise. By showing only the adversarial foreground we make our model more perceptive to the foreground and hence, produce better results. Moreover, with the foreground extraction we are making the model “see” what it needs to know, that helps our model to outperform vanilla implementation.

9. Conclusion

In conclusion, we would like to say that though we have elevated training time, it will never be as important as the accuracy and robustness of the model due to the simple fact that you train once but infer always. Of course, drift detection and retraining happen from time to time in the deep learning lifecycle. In other words, by implementing our technique we may expect elevated training time corresponding to elevated cost but it is a small price to pay to develop an immunity against adversarial attacks.