Adversarial Defenses for CNN-based Camera Model Classifiers

A thesis submitted in the partial fulfilment

of the requirements for the degree of

Master of Technology

by

Satyaki Ghosh

Signal Processing and Machine Learning

Roll No: 204102311

Under the guidance of

Dr. Prabin Kumar Bora

June, 2022

Department of Electronics and Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY, GUWAHATI.

Declaration

I hereby declare that the work presented in this thesis entitled ***Adversarial Defenses for CNN-based Camera Model Classifiers*** *towards partial fulfillment of the requirements for the award of degree of Master of Technology in Signal Processing and Machine Learning at the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, is an authentic record of my own work carried out under the guidance and supervision of* ***Dr. Prabin Kumar Bora*** *and refers other researchers which are duly listed in the reference section. The contents of the thesis, in full or parts, have not been submitted to any other Institute or University for the award of any degree or diploma.*

Satyaki Ghosh

June, 2022
Guwahati.

Department of Electronics & Electrical Engineering,
Indian Institute of Technology Guwahati, Assam.

Certificate

This is to certify that the work contained in this thesis entitled ***Adversarial Defenses for CNN-based Camera Model Classifiers*** *is a bonafide work of* ***Satyaki Ghosh*** *(**Roll No.204102311**), carried out at the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Prabin Kumar Bora**
Professor

June, 2022
Guwahati.

Department of Electronics & Electrical Engineering,
Indian Institute of Technology Guwahati, Assam.

Acknowledgement

First of all, I want to express my gratitude towards my supervisor **Dr. Prabin Kumar Bora** for his constant supervision and valuable guidance. He consistently allowed this project to be my own work. He guided me in the right direction whenever he thought I needed it. His commitment and continuous insistence on revising the basics thoroughly aided me in doing a proper literature survey. I also thank **Mr. Rijuban Rangslang** for guiding me to choose the relevant research papers and understanding them. I also want to thank all the researchers who have done astonishing work in this field, which helped me a lot to understand the problem and its challenges properly.

Finally, I want to thank my parents and also my friends for providing me with unparalleled support and continuous encouragement throughout my academic journey, especially in these tiring times of the pandemic.

Abstract

Various machine-learning and deep-learning models, including Convolutional Neural Networks, misclassify adversarial inputs. These are formed by adding small perturbations to the examples from the dataset, such that these intentional modifications result in the model giving a wrong answer with relatively high confidence. More importantly, especially for CNNs, these attacks are very often transferable to other classifiers. The changes in these adversarial images are almost visually imperceptible. Image forensics tries to identify processing or editing history of an image from the traces left by image manipulations. Here, we are focusing on camera-model identification. With the improvement of compute, several CNN models have become popular in image forensics also as in many other fields, one of them being the DenseNet for classifying camera-models. But these different architectures also suffer from adversarial vulnerability. One of the most effective defences against adversarial images is adversarial training, where the classifier is trained with normal and adversarial perturbed images. But adversarial training takes a very long time due to a lot of gradient computations with respect to the pixel values. So, some fast adversarial training methods, combined with some other adversarial attacks, have been tried on the task of camera-model identification. Another approach that has been tried is a linear filter, which is similar to steganalysis methods. On five camera-models have been used from the VISION dataset, the normal training accuracy was ***91.6****% and against adversarial examples it was* ***43.8****% after adversarial training, which is as good as that obtained in image classification tasks. And with the proposed filter, the robust accuracy was* ***58.1****% but with the training accuracy dropping to* ***89.6****%.*

Keywords: Adversarial Images, Convolutional Neural Networks, Image Forensics, Camera-model identification, Adversarial Attacks, Adversarial Training, Steganalysis

Contents

List of Figures

List of Tables

1. Introduction

Camera-model classification is an important forensic application that tries to find out the brand and model of the camera used to take a given picture. This can be helpful when the origin of some photos related to unlawful activities is being investigated. In social-network sites also, people upload hundreds of thousands of photos daily. So, there also, camera-model classification can be used to identify the author of uploaded illegal images. The meta-data provided in the header file of the image can be edited easily. So, it is necessary to classify the images based on the content of the image itself. The camera-models leave a distinct trace, which can be used by the classifier. CNN models have been very successful in image classification tasks. And specifically for camera-model classification, the DenseNet architecture have been quite useful[9]. So DenseNet has been used as our classifier[3]. For training it, we have used five classes from the VISION dataset, containing images from 30 different smartphone models[10].

Several machine-learning models and deep-learning models like Convolutional Neural Networks are vulnerable to adversarial attacks [11]. They tend to misclassify inputs which are slightly different from the training distribution, but these perturbations are made in such a way that these models wrongly classify them with quite high confidence. For CNNs used to classify images, the modifications in these images are almost impossible to perceive visually. There are several methods for generating these adversarial images like Projected Gradient Descent (PGD) [12] which is one of the strongest attacks and Fast Gradient Sign Method, which is one of the fastest attacks [4]. Moreover, these attacks can be transferred from one model to another, i.e., the adversary may produce adversarial images against a substitute model and transfer them to the actual target or victim model with high attack success rate [13], [14]. When this is done with very little knowledge about the target model, it is called a black-box attack, otherwise it is called a white-box attack.

In order to defend the different machine-learning and deep-learning models against these attacks, several defenses have been proposed. Recently, *Athalye et al.* [15] proved that

many of these popular defenses which use gradient masking can be easily broken. However, adversarial training [12] still stands as one of the most powerful defenses. Here, the classifiers are trained with normal as well as adversarially perturbed images.

Camera-model classification is used in forensic tasks for source identification of images during investigations [16]. So, it is essential to make the models extremely robust against adversarial attacks [17]. For adversarial training, PGD training is considered the best since it is the strongest attack. However, crafting these adversarial examples is computationally expensive. For example, on the VISION dataset, the time taken for full training was approximately 57 hours on a Tesla-V100 GPU. Each epoch almost took 1.5-2 hours and the initial epochs even took as much as 5 hours. So, we could only run it for 25 epochs. So, this is infeasible for larger datasets or with a greater number of camera-classes. With so many brands each having so many camera-models and new ones coming up every week, the process of making robust classifiers should be fast along with being accurate. So, we look towards some other algorithms which take less time.

Some fast implementations of PGD [18] as well as some even faster and robust enough implementations of FGSM [19] have been proposed as a defense. These are the free adversarial training and fast adversarial training methods respectively. The problem with that is it is not as robust as the PGD trained model.

Another defense mechanism can be inspired from steganalysis, where a steganalyst tries to detect if a given image has a message hidden by a steganographer [8]. A linear steganalysis filter has also been used to detect PGD crafted adversarial examples before passing them onto the classifier. Only if the image is benign, it is before they are fed to the robust classifier. Using these tools, we aim to build a robust defense for a forensic application, specifically camera-model identification, which has been shown to be vulnerable to adversarial attacks.

2. Related Works

2.1 Image forensics using CNN

Image forensics tries to identify processing or editing history of an image from the traces left by image manipulations. With the improvement of compute, several CNN models have become popular in image forensics also as in many other fields. The main problem with forensic algorithms is that designing different algorithms for different image manipulations (for example resizing and resampling, median filtering, JPEG compression, contrast enhancement, etc) is time-consuming and laborious. Moreover, only a generalized manipulation detector can detect multiple types of modifications made in an image. Otherwise, a whole set of detectors would have to be applied on the image.

2.2 Camera-model classifiers

When the light first enters through the lens and filters, the color filter array (CFA) first projects each color to R, G, or B and then the sensor captures it. So, the output of the sensor is single-channel. In order to estimate the image, color interpolation finds the missing color information of the missing channels for each sensor, based on the neighbouring pixel values. Some post-processing like white-balancing, compression may further be involved before the image is digitally stored.

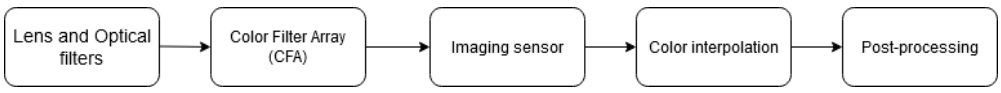


Figure 2.1: Camera-processing pipeline[1]

The camera-processing pipeline has a number of sources as shown in the Figure 2.2. SPN is the noise component which does not get nullified by frame averaging. PRNU is the primary component of SPN. PNU is due to the minute variations in the sensitivity of each sensor, and is the main component of PRNU. PRNU which also contains low-frequency spatial features as LFD, which is not dependent on the sensor (caused by refraction by dust,

vignetting, etc). FPN, the other component of SPN, is the pixel-to-pixel differences when no light is falling on the sensor.

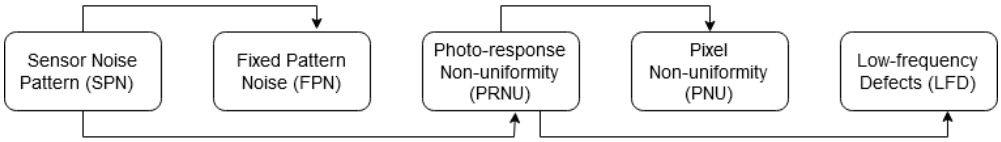


Figure 2.2: Sources of noise in a picture captured by a camera [1]

2.2.1 Constrained-CNN

Bayar et al. proposed the constrained-CNN architecture [2] which can detect detect (a) multiple types of image manipulations and (b)camera-models depending on the application. It has a constrained layer at the beginning which subdues the image's content and learn the manipulation detection features. CNNs generally tend to learn the features which help them to classify objects, but this layer (Fig 2.3) acts as a high-pass with learnable parameters and extracts prediction residual features. From this, the deeper layers learn higher-level forensic features. This works even when there is a difference between train and test distributions.

$$w_k^{(1)}(0,0) = -1$$

$$\sum_{m,n \neq 0} w_k^{(1)}(m,n) = 1$$

where (1) denotes the first layer number in the CNN, k denotes its kth convolutional filter and (m,n) denotes the spatial coordinates within that filter.

The authors got over 97% accuracy for the Dresden database which has 14,000 images from 25 different digital single-lens reflex camera-models (DSLR). We also got a similar accuracy over five camera-models from the Dresden database. In this project, the VISION dataset has been used which has 34,427 photos from 35 smartphone camera-models of 11 brands [10]. In both the cases, only five camera-models were chosen due to the space and time constraints. The training dataset itself was more than 22 GB for just five classes. We have chosen the VISION dataset because now-a-days, photos taken by a smartphone are much more common than DSLR–clicked photos, and they also comprise a very large fraction of the photos uploaded over the social media. The detection of attempted forgery of source camera-model, uploading of illegal photos, etc in such cases of social media will be very useful. So, the VISION dataset has been chosen. However, when the constrained-CNN was tried over the VISION dataset, the accuracy never got beyond 22% which is almost as

good as random guessing with five classes. This shows a marked difference in distribution in the two datasets. The constrained layer extracts the high frequency features from the image, but for photos from mobile devices (as in VISION), this layer seems to have difficulty extracting the features. So, DenseNet architecture [3] was used, which is also quite popular for Camera-model classification, and it worked for VISION [9].

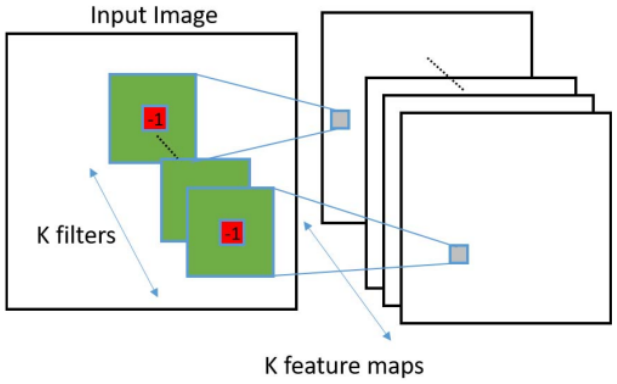


Figure 2.3: Constrained layer [2]

2.2.2 DenseNet

In a standard Convolutional Neural Network, we have an input image. It is passed through the network to get an output prediction in a way where the forward pass as shown in Fig 2.2 below.

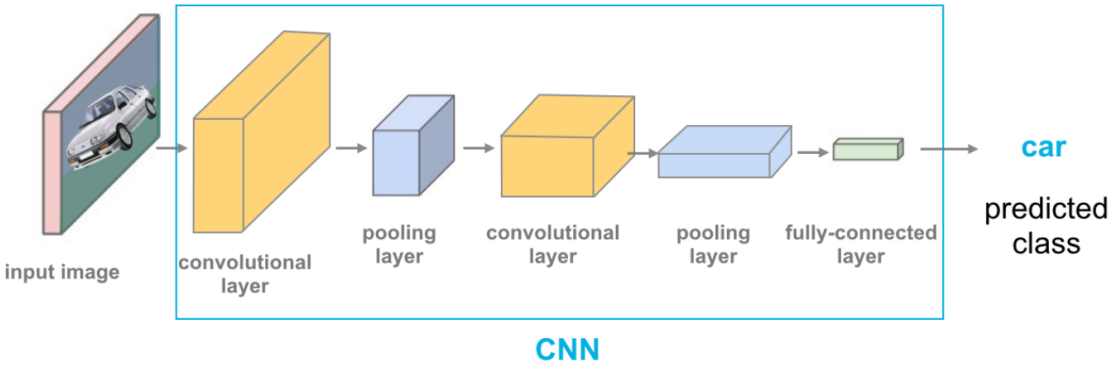


Figure 2.4: CNN [3]

The first layer takes the image as its input. All the other layers take the previous layer's output, produces an output feature map and passes it to the next layer. So, there are L connections for L layers.

However, in DenseNet, there are blocks called denseblocks in which every layer is connected to all the layers after it. So, a denseblock with L layers will have L(L+1)/2 connections.

Each layer gets the output of all the preceding layers' feature maps as input. So, effectively, each layer concatenates its own feature map to the feature map coming from the previous layers and passes it to the next layer. Inside each denseblock, the size of the feature map of every layer is same to facilitate feature map concatenation.

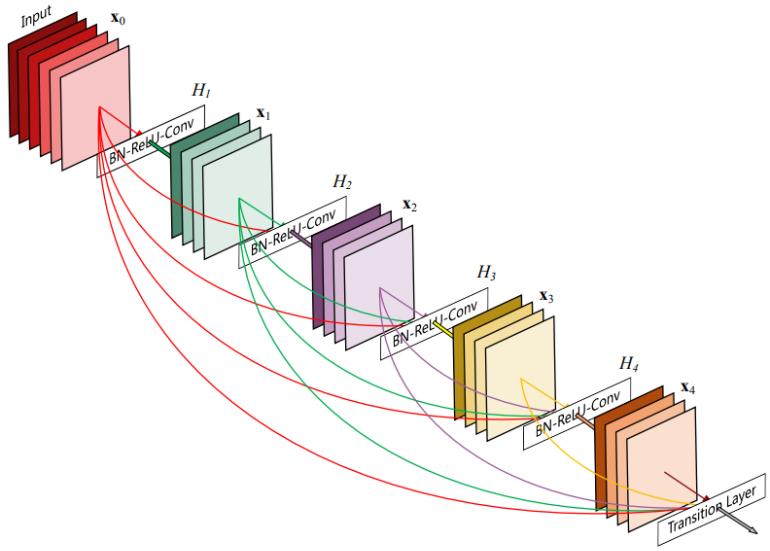


Figure 2.5: A denseblock [3]

Some advantages of these connections are:

- Solves the vanishing gradient problem
- Encourages feature reuse
- Helps feature propagation
- Greatly reduces the number of parameters

There are transition layers in between the denseblocks where there is a batch-norm layer, 1x1 convolution followed by a 2x2 average pooling layer. Each denseblock has multiple denselayers, each of which has two sets of batch-normalization, ReLU and convolution (1x1 and 3x3). Each layer adds K feature-maps to the previous feature maps. As a result, the input the later layers become very high. So, to reduce the computational burden, the 1x1 blocks are used as bottlenecks for dimensionality reduction before the 3x3 layers.

Every DenseNet architecture has four denseblocks. In DenseNet121, the denseblocks have 6, 12, 24, 16 denselayers. For DenseNet201, the corresponding numbers are 6, 12,

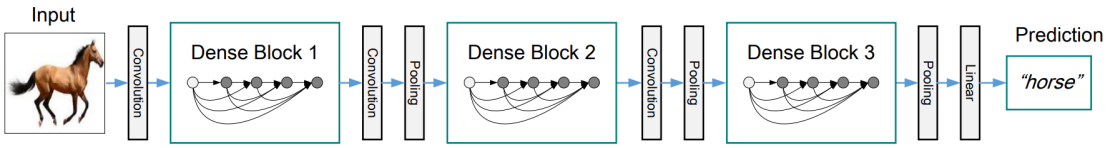


Figure 2.6: DenseNet Architecture [3]

48, 32. The first layer has a 7x7 stride 2 convolutional Layers followed by a 3x3 stride-2 MaxPooling layer. And the last denseblock has a classification block at the end.

This DenseNet architecture has been shown to be very effective in camera-model identification in the Kaggle competition [20] and in [9]. The input patches are of size 224x224x3. We lose the high-frequency forensic micro-patterns and information while resizing an image, which is crucial in the classification. So, random patches are extracted from the images and fed to the DenseNet classifier.

2.3 Adversarial Attacks

Szegedy et al. first found that deep-learning models can be easily made to misclassify inputs when some perturbation is added to it that tends to increase the prediction error of the network [11]. These perturbations are referred to as adversarial attacks. These attacks can of two types based on the target label of the attacked input. If the attacker wants it to be classified into a particular class, it is *targeted attack*, and if the image is modified just to misclassify it into any other class other than its original class, it is called *untargeted attack*. Creating a targeted attack is often more difficult and in some cases may require the image to be modified a bit too much which becomes visually perceptible. Here, the loss function is defined in such a way that the probability of prediction reduces for the original class and increases for the target class. In untargeted attack, the loss function has only one component : negative of the loss between the actual class and predictions. In targeted attack, the loss function has two components : negative of the loss one is between the actual class and predictions, and the other is between the target class and predictions.

These attacks are effective even when they are used in a black-box manner, where the target neural network or machine-leaning model is just used as an oracle to give output the for some inputs crafted on a substitute model. This local model learns from this input-label pairs and the attacks made using this model is also effective on the target model, without having any knowledge about its architecture [14], [13]. Adversarial attacks poses a serious risk to a number of applications, not just limited to images. Some of the possible threats

Figure 2.7: Original images (left) and the adversarial images (right) of each of the five camera-models in each row; the adversarial images are all misclassified

are:

- Different filters like spam filters in Gmail, or filters in Facebook (which prevent people from posting objectionable pictures) could be bypassed by adding a good enough perturbation vector to the original image.
- With autonomous vehicles on the rise, this could pose a risk if cars would misclassify some photo or poster on the road as a pedestrian (or some object to avoid). This could cause serious accidents.

Most of the software packages for adversarial image generation try to minimize the distortion of the image which results in the image being very close to the boundary. But for a black-box attack, the target network's boundary may vary which will make the attack useless. Li et al. showed for *counter-forensic* applications that by increasing the confidence score (the difference between the output (logits) for the targeted class and that of the highest-scored class among other classes), the attacks can be made much stronger.[21] So for binary classification, let class label yϵ0,1 and z_i be the logits. With an input image X, an image $\hat{X}$ is declared adversarial if:

$$z_i - z_{i-1} > c$$

where $c > 0$ is the desired confidence level.

They tried the following three combinations for median filtering and image resizing:

1. Cross-network : Network architectures are different but training dataset is same
2. Cross-training : Network architectures are same but trained on different datasets
3. Cross-network-and-training : Different architectures trained on different datasets

They obtained an Attack-Success-Rate (ASR) of more than 0.8 while PSNR was around 30-40 dB. The ASR starts to decrease with increasing c, and cross-network was more difficult than cross-training.

The attack algorithms for camera-model classification (like DenseNet trained on VISION) work in the same way as object classification (like for ResNets trained on ImageNet). Some of the different attack algorithms are given below [5]:

2.3.1 Fast Gradient Sign Method:

Goodfellow et al. proposed this method as a fast way to craft adversarial images [4].

$$\hat{X} = X + \epsilon * sign(\nabla_X J_\theta(X, y))$$

ϵ is the strength of the attack and the gradient gives the direction of pixel update and y is the label of the original image $\hat{X}$. Compared to L-BFGS, this is faster and more efficient. There is also an iterative version called Iterative-FGSM OR I-FGSM, which updates the pixels not in one go but in several iterations, but finally clipped by ϵ. It uses L^∞ as the distance metric.

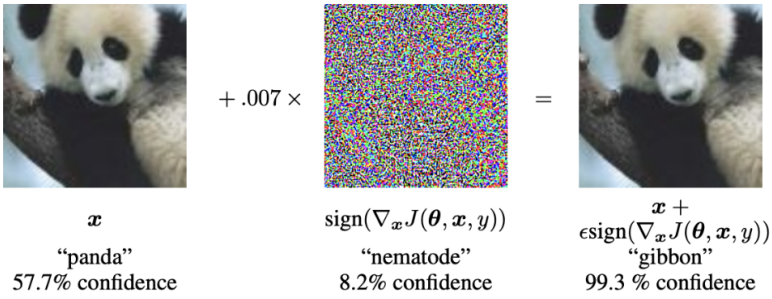


Figure 2.8: An example of FGSM applied to GoogLeNet on ImageNet.ϵ is taken as 0.007 [4]

2.3.2 Projected Gradient Descent:

PGD attack perturbs the image under L^∞ constraints. To calculate$X_{i+1} from X_i$, I-FGSM is used. Then, the image is projected onto the space having constrained L^∞ distortion with the maximum distortion fixed to a certain value α [12].

$$X_{i+1} = \Pi_{X+\alpha}(X_{i+1})$$

2.3.3 L-BFGS:

For a given image X, the objective is to find $\hat{X}$ such that they are equal under some L_2-norm threshold, yet the classifier makes an error. Here l is the target label.

$$Minimize\ c.||X - \hat{X}||_2^2 + loss_{F,l}(\hat{X})\ such\ that\ \hat{X} \epsilon [0,1]^n$$

Cross-entropy is commonly used as the loss function. The constant c is found by solving this problem multiple times and using some 1D-optimization problem like Bisection search.

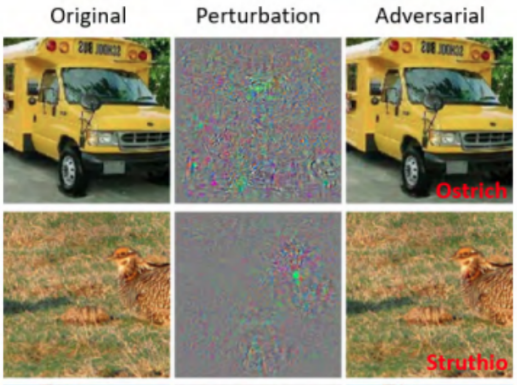


Figure 2.9: An example of adversarial examples is generated for AlexNet [5]

2.3.4 Other attacks:

Some other image perturbations include Poisson noising, Gaussian blurring, adding random noise and removing pixels' least significant bit (LSB). These may misclassify the image, but often these lead to visible changes in the image, unlike the two attack methods mentioned above [22].

2.4 Adversarial Defenses

Various defense techniques have also been proposed over the course of time and some of them have been really successful, like Adversarial training [12], however some methods like gradient masking was initially thought to be powerful, but later on proved to be breakable [15].

2.4.1 Adversarial Training:

This aims to make the neural network more robust by training it with adversarial samples.

$$\min_{\theta} \max_{D(X,\hat{X})<\eta} J(\theta, \hat{X}, y) \tag{2.1}$$

where X is the original image, $\hat{X}$ is the attacked image, θ is the network parameter and J is the loss function of the network. η puts the constraint on the adversarial image beyond which it cannot be modified. The inner maximisation problem tries to make adversarial images (using some attack algorithm like FGSM, PGD, etc) which will fool the network, and the outer minimisation problem tries to update the network parameters to prevent that and minimize the loss. A few examples are:

- **FGSM Adversarial Training:** Here, the FGSM method is used to generate the

attacked images.

$$\tilde{J}(\theta, X, y) = cJ(\theta, X, y) + (1 - c)(\theta, X + \epsilon.sign(\nabla_X J_\theta(X, y)), y)$$

where $X + \epsilon.sign(\nabla_X J_\theta(X, y))$ is the generated adversarial image, and the constant c balances the weights of the two loss functions. The iterative method is also be used to make robust classifiers as they tend to produce stronger adversarial images. This method is weaker than PGD adversarial training.

- **PGD Adversarial Training:** Using PGD attacked images for training makes the model robust against different first-order L_∞ attacks [12]. This also gave the best result in one of the recent Competition on Adversarial Attacks and Defenses (CAADs) for ImageNet. However, computing these images is extremely expensive, i.e., time-consuming.

- **Adversarial Logit Pairing:** This is quite different training approach from the other two, but still uses adversarial images for training. It tries to minimize the cross-entropy between X and $\hat{X}$ (produced by PGD) by including their cross-entropy term in the loss function along with the original loss function. Here c has to be tuned.

$$\tilde{J}(\theta, X, \hat{X}, y) = J(\theta, X, y) + cJ(\theta, X, \hat{X})$$

- **Generative Adversarial Training:** Generative Adversarial Networks can also be used for training robust classifiers. [23] proposes a GAN where the generator takes the gradients of the classifier with respect to the original image and gives adversarial perturbations (similar to FGSM perturbations), which makes the model even more robust. Rob-GAN [24] is an Auxiliary Classifier GAN or AC-GAN architecture which learns the PGD image distribution and after training, the generator outputs fake images which mimic PGD images. These can then be used to train the classifier to make it robust against PGD-like powerful attacks.

2.4.2 Steganalysis-like filters

Steganography is the technique of embedding hidden messages in images, audio files, etc and steganalysis is the detection of such messages from the manipulated files. So, both adversarial detection and steganalysis involves detecting minor changes in the media, and therefore methods of steganalysis can be used in this case of defense against adversarial

images. Using that idea, a linear prediction filter has been used here to try and detect the adversarial examples generated by Projected Gradient Descent method.

A basic idea of steganalysis is that pixels that were modified while the embedding was done will behave differently than pixels that were not changed. So, in a cover image, the surrounding pixels can be used to predict the original pixel values. If pixel i was changed during modification, its original value x_i will deviate from the estimated value $\hat{x}_i$, which can be estimated using the following simple filter [25]:

$$\hat{x}_i = x_i * \begin{pmatrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{pmatrix}$$

The sum of the weighted differences of the original and estimated pixels can be found ($\hat{p}$). Depending on a previously decided threshold, if this sum is greater than the threshold, the image is labelled as adversarial.

$$\hat{p} = \frac{1}{n} \sum_{i=0}^{n-1} w_i (x_i - \hat{x}_i)$$

The weights w_i depends on the predictability of a local region in an image. If the variance is too high, then it will be difficult to predict the pixel correctly from its neighbouring pixels and vice-versa. It has been found experimentally that $w_i^-1 \propto 5 + \sigma_i^2$ [25], where σ_i^2 is the local variance in a patch neighbourhood of the pixel i.

This filter can be used on the image before giving it to the classifier. So, if some image is not benign (unmodified or not adversarial), it will be fed to the CNN (maybe normal or adversarially trained).

2.4.3 Other popular defenses:

- **Data randomization** Deep Neural Networks are quite robust to random disturbances. Data randomization tries to convert the adversarial effects into random effects which the network can handle better. One of them is random input transformation [6]. The images are resized to random sizes and zeros are padded to the image in a random manner before giving as input to the model. Some other randomization includes adding a random layer of noise, randomly masking parts of the output features of a layer from the convolutional filters [5].

- **Ensemble methods** 2C classifers usually generalize well to inputs that it did not

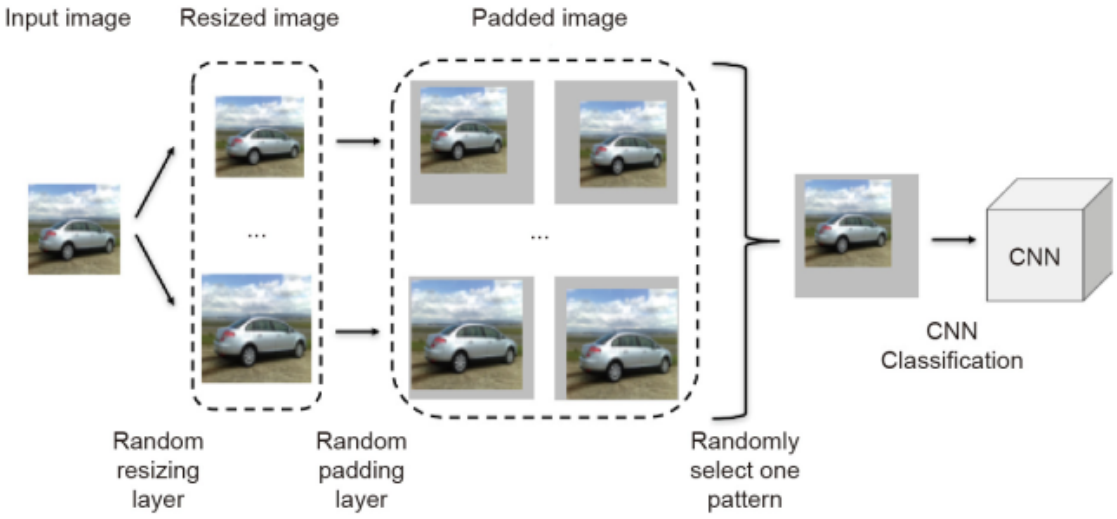
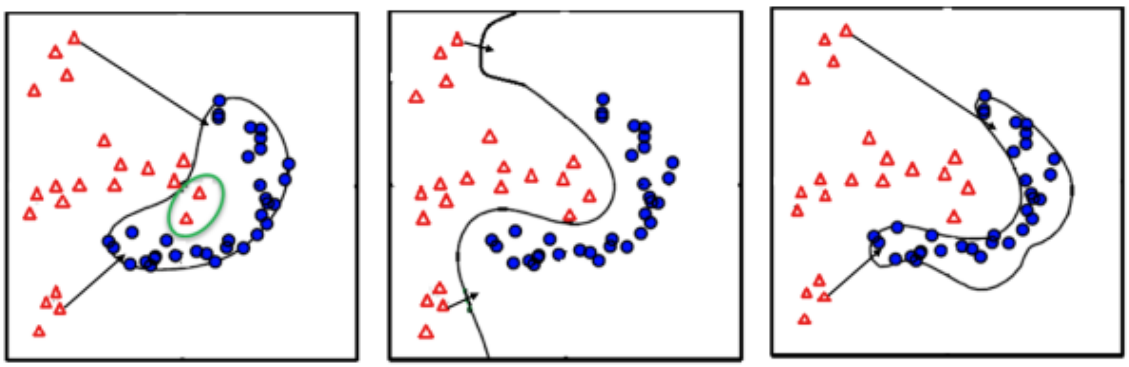


Figure 2.10: Randomization-based defense mechanism [6]: Random resizing and then randomly padding is done on the input.

see during training compared to 1C classifier. However, an adversary may use this to generate inputs which fall in those unseen regions of the feature space (Figure 2.11).



(a) (b) (c)

Figure 2.11: (a) Scheme of 1C by defining a closed region (b) Scheme of 2C when the attacker transferred 'red triangles' to unseen 'blue dots' region (c) Scheme of 1.5C by defining a well-shaped decision region

- **GAN based Defense:** GAN is used to train a generative model learn some given distributions. So, for adversarial defense, GANs are often used to generate a clean version of the attacked image, which can then be reliably fed to the model. Defense-GAN is a very popular algorithm for this [7]. The generator first learns the distribution of the clean images with the help of the discriminator, which is previously trained to classify whether an image comes from the clean image distribution or not. During

test time, generator then projects any input (including adversarial inputs) onto to its range. Thus it basically gives the closest clean image corresponding to the input, which is then given to the classifier. However, Jalal et al. [26] has shown Defense-GAN to be only 3% accurate in presence of their proposed attack method (for PGD attack, it was 55% [15]). They have also made a defense using a generator (like in GANs or as decoders in auto-encoders) to generate the attack and train the network with it.

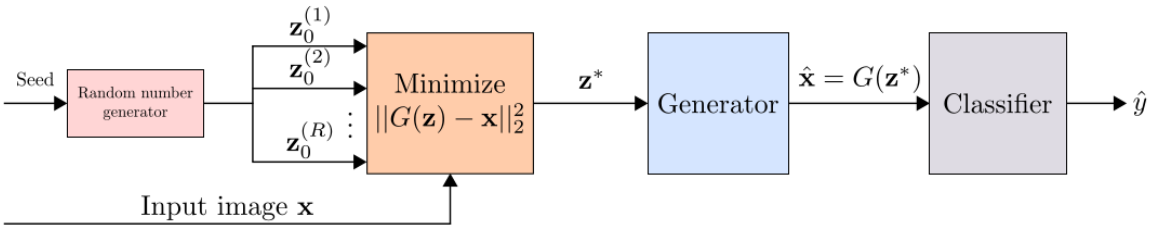


Figure 2.12: Overview of the Defense-GAN algorithm [7]

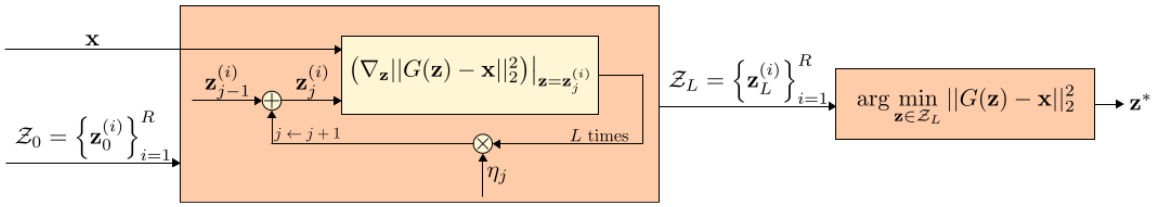


Figure 2.13: L steps of Gradient Descent are used to estimate the projection of the image onto the range of the generator [7]

Among all these different types of defenses, adversarial training has been shown to be the strongest of all and most versatile [15]. And many of the other defenses based on hiding the gradients have been shown to be very much vulnerable to attacks and Defense-GAN, etc have also been proved to be weak [15], [26]. So, in this project, we have tried to make this adversarial training faster for camera-model identification and have also used a linear filter to predict the adversaries before giving as input to the classifier.

3. Proposed Methodology

This project has the following contributions regarding camera-model classification:

- We are proposing the use of the "fast" and "free" adversarial training methods.
- Also, a cyclic learning rate is being used. It prevents the optimizer going in wrong gradient directions initially, and also helps the model weights to converge near the end of the training. The details are given in Experimentation (Section 4.3).
- We are also using steganalysis-inspired linear filter which will be used before the image is fed to the classifier. The filter will reject the adversarial images and only pass the natural images to the adversarially trained classifier.

3.1 Robust Adversarial Training

3.1.1 Adversarial training for free:

Although adversarial training is one of the most effective defense, the main hurdle to its widespread use is its need of high computation power. It becomes almost impractical to use it for large datasets. Fortunately, [19] has proposed a fast implementation of PGD adversarial training and found almost comparable results on CIFAR-10 and CIFAR-100 datasets. This is approximately 7 to 30 times faster than similar strong training procedures and takes slightly more time as natural training. In PGD, the gradients have to be computed to update both the network and the image and this computation causes it to be particularly slow. In the proposed method, the model weights and the image are updated simultaneously in one backward pass, instead of updating them using using gradient computations in separate passes. For a K-iteration PGD, the adversarial image generation loop (inner loop in the equation of adversarial training) runs K-times for every single outer loop, which updates the network parameters.

In this "free method", the gradient computation of the loss with respect to the network and the image are both done simultaneously during the backward pass, and the corresponding values are updated. To make multiple change to a single image, every mini-batch is sent through the forward pass m times consecutively. This is done to produce stronger iterative adversarial images. Quite robust models can be built with m less than 10. . For five classes from the VISION dataset, quite robust models can be built with in less than 10 hours. The best model, obtained at the 77^{th} epoch, had 43.8% robust accuracy and 91.6% normal accuracy which is almost as good as the accuracy reported for object-detection tasks.

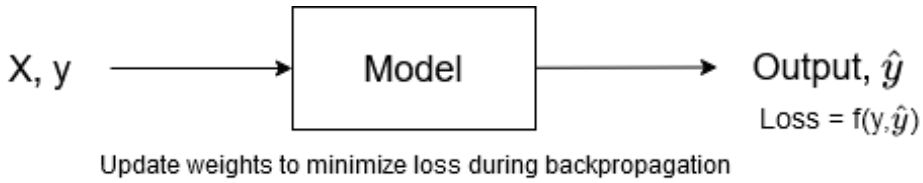


Figure 3.1: Normal training

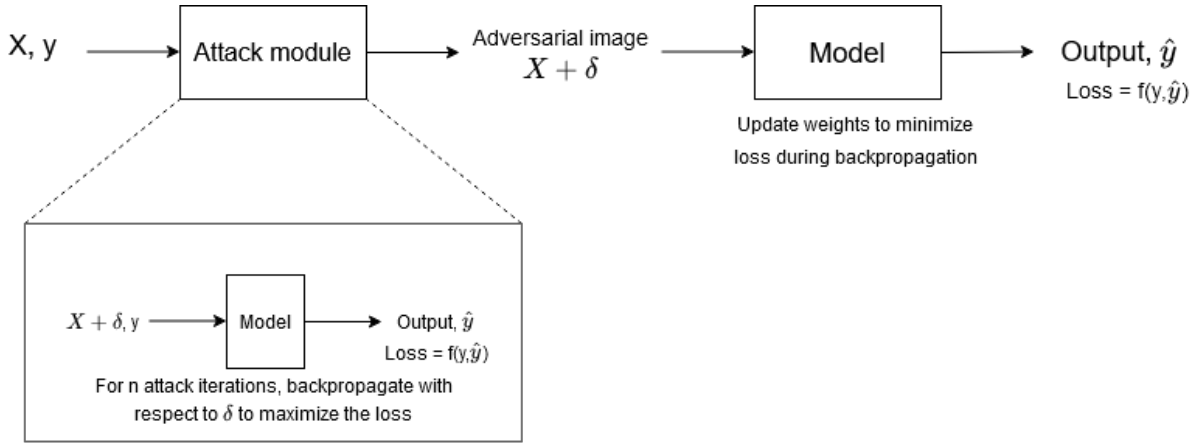


Figure 3.2: PGD Adversarial training

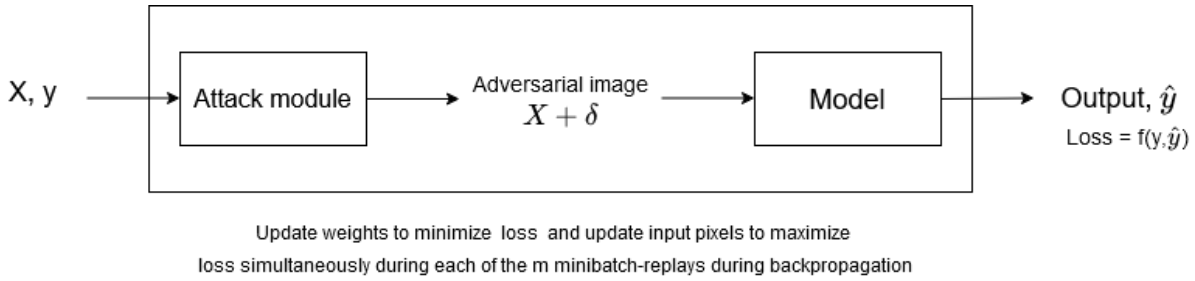


Figure 3.3: Free Adversarial training

3.1.2 Fast Adversarial training:

Although PGD is the most preferred attack, FGSM is a much faster attack. Wong et al. [19] found a way to make FGSM adversarial training almost as robust as PGD training by using random initialization for the perturbation from a uniform distribution. The paper claims to make robust classifier for CIFAR-10 in 6 minutes which [12] reported as 80 hours and the "free" adversarial training paper [18] reported as 10 hours.

For VISION dataset, it took nearly 3 hours to train, but the accuracy was significantly less than that of the "free" method.

3.2 Linear filter as a detector

Often, the adversarial images are crafted with the constraint on distortion, because a highly manipulated image looks different from the original image. Also, if you don't have the original image and only the manipulated image is given by the attacker, and if a dog image looks already like a cat, a human can detect it.

But not always is a human actually going to see it and so a large perturbation can push a data point very deep into a different class's region and still fool the classifier. Also, in camera-model identification (unlike image classification), it is difficult to classify just by looking at it when the source is unknown. Hence, we may be too dependent on the classifier itself. So, a detector which can filter the inputs based on their perturbation magnitude may work as the human in image classification.

An adversary detector can be attached before the classifier. We have used a linear prediction filter inspired from steganalysis. The images detected as adversarial by the filter will not be passed onto the classifier.

A basic idea of steganalysis is that pixels that were modified while the embedding was done will have some different nature than pixels that were not changed [8]. So, in an original image, the surrounding pixels can be used to predict the original pixel values. But in a stego image, the difference between the original pixel value and the estimated value will be quite high. If pixel i was changed during modification, its original value x_i will deviate from the estimated value $\hat{x}_i$, which can be estimated using the following simple filter [25]:

$$\hat{x}_i = x_i * \begin{pmatrix} -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ -\frac{1}{4} & \frac{1}{2} & -\frac{1}{4} \end{pmatrix}$$

We take mean (as $\hat{p}$) of the weighted differences of the original and estimated pixel value for every pixel in the image. A threshold is decided equal to the maximum value of $\hat{p}$ seen during training. Then during testing, any $\hat{p}$ value greater than that threshold will indicate the presence of adversarial perturbations. A small $\hat{p}$ indicates a normal image.

$$\hat{p} = \frac{1}{n} \sum_{i=0}^{n-1} w_i (x_i - \hat{x}_i)$$

The prediction $\hat{p}$ is generated from the surrounding pixels. So, if in some region, that predictability is low even in normal images, then the contribution in the sum for that region should be low, i.e., the w_i for the pixels i in that region should be small.

It has been found experimentally that $w_i^{-1} \propto 5 + \sigma_i^2$ [25], where σ_i^2 is the local variance in a patch neighbourhood of the pixel i.‘‘‘‘‘‘‘‘‘

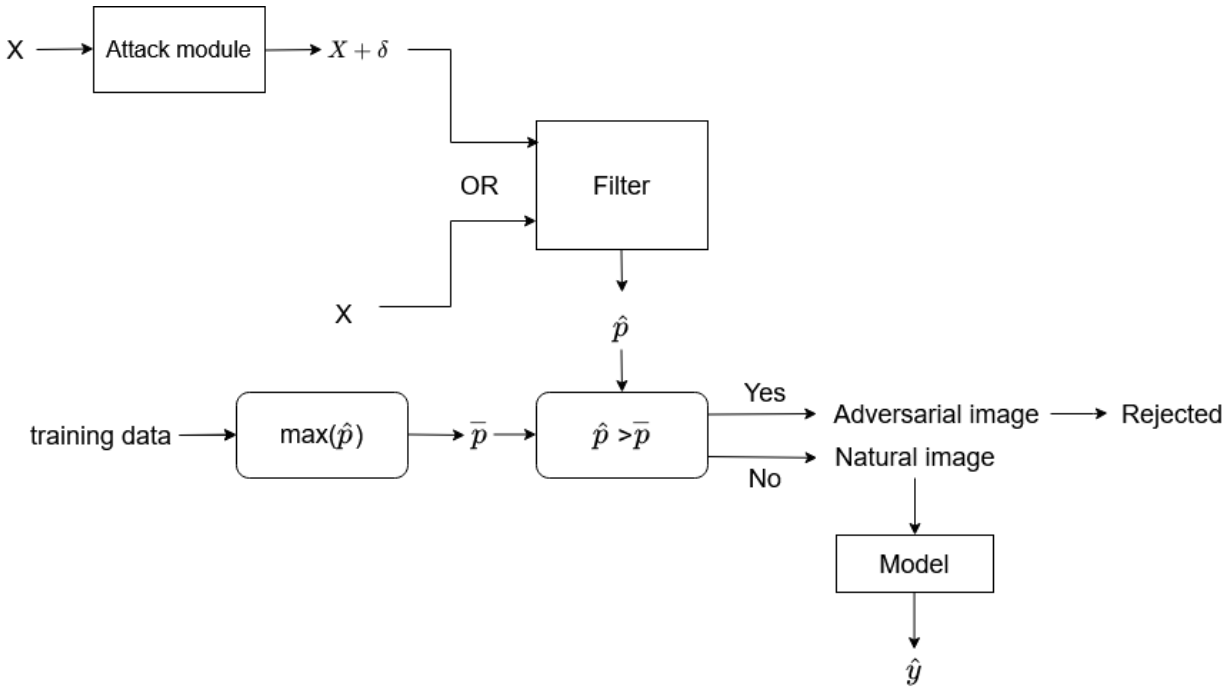


Figure 3.4: Block diagram of the process [8]

This filter correctly identifies almost all the adversarial images. So, the true positive rate is very high (when adversarial images are considered positive). With increasing α in PGD attack, the adversarial images are more accurately detected.

4. Experimental Results

4.1 Dataset

The VISION dataset has 34,427 images and 1914 videos, all taken by portable devices like smartphones and tabs. It has photos taken by 35 smartphone models from 11 brands. The photos are in both native format and compressed format (as done in social-media platforms). So, we have taken from both types of photos. There are natural images (images of common objects, scenes, etc) and flat images (images of sky or sea having very less variation in content). For our project, we have used only the natural images. Portable devices are the most common devices used for taking photos and videos. Hundreds of thousands of these photos are also daily uploaded in the social-media platforms like Facebook, Instagram, etc. These smartphones have digital stabilization while taking photos and videos. Unfortunately, this causes problem in video source identification using the sensor information. The social media platforms have strong compression ratios which is also a hurdle in camera-model identification. The same sensor captures photos and videos in these devices.

In this project we have used natural images of five camera-models, namely

- Samsung GalaxyS3Mini
- Apple iPhone4s
- Huawei P9
- LG D290
- Apple iPhone5c

Two camera models of Apple were chosen because there was a total of 13 models from Apple among the 35 models in the dataset, which was by far the highest number. The second-highest was Samsung having eight models.

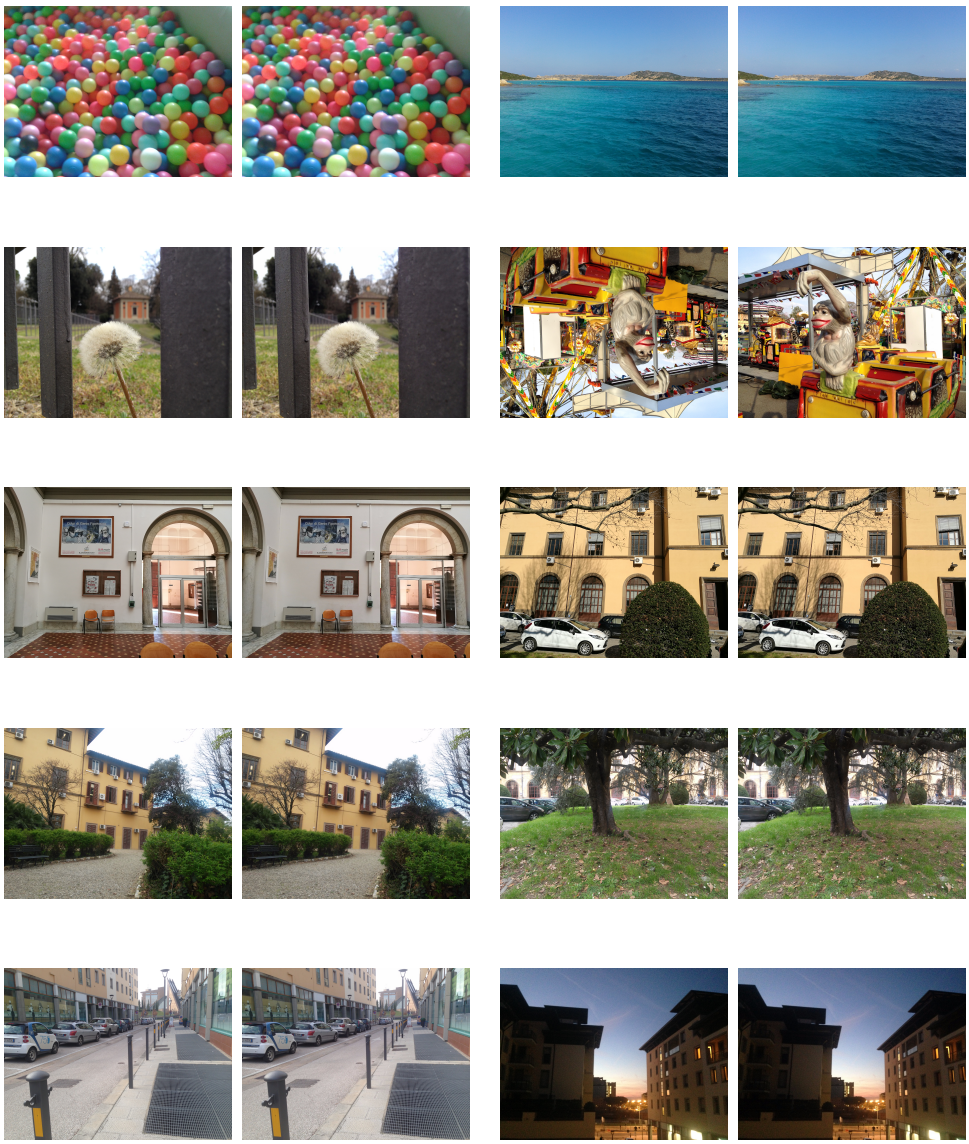


Figure 4.1: Original (left) and WhatsApp-uploaded (right) photos taken by the five camera-models (Samsung GalaxyS3Mini, Apple iPhone4s, Huawei P9, LG D290 and Apple iPhone5c) in each of the above five rows

Brand	Model	ID	Resolution	Number of images	Flat	Nat
Samsung	Galaxy S3 Mini	D01	2560 × 1920	283	78	205
Apple	iPhone4s	D02	3264 × 2448	307	103	204
Huawei	P9	D03	3968 × 2976	355	118	237
LG	D290	D04	3264 × 2448	368	141	227
Apple	iPhone5c	D05	3264 × 2448	463	113	350

Table 4.1: Details of the images taken by each of the five camera-models

From each camera-model, 150 JPEG colour images are chosen and the 100 patches are extracted randomly from the image. So, for each model, we have 15,000 patches and thus a total of 75,000 patches for five camera-models. This dataset is then divided in the ratio of 80:20 for the train set and the test set. Thus, the train set and test set have 50,000 and 15,000 patches respectively.

4.2 Camera-model classification with DenseNet

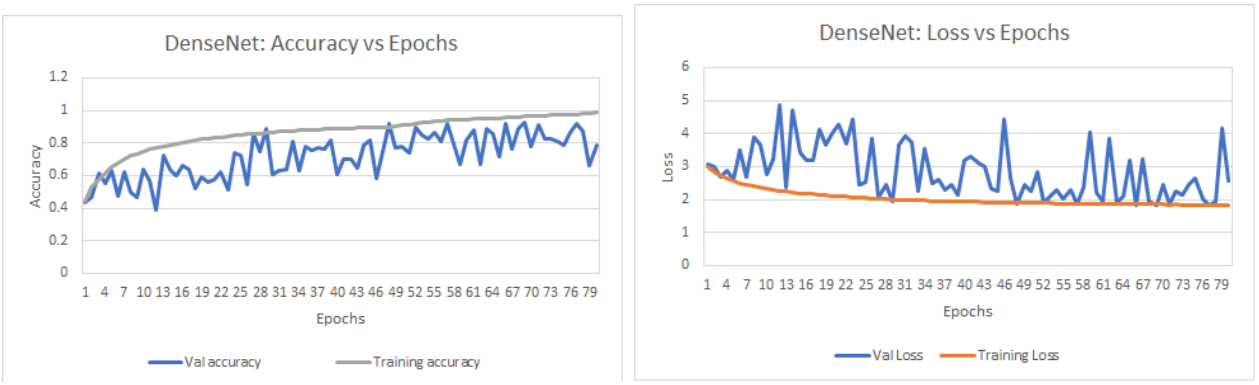
As discussed in Chapter 2, DenseNet is quite successful at our task of identifying the camera-model. It takes input of shape 224×224×3. So, we used it train for 80 epochs with a batch-size of 32 and a momentum of 0.9. We started with a learning rate of 0.001, which got halved every five epochs. The pixel values were normalized between 0 and 1 by dividing with 255. The loss function used was sparse categorical cross-entropy. The best accuracy obtained was 98.5%.

The confusion matrix for the classification with natural images using DenseNet is given below with the following camera-models in order: Samsung GalaxyS3Mini, Apple iPhone4s, Huawei P9, LG D290, Apple iPhone5c.

$$\begin{bmatrix} 2971 & 17 & 12 & 4 & 6 \\ 14 & 2941 & 11 & 12 & 32 \\ 5 & 17 & 2913 & 34 & 22 \\ 7 & 30 & 37 & 2938 & 8 \\ 12 & 18 & 24 & 13 & 2982 \end{bmatrix}$$

The accuracy and loss curves of the DenseNet training on the five classes of VISION is given below:

The accuracy reaches a maximum value of 98.54% on patches and 99.6% on images



(a) (b)

Figure 4.2: (a) Accuracy vs Epochs and (b) Loss vs Epochs of the DenseNet-121 model trained on VISION for camera-model identification

overall.

4.3 Formulating attacks:

After the DenseNet has been successfully trained to classify the camera-models, we have to see the adversarial vulnerability of the model. For that, we need to generate the adversarial images which is done using the FGSM and PGD algorithms. We use Data-Loader to load the patches in batches and then use the above algorithms to generate adversarial images to train the DenseNet. For free adversarial training, we used a batch-size of 128. We have used a cyclic learning rate where it started from 0, went up to 0.004 in steps of 0.001 and then again went back to 0 in steps of 0.001. The training was done for 80 epochs, with a momentum of 0.9. The maximum change in the pixel value was kept as 8. We used 8 minibatch-replays, which are the consecutive gradient updates on a single batch of patches, while the attack was being improved at each replay. Mixed precision training was done using the APEX library of PyTorch. For fast adversarial training, the values of the hyperparameters were kept same. The delta initialization was done randomly from a uniform distribution. The step-size for pixel value update was kept as 10.

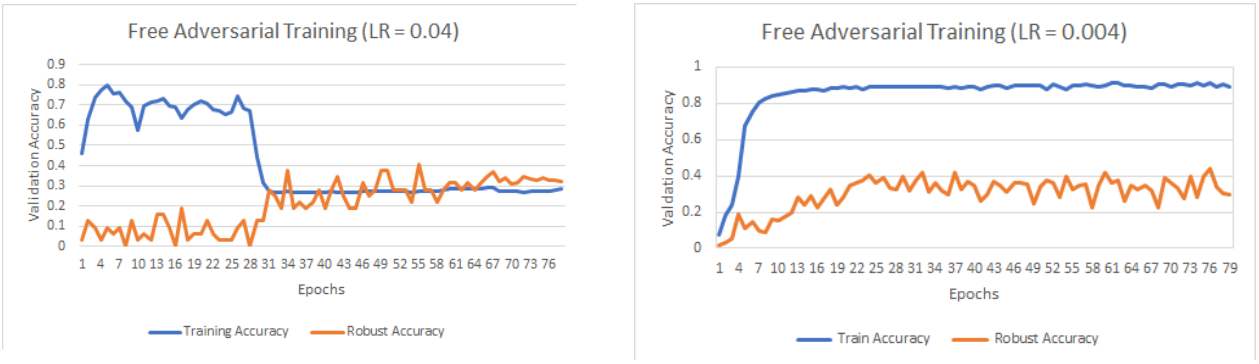
4.4 Defense with adversarial training

With free adversarial training, the best model, obtained at the 77th epoch, had 43.8% robust accuracy and 91.6% normal accuracy which is almost as good as the accuracies reported for object-detection tasks. This was only after the max learning rate was reduced to 0.004 from the initial 0.04. With the latter value, the accuracy would suddenly drop to around 30% and never go back up. This happened with each of the other adversarial

training methods. The adversarial images produced in every batch are used to train the model, which is also evaluated on normal images and PGD attacked images at the end of each epoch.

For the fast adversarial training, the normal accuracy never improved over 85% and the best robust accuracy was 43.18%. The sudden drop in accuracy with a bigger learning rate happened in this case also and even in PGD adversarial training. The values are provided in Table 4.1.

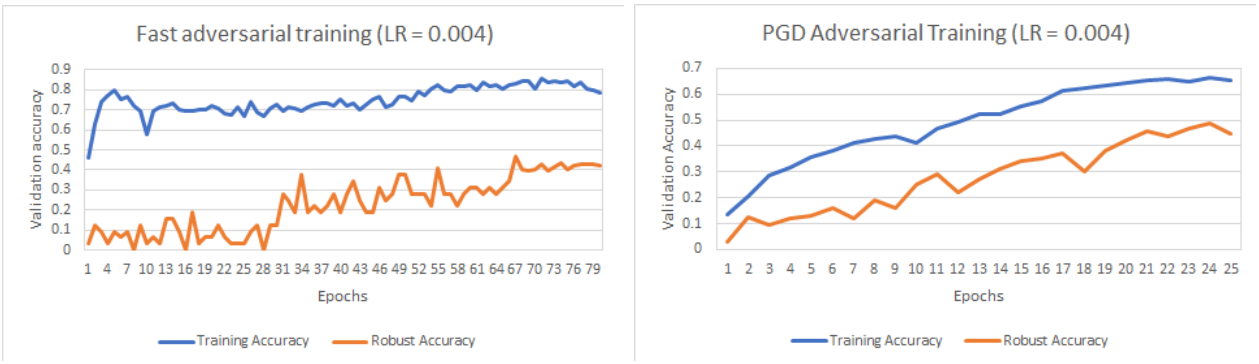
The training and validation accuracy curves for free adversarial training with learning rates 0.04 and 0.004 are given below: With the 0.04 learning rate, the model diverges and



(a) (b)

Figure 4.3: Accuracy vs Epochs for DenseNet-121 during free adversarial training (with cyclic learning rate) with (a) max learning rate = 0.04, (b) max learning rate = 0.004

the loss increases dramatically and it never goes back. So, a smaller learning rate was tried which finally worked. This problem was there in fast and PGD training also, so only the accuracy curves of the 0.004 learning rate for those methods are given below:



(a) (b)

Figure 4.4: Accuracy vs Epochs for DenseNet-121 during (with cyclic learning rate) and learning rate = 0.004 (a) fast adversarial training and (a) PGD adversarial training

4.5 Applying the detector with trained classifier

Training time for the filter is negligible. We just have to find the maximum $\hat{p}$, so that the filter can take a decision based on the the $\hat{p}$ value of the test image. Initially, the estimates of $\hat{x}_i$ are made by convolving with the kernel. Then the weight-matrix is found by estimating the local variance at each pixel. We calculate the mean of a 3×3 neighbourhood and then sum the squared differences with the mean (except for the central pixel). From that, the w_i is estimated as $w_i^{-1} \propto 5 + \sigma_i^2$ where σ_i^2 is the local variance. Then the $\hat{p}$ can be calculated for individual images, and we take the maximum of the training set as the threshold.

During test time, any image with $\hat{p}$ greater than that threshold is rejected as an adversarial image.

The PGD training takes way too much time, so we could only run it for 25 epochs. And the fast adversarial training does not perform good enough. The steganalysis filter was used along with the DenseNet trained with free adversarial training with a learning rate of 0.004. All the results are tabulated in Table 4.2. The confusion matrix for the filter detecting natural and adversarial images is given below:

$$\begin{bmatrix} 6598 & 902 \\ 577 & 6923 \end{bmatrix}$$

4.6 Discussions

The DenseNet performed really well as camera-model classifier on the VISION dataset. The adversarial vulnerability was shown when it was given PGD attacked images and the accuracy dropped to 1-2%. As defenses, of all the adversarial training methods, the free adversarial training method seemed most optimal (with respect to accuracy values and training time) for real-life usage when we may have to classify for hundreds of camera-models. The best results were obtained by combining it with the filter.

Model	Normal accuracy (%)		Robust accuracy (%)		Training time for 80 epochs (min)
	Patch	Full	Patch	Full	
DenseNet	98.54	99.6	1.2	2.1	85
DenseNet + Free (0.04)	27.18	31.3	28.12	30.82	598
DenseNet + Free (0.004)	91.6	94.3	43.8	49.8	622
DenseNet + Fast (0.004)	80.46	82.28	43.18	45.2	212
DenseNet + PGD (0.004)	66.46	70	49.2	52.7	3420
DenseNet + Free (0.004) + Stega-filter	89.6	92.1	**58.1**	**63.2**	622

Table 4.2: Best Accuracy values of the different defense methodologies. Both patch-level and image-level accuracies are provided against both normal images and PGD-adversarial images. Except for PGD training, which was done for 25 epochs, all other training were done for 80 epochs

5. Conclusion and Future Work

5.1 Conclusion

- From the above experimentation, it is observed that the constrained-CNN did not work with VISION the way it did with Dresden. It faced some difficulty in collecting the features from the images taken by portable devices like smartphones. So, DenseNet, which has been used in other camera-model identification tasks, was chosen and it indeed gave good results.

- The accuracy of the DenseNet against adversarial images was around 1-2%. As a defense, adversarial training is the most common method. PGD training was taking too long, so we tried the fast and free adversarial training methods. Of them, fast training (which is based on FGSM) did not give good enough results. Free training (based on PGD) gave better results, but also took almost three times more duration to train. So, PGD-related training seems more reliable for camera-model identification (and maybe other forensic applications), and if we have time, we should go for free training instead of fast or FGSM based training.

- Since this is a forensic application, it is of utmost importance that the adversarial images get correctly classified or at least they get flagged as having been modified. So, we try a linear filter estimates the pixel values from the surrounding pixels and calculates the weighted mean of the difference between the original and estimated pixel values. The images identified as being adversarial will not be passed on to the classifier, otherwise, the adversarially-trained classifier will predict its label (camera-model here). The filter correctly detects almost all the adversarial images, but also incorrectly flags some normal images as adversarial. The use of this filter definitely makes the overall classification process more robust.

5.2 Future Work

- For the images flagged as adversarial, we may not completely reject them, as they may actually be natural. If individual images are given high importance, a human can manually look at it and decide. And if many images have to be classified we can assign weights (inversely proportional to the difference $(\hat{p} - \overline{p})$) to the image outputs as to how they contribute the overall accuracy. Or there can be two thresholds based on the $\hat{p}$ value. Any image above the upper threshold is rejected, any image below the lower one will be accepted. And for the images in between the two thresholds, we can use the above weight-assignment strategy.

- We can look at some other defenses used in image classification. TRADES[27] and Robust Manifold Defense [26] have found success recently. It was tried, but the results were not good enough. It may need some modification for camera-model identification or forensic tasks in general. We may try to improve it by trying out changes in the algorithms.

- FGSM and PGD are the most popular attacks. But another that is used is Auto-attack and it is non-parameterized. This can be used to generate adversarial images during the training of the model itself, just as FGSM or PGD is used.

- With enough time, this experimentation can be run for more number of models of the VISION dataset or maybe some other camera-model datasets.

- This same identification can be extended to videos, but there the perturbations will be added framewise, and so some extra method may have to be used that also takes care of the time factor.

- Due to adversarial training, the normal accuracy drops quite a lot. So, a trade-off may be studied to find the best combination of normal and robust accuracy for camera-model identification as done in [27].

- We may try to reduce the False Positive and False Negative for the filter (here positive is natural image and negative is the adversarial image). In this case, False Negative is more crucial as we do not want a misclassification by the model.

- The weights in the filter kernel can be made to be learnable instead of just predicting it from the variance of the surrounding pixels.

Bibliography

[1] J. T. A. Andrews, Y. Zhang, and L. D. Grin, "Conditional adversarial camera model anonymization," *ECCV*, 2020.

[2] B. Bayar and M. C. Stamm, "Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection," *IEEE Transactions on Information Forensics and Security*, 2018.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *ICLR*, 2015.

[5] E. Nowroozi, A. Dehghantanha, R. M. Parizi, and K.-K. R. Choo, "A survey of machine learning techniques in adversarial image forensics," *Computers & Security*, 2020.

[6] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," *arXiv: 1711.01991*, 2017.

[7] R. C. P Samangouei, M Kabkab, "Defense-gan: Protecting classifiers against adversarial attacks using generative models," *ICLR*, 2018.

[8] P. Schottle, A. Schlogl, C. Pasquini, and R. Bohme, "Detecting adversarial examples - a lesson from multimedia security," *EUSIPCO*, 2018.

[9] A. M. Raf, U. Kamal, R. Hoque, and A. Abrar, "Application of densenet in camera model identification and post-processing detection," *CVPR*, 2018.

[10] D. Shullani, M. Fontani, M. Iuliani, O. A. Shaya, and A. Piva, "Vision: a video and image dataset for source identification," *EURASIP Journal on Information Security*, 2018.

[11] C. Szegedy, I. Sutskever, I. Goodfellow, W. Zaremba, R. Fergus, and D. Erhan, "Intriguing properties of neural networks," *ICLR*, 2014.

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *ICLR*, 2018.

[13] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," *Asia-CCS*, 2017.

[14] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv:1605.07277*, 2016.

[15] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," *PMLR*, 2018.

[16] B. Bayar and M. C. Stamm, "Design principles of convolutional neural networks for multimedia forensics," *Media Watermarking, Security, and Forensics*, 2017.

[17] F. Marra, D. Gragnaniello, and L. Verdoliva, "On the vulnerability of deep learning to adversarial attacks for camera model identification," *Signal Processing: Image Communication*, 2018.

[18] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, and J. Dickerson, "Adversarial training for free!" *NeurIPS*, 2019.

[19] E. Wong, L. Rice, and J. Kolter, "Fast is better than free: Revisiting adversarial training," *ICLR*, 2020.

[20] "IEEE's signal processing society - camera model identification," *https://www.kaggle.com/c/sp-society-camera-model-identification*, 2018.

[21] W. Li, B. Tondi, R. Ni, and M. Barni, "Increased-confidence adversarial examples for improved transferability of counter-forensic attacks," *arXiv:2005.06023*, 2020.

[22] J. Bernacki, "Robustness of digital camera identification with convolutional neural networks," *Multimedia Tools and Applications volume*, 2021.

[23] H. Lee, S. Han, and J. Lee, "Generative adversarial trainer: defense to adversarial perturbations with gan," *arXiv: 1705.03387*, 2017.

[24] X. Liu and C. Hsieh, "Rob-gan: generator, discriminator, and adversarial attacker," *arXiv:1807.10454v3*, 2018.

[25] A. D. Ker and R. Bohme, "Revisiting weighted stego-image steganalysis," *Security, Forensics, Steganography, and Watermarking of Multimedia Contents*, 2008.

[26] A. Jalal, A. Ilyas, C. Daskalakis, and A. Dimakis, "The robust manifold defense: Adversarial training using generative models," *arXiv:1712.09196*, 2019.

[27] H. Zhang, H. Zhang, and J. Jiao, "Theoretically principled trade-off between robustness and accuracy," *ICML*, 2019.