ECE-GY 6143 - Machine Learning Project

Analyzing impacts of Attacks and Counters on Image Classifiers

Rigveda Vangipurapu - rv2205
Tanmay Khot - tsk9863

Quad Chart

Objective	Work done
• To perform image classification for a custom dataset using Deep Learning • Understand the impact of adversarial attacks, image perturbation, and image compression for performing image classification • Machine learning (Deep Learning specifically) methods work better for pattern recognition in images compared to other algorithmic approaches.	• Created a custom dataset having 5 classes, 500 images per class using Flickr API and another dataset having only adversarial attacked images • Used Transfer learning to perform image classification using VGG and ResNet models and compare their performance • Used Fast Gradient Method and Projected Gradient Descent Method adversarial attacks on images • Implemented methods for adding random noise and image blurring • Implemented PCA and JPEG compression for image compression
Results	**Future scope**
• PGD performs better than FGSM in terms of fooling the classifier whereas FGSM produces more similar images to the original input • PCA reconstructed images fool the classifier even though they appear very similar visually compared to original images • Blurring the images in order to suppress the effect of adversarial attacks did not work as expected • A new dataset was generated containing adversarially attacked images which can be used for training and testing future image classification models	• Train models with adversarial samples to make them more robust • Experiment with more adversarial attacks • Test on a larger dataset with more image classes • Try image compression methods like JPEG compression with attacks • Try advanced adversarial attack defense methods such as Defensive Distillation, Binary Input Detector, Binary Activation Detector

Introduction

The task of identifying what an image represents is called image classification. It involves assigning labels to images according to their types
Today, when it comes to image data, ML algorithms can interpret images the same way our brains do and hence are widely used for image classification. An image classification model is trained to recognize various classes of images.

Whilst this is a robust solution, there can be many potential problems that the image classification algorithms could face, which constitute their drawbacks.

When it comes to real-life implementation and deployment of image classification models, sometimes small errors could have a huge negative impact. For example, if a machine learning model is deployed as part of a self-driving car architecture, minor misclassification could result in major mishaps.

Many deep learning models are reliable with very high accuracy. Yet making alterations to inputs in the form of tiny changes that are typically imperceptible to humans can flummox the best neural networks around.

Few such techniques used to trick the deep learning models are:

- adversarial attacks
- image perturbation

We then analyze the impact of the above ‘attacks’ on the performance of image classifiers.
We also analyze how we can counter the adversarial attacks such as:

- image compression
- blurring

Data Engineering

Entire data for this project has been sourced from Flicker.com. Flickr is a popular photo-sharing and hosting service with advanced and powerful features. It supports an active and engaged community where people share and explore each other's photos.
Flickr also supports data exchange using python with the help of FlickrAPI.

We chose the following classes as a part of this project:

- ambulance
- golden retriever
- Persian cat
- school bus
- traffic light

500 images for each of the following classes were downloaded from Flickr using Flickr API, ordered by the relevance of the data.
The first layer of the models we used (ResNet and VGG) needs the dimensionality of the data to be (224,224), so every image was resized accordingly before saving them.

Each image was manually eyeballed to identify and remove irrelevant images that might have accidentally been added while downloading the data. These images were then stored in separate folders according to the classes to that they belong.

Once all the images were saved, each image is converted into an array, reshaped, and preprocessed before feeding into the classifier.

Work Done

Basic image classification:

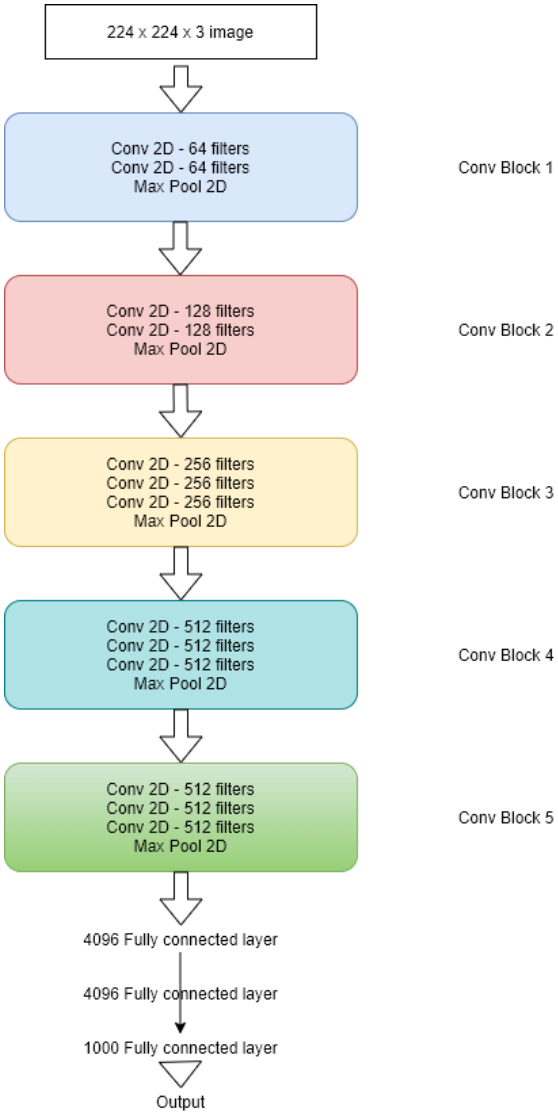
Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to implement neural networks easily.

We selected 2 pre-trained deep learning networks from Keras for this project:

- VGG16
- ResNet

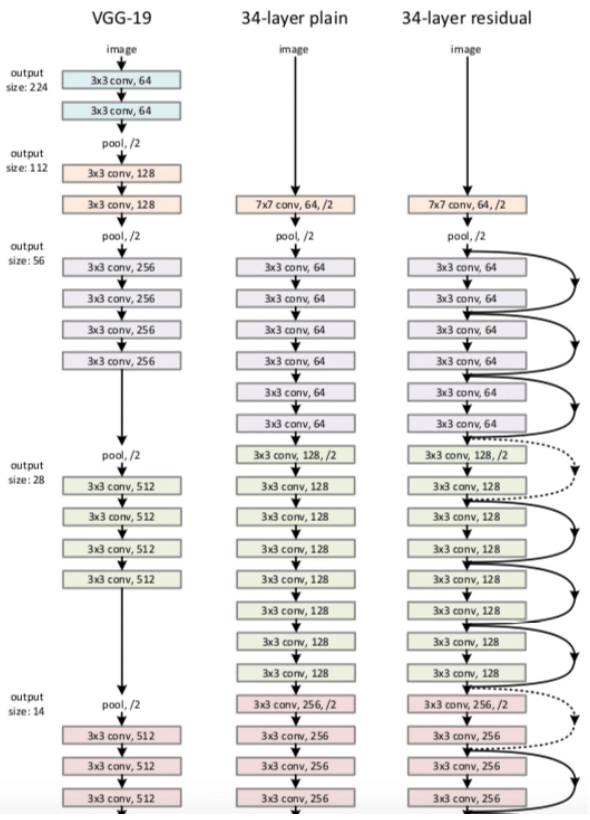
These models are trained on the Imagenet dataset. ImageNet is a large dataset of annotated photographs. There are a little more than 14 million images in the dataset, a little more than 21 thousand groups or classes (synsets), and a little more than 1 million images that have bounding box annotations.

VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The network has learned rich feature representations for a wide range of images.



source : https://towardsdatascience.com/creating-vgg-from-scratch-using-tensorflow-a998a5640155#:~:text=VGG%2016%20architecture%20and%20implementation%20using%20Tensorflow%3A&text=VGG%20network%20uses%20Max%20Pooling,with%20a%20stride%20of%202.

Resnet is a convolutional neural network that can be utilized as a state-of-the-art image classification model. The Resnet models have been pre-trained on the ImageNet dataset as well. Tiny ImageNet alone contains over 100,000 images across 200 classes. The large ImageNet dataset contains a vast array of image classes. Skip-connections are one of the most salient features of the ResNet architecture. Skip Connections (or Shortcut Connections) as the name suggests skips some of the layers in the neural network and feeds the output of one layer as the input to the next layers, thus avoiding degradation in training accuracy.



source: https://adventuresinmachinelearning.com/introduction-resnet-tensorflow-2/

We performed several experiments based on image classification using these 2 models. Transfer learning was used to perform the initial classification and the results were stored in resvgg and resresnet dictionaries respectively. These will be used in the future the compare the effects of various attacks on the classification.

Attacking the image:

1. **Image perturbation - Adding random noise**

Image noise is an undesirable by-product of image capture that obscures the desired information. Image noise is a random variation of brightness or color information in images.

We attempted to deliberately generate noise in the images and analyze the effect on both the classifiers.

The addition of random noise was implemented as follows:

- 4 random numbers: r1, r2, r3, and r4 were generated.
- r1 value ranges from 0 to 224
- r2 value ranges from 0 to 224
- r3 value is picked as among 1,2,3
- r4 value ranges from 0 to 224

r1, r2, and r3 are used to access one particular pixel of the image. This pixel value is then changed to r4.

This change is done to n% of the pixels to see the results.
The value of n is varied from 0 to 40 and the corresponding score, accuracy, and structural similarity index measure (SSIM) are calculated. Corresponding graphs are plotted to show the trends.

2. **Adversarial attacks**

Adversarial machine learning, a technique that attempts to fool models with deceptive data, is a growing threat in the AI and machine learning research community. The most common reason is to cause a malfunction in a machine learning model. An adversarial attack might entail presenting a model with inaccurate or misrepresentative data as its training or introducing maliciously designed data to deceive an already trained model. In our project, we have used pre-implemented adversarial attacks from the Adversarial Robustness Toolbox library.

We planted 2 adversarial attacks:

- FGSM
- PGD

FGSM

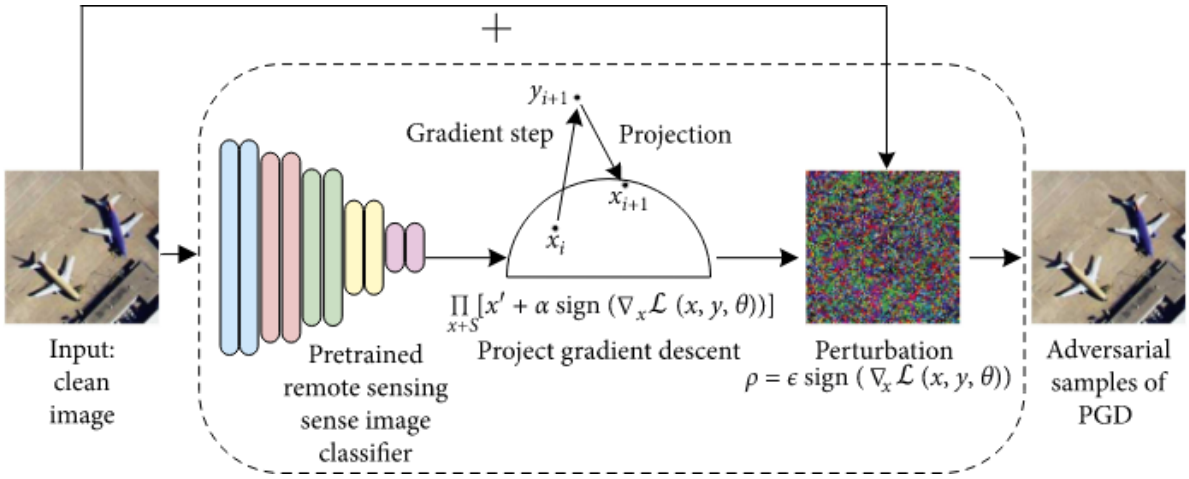
One of the first and most popular adversarial attacks to date is referred to as the Fast Gradient Sign Attack (FGSM). The attack is remarkably powerful and yet intuitive. It is designed to attack neural networks by leveraging the way they learn, gradients. The idea is simple, rather than working to minimize the loss by adjusting the weights based on the backpropagated gradients, the attack adjusts the input data to maximize the loss based on the same backpropagated gradients. In other words, the attack uses the gradient of the loss w.r.t the input data, then adjusts the input data to maximize the loss.

$$X_{Adversarial} = X + \varepsilon \, . sign\left(\nabla_X J(X, Y)\right),$$

where ε is small number and ∇ is the gradient of cost function with respect to X

PGD

The Projected Gradient Descent (PGD) attack is a white-box attack which means the attacker has access to the model gradients i.e. the attacker has a copy of your model's weights. This threat model gives the attacker much more power than black-box attacks as they can specifically craft their attack to fool your model without having to rely on transfer attacks that often result in human-visible perturbations. PGD can be considered the most "complete" white-box adversary as it lifts any constraints on the amount of time and effort the attacker can put into finding the best attack.



source: https://www.hindawi.com/journals/scn/2021/6663028/

Attempts to counter the attacks:

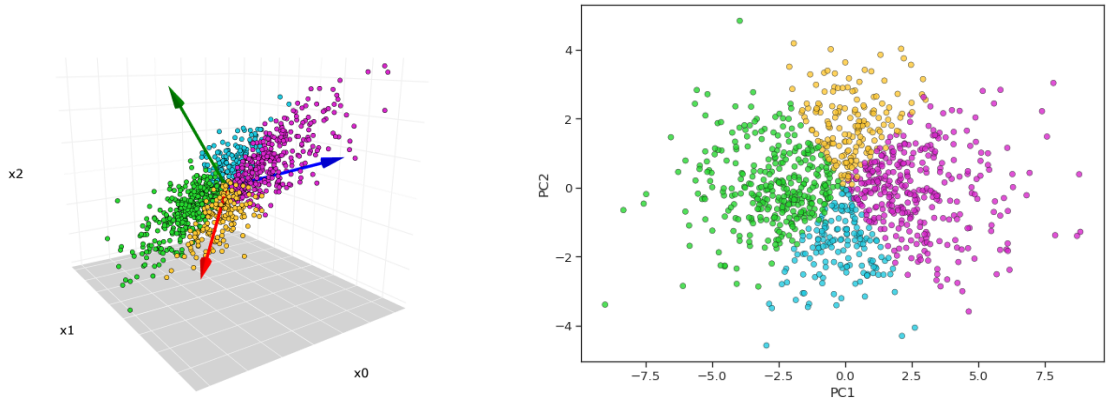
- **Image compression - PCA**

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. It is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

One of the use cases of PCA is that it can be used for image compression — a technique that minimizes the size in bytes of an image while keeping as much of the quality of the image as possible.

It is also known as a general factor analysis where regression determines a line of best fit.

PCA reduces the images to a lower-dimensional representation and the original image can be obtained back from the compressed image using some mathematical calculations.



source: https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

The transition relations can be written as:

$$\mathbf{u} = \frac{1}{\sqrt{\lambda}}\mathbf{Z}^{\mathsf{T}}\mathbf{N}^{1/2}\mathbf{v}$$

- **Blurring**

$$\mathbf{v} = \frac{1}{\sqrt{\lambda}}\mathbf{N}^{1/2}\mathbf{Z}\mathbf{u}$$

Blurring was used as an attempt to remove the noise from the attacked images.

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

The blur method of OpenCV was to used blur the images based on a given kernel size k.

The value of k is varied over a set of values and the corresponding score, accuracy, and structural similarity index measure (SSIM) is calculated. Corresponding graphs are plotted to show the trends.

Metrics used

- **Accuracy**

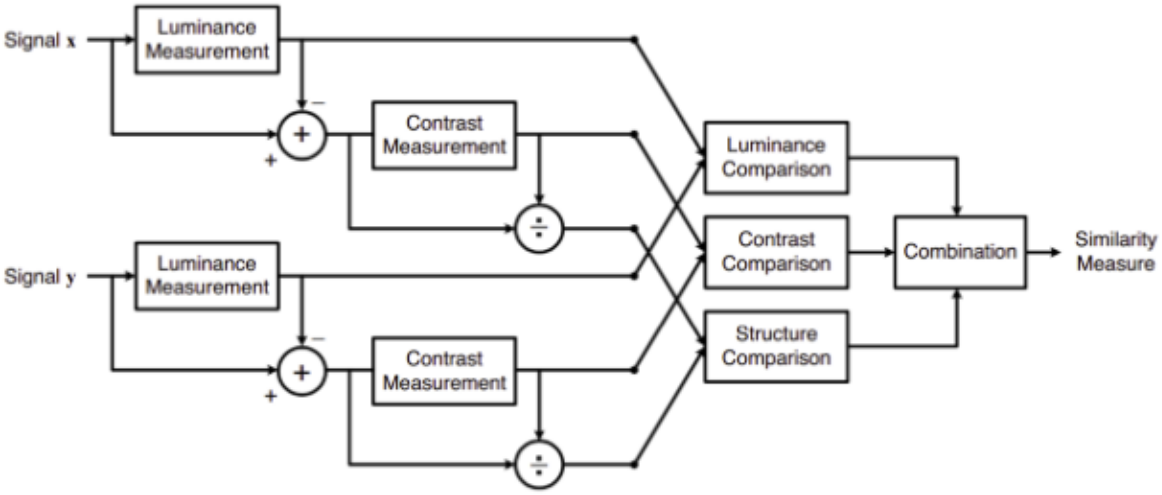
  This is a very straightforward way of computing the correctness.
  The original labels generated by the model are stored. After any alteration is done, a comparison is made between the original classification result and the new image class label. If both the labels match, the score is +1, else the score is 0.

  After computing score for the whole data, accuracy can be computed as

  $$score \ / \ total \ images$$

- **SSIM**

  The Structural Similarity Index (SSIM) metric extracts 3 key features from an image: Luminance, Contrast, and Structure. The combination of considering multiple factors gives us a better overall result to know if the two images are similar. Mean Squared error fails to work even if th image is shifted by one pixel since the relative position of the pixels will vary. We used SSIM metric in our project to compare the new generated images from reconstruction using PCA, JPEG Compression, adversarial attacks, blurring and random noise addition with the original image.

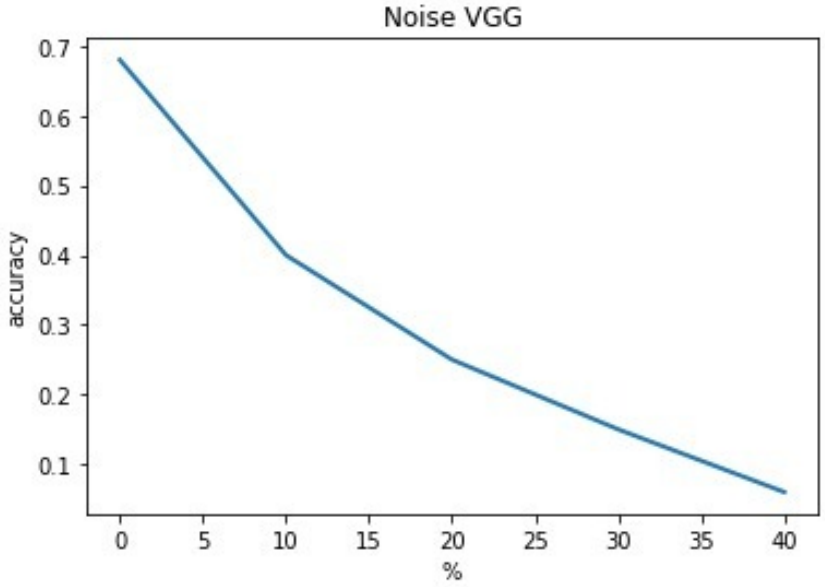


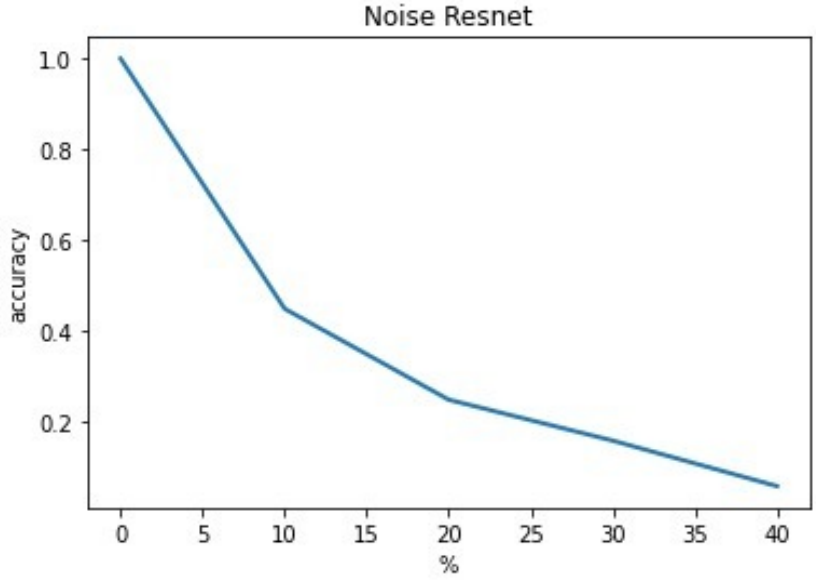
source: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

We chose SSIM over MSE because SSIM considers several factors into consideration other than comparing direct pixel values. Moreover, it is easy to interpret since its outputs are always between 0-1, 1 for the same images.

Results

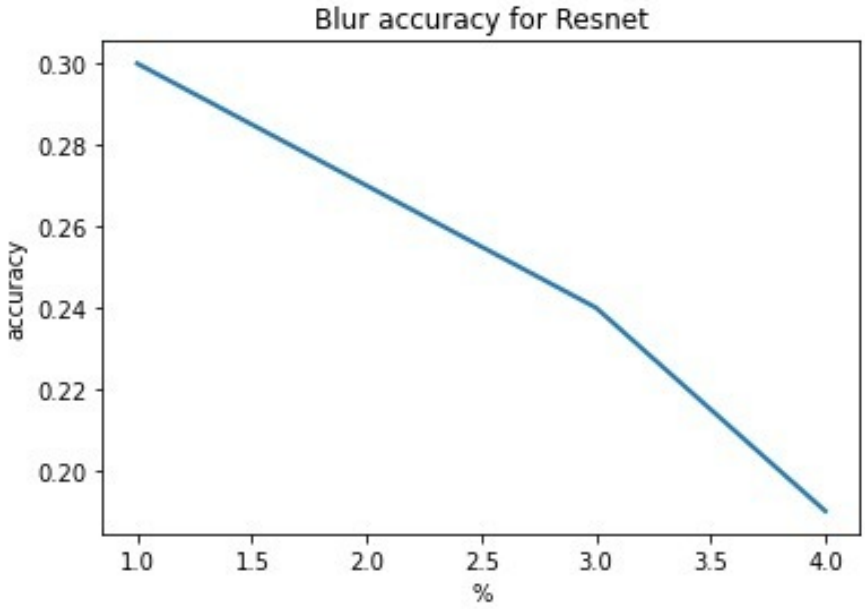
These are the effects of adding noise to images and using the preexisting classifiers to reclassify them

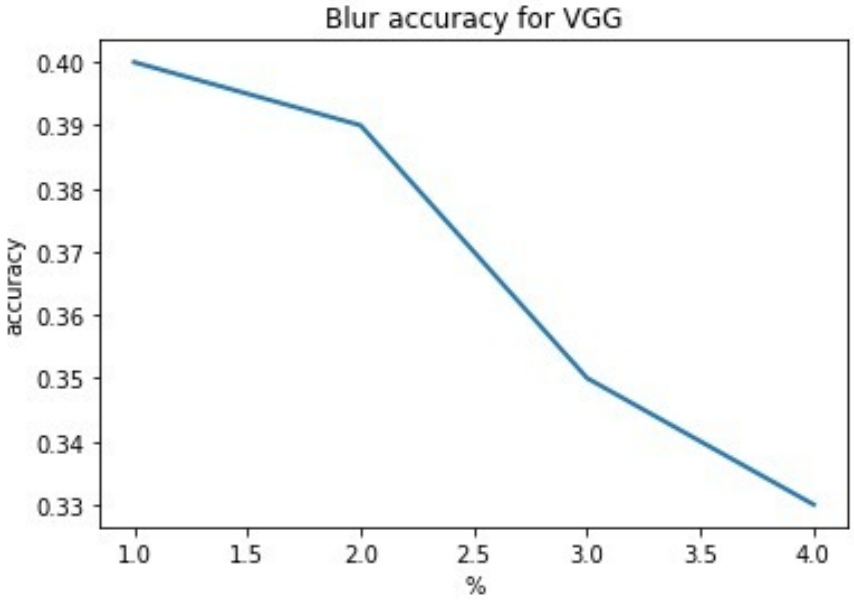




In the case of blurring, we expected that blurring the image would reduce the impact of the attacks and thus result in better classification of the attacked images.

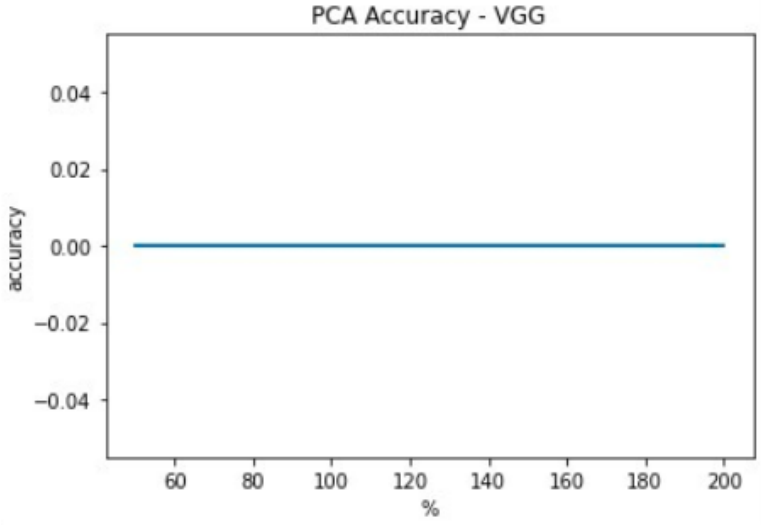
But after implementation, we observed that the classification results were poor even after blurring the images.

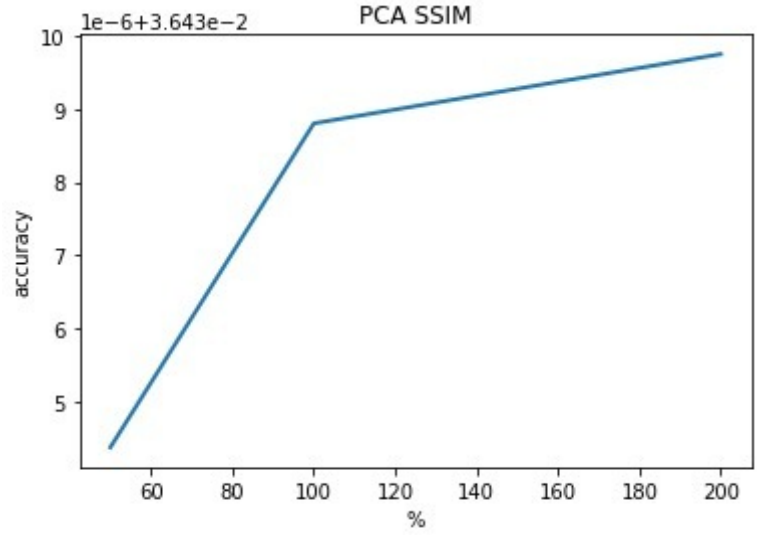


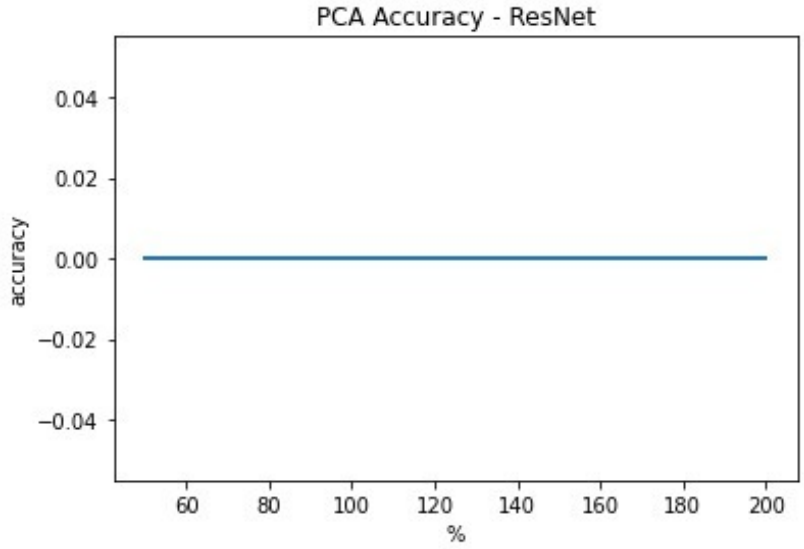


In the case of PCA, we expected that while the image is being restored from lower-dimensional representation to the original dimension, the effect of the attacks will be minimized and the misclassifications would be minimized.

While implementing PCA, it failed to correctly classify the images even before the attack was planted, so the approach seemed redundant since the base accuracy itself was very low.

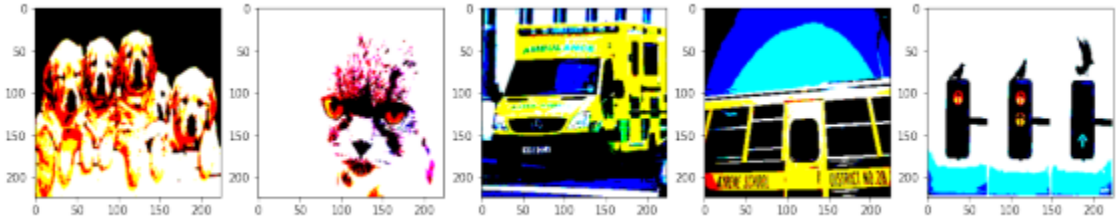

PCA Accuracy - VGG
0.04
0.02
0.00
−0.02
−0.04
accuracy
60
80
100
120
140
160
180
200
%


PCA SSIM
1e−6+3.643e−2
10
9
8
7
6
5
accuracy
60
80
100
120
140
160
180
200
%


PCA Accuracy - ResNet
0.04
0.02
0.00
−0.02
−0.04
accuracy
60
80
100
120
140
160
180
200
%

Original data

Adversarial Samples

Random noise


Conclusion

1. Our idea was, to use PCA to reduce the dimensions of adversarially attacked images and to check if the new generated image still had an effect of the attack. According to our experiments, PCA retains the visual features of the image but completely changes the structure of the images internally which can fool the classifiers easily. Thus, if the images are to be used for any ML process, the images should not be compressed using PCA. Methods like JPEG Compression could perform better

2. Our idea was to blur adversarially attacked images to supress the effect of the attacks, however blurring didn't help much as it blurred the image too much.

3. Adversarial attacks produce images that are visually very similar to the original images and in fact have a very high SSIM score as well. In order to improve the model's classification towards adversarial attacks, the models should be trained with adversarial samples to improve its robustness

4. PGD is more powerful than FGSM according to the results. The more accuracy is significantly less on images attacked by PGD and the SSIM score for PGD images is still relatively high

5. In many cases it was observed that ResNet outperforms VGG which was expected.

6. A new dataset containing adversarial images can be generated using the code segments provided. This can help in training and testing new models.

Future Scope

With more computational power and access to a larger dataset, we can train models with adversarial samples to make them more robust. Along with this, we can also experiment with more adversarial attacks and understand their impact. For image compression, methods like JPEG compression could perform better than PCA. There is also a scope to experiment with some advanced defence methods such as Defensive Distillation, Binary Input Detector, Binary Activation Detector

References

1. https://www.nature.com/articles/d41586-019-03013-5
2. https://towardsdatascience.com/main-challenges-in-image-classification-ba24dc78b558
3. https://nanonets.com/blog/machine-learning-image-processing/#what-is-image-processing-and-why-is-it-important
4. https://www.tensorflow.org/lite/examples/image_classification/overview
5. https://venturebeat.com/2021/05/29/adversarial-attacks-in-machine-learning-what-they-are-and-how-to-stop-them/
6. https://www.mathworks.com/help/deeplearning/ref/vgg16.html;jsessionid=3bfab5a24314fa96dcbf44e4f1af#:~:text=VGG%2D16%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals.
7. https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/#:~:text=The%20ImageNet%20dataset%20is%20a,for%20developing%20computer%20vision%20algorithms.
8. https://towardsdatascience.com/creating-vgg-from-scratch-using-tensorflow-a998a5640155#:~:text=VGG%2016%20architecture%20and%20implementation%20using%20Tensorflow%3A&text=VGG%20network%20uses%20Max%20Pooling,with%20a%20stride%20of%202.
9. https://builtin.com/data-science/step-step-explanation-principal-component-analysis
10. https://pytorch.org/tutorials/beginner/fgsm_tutorial.html
11. https://www.hindawi.com/journals/scn/2021/6663028/
12. https://pca4ds.github.io/formulas-for-pca.html
13. https://towardsdatascience.com/dimensionality-reduction-of-a-color-photo-splitting-into-rgb-channels-using-pca-algorithm-in-python-ba01580a1118https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e
14. https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf
15. https://adversarial-robustness-toolbox.readthedocs.io/en/latest/