Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Comparative Evaluation of Adversarial Attacks and its Defense Mechanisms on Deep Neural Network

Prabhudev Bengaluru Kumar

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Nico Hochgeschwender
M.Sc. Deebul Nair

August 2021

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Prabhudev Bengaluru Kumar

Abstract

Deep Learning (DL) provides advanced methods for safety-critical applications. DL is the engine behind many real-time applications, including autonomous systems, visual recognition, computer vision. However, the introduction of adversarial attacks makes it risky to employ DL methods. A process of injecting adversarial noise into input data to confuse a model is an adversarial attack. This research work aims to examine adversarial attacks and evaluation of defense mechanisms against them.

Based on extensive research and keeping realistic scenarios in mind, this work chooses an adversarial patch attack as the attack method. An adversarial patch attack involves attaching a constructed adversarial patch on input data to mislead a model. A thorough study is conducted to identify defense mechanisms and evaluate them to combat adversarial patch attacks. Adversarial training, abstention class and evidential uncertainty estimation are the defense mechanisms performed in this research using Deep Neural Networks (DNNs), namely MobileNetV2, ResNet50, and VGG16 on Beans, Imagenette, and RoboCup@Work datasets.

In adversarial training defense, an adversarial dataset is constructed to train the DNNs. Experimental investigation reveals that the adversarial training defense fails to provide protection against the adversarial patch attack. On the other hand, a new class (hot class/abstention class) is created in abstention class defense and is included in the training dataset. Experiments show that this defense can protect against the adversarial patch attack if the right elements are chosen and added to the abstention class. The evidential uncertainty estimation defense also provides protection against the adversarial patch attack by demonstrating the uncertainty present in the confidence level of model predictions after the adversarial patch attack.

Acknowledgements

I would first like to express my deepest and sincerest gratitude to my supervisors Prof. Dr. Nico Hochgeschwender and M.Sc. Deebul Nair, for providing me with an excellent opportunity to work on this interesting Research and Development project and for their motivation, patience and guidance through all aspects of this project.

I am grateful to M.Sc. Deebul Nair for his mentorship, constructive feedback, support, and guidance to help me become technically more competent.

I would like to thank my colleagues Adithya, Kaushik, Ganesamanian, Santosh, Aswinkumar, Vishnu, Manoj Kolpe Lingappa, Proneet and Krishna for their constant motivation, constructive feedback and support.

Finally, I thank my family for their unconditional love, support and blessings.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation

Deep Learning (DL) provides cutting edge technologies for wide area of Artificial Intelligence (AI) applications [14] [13] [22] [1] [34]. Due to the ability of handling complex computations as well as the increase in powerful infrastructure and availability of data, DL is efficiently employed to increase the accuracy in real-world applications like bioinformatics [38], autonomous systems [26], cybersecurity [5], biometrics [58]. The introduction of adversarial attack poses a real danger to the performance of DL techniques. The adversary (attacker) generates adversarial noise and adds it to the input to fool DL techniques. For instance, in autonomous vehicles, the trajectory of the vehicle can be changed by introducing adversarial billboards [47]. The adversary prints the adversarial patterns on the billboard, and this billboard is placed beside the road. When an autonomous vehicle captures the image of the billboard, the steering angle of the vehicle is changed, which may lead to an accident. This adversarial approach is depicted in Figure 1.1.

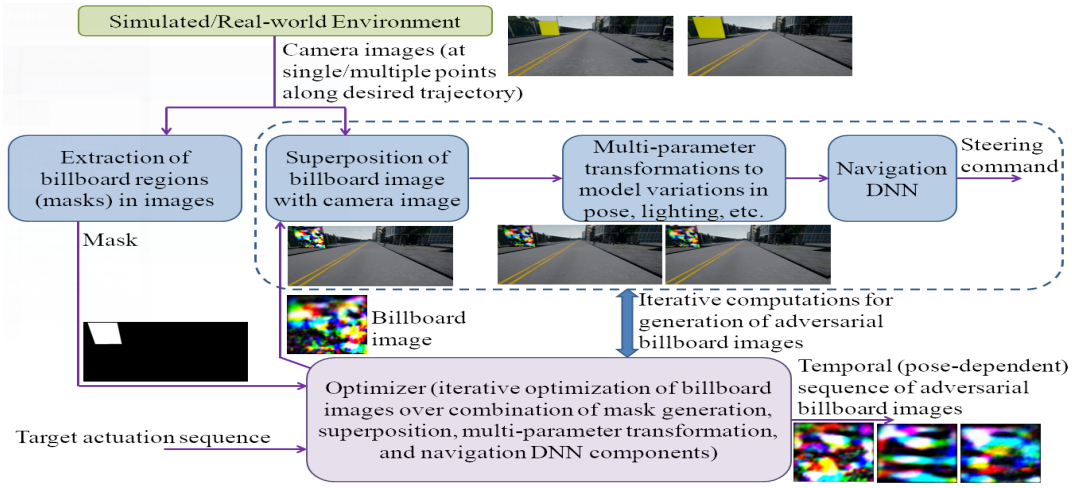


Figure 1.1: Adversarial attack on autonomous system. Image source : [47]

Similarly, in the field of object detection, a vehicle can avoid being detected from the camera by

attaching adversarial patterns [23]. As depicted in Figure 1.2, a screen is attached to the vehicle, and adversarial patterns are dynamically displayed on this screen w.r.t the position of the vehicle and the camera. Here, the vehicle can escape from camera detection easily. When the camera captures the vehicle's image with an adversarial pattern attached, the vehicle will not be detected or misclassified as a different version of car.

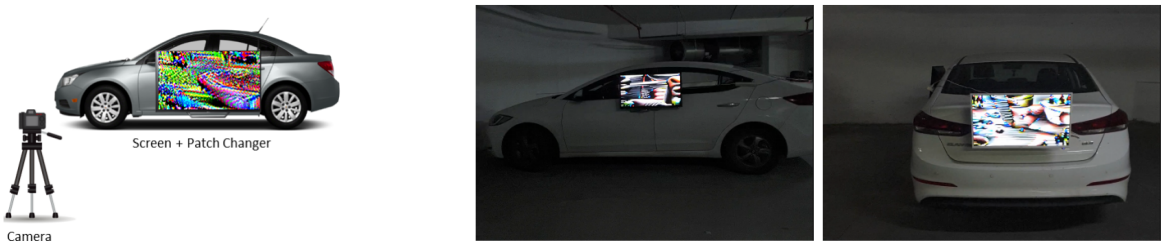


Figure 1.2: Adversarial attack on object detection. Image source : [23]

Similarly, in biometrics [25], the adversary can make adversarial attacks to gain authentication to the system, which raises security issues. The adversarial approach on this application is shown in Figure 1.3

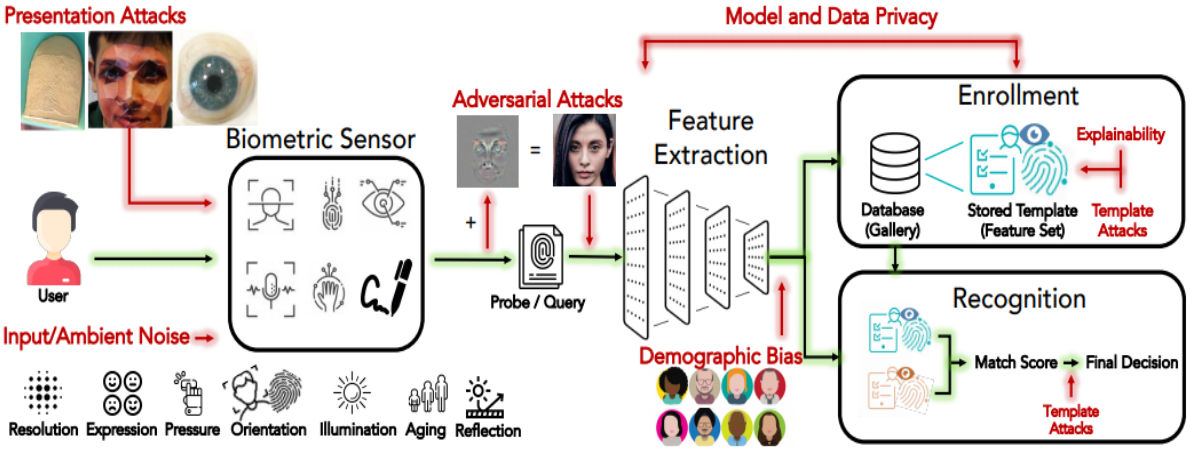


Figure 1.3: Adversarial attack on biometrics. Image source : [25]

An adversarial attack performed in the case of autonomous systems and object detection is a physical real-world scenario kind of attack where the adversary performs the attack before the target object gets captured by the input device, like generating a noise pattern and attaching it to the target object. That is, before the generation of data, an adversarial attack takes place. Another kind of adversarial attack is a digital-world attack where the adversary modifies the data captured from the input device. That is, after the generation of data by the input device, an adversarial attack takes place. There are several adversarial attacks and defense mechanisms introduced by the DL research community [69] [10] [44] [11] [30]. This research work concentrates on physical real-world based attacks and defense mechanisms to protect against

them. By keeping real-world applications in mind, this research work studies physical real-world attacks instead of digital-world attacks. The physical real-world attack chosen for this research is an adversarial patch attack. The adversary creates an adversarial patch and attaches it to the target images to fool the DL techniques. This research aims to helps future researchers to identify convenient defense mechanisms to protect against physical real-world adversarial attacks and perform further research in this field.

1.2 Challenges and Difficulties

The following is the list of challenges and difficulties faced during the progress of this research work:

- Tuning Deep Neural Network (DNN) to get high accuracy. The first challenge was to tune the hyper-parameters of three considered DNNs to get good accuracy w.r.t to each considered dataset and performed experimentation.
- Creating adversarial dataset to perform adversarial training defense mechanism: There is a need for the adversarial dataset to perform adversarial training defence. While creating an adversarial dataset, different patches targeting each respective class has to be generated. Then each generated patch has to be attached to the respective class, which is a difficult and length procedure.
- Identifying the images suitable for abstention class, which is added to the dataset to provide defense mechanism: The abstention class is added to the original dataset to provide defense mechanism. The images belonging to this abstention class must be carefully selected as they control the defence's performance.
- Choosing the number of elements needed to create an abstention class. What must be the size of an abstention class? Size of the abstention class matters as it may affect the training of the model.
- Executing evidential uncertainty estimation defense mechanism on considered DNNs. Careful observation was needed while training a DL model with an evidential uncertainty estimation method as sometimes, loss function output can sometimes be Not a Number (NaN).

1.3 Problem Statement

The purpose of this research work is to identify physical real-world adversarial attacks and perform state-of-the-art defense techniques to protect against the attack. The comparative evaluation of defense mechanisms is performed to identify which defense method works better. An adversarial patch attack is executed on three Deep Neural Networks (DNN) that is MobileNetV2, ResNet50 and VGG16 using Beans dataset, Imagenette dataset and RoboCup@Work dataset. A real-life scenario is taken into consideration with varied patch sizes of 20%, 25%, 35% and 40%. To summarize, this R&D work gives answers to the research question (RQ) mentioned below:

RQ1 What are the existing state-of-the-art defense mechanisms against adversarial patch attacks in deep learning?

RQ2 Can an adversarial patch attack always perform the targeted attack?

RQ3 Is adversarial training defense effective against adversarial patch attacks?

RQ4 Could the defense against adversarial patch attacks be improved by adding an abstention class to a dataset?

RQ5 Can the evidential uncertainty estimation method be used to defend against adversarial patch attacks?

2

Background

This chapter describes the concept of adversarial attack, adversarial attack scenarios, adversarial attack types, and adversarial threat models.

2.1 Adversarial Attack

The adversarial attack is a process of creating adversarial noise and then crafting adversarial data by adding the created noise to the input data and passing it to the Deep Neural Network (DNN) to classify the input images wrongly. Suppose X is the dataset with labels Y and $f(.)$ is a DNN. For each $x \in X$ where $f(x) = y$ st $y \in Y$, noise $\bar{x}$ is created. Then this noise added to the data $x = x + \bar{x}$. When this data is passed to the DNN, it wrongly classifies the x s.t $f(x) \neq y$.

The example of an adversarial attack is depicted in Figure 2.1. As depicted in the figure, if the image of the tiger is passed to the classifier, then gets classified correctly as a tiger, but if this input tiger image is corrupted by adversarial perturbation and is given as input to the classifier, then this leads to an adversarial attack which makes the classifier wrongly classify the tiger image as bustard.

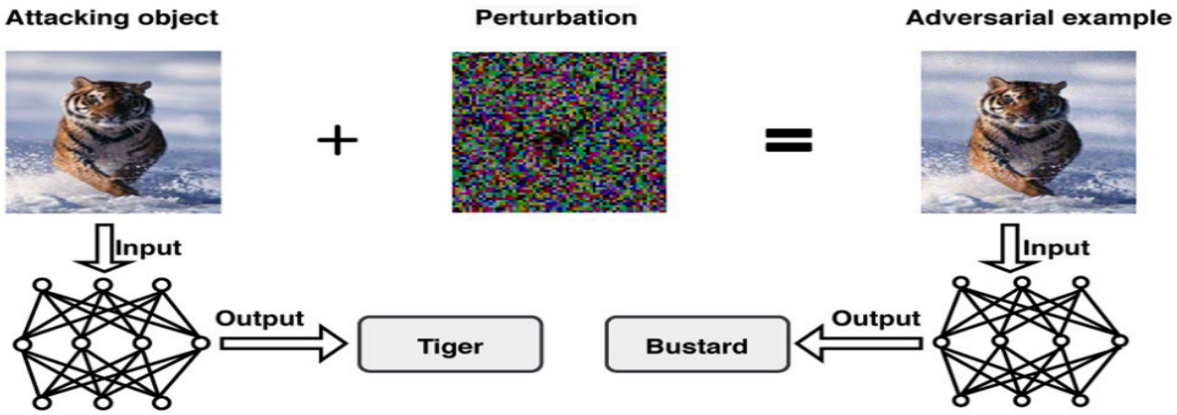


Figure 2.1: Example of Adversarial attack. Image source : [49]

2.2 Adversarial Attack Scenarios

This sections describes different scenarios of adversarial attack

2.2.1 Digital-World Attack

In a digital-world attack scenario, the attacker perturbs the data which is produced by the input device. That is, once the input device captures the target object and it creates the data. The attacker modifies this produced data and then sent it to the DNN to misclassify the target object. The pipeline of digital-world attack is depicted in Figure 2.2.

As depicted in the pipeline diagram, the attacker does modification after input data is generated by the input device. That is, at first, the capturing device captures the image of the target object, and then while this captured image is being fed to the DNN, the attacker intervenes and performs modification to this captured image to produce an adversarial image (modified image). Then this adversarial image is fed to the DNN, which incorrectly classifies the image. As shown in the pipeline diagram, the camera captures the image of the target object, which is building. While this building image is being passed to a DNN, the attacker intervenes and adds adversarial noise to the building image to produce an adversarial image. When this adversarial image is passed to the DNN, misclassification takes place.

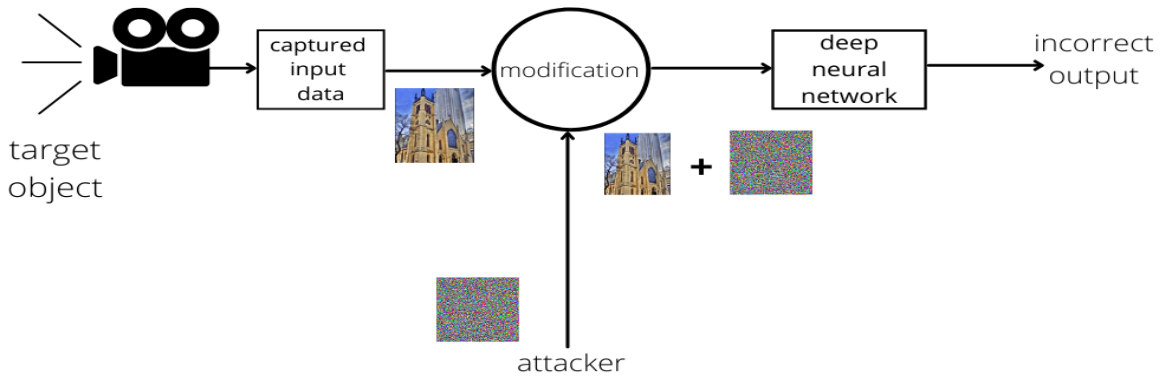


Figure 2.2: Pipeline of digital-world attack. Image source: Imagenette dataset mentioned in Section 5.1.2 is used to create this Figure

2.2.2 Physical Real-World Attack

In physical real-world attack scenario, the attacker performs adversarial attack before the input device captures the target object. That is, the attacker modifies the target object or the input device before the data is collected. Once the input device generates the data, it is sent to the DNN which misclassifies the target object. The pipeline of physical real-world adversarial attack is shown in Figure 2.3.

As depicted in the pipeline diagram, at first, the attacker produces an adversarial patch and attaches the patch to the target object to initiate an adversarial attack. The capturing device captures this modified target object. When this captured image is fed to the DNN, misclassification of the image takes place. As depicted in the pipeline Figure 2.3, a patch is generated by the attacker, and then this patch is placed on the target building object. Then the camera captures building with a patch attached to it. When this captured patched building image is passed to the DNN, the image gets misclassified.

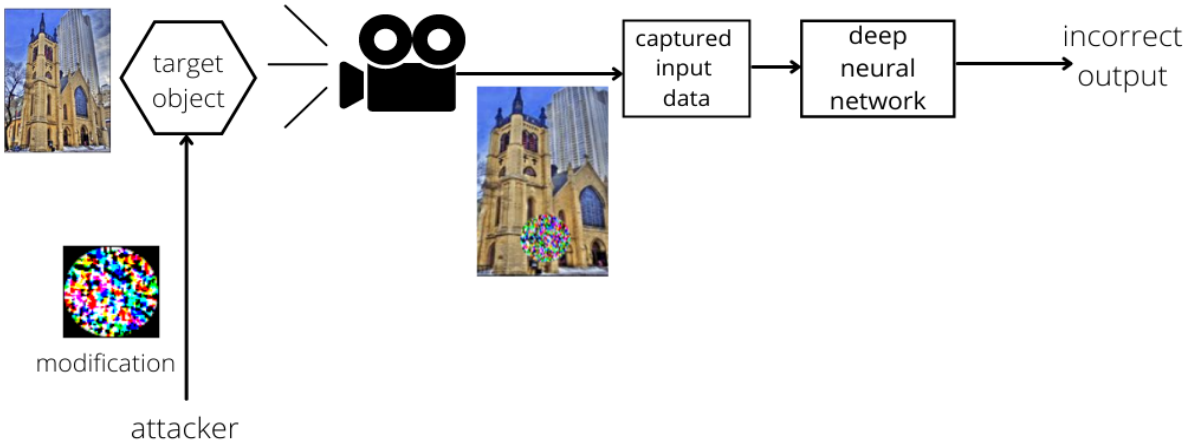


Figure 2.3: Pipeline of physical real-world attack. Image source: Imagenette dataset mentioned in Section 5.1.2 is used to create this Figure

2.3 Type of Adversarial Attacks

This section shows different adversarial attack types

2.3.1 Non-Targeted Adversarial Attack

In a non-targeted adversarial attack, the aim is to classify the input data wrongly [19] [48]. In this scenario, The final output is not desired by the attacker.

2.3.2 Targeted Adversarial Attack

In a targeted adversarial attack, the aim is to misclassify the target object as the attacker's desired label class [28] [9] [60]. The attacker modifies the input data so that it produces selected wrong output rather than random incorrect output when misclassification occurs.

2.4 Threat Model in Adversarial Attack

This section describes about different threat models in adversarial attack

2.4.1 Black-Box Model Attack

In a black-box model attack, the attacker is clueless about the DNN. The attacker has no information about DNN structure and has no idea about its parameters.

2.4.2 Grey-Box Model Attack

In a grey-box model attack, the attacker know about the DNN structure but has no clue about the model's parameters. Since the attacker knows the model's design, the performance of a grey-box attack is better than a black-box attack.

2.4.3 White-Box Model Attack

In a white-box model attack, the attacker has complete information about the DNN structure and its parameter. Hence white-box attack's performance is better than the grey-box attack and the black-box attack. This research work performs a white-box model adversarial attack.

2.5 Adversarial Attack on different field of Applications

This section provides information on adversarial attacks on main application fields

2.5.1 Classification

- The objective of adversarial attacks on classification application is to make a network wrongly classify the given input [16], [59].
- Both digital-world and real-world adversarial attacks can be performed on the classification field.
- An adversary intercepts the input to modify or add perturbation to the input to create an adversarial sample. When this adversarial sample is passed to the classifier, it incorrectly classifies the adversarial sample.
- For example, in the task of radio signal classification, an attacker introduces perturbation to the radio signal input, which inturn causes misclassification [52].
- Fast Gradient Sign Attack (FGSM) [17], Carlini and Wagner attacks (C&W) [8], adversarial patch attack [7] are some of the attacks which can be employed in the field of classification.

2.5.2 Regression

- The intention of adversarial attacks on regression is to produce continuous incorrect output by introducing perturbation into the input of regression model [39][37].
- In stock market prediction, an adversary can manipulate the market data to produce wrong predictions [42].
- FGSM, C&W, Jacobian-based Saliency Map Attack (JSMA) [45] are few adversarial attacks on regression.

2.5.3 Object Detection and Semantic Image Segmentation

- The motivation behind adversarial attack on semantic segmentation is to link pixels of images to wrong class labels [4].
- An adversarial attack on the field of object detection can be made for two purposes
  - misdetection: the purpose is to make incorrect object detection. For example, making a model wrongly detect a cat as a dog in an image.
  - evading detection: the purpose is to avoid detection of an object by the detection model. For example, a speed bike on a highway trying to evade a speed limit detection camera.
- Adversarial attacks like dynamic adversarial patch [23], dpatch attack [31], Dense Adversary Generation (DAG) [68] are some of the attack employed in the field of object detection and semantic image segmentation.

3

Adversarial Patch Attack

This chapter examines adversarial patch attacks, the generation of adversarial patches, and the experiment of adversarial patch attack on Beans, Imagenette and RoboCup@Work datasets.

3.1 Overview

The research work conducted here is based on the paper "*Adversarial Patch by Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, Justin Gilmer*" [7]. This paper introduces a new attack called an adversarial patch attack to misclassify images. In this research work, the adversarial patch attack method proposed by [7] is used to for experiments.

Prior works focused on performing the imperceptible changes to the model inputs to perform an adversarial attack. So, the attacker required access to the input dataset. But adversarial patch paper introduces a white-box attack that focuses on creating an image-independent patch and then place it on the target object to make the classifier misclassify the object. "*The patch created is extremely salient to a neural network*" [7] i.e., the created patch contains extremely prominent features of a neural network. To put it another way, the objective is generation of adversarial patch that incorporates all crucial attributes that a neural network learns. Considering all vital attributes, in turn, helps fool the neural network effectively. In this type of adversarial attack, a patch is generated and attached to input images to produce patched images. When these patched images are fed into the classifier, it misclassifies the images. To perform the adversarial patch attack, a patch is to be created. The patch is generated using a variant of the framework, Expectation Over Transformation (EOT) [3]. The previous works try to produce adversaries by "*maximizing the log-likelihood of the target*" [3] around the input. This adversarial patch paper tries to maximize the expectation of log-likelihood around the distribution of transformed images. The attacker feeds some images into the model, and in each iteration, an objective function shown in Equation 3.1 is optimized to train the patch. That is, in each iteration, the expectation of log-likelihood is calculated and maximized over the distribution of images.

$$\hat{p} = \arg\max_{p} E_{x\sim X, t\sim T, l\sim L}[log\, Pr(\hat{y}|A(p, x, l, t))] \tag{3.1}$$

where

X - training set of images

T - distribution over transformation of patch
L - distribution over location in image

Here the patch is generated by the expectation of log-likelihood over images; hence the produced patch is universal and it functions no matter what the background is. After creating the patch, a part of the input images is replaced by the generated patch to produce the patched images. An operator for patch application, A(p,x,l,t) is defined for this procedure. This operator attaches the patch p on the image $x \in R^{(wxhxc)}$ at the location l by performing transformation t. Then these patched images are provided as input to make the classifier wrongly classify the patched inputs.

3.1.1 Generation of Adversaries

In prior works, for any given inputs X and possible labels Y, adversaries generated targeting a class $\hat{y}$ for an input $x' \in X$ by calculating log-likelihood of the target label and then maximizing it around single original input.

$$\arg\max_{x'} log\, Pr(\hat{y}|x')$$

The adversaries generated by this type of approach fails in real-world scenarios such as different camera angles and different distance from the camera to the target object [32][33]. To overcome this issue, this paper [7] generates adversaries by maximizing the expectation of log-likelihood, calculated by targeting a class $\hat{y}$ for a set of transformed inputs x^t.

$$\arg\max_{x^t} E_{x^t \sim A(p,x,l,t)} log\, Pr(\hat{y}|x^t)$$

Here, in each iteration, patch p is visualised by calculating expectation of log-likelihood and is applied at a location $l \in L$ on all inputs $x \in X$ by transforming $t \in T$.

$$\arg\max_{p} E_{x \sim X, t \sim T, l \sim L}[log\, Pr(\hat{y}|A(p,x,l,t))]$$

The flowchart of how prior works performed adversarial attack is depicted in Figure 3.1. At first, an adversary takes image x with true label y s.t $f(x) = y$ and feeds it to the classifier $f(.)$ to calculate log-likelihood value x' of the image with respect to the target class y'. Then the attacker adds this value x' to the image x ($x = x + x'$). This procedure is repeated in many iterations, and in each iteration, the log-likelihood value x' is maximised. Here in each iteration, the log-likelihood value x' is calculated and maximised around a single input image. Once all iterations are completed, the image x will be misclassified by the classifier to the target label y' s.t $f(x) = y'$.

The flowchart of how adversarial patch paper performs adversarial attack is shown in Figure 3.2. At first, an adversary takes a set of images x with true labels y s.t $f(x) = y$ and feeds it to the classifier to calculate log-likelihood value x' around the set of input images x with respect to the target label class y'. This log-likelihood value is visualised as a patch and attached to the input images x to produce transformed patched images. Here, it is called transformed images because while attaching the patch to

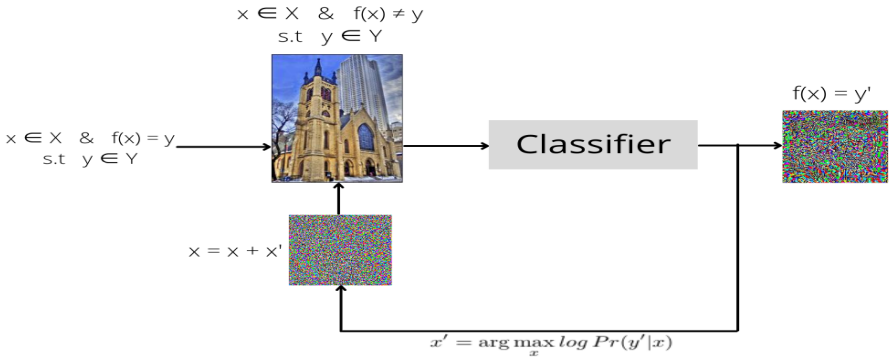


Figure 3.1: Flowchart of how prior works performed adversarial attack. Image source: Imagenette dataset mentioned in Section 5.1.2 is used to create this Figure

the image, transformation is done on both patch and images. In the next iteration, these transformed patched images are passed to the classifier to calculate log-likelihood value x'. They are maximised around the transformed images obtained from the previous iteration with respect to the target label class y'. This procedure is repeated for many iterations, and in each iteration, log-likelihood value x' is maximised. Once all iterations are complete, when final transformed patched images are fed to the classifier, misclassification occurs, and all the images will be incorrectly classified to the target class $f(x) = y'$

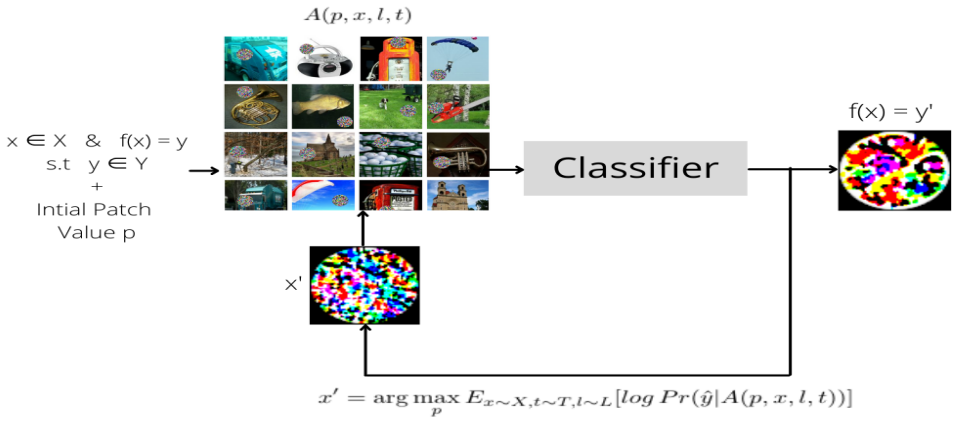


Figure 3.2: Flowchart of how adversarial patch paper performs adversarial attack. Image source: Imagenette dataset mentioned in Section 5.1.2 is used to create this Figure

3.2 Adversarial Patch Attack Method

In this research work to perform an adversarial patch attack, the source code provided by the python library called "*Adversarial Robustness Toolbox (ART)*" [43] is used. This library provides tools for researchers and developers to perform an adversarial attack, defend and evaluate the machine learning model. An adversarial patch is generated by using ART's Adversarial Patch module. At first, a classifier with TensorFlow framework is created by using the Deep Learning (DL) model, which needs to be attacked. Then generate a function of the Adversarial Patch module, which uses the input parameters, a set of images, and one-hot encoding of the target class is used to train the classifier.

3.2.1 Generation of Adversarial Patch

In each iteration, the loss function's gradient is calculated around the transformed image (patch) and is used to tune the weights of the DNN to maximise the loss. After each iteration, the calculated loss is visualized as a patch by converting it to NumPy. The gradients of loss for the next iteration are calculated relative to this patch. Hence, in each iteration, the patch gets optimized. Once the classifier is trained, the final loss is converted to NumPy and visualized as the patch. For this patch, loss value and one-hot encoded target class value is added as scalar data. An example of a generated patch is depicted in Figure 3.3. Categorical cross-entropy is the loss that is calculated. Adam optimizer is the optimizing algorithm used to train the classifier. Then this adversarial patch is attached to test data to obtain the patched dataset using the function apply_patch of Adversarial Patch module. This function takes the parameters, original images and scale of the patch w.r.t the input size of the classifier as the input. It attaches the adversarial patch of input scale on the original images at a random location to produces patched images. Finally, these patched images dataset is provided as input to the model to perform the attack.

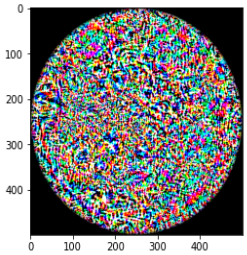


Figure 3.3: Sample of patch image generated on MobileNetV2 model using Beans dataset

The process of how a patch is generated is shown in Figure 3.4. At first, initial patch values, which is a slice of the classifier's clip value, are attached to the set of images at a random location after performing transformation to generated patched images. Then these patched images are provided as input to the classifier. Loss is calculated w.r.t target class. Next, the gradients of the classifier are updated using the loss. Loss is sent backwards to optimize the patch. This process of optimizing loss by sending it back

takes place till all the iterations are complete. Once all the iterations are complete, the final patch is generated by visualising the last loss.

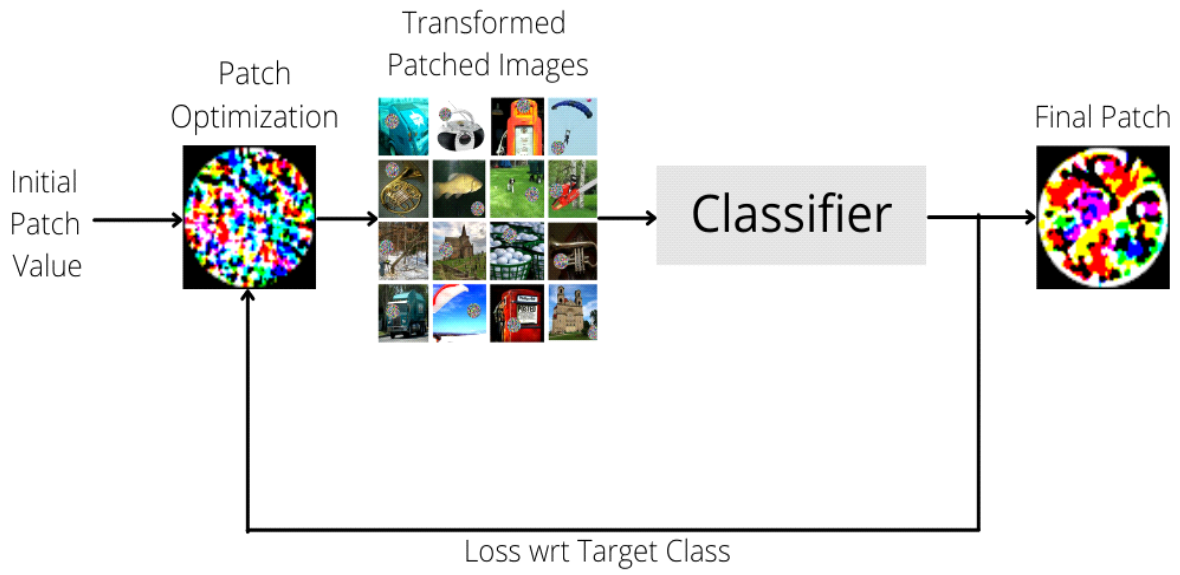


Figure 3.4: Flow chart of patch generation process. Image source: Imagenette dataset mentioned in Section 5.1.2 is used to create this Figure

3.2.2 Pseudocode of How Patch is Generated

Pseudo-code of generate function used to create a patch is shown in Pseudocode 1. As shown in the pseudocode, the generate function takes a set of images (X) as input along with a one-hot encoding value of the target label (Y). Here, the number of images and target labels are equal. The initial patch value is calculated at first. Then a new TensorFlow dataset is created by slicing X and Y with respect to the initialized batch size. Then 500 iterations are done; in each iteration, firstly, the patch value is generated by training the classifier providing $x \in X$ with respect to target value $y \in Y$ along with the initial value of patch. Secondly, this patch value is visualized as an image and randomly overlayed on the input set of images X to produce patched images. Thirdly, loss of classifier is obtained with respect to produced patched images. Finally, obtained loss and $mean(Y)$ is appended to the constructed patch image as scalar data. Once all 500 iterations are completed, the patch is passed out as output parameters from the generating function.

Pseudocode 1 Pseudo code for generate function of adversarial patch module to generate patch

Input images X, one hot-encoding of target labels Y, length(X) = length(Y)
Output patch

batch_size ← *16*
maximum_iteration ← *500*
clip_value ← *(0, 255)*
patch_value ← *np.ones(input_shape(X) * mean(clip_values)*
tensorflow_dataset ← *tf.data.Dataset.from_tensor_slices(X, Y)*
▷ TensorFlow API method
for *iteration = 0 to maximum_iteration* **do**
    for $x, y \in tensorflow\_dataset$ **do**
        patch_value = train(x, y, patch_value) ▷ training with base classifier
    end for
    patch_image = patch_value.numpy().transpose((2, 0, 1)) ▷ Converting to visualize as patch
    patched_images = random_overlay(patch_image, X)
    loss = compute_losses(patched_images, Y) ▷ compute loss w.r.t base classifier
    patch ← *add_scalar(loss, mean(Y))*
end for

The pseudocode of the training function executed inside the generate function is shown in Pseudocode 2. As depicted, the train function takes an image x, a target label y from the TensorFlow dataset and patch value as input. At first, the patch value is visualised and randomly overlayed on the input image x to create a patched image. Next, prediction of this patched image is made with respect to the base classifier that gives prediction value y_{pred}. Cross entropy loss is calculated between the input target label y and y_{pred}. Then this loss is used to update the weights of the classifier. Finally, the loss calculated is provided as output from the train function.

Pseudocode 2 Pseudocode for train function implemented in generate function of adversarial patch module

Input image x, one hot-encoding of target label y, patch_value
Output loss

patch_image = patch_value.numpy().transpose((2, 0, 1)) ▷ converting to visualize as patch
patched_image = random_overlay(patch_image, x)
y_pred ← *classifier.predict(patched_image)* ▷ prediction w.r.t base classifier
loss_per_example ← *categorical_crossentropy(y_true = y, y_pred)*
loss ← *tf.reduce_mean(loss_per_example)*

3.3 Experiment of Adversarial Patch Attack

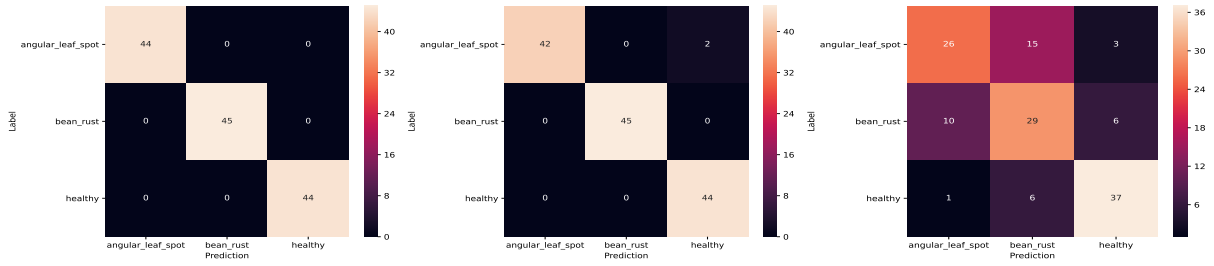
In this work, an experiment of adversarial patch attack is performed on three DNN models, i.e., MobileNetV2, ResNet50 and VGG16, using the Beans dataset, Imagenette dataset and RoboCup@Work dataset. The details about the datasets, DNNs and hyper-parameters used to perform the experiment in this section is provided in Chapter 5 .

Beans Dataset

The MobileNetV2, ResNet50 and VGG16 model is trained with the Beans training dataset. The hyper-parameter learning rate of MobileNetV2 and ResNet50 is 0.0001, and the learning rate of the VGG16 model is 0.00001. The accuracy of all three models before the attack is shown in Table 3.1. The confusion matrix for all the three models before the attack is illustrated in Figure 3.5.

Model	Training Accuracy (%)	Validation Accuracy (%)
MobileNetV2	100	97.66
ResNet50	100	96.88
VGG16	67.41	69.53

Table 3.1: Training and validation accuracy of MobileNetV2, ResNet50 and VGG16 model on Beans dataset before adversarial patch attack

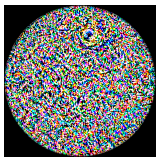


(a) MobileNetV2 (b) ResNet50 (c) VGG16

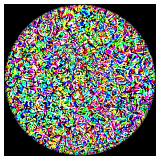
Figure 3.5: Confusion matrix of (a) MobileNetV2, (b) ResNet50 and (c) VGG16, after classification of Beans dataset before adversarial patch attack

To perform an adversarial patch attack on the three models used to classify the Beans dataset, 39 randomly chosen images from the Beans dataset are used to generate a patch. The generated patch w.r.t the individual model is depicted in Figure 3.6. Then the patch is overlaid on the rest of the test images at a random position to obtain the set of patched images. The scale percentage of the patch overlaid on the images is considered as 20, 25 and 35. Finally, the patched images are given as the input parameter to the trained models of the beans dataset to perform an adversarial patch attack which makes the model misclassify the image. The confusion matrix of all three models after performing adversarial attack targeting the healthy class of scale percentage 25 is shown in Figure 3.6. As shown, in the case of the MobieleNetV2 model (refer Figure 3.6(d)), the intensity of attack is less. In case of the ResNet50

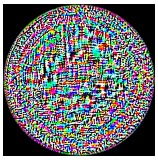
model (refer Figure3.6(e)), a patch attack has occurred targeting bean rust class instead of healthy class, but when the attack was performed against VGG16, all the images have successfully classified as the targeted healthy class as shown in 3.6(f). Here, both targeted and non-targeted patch attack happens in the case of Beans data. The accuracy of all three models after an adversarial patch attack decreases and is given in Table 3.2.



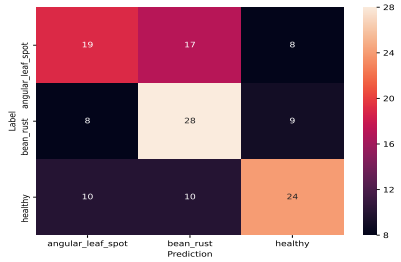
(a) Patch generated by MobileNetV2



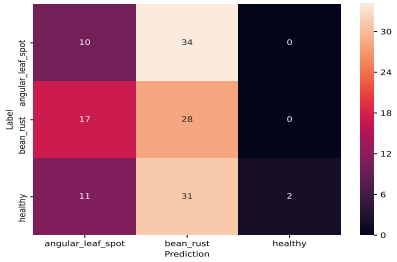
(b) Patch generated by ResNet50



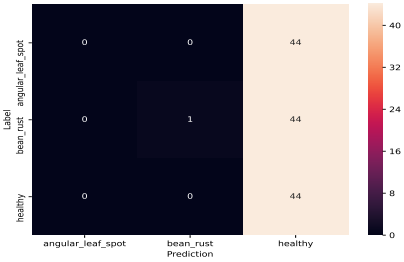
(c) Patch generated by VGG16



(d) Confusion matrix of MobileNetV2

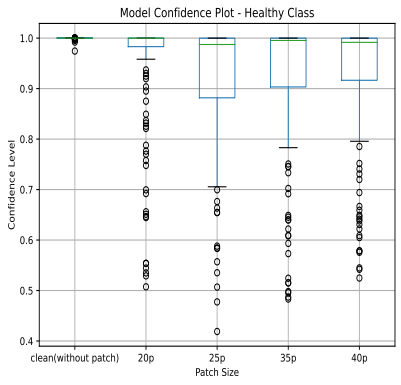


(e) Confusion matrix of ResNet50

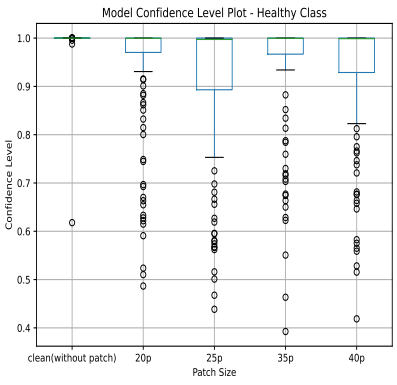


(f) Confusion matrix of VGG16

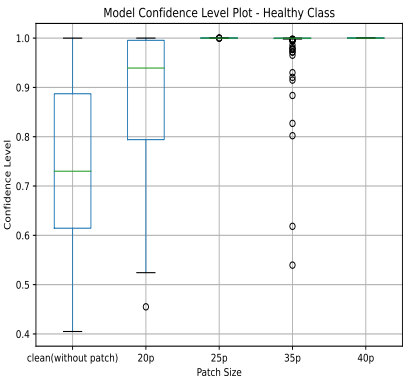
Figure 3.6: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting healthy class



(a) MobileNetV2 model



(b) ResNet50 model



(c) VGG16 model

Figure 3.7: Confidence of all models before and after adversarial patch attack targeting healthy class

Sl. No.	Model	Patch Scale (%)	Targeting Class (%)		
			angular leaf spot	bean rust	healthy
1	MobileNetV2	35	36.09	36.84	45.86
		25	55.63	53.38	53.38
		20	66.91	66.91	65.41
2	ResNet50	35	35.33	27.59	30.82
		25	35.33	27.81	30.07
		20	36.84	32.33	35.33
3	VGG16	35	33.08	33.83	33.08
		25	42.12	38.34	33.93
		20	48.10	51.12	40.60

Table 3.2: Accuracy of the models after adversarial patch attack on Beans dataset (decrease in accuracy can be observed)

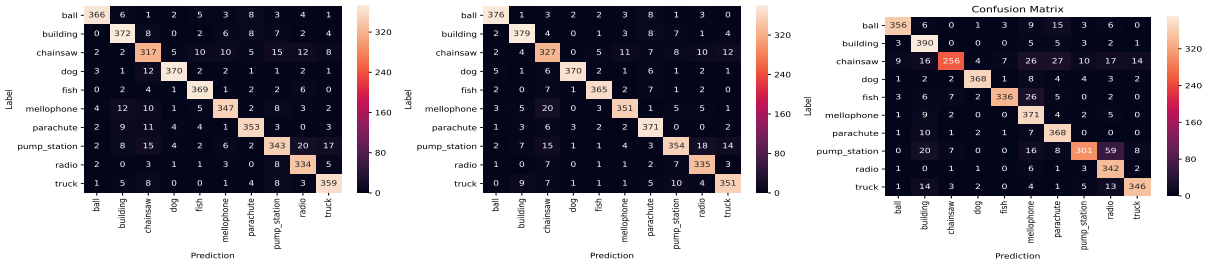
Firstly, as shown from the obtained results, the model's accuracy is high before the attack, and once the attack is made, the accuracy decreases. Secondly, the scale value of the patch is also an essential factor while performing the attack. One more interesting observation which can be noticed from the results is while performing an adversarial attack on ResNet50 model targeting healthy class, the attack is successful in misclassifying certain images, but it fails to perform targeted attack as a high percentage of images were classified as bean rust class (refer Figure 3.6(e)). Whereas while the attack is performed on the VGG16 model, the attack is successful in both misclassification of images and in performing targeted attack(refer Figure 3.6(f)). The Confidence level of the considered models before the attack and after adversarial patch attack with different patch sizes considered is shown in Figure 3.7. As depicted, before the attack, the confidence level in predictions of the MobileNetV2 model and ResNet50 model is high, and after the adversarial attack, the predictions confidence level decreases. But in the case of the VGG16 model, before the attack, predictions confidence is between 60-90%, but after the attack, the confidence increases. From the accuracy table depicted in Table 3.2, the considered model's accuracy is in inverse proportion with patch size.

Imagenette dataset

The MobileNetV2, ResNet50 and VGG16 model is trained with Imagenette training dataset. The hyper-parameter learning rate of MobileNetV2 and ResNet50 is 0.0001, and the learning rate of VGG16 is 0.0001. The accuracy of the three models before the attack is depicted in Table 3.3. The confusion matrix of the classification w.r.t the models before the attack is depicted in Figure 3.8.

Model	Training Accuracy (%)	Validation Accuracy (%)
MobileNetV2	99.82	89.94
ResNet50	99.62	91.18
VGG16	99.76	89.49

Table 3.3: Training and Validation accuracy of MobileNetV2, ResNet50 and VGG16 model on Imagenette dataset

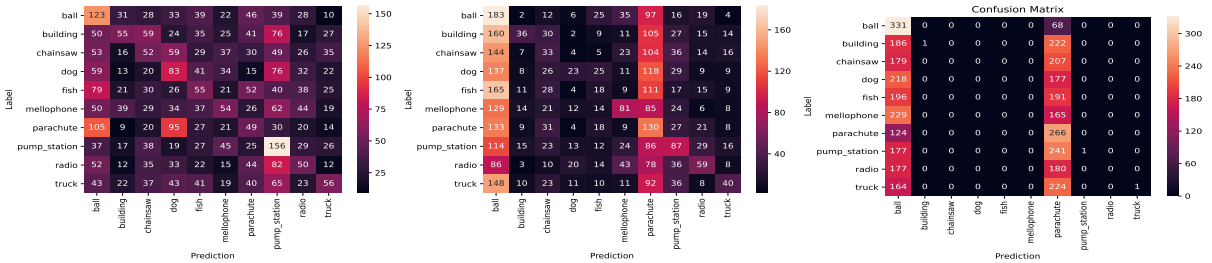


(a) MobileNetV2 (b) Patch generated by ResNet50 (c) Patch generated by VGG16

Figure 3.8: Confusion matrix of (a) MobileNetV2, (b) ResNet50 and (c) VGG16 after classification of Imagenette dataset before adversarial patch attack

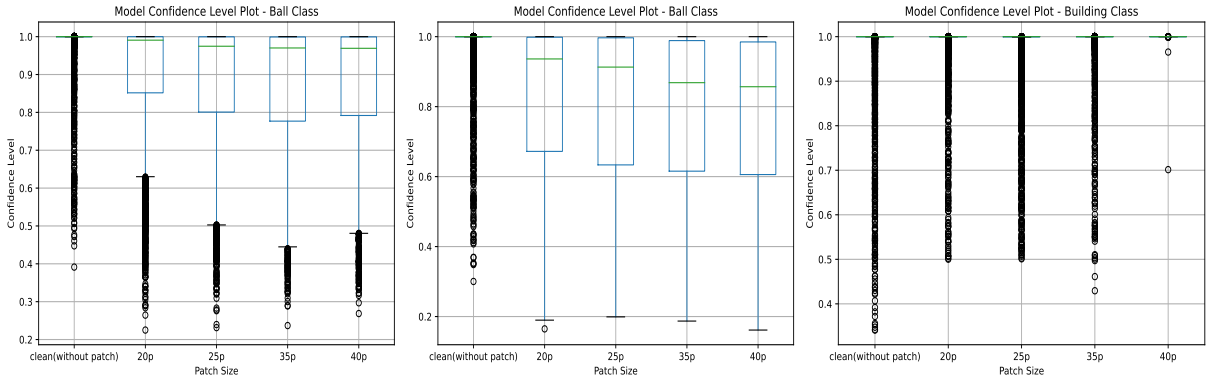
To perform an adversarial patch attack on the three models used for classification of Imagenette dataset, 39 randomly chosen images from the same dataset to generate a patch. The generated patch w.r.t the individual model is depicted in Figure 3.9. Then the patch is overlaid on the rest of the test images at a random position to obtain the set of patched images. The scale percentage of the patch overlaid on the images is considered as 20, 25 and 35. Finally, the patched images are given as the input parameter to the trained models of the Imagenette dataset to perform an adversarial patch attack which makes the model misclassify the image. The confusion matrix of the three models after performing adversarial attack targeting building class of scale percentage 25 is shown in Figure 3.9. The accuracy of all three models after an adversarial patch attack is given in Table 3.4.

(a) Patch generated by MobileNetV2 (b) Patch generated by ResNet50 (c) Patch generated by VGG16



(d) Confusion matrix of MobileNetV2 (e) Confusion matrix of ResNet50 (f) Confusion matrix of VGG16

Figure 3.9: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting building class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 3.10: Confidence of all models before and after adversarial patch attack targeting building class

Sl. No.	Model	Patch Scale (%)	Targeting Class (%)				
			ball	chainsaw	mellophone	parachute	radio
1	MobileNetV2	35	11.36	11.97	11.84	12.68	11.26
		25	18.67	19.49	19.94	19.46	20.40
		20	29.19	30.26	28.10	29.04	30.03
2	ResNet50	35	11.97	10.82	11.26	12.61	12.12
		25	17.57	16.33	18.29	17.88	17.57
		20	25.52	23.08	24.56	23.66	23.71
3	VGG16	35	14.64	13.42	14.75	9.93	12.00
		25	15.28	13.55	12.99	11.38	15.54
		20	14.29	16.33	15.87	14.75	13.35

Table 3.4: Accuracy of the models after adversarial patch attack on Imagenette dataset (decrease in accuracy can be observed)

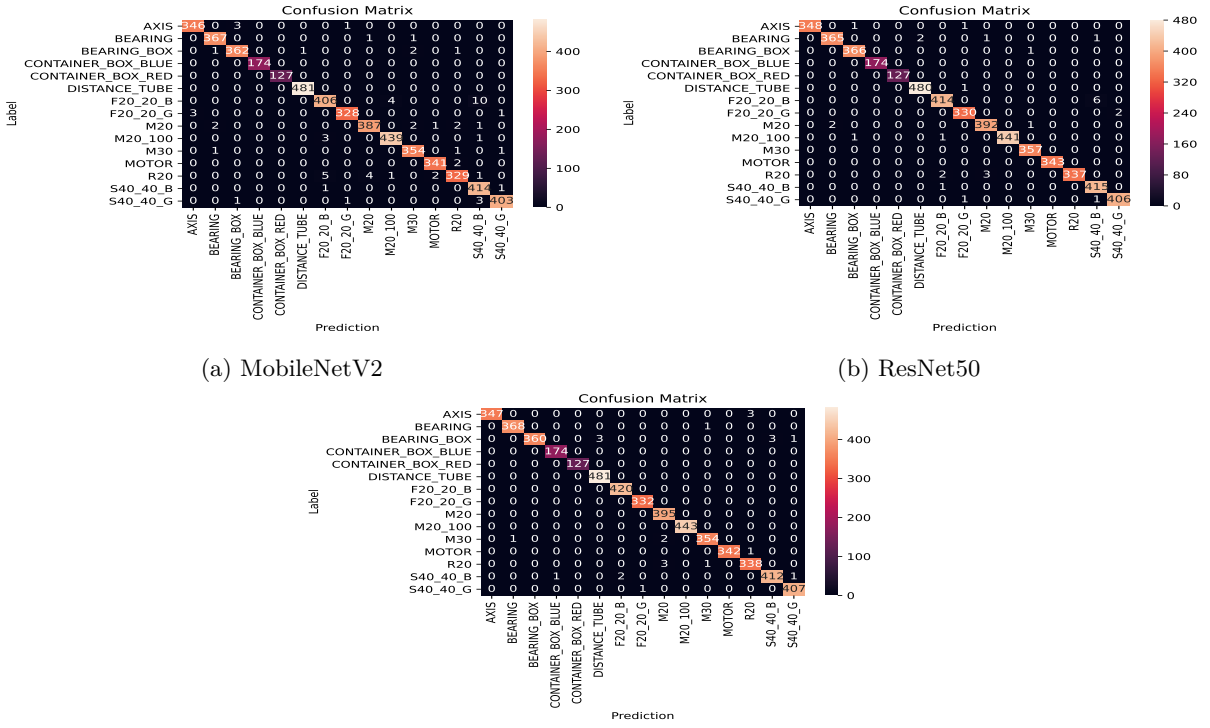
The obtained results show that after performing an adversarial patch attack, the accuracy of the three DL models decreases. And as can be noticed, the scale value of the patch plays an essential role while performing the attack. Similar to the results of an adversarial attack on the Beans dataset, an interesting observation which can be noticed here is while performing an adversarial attack on ResNet50 and VGG16 model targeting building class, a high percentage of images are classified as ball class and parachute class (refer Figure 3.9(e) and 3.9(f)). Here target attack has occurred on two classes which is ball and parachute. Then while the attack is performed on the MobileNetV2 model, the attack is successful in misclassifying the images, but it fails to perform the targeted attack (refer Figure 3.9(d)). The Confidence level of the condsidered models before the attack and after adversarial patch attack with different patch size considered is shown in Figure 3.10. As depicted, before the attack, confidence level in predictions of MobileNetV2 model and ResNet50 model is high, and after adversarial attack, the predictions confidence level decreases. But in case of VGG16 model, predictions confidence both before and after the attack, remains high. From the accuracy table shown in Table 3.4, it can be observed that accuracy of the considered models increases with decrease in the patch size of the attack.

RoboCup@Work dataset

The MobileNetV2, ResNet50 and VGG16 model is trained with the RoboCup@Work training dataset. The hyper-parameter learning rate of MobileNetV2 and ResNet50 is 0.0001, and the learning rate of VGG16 is 0.00001. The accuracy of the three models before the attack is shown in Table 3.5. The confusion matrix of the classification with respect to the models before the attack is shown in Figure 3.11.

Model	Training Accuracy (%)	Validation Accuracy (%)
MobileNetV2	99.54	98.76
ResNet50	99.74	99.46
VGG16	99.90	99.55

Table 3.5: Training and Validation accuracy of MobileNetV2, ResNet50 and VGG16 model on RoboCup@Work dataset



(a) MobileNetV2

(b) ResNet50

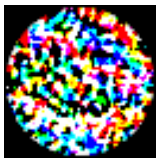
(c) VGG16

Figure 3.11: Confusion matrix of (a) MobileNetV2, (b) ResNet50 and (c) VGG16 after classification of RoboCup@Work dataset before adversarial patch attack

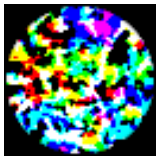
To perform an adversarial patch attack on the three models used to classify RoboCup@Work dataset, 39 randomly chosen images from the RoboCup@Work dataset are used to generate a patch. The generated patch with respect to the individual model is depicted in Figure 3.12. Then the patch is overlaid on the rest of the test images at a random position to obtain the set of patched images. The scale percentage of the patch overlaid on the images is considered as 35, 25 and 20. Finally, the patched images are given as the input parameter to the trained models of the RoboCup@Work dataset to perform an adversarial patch attack, making the model misclassify the image. The confusion matrix of all the models after performing adversarial attack targeting bearing box class of scale percentage 25 is shown in Figure 3.12. The accuracy of all three models after an adversarial patch attack is given in Table 3.6.

Sl. No.	Model	Patch Scale (%)	Targeting Class (%)				
			Bearing Box	Distance Tube	F20_20_B	M20	Motor
1	MobileNetV2	35	7.45	6.93	7.06	7.71	7.40
		25	9.10	9.56	9.56	9.05	9.95
		20	12.60	12.67	11.55	13.44	12.82
2	ResNet50	35	7.26	7.21	7.75	7.09	7.23
		25	8.09	8.22	8.86	8.79	9.03
		20	8.47	8.45	8.52	8.82	9.77
3	VGG16	35	10.61	9.09	10.55	9.20	8.09
		25	12.47	9.95	10.85	10.10	12.67
		20	13.87	8.90	11.40	12.19	14.23

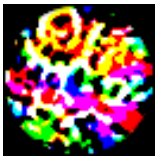
Table 3.6: Accuracy of the models after adversarial patch attack (decrease in accuracy can be observed)



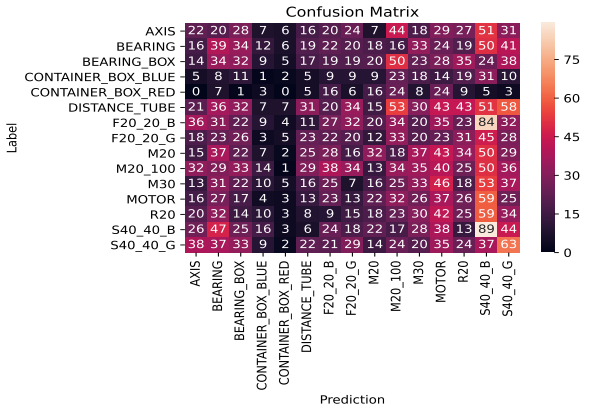
(a) Patch generated by MobileNetV2



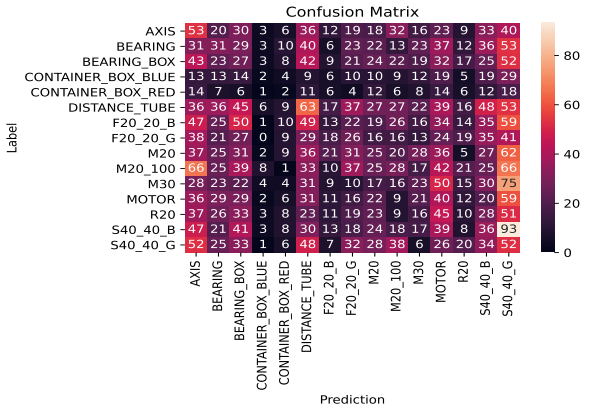
(b) Patch generated by ResNet50



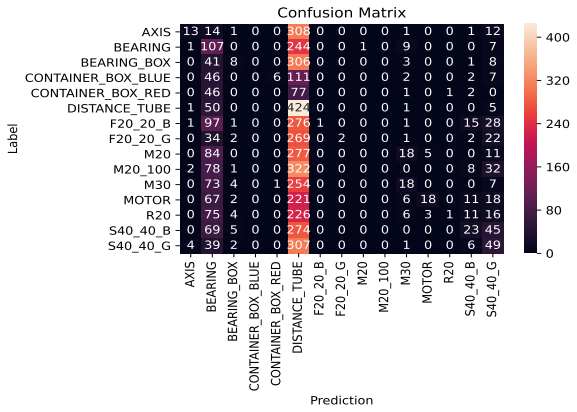
(c) Patch generated by VGG16



(d) Confusion matrix of MobileNetV2

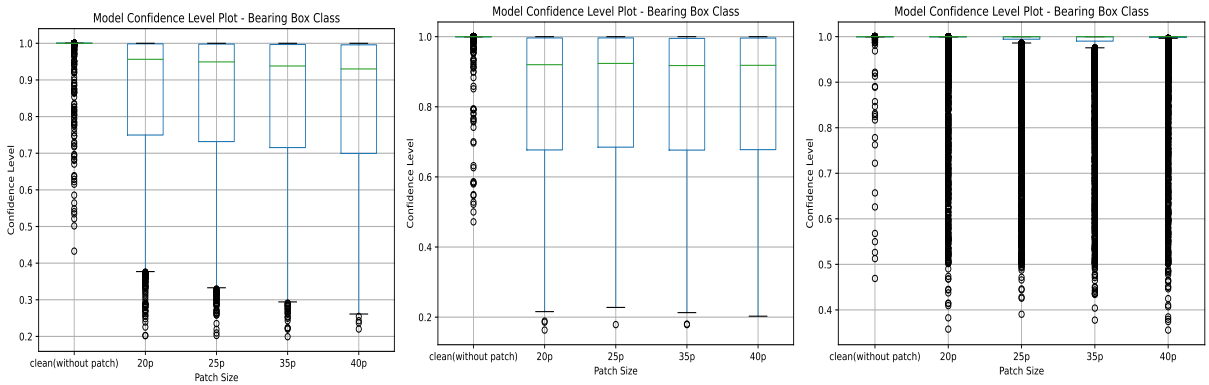


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 3.12: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bearing box class



(a) ResNet50 model (b) VGG16 model

Figure 3.13: Confidence of all models before and after adversarial patch attack targeting bearing box class

The results obtained show that after performing an adversarial patch attack, the accuracy of DL models decreases. As can be noticed, while performing an adversarial attack on MobileNetV2 and ResNet50 model targeting bearing box class, the attack is successful in misclassifying the images but it fails to perform the targeted attack (refer Figure 3.12(d) and 3.12(e)). Then while the attack is performed on the VGG16 model targeting bearing box class, a high percentage of images are classified as distance tube class (refer Figure 3.12(f)), here targeted patch attack has occurred on distance tube class instead of bearing box class. The confidence level of the considered models before the attack and after adversarial patch attack with different patch sizes considered is shown in Figure 3.13. As depicted, before the attack, the confidence level in predictions of the MobileNetV2 model and ResNet50 model is high, and after the adversarial attack, the predictions confidence level decreases. But in the case of the VGG16 model, predictions confidence both before and after the attack remains high. From the accuracy table depicted in Table 3.6, it is evident that the accuracy of the considered models increases with a decrease in the patch size of the attack.

4

State of the Art

In this chapter, state-of-the-art defense techniques against adversarial patch attacks are described.

The defense mechanisms against adversarial attacks are constructed in three different way as depicted in Figure 4.1:

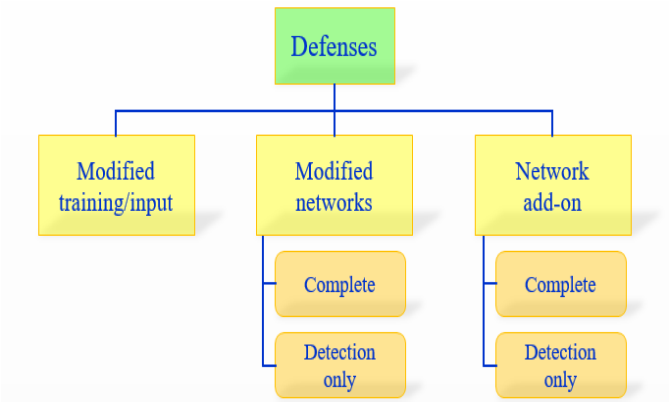


Figure 4.1: Different ways of categorization adversarial defense mechanisms. Image source : [2]

- Modify training/testing data
  - In this way of construction, defense is independent of the neural network. This method aims to detach perturbations from the given input by refactoring the input data.
  - Adversarial training defense [55], foveation-based defense [33], data compression [15] are some of the techniques which belongs to this category.
- Modify the neural network architecture
  - In this way of construction, neural network architecture is modified to provide the defense. Modification of network may be changing loss functions, weights of neurons, changing activation functions, adding or removing layers.

  - Gradient masking/regularization [51], uncertainty estimation method, defense distillation [46], deep contractive networks [20] are some of the mechanisms which belongs to this category.

- External methods/models are used as add-ons with the underlying neural network

  - In this way of construction, the given input is first passed through the external methods/models used as add-ons to remove adversaries present in the input. Then after removing the adversaries, the input is passed to the underlying network for classification.
  - MagNet defense technique [36], GAN-based defense [53] are some of the mechanisms which belong to this category.

The modified neural network way and external methods/models add-on can be divided into complete type, and detection only type. In complete type, defense identifies the adversaries, removes adversaries from the input data, and passes the input for classification. But in detection only type, defense identifies the adversaries and informs the network about the presence of adversaries so that particular adversarial input can be ignored or eradicated.

RQ1. What are the existing state-of-the-art defense mechanisms against adversarial patch attacks in deep learning?

4.1 Local Gradients Smoothing (LGS) Defense

This method was proposed in the paper "*Local Gradients Smoothing: Defense against localized adversarial attacks*" [41]. When an adversarial patch attack is performed, the patched region on the adversarial images contains high-frequency noise. The motivation of this method is to identify high-frequency noise locations present on adversarial images and suppress them without affecting low-frequency locations. At first, the first-order magnitude of image gradients is calculated. Then this gradient is normalized and applied to the original image, which inturn helps detect noise perturbations on the image, and then noise suppression is done. This way, high-frequency locations are identified and suppressed.

This defence mechanism's idea is that those high-frequency locations do not contain valuable information needed for final classification. Hence, there is a high probability that these locations include perturbations. So the solution is to find these locations and suppress them. But this may also suppress low-frequency locations present on the image, which contains information needed for final classification. To overcome this problem, after applying the normalized gradient map on the image, a block-wise approach is followed in which the image is divided into overlapping blocks. The frequency of the overlapping blocks is estimated and compared to a value (threshold value). If the frequency is more than the threshold value, then that block is suppressed.

The Figure 4.2 shows the working of the LGS defense mechanism. LGS first calculates first-order magnitude gradients of image and normalizes it. Then it is applied on top of the input adversarial image as depicted in the Figure 4.2. Then block-wise approach is employed to find high-frequency regions based on comparison with threshold value. The blocks which has frequency higher than threshold value is suppressed. This is the working of LGS defense mechanism against adversarial patch attack.

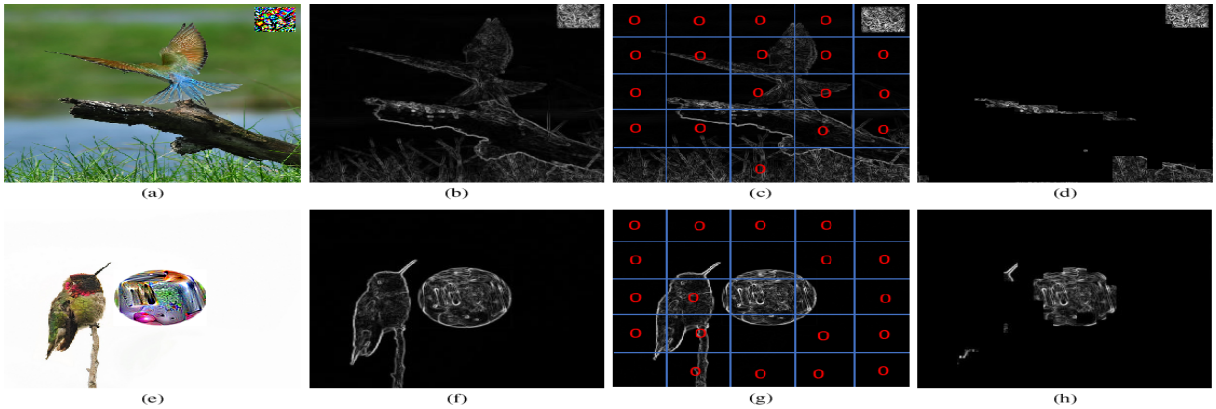


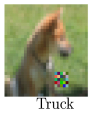
Figure 4.2: Working of Local Gradient Smoothing (LGS) defense. (a) and (e) represents patch-based adversarial image, (b) and (f) represents normalised gradient map, (c) and (g) represents block-wise approach applied to find high-frequency locations, (d) and (h) represents high-frequency regions which needs to be suppressed. Image source : [41]

4.2 Minority Reports Defense

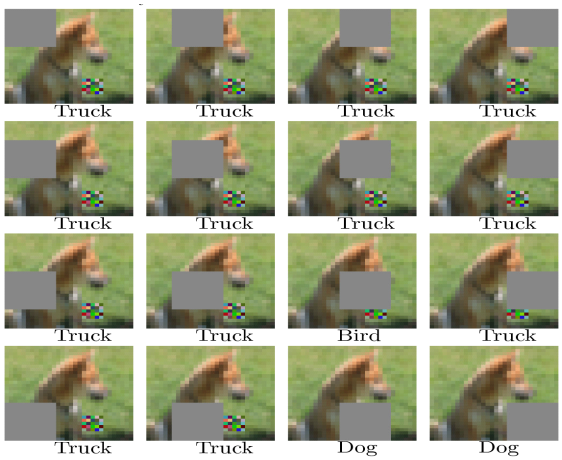
This method was proposed in the thesis report of "*Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches*" [35]. The idea behind this defense mechanism is to occlude (with a uniform grey rectangle image) the patch region of the adversarial image to produce an occlusion image and then classify the occlusion image. Patch present on adversarial images generated by adversarial patch attack influence the prediction of the classifier. Hence, by performing occlusion and covering the patch region, the influence of patch on the predictions of the classifier can be reduced.

But the issue is, the position of the patch overlaid on the adversarial image is unknown. Hence, this technique first identifies possible areas of the patch and then occlude it, creating several occlusion images. That is, for a single image, different possible patch locations are identified. On each possible patch location, an occlude image is applied on top of it and treated as a new occlusion image. A critical factor in this defense mechanism is that the size of the occlude region (grey rectangle image) is larger than the adversarial patch region. If occlude areas are larger, they can completely overlap the adversarial patch region, and hence this adversarial patch will have no influence on the classifier. Thereby, the image will be rightly classified. This defense is called "*minority reports defense because there will always be a minority of predictions that cannot be influenced by the attacker and (presumably) vote for the correct label*" [35]. Another essential factor of this defense is the training dataset. Occlude region is added to normal training images to create a training set of occlusion images. If and only if the training performance of the classifier on a training set of occlusion images is good, and if the classifier can provide good prediction results on occlusion images, then this defense mechanism succeeds or else this defense mechanism does not work.

Figure 4.3 shows how occlusion is performed. As depicted, the input is an adversarial patched dog image (refer Figure 4.3a). All possible locations of the adversarial patch present on this input image are identified. Occlude region is applied on all identified regions separately, creating a set of occlusion images.



(a) Adversarial patch attack on dog image



(b) Occlusion of adversarial images with classification prediction label. Here, in the 4th row, 3rd and 4th image are not influenced by adversarial patch attack as patch is completely covered by occlude region (grey squares).

Figure 4.3: Working of occlusion defense. Image source: [35]

Then these sets of occlusion images are passed to the classifier, output predictions of the classifier are represented in the form of a grid. As depicted in Figure 4.3b, in the 4^{th} row, 3^{rd} and 4^{th} image have classified correctly as a dog since the occlusion region have completely covered the patch region of the image. The intuition here is that when an attack happens, most occlusion images get mispredicted for the same incorrect class label. And there exist a minority of occlusion images that get predicted correctly for the true class label. This minority set of images are treated as non-adversarial images. Since majority result is ignored and minority results are considered, this defense is named minority reports defense.

4.3 Certified Defense

This defense mechanism was proposed in the paper "*Certified Defenses for Adversarial Patches*" [12]. The intuition for this defense is to have a verifier which generates a certifiable certificate to guarantee that the input is not an adversarial image. That is "*Given a model and an input, the verifier outputs a certificate if it is guaranteed that the image can not be adversarially perturbed. This is done by checking whether there exists any nearby image (within a prescribed l_p distance) with a different label than the image being classified*" [12]. The verifier outputs a certifiable radius, the lower bound distance between the present input image and the nearest another class image. Then the neural network is trained to tighten this lower bound. Here, the process of training a network to tighten the lower bound is called certificate training.

Interval bound propagation (IBP) [18] is used as the verifier in this defense mechanism. Certificate training helps IBP to produce strong certificates. IBP produces the upper bound and the lower bound interval of activation of the layer neurons. While the training of the network, certificate loss is calculated and minimized to increase the performance of the certificate. To defend against adversarial patch attack, the network is given all possible patch images to train with all possible patch locations. Each patch image

is trained with all possible locations. The pixels inside the patch region are re-assigned to a value with upper bound as 1 and lower bound as 0, and the pixels outside the patch region remain the same as the original pixel value. Using this bound interval, the network is trained to generate the certificate.

The issue with this approach is the long training time required. That is, the network needs to be trained with all possible locations for each patch image. This requires a long training time. To overcome this issue, two methods have been proposed by the same paper [12] to perform certificate training. One is random patch certificate training in which, instead of considering all patch images, a set of random patches are selected and used to train the model. The other one is guided patch certificate training in which the U-net [50] method is used to choose few locations of patch images instead of selecting all possible locations of patch images.

4.4 PatchGuard Defense

PatchGuard defense is proposed in the paper "*PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking*" [67]. The intuition for the defense mechanism is to use a small receptive fields in Convolutional Neural Network (CNN) to put a limit on how many features can be corrupted in the presence of an adversarial patch attack. The receptive field is the place on the input image at a point of time where the convolutional kernel is present. The large receptive field captures even the small patch regions, which inturn influence local predictions. Hence, this defense mechanism uses small receptive fields to overcome this issue, which ensures that an adversarial patch corrupts fewer local predictions. High feature values present in these corrupted regions can be detected when a small amount of local features are affected. Then a robust masking process is applied to mask off these corrupted regions and limit the influence of the patch on final network predictions.

The working of the PatchGuard defense mechanism is depicted in Figure 4.4. An input image is provided to the CNN, which has small receptive fields. Features are extracted from these small receptive fields. Small receptive fields help to generate high-feature values from adversarial patch regions. Robust masking identifies high-feature values, treats them as abnormal elements and masks these abnormal elements. In Figure 4.4, an adversarial image of fish is given as input to the CNN, which has small receptive fields and then features are extracted and represented in matrix form. As shown, components extracted from adversarial regions contains a high-feature value, and these high-feature values are highlighted with a red colour matrix. Then aggregation of initial feature values is done, which is called insecure aggregation. The final prediction leads to truck class (as highlighted with red colour in the depicted figure). To prevent this insecure aggregation, a robust masking technique is used to identify and mask off the extracted high-feature value. Once high-feature values are masked by robust masking, remaining feature values are aggregated to make the final prediction which leads to the true class label, fish.

The two crucial parts of this PatchGuard defense mechanism are to use small receptive fields and execution of robust masking process. By using small receptive fields, this defense mechanism reduces the possibility of an adversarial patch corrupting a large number of features extracted from the given input image. Small receptive fields also help to obtain high-feature values from the patch region. On the other hand, robust masking helps identify high-feature value and mask it, get true feature-values of the

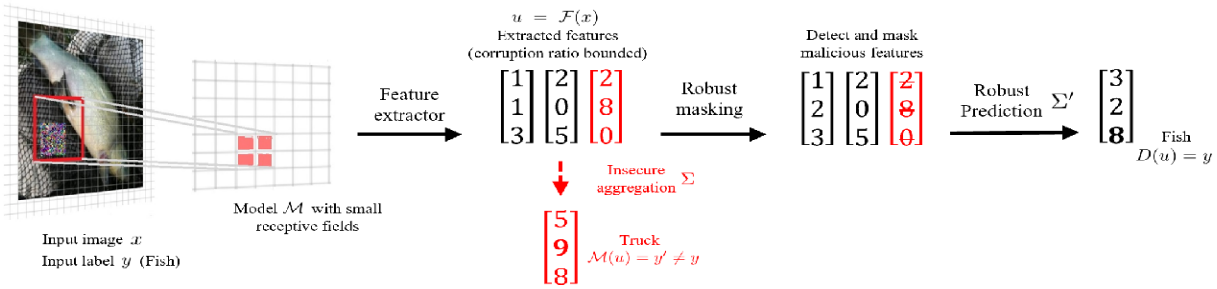


Figure 4.4: Overview of PatchGuard defense. Image source: [67]

image, and preserve the image's actual label class after prediction. But the size of receptive fields must be balanced carefully while using this defense mechanism as choosing the wrong receptive field size leads to ineffective defense. Similarly, sometimes robust masking can wrongly mask the feature-values which lead to false positive (FP). Even when partial-corrupted feature values are extracted from the image, designing of robust masking technique becomes difficult.

4.5 Clipped BagNet (CBN) Defense

Clipped BagNet (CBN) is presented in the paper "*Clipped BagNet: Defending Against Sticker Attacks with Clipped Bag-of-features*" [70]. CBN is built upon the classifier BagNet-33 [6], BagNet-33 is the classifier with receptive field size of 33 x 33. The part of the input image obtained from the receptive field is examined separately in BagNet, and from each image, part prediction is obtained and aggregated to get the final class prediction. The idea behind CBN is to add bounds to the BagNet's aggregation step value to mitigate the impact of adversarial patch on final network prediction. BagNet classifier generates high-dimensional logits vector from each part of the input image, the receptive field is focused on and averages it, to obtain global logits vector. Then, the global logits vector is passed to softmax to produce final classification predictions. "*BagNet's structure as an average of per-patch logits forms a foundation*" [70] for this CBN defense mechanism.

BagNet is not robust in handling a large change in high-dimensional logits vector value. Hence, CBN was proposed to overcome this limitation and can be used to defend against adversarial patch attacks. In CBN, once high-dimensional logits vector is generated from the receptive field part of the image. Clipping operation is done to bound the per-patch logits. After bounding the high-dimensional logits vector, its average is taken and passed to softmax to obtain final predictions with bound limits. The clipping function used in CBN is the tanh function, which helps derive upper bounds and lower bounds of the final global logits vector. But this defence mechanism has no idea about the patch location on the input image. So, all possible patch locations are considered and iterated to find which image location can alter the final classification.

5

Datasets and Methods

This chapter describes datasets used to perform experiments in this research work and DNN methods on which experiments are done.

5.1 Datasets

This research evaluates three DNN methods on three image datasets: Beans dataset, Imagenette dataset and RoboCup@Work dataset, the description of these datasets is given below sections.

5.1.1 Beans Dataset

Beans dataset [29] is Tensorflow dataset which contains the images of beans leaves captured using smartphone cameras. This dataset has three classes: Angular Leaf Spot class (refer Figure 5.1(a))and Bean Rust class (refer Figure 5.1(b))depicted as disease classes and Healthy class (refer Figure 5.1(c)). The size of this image dataset is 500x500 resolution. The sample of this dataset is shown in Figure 5.1.

(a) Angular Leaf Spot class (b) Bean Rust class (c) Healthy class

Figure 5.1: Sample of Beans dataset with class names. Image source : [29]

5.1.2 Imagenette Dataset

Imagenette dataset [24] is also Tensorflow dataset and is a subset of the Imagenet dataset. This dataset has ten classes obtained from Imagenet data. Since Imagenet data is huge and takes a lot of time to train,

this dataset was created. The size of this image dataset used in this research work in 160x160 resolution. Figure 5.2 portrays the images of this dataset.

(a) Ball class (b) Buildings class (c) Cutting Machine class/ Chainsaw class (d) Dog class

(e) Fish class (f) Musical Instrument class/ Mellophone class (g) Parachute class (h) Pump class

(i) Radio class (j) Truck class

Figure 5.2: Sample of Imagenette dataset with class names. Image source : [24]

5.1.3 RoboCup@Work Dataset

RoboCup@Work dataset is a dataset created by the RoboCup team of Hochschule Bonn-Rhein-Sieg. This dataset has fifteen classes of images captured with respect to different angles and light conditions. The size of this image dataset used in this research work is 64x64 resolution. The images of this dataset with its class is shown in Figure 5.3.

(a) Axis class (b) Bearing class (c) Bearing Box class (d) Container Box Blue class

(e) Container Box Red class (f) Distance Tube class (g) F20_20_B class (h) F20_20_G class

(i) M20 class (j) M20_100 class (k) M30 class (l) Motor class

(m) R20 class (n) S40_40_B class (o) S40_40_G class

Figure 5.3: Sample of RoboCup@Work dataset with class names

5.2 Methods

The following three TensorFlow DNNs are used in this research work, and all three are designed to perform multiclass classification of images:

5.2.1 MobileNetV2

MobileNetV2 architectural model is based on Convolutional Neural Network (CNN), and it has an inverted residual network with bottleneck features. For this R&D work, the MobileNetV2 model is downloaded from TensorFlow and utilised [54]. Specific hyperparameter values of this model remain constant while training on all three datasets is depicted in Table 5.1.

5.2.2 ResNet50

ResNet50, a variant of the ResNet model, is also based on CNN. For this R&D work, the pre-trained ResNet50 model is downloaded from TensorFlow and utilised [21]. The hyperparameter values, which remains the same while training on the datasets is depicted in Table 5.1.

5.2.3 VGG16

Adding to the models mentioned above, one more CNN based model used in this research work is VGG16. Similar to the other two models, the pre-trained TensorFlow VGG16 model is downloaded and utilised [57]. The hyperparameter values used while training on the datasets is depicted in Table 5.1.

Hyperparameter	**Value**		
	MobileNetV2	**ResNet50**	**VGG16**
Pretrained weight	Imagenet	Imagenet	Imagenet
Alpha	0.35	-	-
Batch size	32	32	32
Dropout	0.3	0.3	0.5
Training epochs	100	100	100
Optimizer	Adam	Adam	Adam
Activation function	Relu	Relu	Relu
Learning rate	0.0001	0.0001	0.00001

Table 5.1: Hyperparameter specification for MobileNetV2, ResNet50 and VGG16 model

6

Experimentation & Evaluation

This chapter describes different experiments conducted in this research work and their results.

6.1 Adversarial Patch Targeted Attack

RQ2. Can an adversarial patch attack always perform the targeted attack?

An adversarial patch attack can perform both targeted attack and non-targeted attack. But is it possible to achieve a targeted attack every time? The objective of this experiment is to check whether is it possible to perform a targeted adversarial patch attack at all times.

6.1.1 Hypothesis

Hypothesis: an adversarial patch attack can always perform targeted attack

To perform this experiment, a hypothesis is considered. The hypothesis states that it is possible to achieve a targeted adversarial attack every time.

6.1.2 Observation

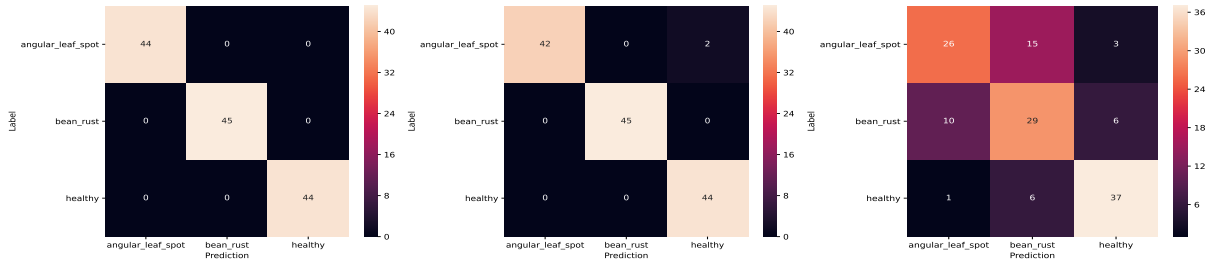
In this experimentation, an adversarial patch attack is performed on all three DL models with respect to the Beans dataset, Imagenette dataset and RoboCup@Work dataset, considering a patch size of 25%.

Beans Dataset

The confusion matrix of all the three DNN trained on the Beans dataset before the attack is illustrated in Figure 6.1. As depicted in the confusion matrix, images are adequately classified. The patch generated to perform the adversarial attack targeting angular leaf spot class w.r.t considered models is illustrated in Figure 6.2. Then after the attack, firstly, from Figure 6.2(d) when an adversarial patch attack is targeting angular leaf spot class, the targeted attack does not happen, even intensity of the attack is less on the MobileNetV2 model. But in the case of the ResNet50 model (refer Figure 6.2(e)), the targeted attack happens, but the bean rust class is targeted instead of angular leaf spot. Similarly, in the case of the VGG16 model (refer Figure 6.2(f)), the targeted adversarial attack occurs on a healthy class instead of an angular leaf spot class.

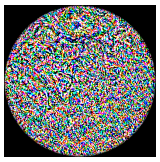
Secondly, the patch generated to perform adversarial patch attack targeting bean rust class is shown in the Figure 6.3 with respect to each model. After the attack, in the case of the MobileNetV2 model, as depicted in Figure 6.3(d) bean rust class is successfully targeted by the attack, but the intensity of attack is low. Then in the case of the ResNet50 model (refer Figure 6.3(e)), the targeted attack happens but not entirely as some of the images are classified as angular leaf spot class. But in the case of the VGG16 model, as depicted in Figure 6.3(f), the adversarial attack is successful in targeting the bean rust class with high accuracy.

Finally, the patch generated to perform adversarial patch attack targeting healthy class is shown in the Figure 6.4 with respect to each model. After the attack, as shown, the intensity of the targeted attack is low in the MobileNetV2 (refer Figure 6.4(d)) case, but in ResNet50 case, non-targeted adversarial attack occurs but targeted adversarial attack does not occur as depicted in Figure 6.4(e). In the case of the VGG16 model, as depicted in Figure 6.4(f) targeted adversarial attack perfectly occurs targeting healthy class.

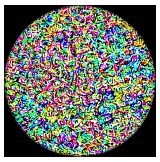


(a) Confusion matrix of MobileNetV2 (b) Confusion matrix of ResNet50 (c) Confusion matrix of VGG16

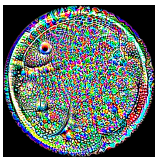
Figure 6.1: Confusion matrix of considered models before adversarial patch attack



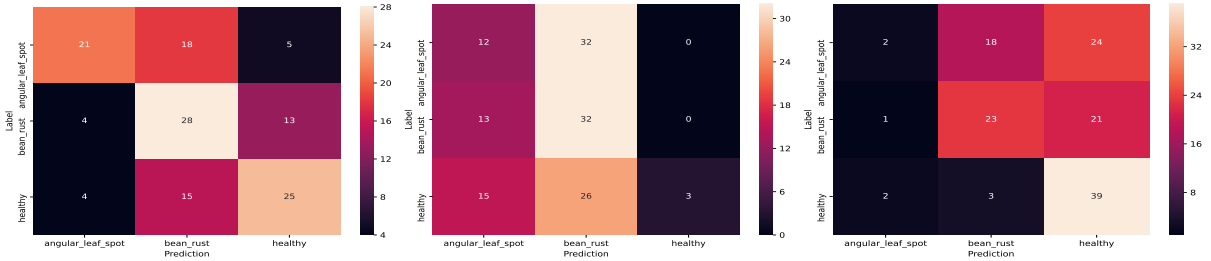
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

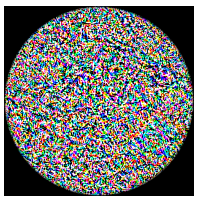


(c) Patch generated by VGG16 model

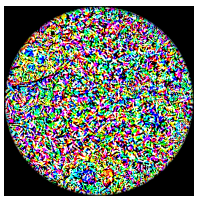


(d) Confusion matrix of MobileNetV2 (e) Confusion matrix of ResNet50 (f) Confusion matrix of VGG16

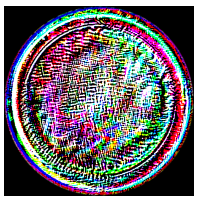
Figure 6.2: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting angular leaf spot class



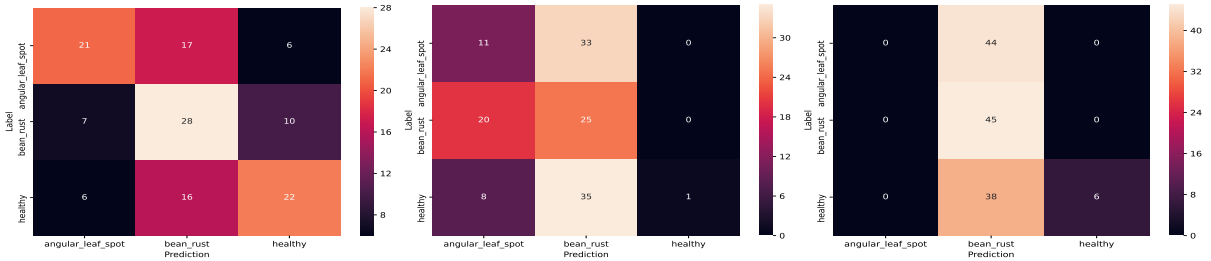
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

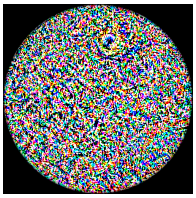


(c) Patch generated by VGG16 model

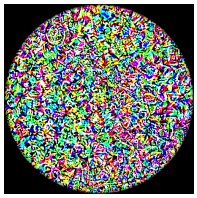


(d) Confusion matrix of MobileNetV2 (e) Confusion matrix of ResNet50 (f) Confusion matrix of VGG16

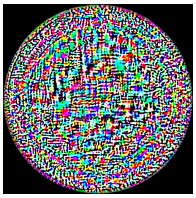
Figure 6.3: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bean rust class



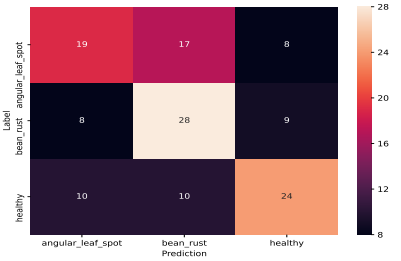
(a) Patch generated by MobileNetV2 model



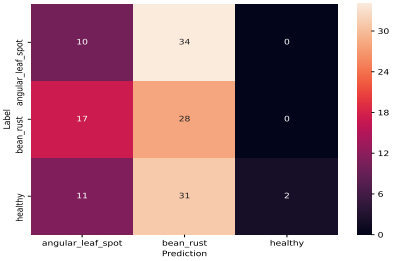
(b) Patch generated by ResNet50 model



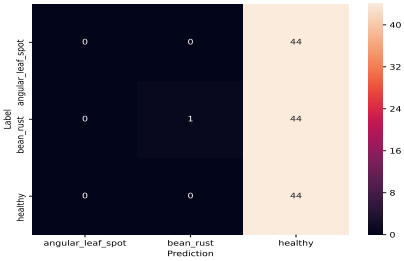
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2



(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.4: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting healthy class

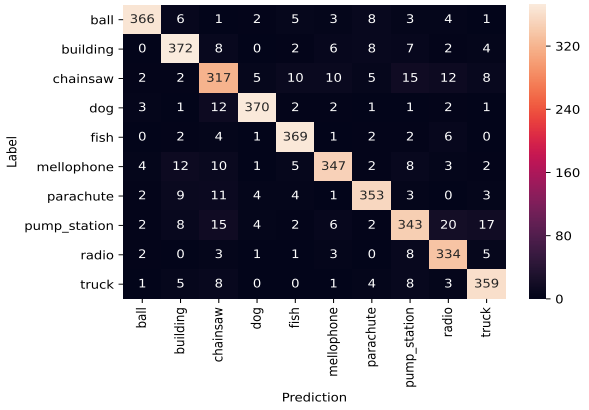
Imagenette Dataset

The confusion matrix of the networks trained on Imagenette dataset before the attack is shown in Figure 6.5. As depicted in the confusion matrix, images have been classified correctly before the attack. The patch generated to perform the adversarial attack targeting ball class w.r.t the considered models is illustrated in Figure 6.6. Then after the attack, firstly, from Figure 6.6(d) when targeting ball class, non-targeted adversarial patch attack happens but targeted adversarial attack does not happen, and even intensity of attack is less on MobileNetV2 model. But in the case of the ResNet50 model (refer Figure 6.6(e)), instead of targeting only the ball class, the attack is also targeting parachute class. That is, the adversarial patch attack is targeting two classes, ball and parachute. Similarly, even in the case of the VGG16 model (refer Figure 6.6(f)), the attack is targeting both ball class and parachute class with high intensity.

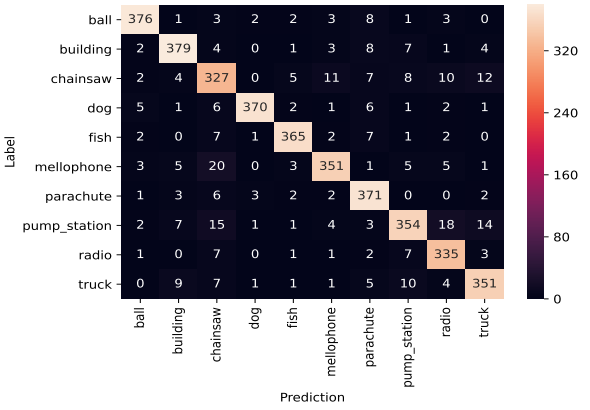
Then, the patch generated to perform an adversarial patch attack targeting the mellophone class is shown in Figure 6.7 with respect to each model. After the attack, in the case of the MobileNetV2 model, as depicted in Figure 6.7(d) non-targeted adversarial patch attacks occur, but the targeted attack does not happen. Similarly, in the case of the ResNet50 model (refer Figure 6.7(e)), even when trying to target mellophone class, targeted adversarial patch attack occurs, but the attack is targeting ball and parachute class instead of mellophone class. But as depicted in Figure 6.7(f), in the case of VGG16, the targeted attack is not happening on mellophone class, but the targeted attack has occurred on parachute class.

Then, the patch generated to perform adversarial patch attack targeting parachute class is shown in

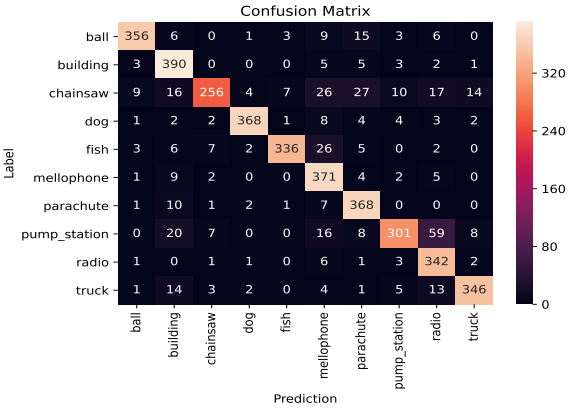
the Figure 6.8 with respect to each model. After the attack as shown, targeted adversarial attack does not occur, but non-adversarial attack occurs in MobileNetV2 (refer Figure 6.8(d)) case, but in ResNet50 case targeted adversarial attack occurs on parachute class as well as ball class instead of just parachute class as depicted in Figure 6.8(e) and the intensity of a targeted attack on ball class is more when compared to the intensity of targeted attack on parachute class. In the case of VGG16 (refer Figure 6.8(f)), targeted attack on parachute class is successful, and even the percentage of attack is high.



(a) Confusion matrix of MobileNetV2

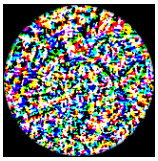


(b) Confusion matrix of ResNet50

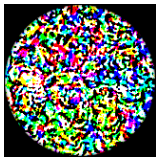


(c) Confusion matrix of VGG16

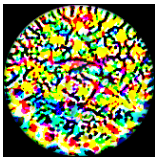
Figure 6.5: Confusion matrix of all the models before adversarial patch attack



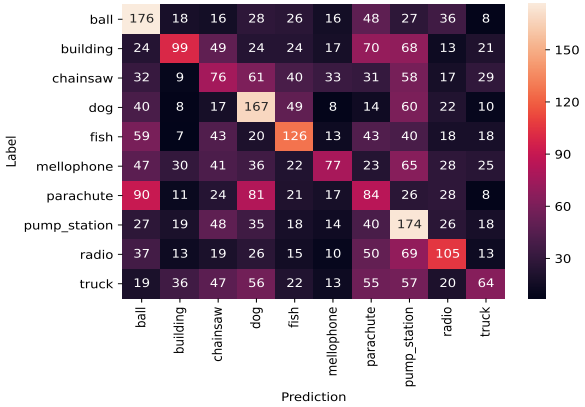
(a) Patch generated by MobileNetV2 model



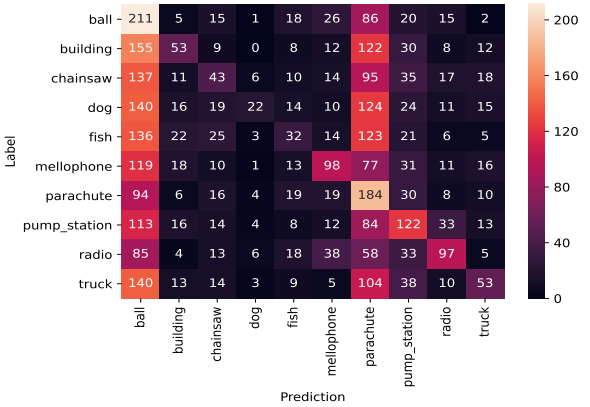
(b) Patch generated by ResNet50 model



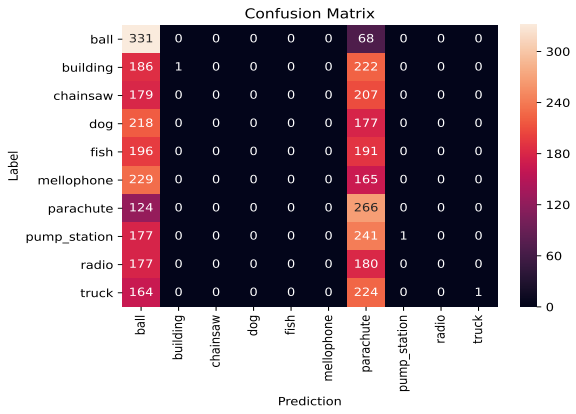
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

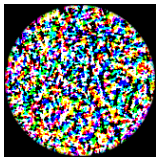


(e) Confusion matrix of ResNet50

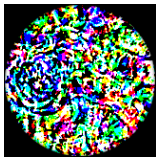


(f) Confusion matrix of VGG16

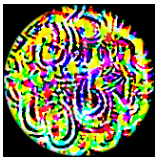
Figure 6.6: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting ball class



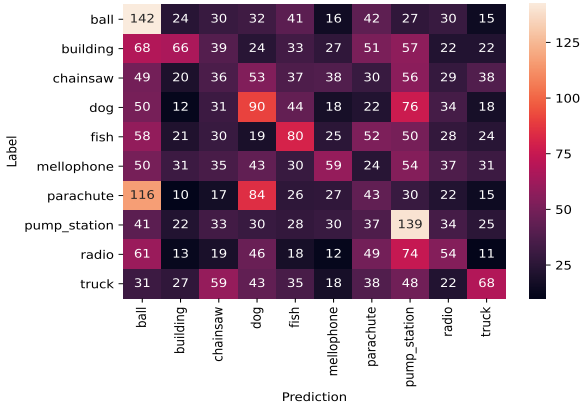
(a) Patch generated by MobileNetV2 model



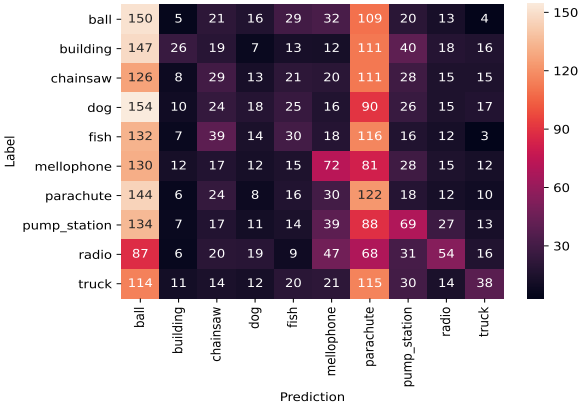
(b) Patch generated by ResNet50 model



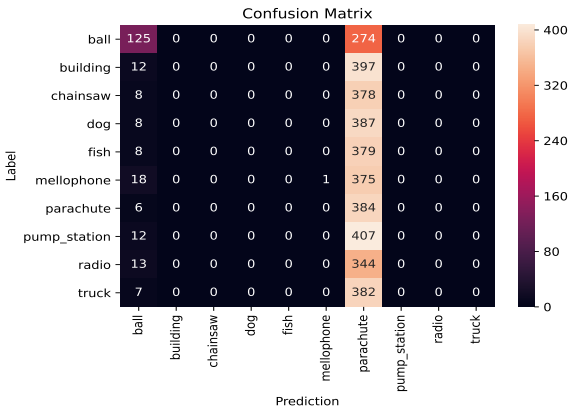
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

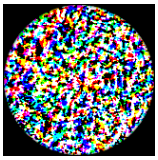


(e) Confusion matrix of ResNet50

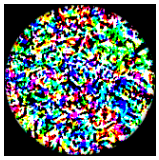


(f) Confusion matrix of VGG16

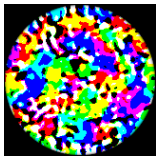
Figure 6.7: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting mellophone class



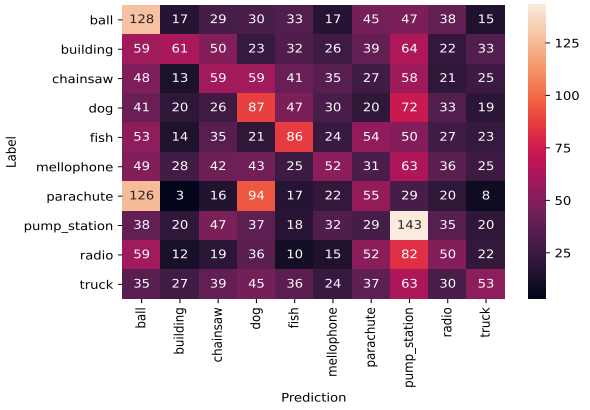
(a) Patch generated by MobileNetV2 model



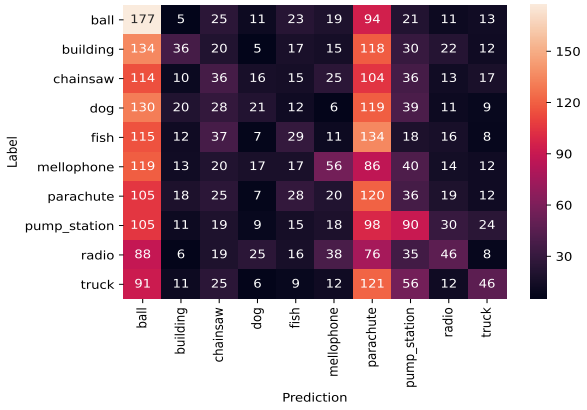
(b) Patch generated by ResNet50 model



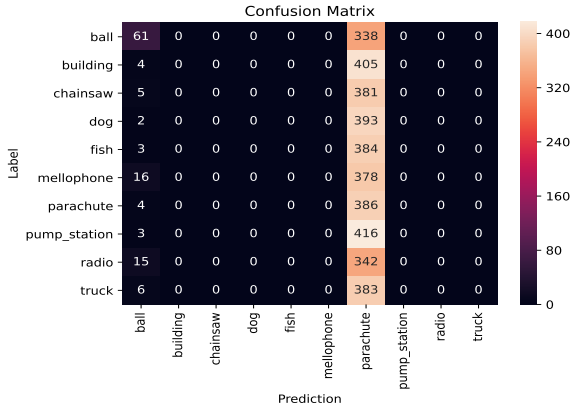
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2



(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.8: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting parachute class

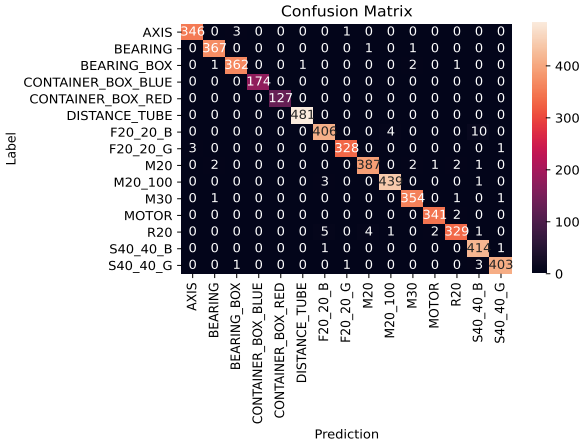
RoboCup@Work Dataset

The confusion matrix of the networks trained on the RoboCup@Work dataset before the attack is shown in Figure 6.9. As shown in the confusion matrix, images have been classified properly before the attack. The patch generated to perform the adversarial attack targeting bearing box class w.r.t each model as depicted

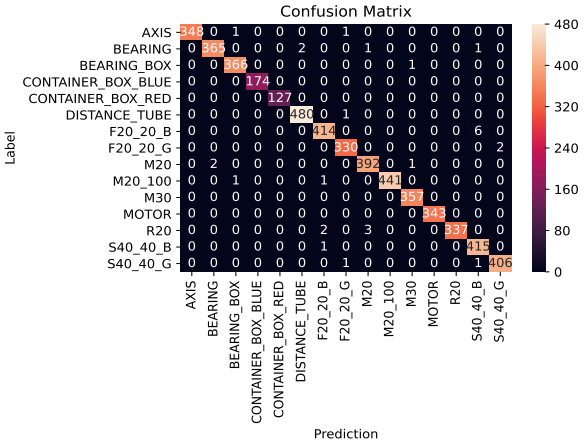
in Figure 6.10. Then after the attack, firstly, from Figure 6.10(d) and 6.10(e), that when targeting bearing box class, non-targeted adversarial patch attack happens but targeted adversarial attack does not happen and even accuracy of attack is less on MobileNetV2 model and ResNet50 model. But in the case of the VGG16 model (refer Figure 6.10(f)), the attack is targeting distance tube class with high intensity and bearing class with low intensity, but the targeted attack did not occur on the intended bearing box class.

Then, the patch generated to perform adversarial patch attack targeting distance tube class is shown in Figure 6.11 with respect to each model. After attack, in case of MobileNetV2 model and ResNet50 model as depicted in Figure 6.11(d) and 6.11(e), non-targeted adversarial patch attacks occur but the targeted attack does not occur. But in the case of the VGG16 model (refer Figure 6.11(f)), the attack is targeting distance tube class with high intensity; here attack is successful in targeting the intended distance tube class.

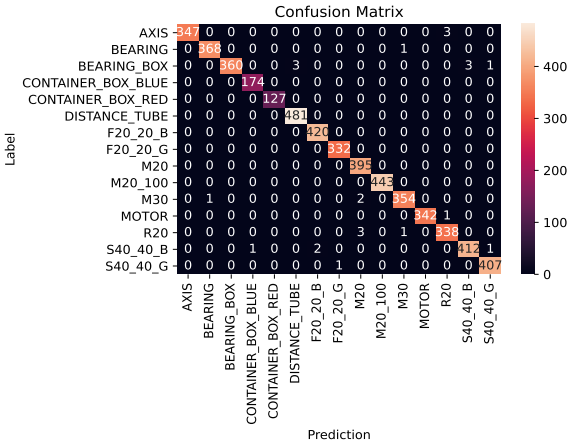
Then, patch generated to perform adversarial patch attack targeting motor class is shown in the Figure 6.12 with respect to each model. After the attack as shown, targeted adversarial attack does not occur but non-adversarial attack occurs in MobileNetV2 (refer Figure 6.12(d)) case and ResNet50 (refer Figure 6.12(e)) case. But in the case of VGG16 model (refer Figure 6.12(f)), targeted attack occurs but not on intended motor class but on distance tube class.



(a) Confusion matrix of MobileNetV2

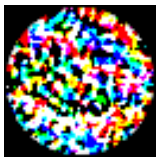


(b) Confusion matrix of ResNet50

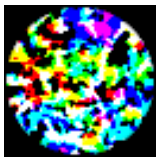


(c) Confusion matrix of VGG16

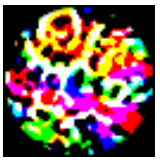
Figure 6.9: Confusion matrix of all the models trained on RoboCup@Work dataset before adversarial patch attack



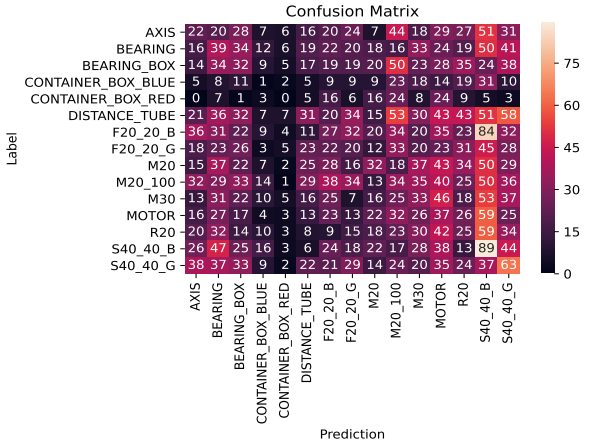
(a) Patch generated by MobileNetV2 model



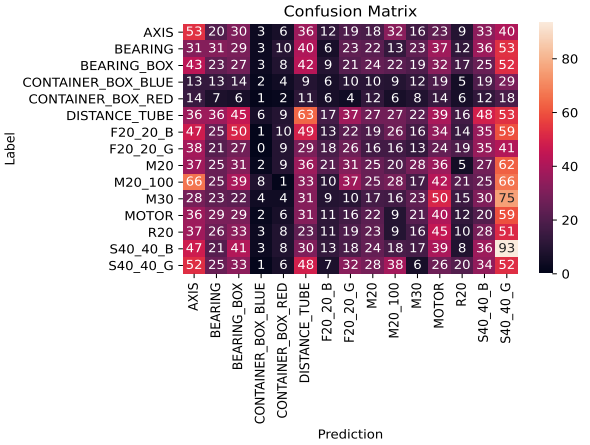
(b) Patch generated by ResNet50 model



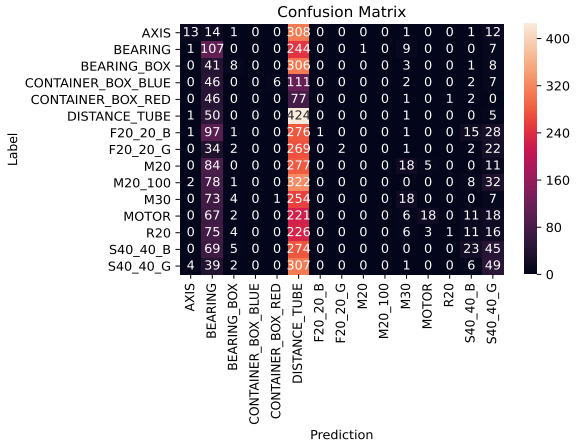
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

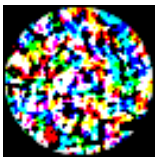


(e) Confusion matrix of ResNet50

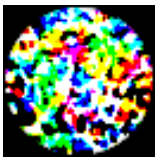


(f) Confusion matrix of VGG16

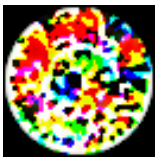
Figure 6.10: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bearing box class



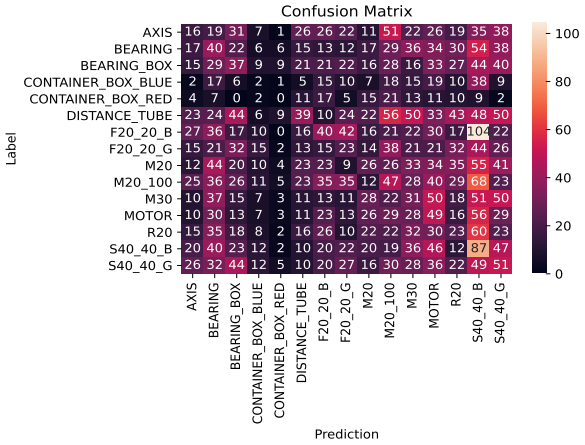
(a) Patch generated by MobileNetV2 model



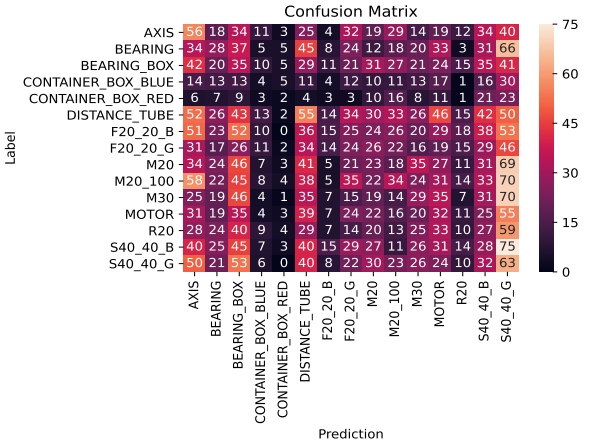
(b) Patch generated by ResNet50 model



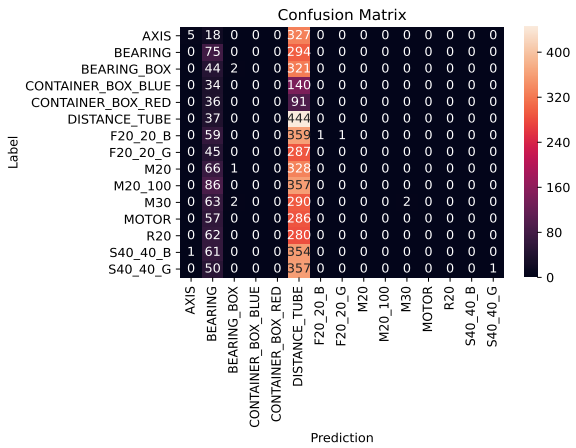
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

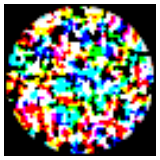


(e) Confusion matrix of ResNet50

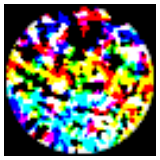


(f) Confusion matrix of VGG16

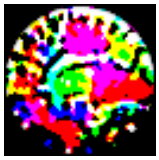
Figure 6.11: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting distance tube class



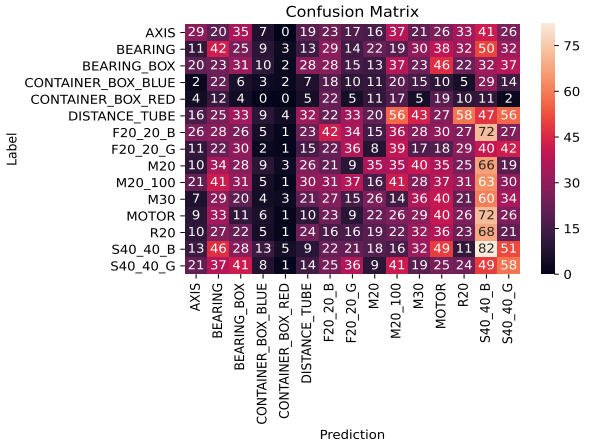
(a) Patch generated by MobileNetV2 model



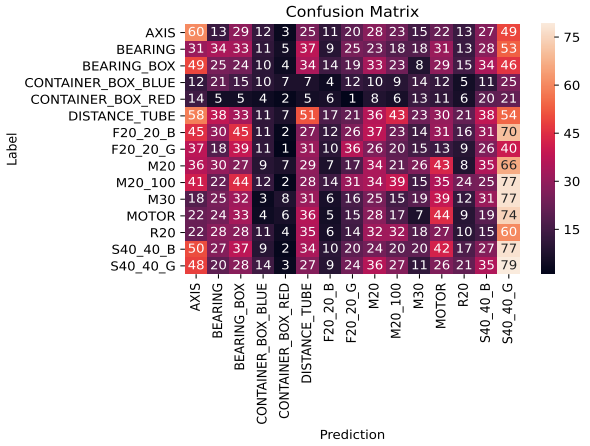
(b) Patch generated by ResNet50 model



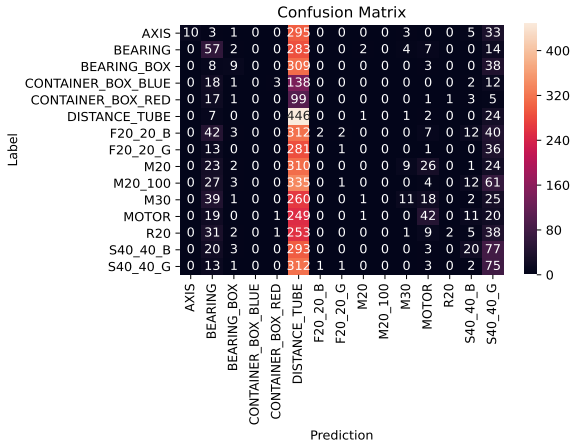
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2



(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.12: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting motor class

6.1.3 Verdict

For the considered datasets and DNN models, the results from the experiments don't adhere to the hypothesis considered in Section 6.1.1. Hence, the hypothesis considered is false and cannot be accepted. From this, it can be concluded that an adversarial patch attack cannot always perform the targeted attack.

6.2 Adversarial Training Defense against Adversarial Patch Attack

RQ3. Is adversarial training defense effective against adversarial patch attacks?

The main idea of adversarial training defense is to use adversarial examples for training the classifier. The intuition here is that by training a classifier with adversarial examples, the features of adversarial examples can be learned, which helps defend against new unseen adversarial examples. In prior works [27] [62] [63], adversarial training was implemented against digital-world attack. In this research work, adversarial training defense is used against a physical real-world attack, an adversarial patch attack. The objective of this experiment is to check whether adversarial training will be effective against an adversarial patch attack or not.

6.2.1 Hypothesis

Hypothesis: adversarial training defense is effective against an adversarial patch attack

To perform this experiment, a hypothesis is considered. The hypothesis states that an adversarial training defense mechanism can provide defense against an adversarial patch attack.

6.2.2 Preparation

To perform adversarial training defense, firstly, a set of adversarial examples has to be constructed. Then a classifier has to been trained with constructed adversarial examples to learn the features of adversarial examples.

For Beans dataset, an adversarial patch attack is performed on MobileNetV2 model to generate a patch. Then this patch is attached on a set of training images of Beans dataset to produce adversarial examples. The size of patch considered in this experimentation to create adversarial training data is 25%. Then this adversarial examples is combined with original training data to create new training dataset. Then by using this new dataset, MobileNetV2 model is trained. In similar fashion, for ResNet50 model and VGG16, a new dataset is created and is used to train the respective model. Similarly, for Imagenette dataset and RoboCup@Work dataset, new adversarial training datasets are created w.r.t MobileNetV2 model, ResNet50 model and VGG16 model and then the created datasets are used to train the respective model. Here, adversarial examples generated for one model is not used to train the other model. For instance, if the MobileNetV2 model generates an adversarial example, this example is not used to train the ResNet50 model and VGG16 model, and vice versa. Similarly, adversarial examples of one dataset is not mixed with other datasets. The samples of adversarial examples constructed from MobileNetV2 model using Beans dataset, Imagenette dataset and RoboCup@Work dataset is depicted in Figure 6.13

(a) Adversarial example of Beans dataset

(b) Adversarial example of Imagenette dataset

(c) Adversarial example of RoboCup@Work dataset

Figure 6.13: Samples of adversarial examples generated by MobileNetV2 model with patch size 25%, used for adversarial training defense

6.2.3 Observation

In this experimentation, after training MobileNetV2, ResNet50 and VGG16 model with respective constructed adversarial datasets (Beans adversarial dataset, Imagenette adversarial dataset and RoboCup@Work adversarial dataset), an adversarial patch attack is performed on all three models with respect to the unseen testing set of Beans dataset, Imagenette dataset and RoboCup@Work dataset considering a patch size of 20%, 25%, 35% and 40%.

Beans Dataset

The confusion matrix of the adversarial trained MobileNetV2 model and ResNet50 model on the Beans dataset before the attack is shown in Figure 6.14. As illustrated in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting angular leaf spot class w.r.t each model is depicted in Figure 6.15. Then after the attack, firstly, Figure 6.15(c) depicts that when targeting angular leaf spot class, non-targeted adversarial patch attack happens even after adversarially training the MobileNetV2 model. In the ResNet50 model's case (refer Figure 6.15(d)), even though performing adversarial training, targeted patch attack occurs, and bean rust class is targeted with high intensity.

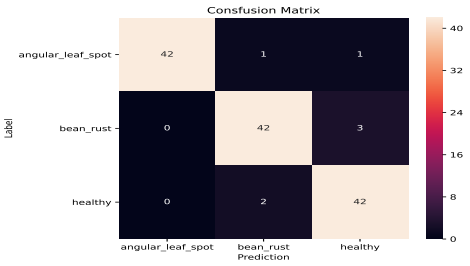
Secondly, the patch generated to perform adversarial patch attack targeting bean rust class is shown in the Figure 6.16 w.r.t each model. After the attack, in case of MobileNetV2 model (refer Figure 6.16(c)), adversarial attack occurs with less intensity. In the case of the ResNet50 model, the targeted attack is successful with high intensity on the bean rust model, as depicted in Figure 6.16(d).

Finally, the patch generated to perform adversarial patch attack targeting healthy class is shown in the Figure 6.17 w.r.t each model. When attack is performed, untargeted adversarial attack takes place in the case of MobileNetV2 model (refer Figure 6.17(c)). Similarly, in the case of the ResNet50 model (refer Figure 6.17(d)), targeted attack occurs on bean rust class instead of healthy class.

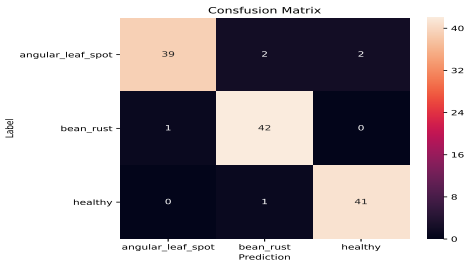
Figure 6.18 provides confidence levels of final predictions of the considered methods before the attack and after the attack with different patch sizes considered. As depicted in Figure 6.18, the confidence

of the adversarially trained models is high before the attack. Once adversarial attacks occur targeting the angular leaf spot class, the confidence of the models decreases. Similar confidence level results are obtained even when the adversarial attack is performed, targeting other classes.

From the observations of results, inference can be made that efficiency of adversarial training is too low when adversarial patch attack was performed on the two models using Beans dataset.

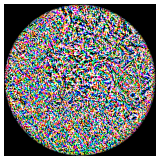


(a) Confusion matrix of MobileNetV2

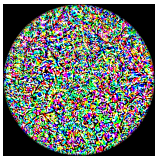


(b) Confusion matrix of ResNet50

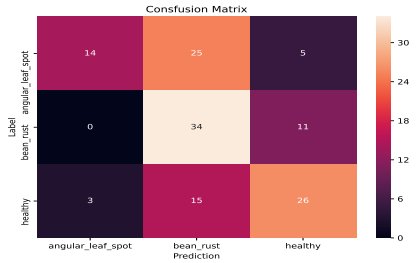
Figure 6.14: Confusion matrix of adversarially trained MobileNetV2 model and ResNet50 model on Beans dataset before adversarial patch attack



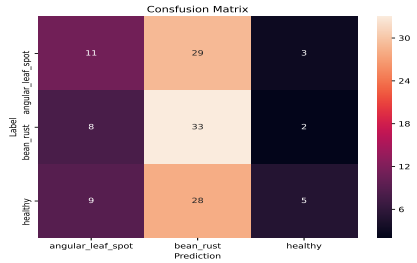
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

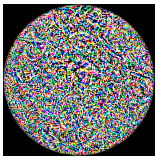


(c) Confusion matrix of MobileNetV2

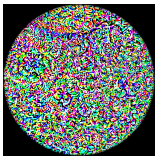


(d) Confusion matrix of ResNet50

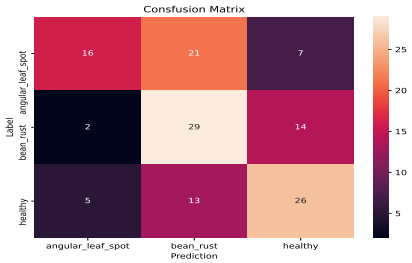
Figure 6.15: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting angular leaf spot class



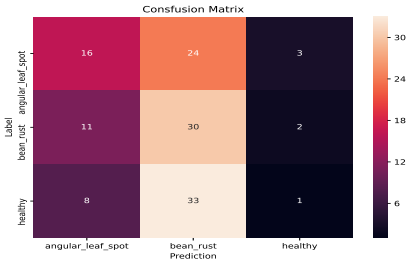
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

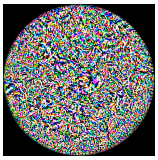


(c) Confusion matrix of MobileNetV2

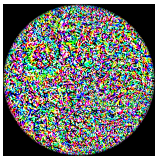


(d) Confusion matrix of ResNet50

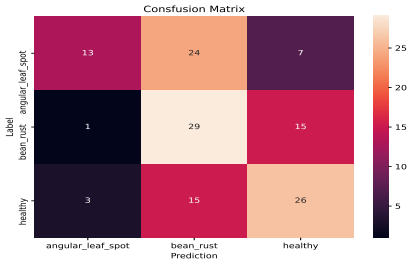
Figure 6.16: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bean rust class



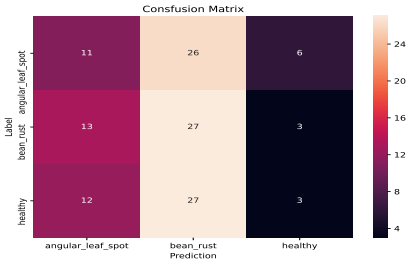
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

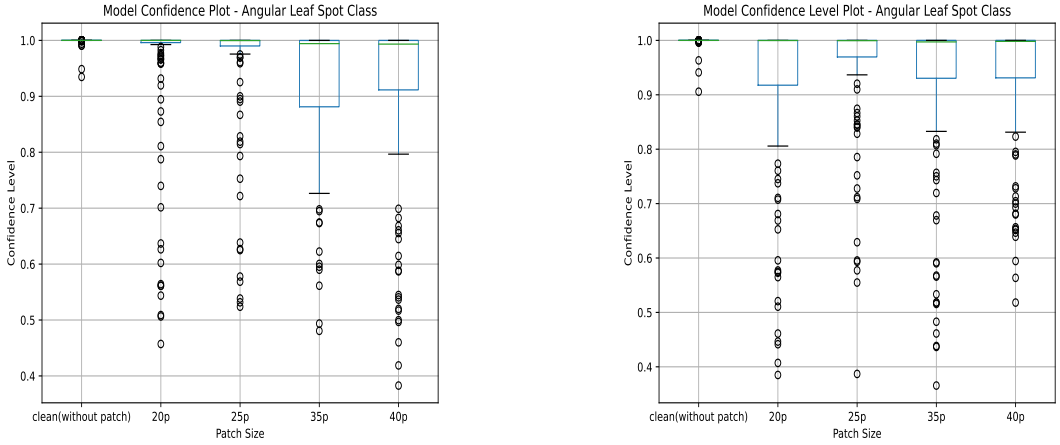


(c) Confusion matrix of MobileNetV2



(d) Confusion matrix of ResNet50

Figure 6.17: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting healthy class



(a) MobileNetV2 model

(b) ResNet50 model

Figure 6.18: Confidence of adversarial trained models before and after adversarial patch attack targeting angular leaf spot class

Imagenette Dataset

The confusion matrix of adversarially trained models on the Imagenette dataset before the attack is depicted in Figure 6.19. As depicted in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting ball class w.r.t each model is depicted in Figure 6.20. Then after the attack, firstly, Figure 6.20(d) depicts that when targeting ball class, a non-targeted adversarial patch attack occurs even though the MobileNetV2 model is adversarially trained. Similarly in case of ResNet50 model as depicted in Figure 6.20(e), non-targeted adversarial attack happens. But in the case of the VGG16 model (refer Figure 6.20(f)), the attack is successfully targeting ball class as intended, and there is zero defense provided by adversarial training.

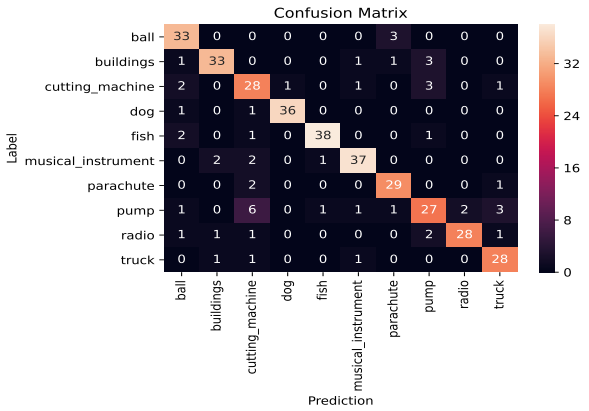
Secondly, the patch generated to perform adversarial patch attack targeting parachute class is shown in the Figure 6.21 w.r.t each model. After the attack, both in case of MobileNetV2 model (refer Figure 6.21(d)),and ResNet50 model (refer Figure 6.21(e)), non-targeted adversarial attack is successful inspite of performing of adversarial training defense. But in the case of the VGG16 model (refer Figure 6.21(f)), the attack is targeting ball class with high intensity and intended parachute class with low intensity, which shows adversarial training is completely unsuccessful.

Then, the patch generated to perform adversarial patch attack targeting radio class is shown in the Figure 6.22 w.r.t each model. Then when adversarial patch attack targets radio class, non-targeted adversarial attack takes place in the case of both MobileNetV2 model (refer Figure 6.22(d)) and ResNet50 model (refer Figure 6.22(e)). But in the case of the VGG16 model (refer Figure 6.22(f)), targeted attack occurs but not on intended radio class but on ball class, and here also adversarial training fails to provide any defense.

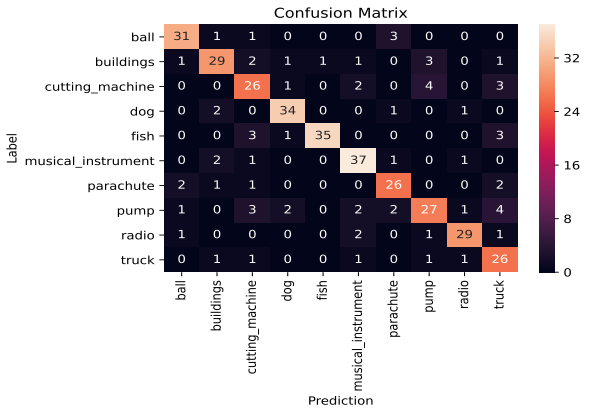
Figure 6.23 provides confidence levels of final predictions of DNNs before the attack and after the

attack with different patch sizes considered. As depicted in Figure 6.23, the confidence level of the adversarially trained models is high before adversarial patch attack in the case of the MobileNetV2 model and ResNet50 model. Once an adversarial attack happens to target the radio class, the confidence level of the models decreases drastically. But in the case of the VGG16 model, the confidence level increases after an adversarial patch attack. A similar result in confidence level are depicted when an adversarial attack is made targeting other classes.

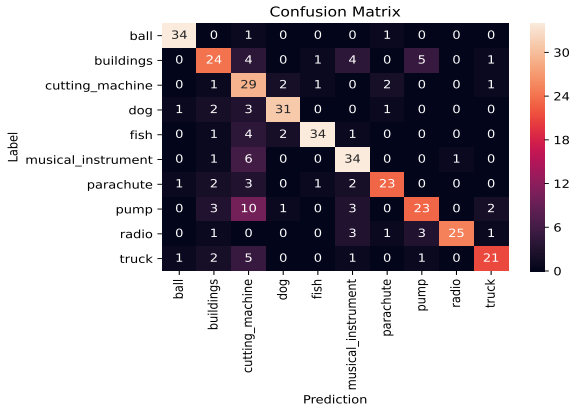
From the above observations, in the case of the Imagenette dataset, adversarial training defense cannot stop adversarial patch attacks on the considered DNNs.



(a) Confusion matrix of MobileNetV2

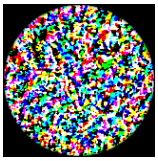


(b) Confusion matrix of ResNet50

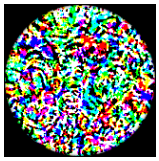


(c) Confusion matrix of VGG16

Figure 6.19: Confusion matrix of adversarially trained models on Imagenette dataset before adversarial patch attack

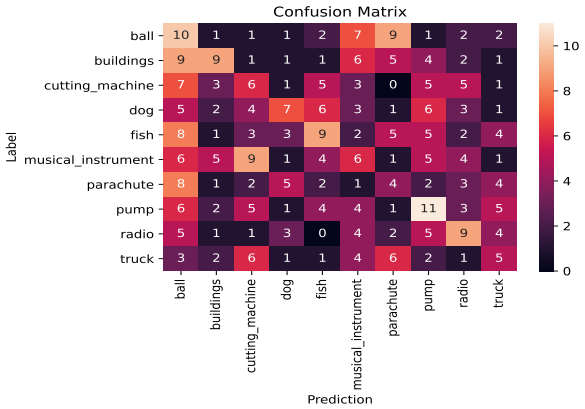


(a) Patch generated by MobileNetV2 model

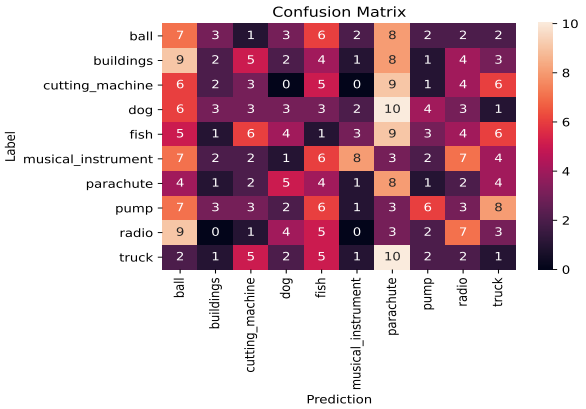


(b) Patch generated by ResNet50 model

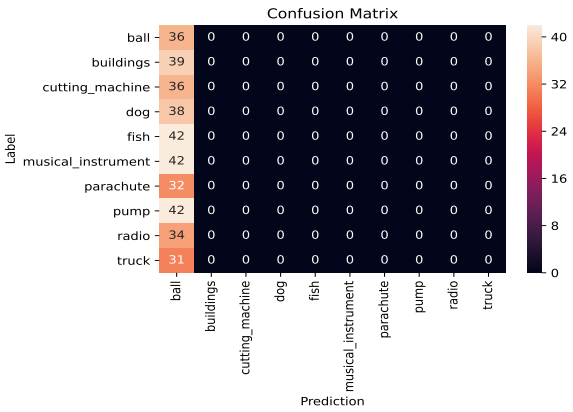
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

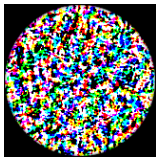


(e) Confusion matrix of ResNet50

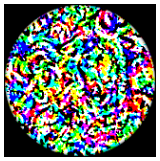


(f) Confusion matrix of VGG16

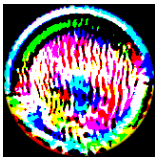
Figure 6.20: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting ball class



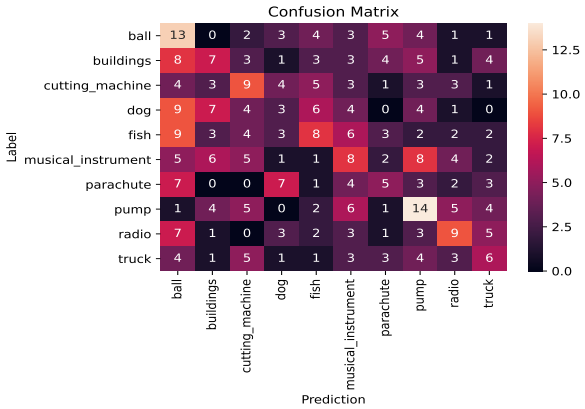
(a) Patch generated by MobileNetV2 model



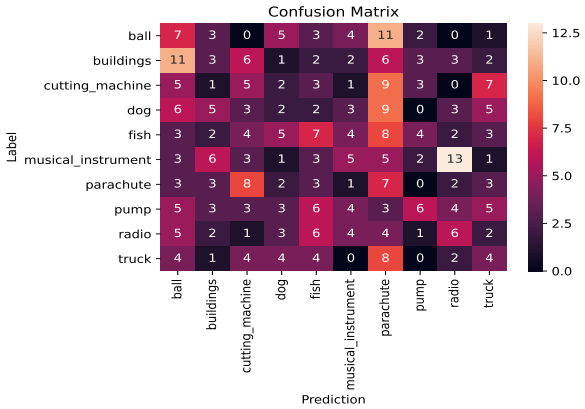
(b) Patch generated by ResNet50 model



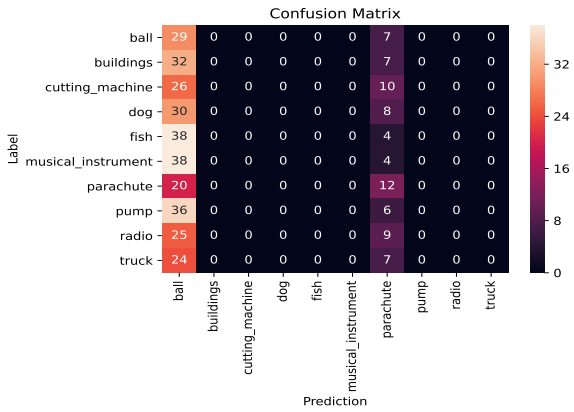
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

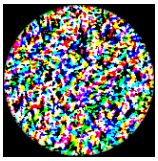


(e) Confusion matrix of ResNet50

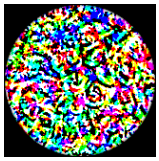


(f) Confusion matrix of VGG16

Figure 6.21: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting parachute class

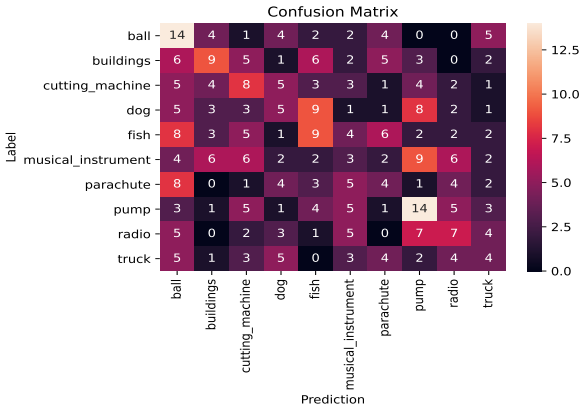


(a) Patch generated by MobileNetV2 model

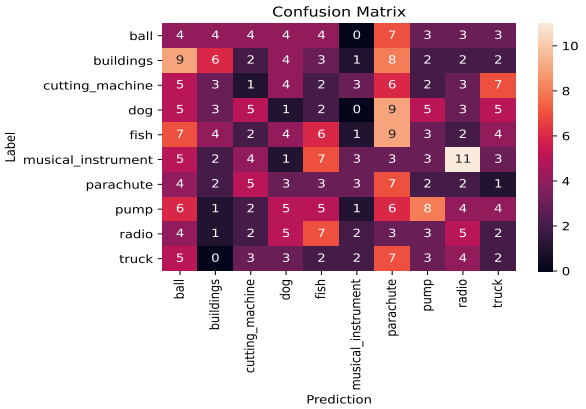


(b) Patch generated by ResNet50 model

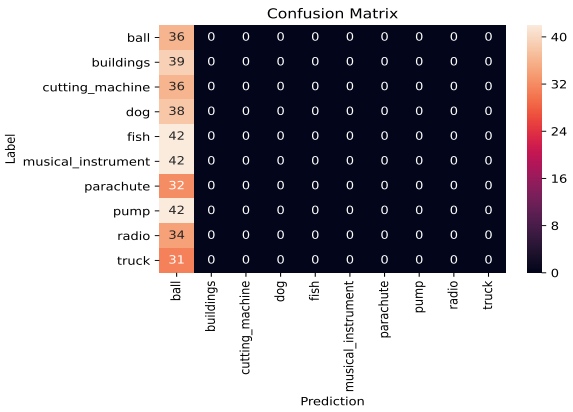
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

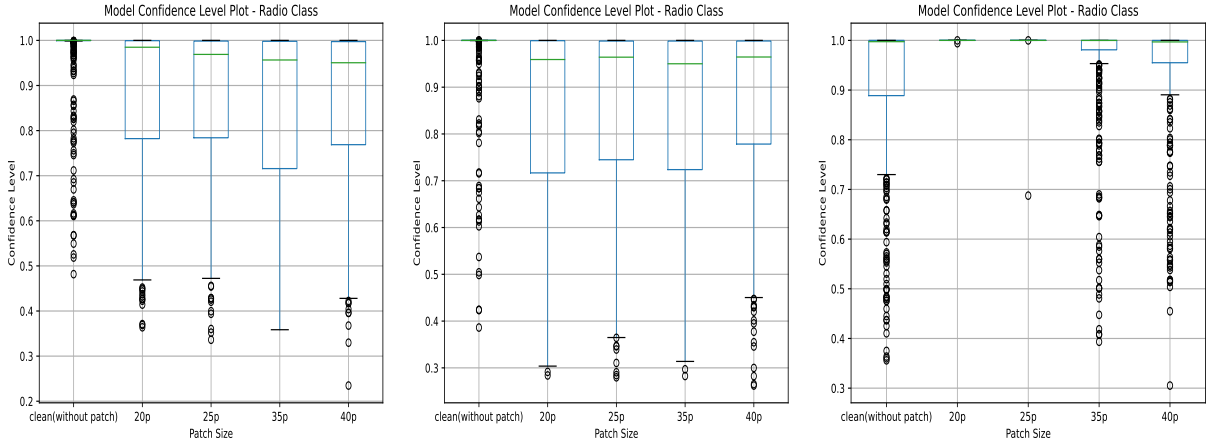


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.22: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting radio class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 6.23: Confidence of adversarial trained models after adversarial patch attack targeting radio class

RoboCup@Work Dataset

The confusion matrix of adversarially trained models on the RoboCup@Work dataset before the attack is depicted in Figure 6.24. As shown in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting distance tube class w.r.t each model is depicted in Figure 6.25. Then after the attack, firstly, Figure 6.25(d) depicts that when targeting distance tube class, non-targeted adversarial patch attack happens and as can be observed from the confusion matrix, the intensity of attack is too low on the classes bearing, bearing box, contained box blue, container box red and distance tube. Similarly, in the case of the ResNet50 model, as depicted in Figure 6.25(e), a non-targeted adversarial attack happens even after adversarial training the model. But in the case of the VGG16 model (refer Figure 6.25(f)), targeted attack occurs targeting M20 and motor class instead of intended distance tube class.

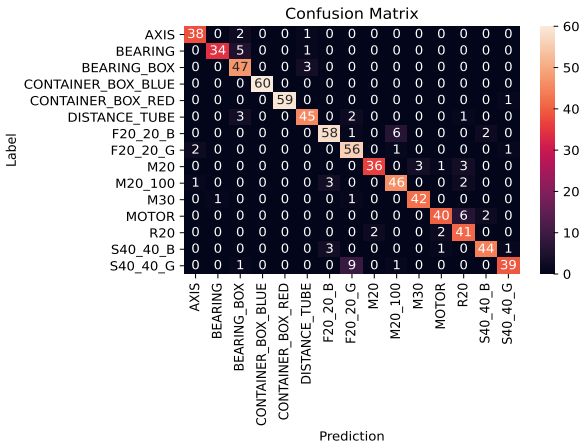
Secondly, the patch generated to perform adversarial patch attack targeting F20_20_B class is shown in the Figure 6.26 w.r.t each model. Then when adversarial patch attack targets F20_20_B class, both in case of MobileNetV2 model (refer Figure 6.26(d)),and ResNet50 model (refer Figure 6.26(e)), non-targeted adversarial attack is successful even in the presence of adversarial training defense. But in the case of VGG16 model (refer Figure 6.26(f)), patch attack is targeting M20 class instead of F20_20_B class.

Then, the patch generated to perform adversarial patch attack targeting motor class is shown in the Figure 6.27 w.r.t each model. Then when adversarial patch attack targets motor class, non-targeted adversarial attack takes place in the case of both MobileNetV2 model (refer Figure 6.27(d)) and ResNet50 model (refer Figure 6.27(e)). But in the case of VGG16 model (refer Figure 6.27(f)), targeted attack occurs on both M20 and motor class. But as depicted in the confusion matrix, the intensity of attack is more in M20 class when compared to the motor class.

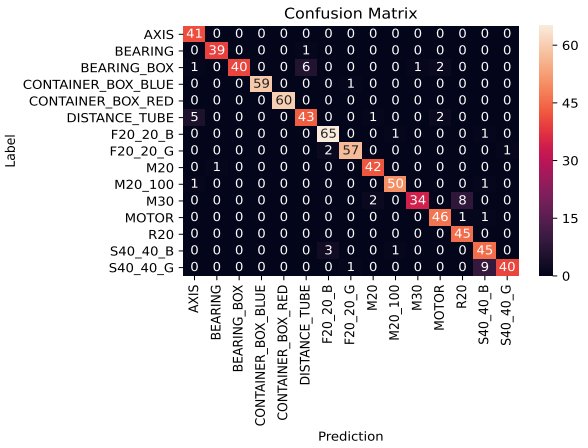
Figure 6.28 provides confidence levels of final predictions of DNNs before the attack and after the attack

with different patch sizes considered. As depicted in Figure 6.28, the confidence level of the MobileNetV2 and ResNet50 model decreases drastically once an adversarial patch attack occurs targeting the F20_20_B class. Once an adversarial attack happens on the VGG16 model when patch size is 20% and 25%, the confidence level decreases, but when patch size is 35% and 40%, the confidence level remains the same. Similar results in confidence level are depicted when an adversarial attack is made targeting other classes.

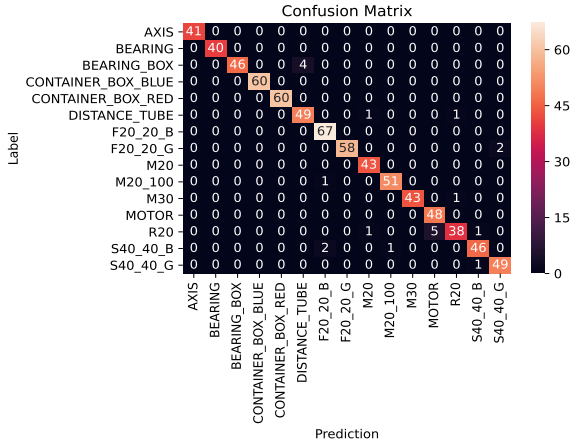
From the above observations, on the RoboCup@Work dataset, adversarial patch attack occurs even after performing an adversarial training defense mechanism.



(a) Confusion matrix of MobileNetV2

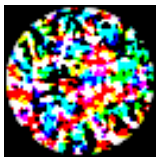


(b) Confusion matrix of ResNet50

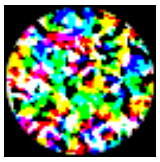


(c) Confusion matrix of VGG16

Figure 6.24: Confusion matrix of adversarial trained models on RoboCup@Work dataset before adversarial patch attack

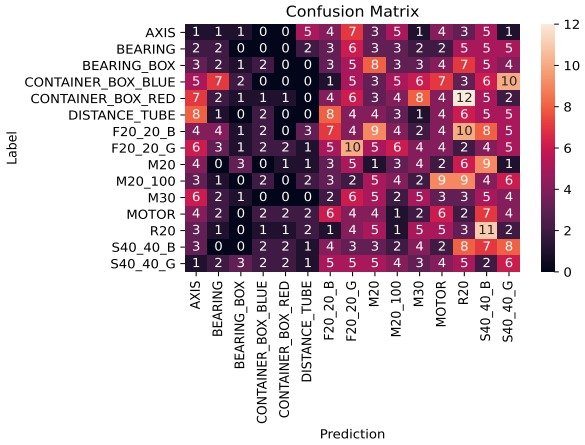


(a) Patch generated by MobileNetV2 model

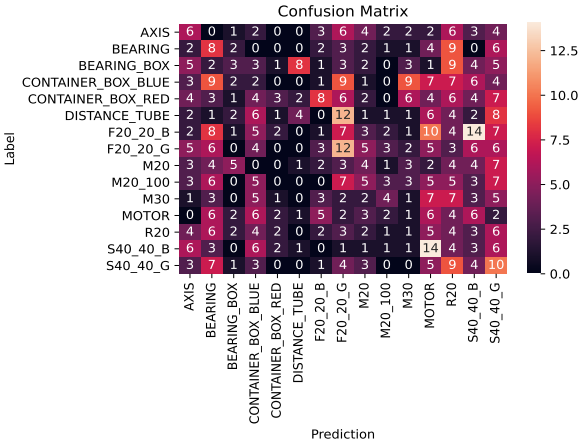


(b) Patch generated by ResNet50 model

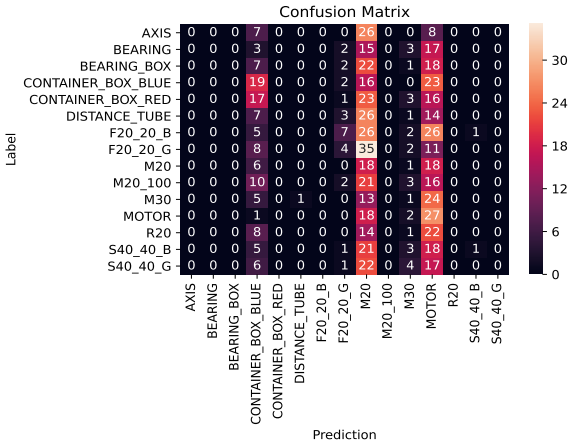
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

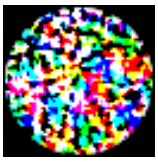


(e) Confusion matrix of ResNet50

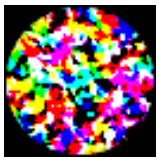


(f) Confusion matrix of VGG16

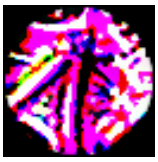
Figure 6.25: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting distance tube class



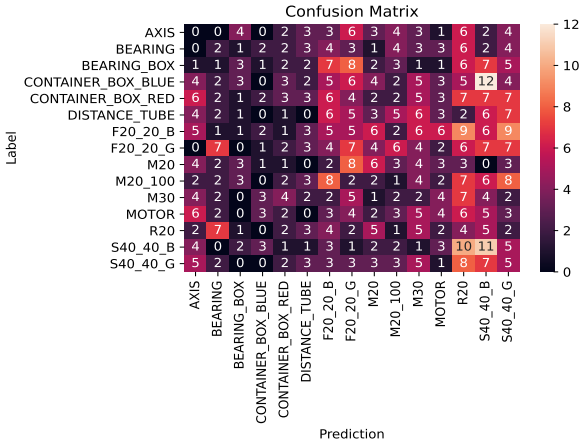
(a) Patch generated by MobileNetV2 model



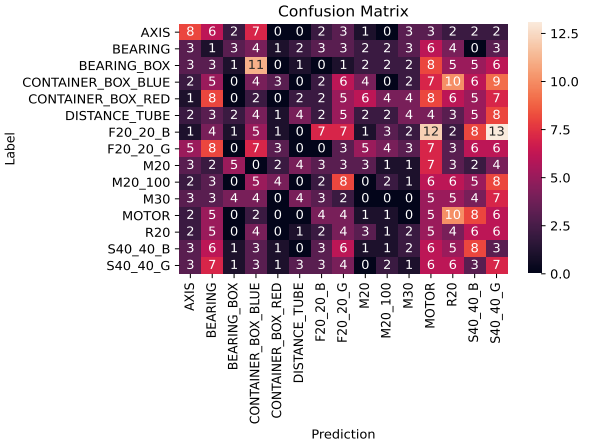
(b) Patch generated by ResNet50 model



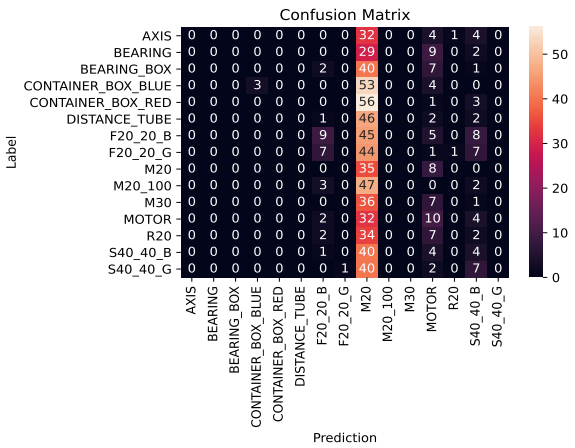
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

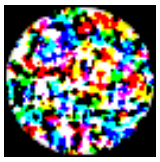


(e) Confusion matrix of ResNet50

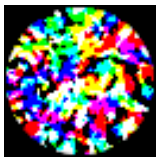


(f) Confusion matrix of VGG16

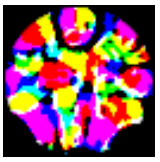
Figure 6.26: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting F20_20_B class



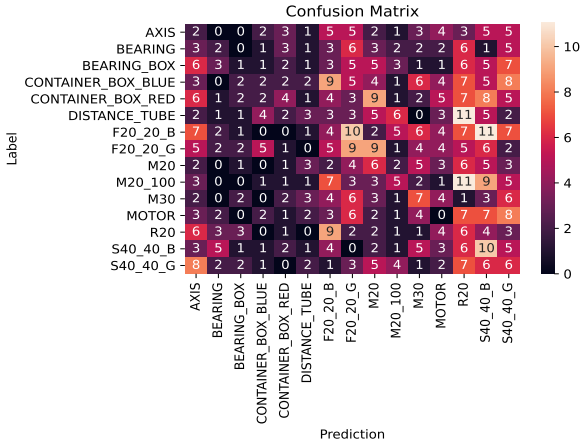
(a) Patch generated by MobileNetV2 model



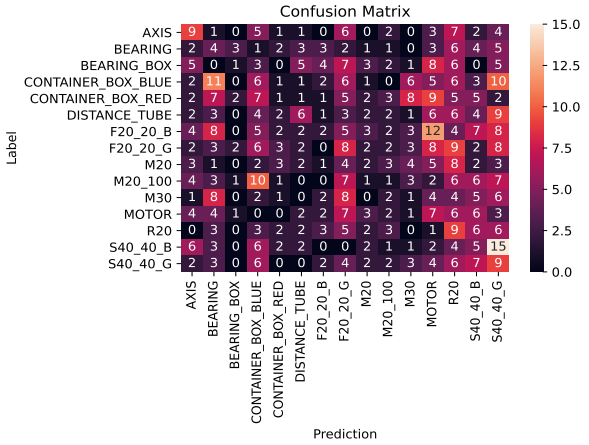
(b) Patch generated by ResNet50 model



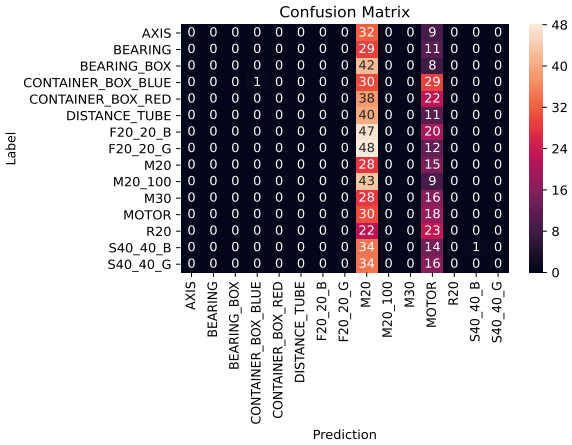
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

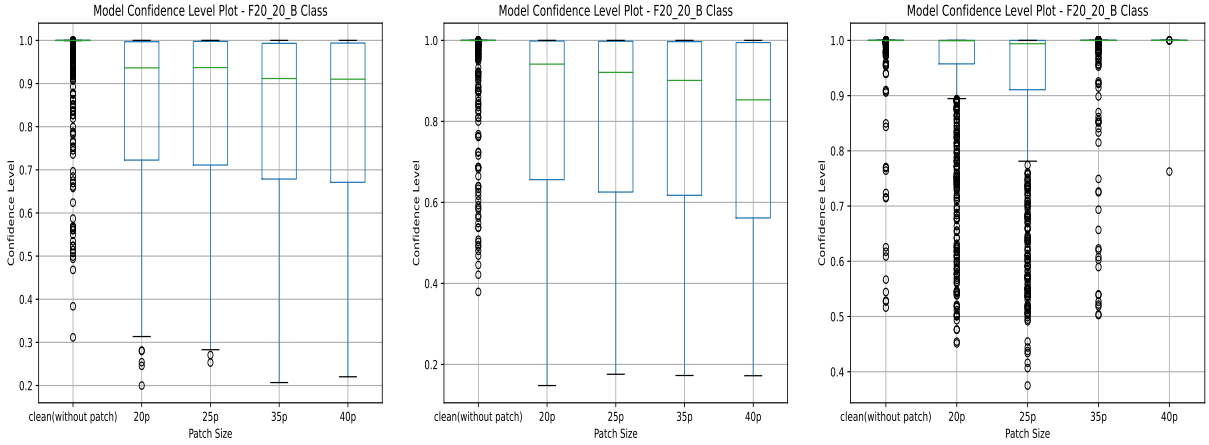


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.27: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting motor class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 6.28: Confidence of adversarial trained models after adversarial patch attack targeting F20_20_B class

6.2.4 Verdict

Based on the considered datasets and models, the results of the experiments do not adhere to the hypothesis considered in Section 6.2.1. Hence, hypothesis considered is false and cannot be accepted. The conclusion of the experiment is the adversarial training defense mechanism is ineffective against an adversarial patch attack.

6.3 Abstention Class Defense against Adversarial Patch Attack

RQ4. Could the defense against adversarial patch attacks be improved by adding an abstention class to a dataset?

The main idea behind abstention class defense is to add an extra class to the training process of the classifier. The intuition here is that by adding an extra abstention class to the training dataset in the training process, a classifier can learn more resilient features of the dataset, which helps differentiate between normal examples and adversarial examples that in turn helps to defend against new unseen adversarial examples during an adversarial attack. During this process, an abstention class is also called a hot class. The usage of extra class to identify outliers of an adversarial patch attack is an idea obtained from the papers [40] [61]. Both [40] and [61] added an extra abstention data to the training data, to provide defense against Out-Of-Distribution (OOD) attack. Similarly, the paper [35] came up with background class defense against digital-world adversarial attacks. In this experimentation, an abstention class will be added to the training dataset during the training process of the considered models to defend against a real-world physical attack that is an adversarial patch attack. The objective of this experiment is to check whether adding an extra abstention class to the training dataset can provide defense against adversarial patch attacks or not.

6.3.1 Hypothesis

Hypothesis: adding abstention class to the training dataset will provide defense against an adversarial patch attack

To perform this experiment, a hypothesis is considered. The hypothesis states it is possible to improve the defence against an adversarial patch attack by adding an extra abstention class to the training dataset.

6.3.2 Preparation

First, to perform abstention class defense, an abstention class has to be added to the training dataset. Then a classifier has to be trained with a training dataset containing abstention class to learn the resilient features found in adversarial examples. The very important work here is to identify the elements that belong to the abstention class. That is, what elements does an abstention class should contain? In this research work, a few experiments were done to identify the elements that may belong to the abstention class. Here, for each considered dataset, an abstention class was constructed and was added.

Experiment 1: adding random images to the abstention class

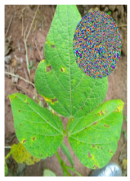
In this experiment, similar to [35], random images were added to the abstention class. The intuition here is that by adding less number of random images, a classifier can be trained to identify unusual behaviour present in adversarial examples. Some samples of random images which are added to the abstention class (hot class) are depicted in Figure 6.29. All three DNN models were trained with the dataset containing abstention class. Then when an adversarial patch attack was performed, all the trained models failed to provide any defense against adversarial patch attack. It showed that adding random images to the abstention class does not improve the defense against adversarial patch attacks.

Figure 6.29: Random image samples added to abstention class. Image source : first picture is taken from [64], second picture from [66] and third picture from [65]

Experiment 2: adding adversarial images to the abstention class

In this experiment, adversarial examples were constructed and were added to the abstention class. The intuition here is that by adding adversarial examples to the abstention class of the training dataset, a classifier can learn resilient features of adversarial patterns present in adversarial examples that in

turn may help differentiate between normal examples and adversarial examples during the testing phase. Some of the adversarial images added to the abstention class are depicted in Figure 6.30. Then all three considered DNN models were trained on the dataset containing abstention class. Then when an adversarial patch attack was executed, all DNN models failed to differentiate between normal examples and adversarial examples. Hence, adding adversarial images to the abstention class did not provide any defense against adversarial patch attacks.



(a) Adversarial example of Beans dataset added to abstention class

(b) Adversarial example of Imagenette dataset added to abstention class

(c) Adversarial example of RoboCup@Work dataset added to abstention class

Figure 6.30: Adversarial examples of respective datasets added to their respective abstention class

Experiment 3: adding subset of all classes present in dataset, to the abstention class

In this experiment, a subset of all classes present in the dataset was collected and was added to the abstention class during the training phase of the classifier. That is, here, the abstention class contains the subset images of all the other classes present in that particular dataset. The intuition is here is that by adding subset images of all classes to the abstention class, resilient features of all the classes can be obtained by a single abstention class. In this experiment, a subset of all classes present in the dataset is collected and added to a single abstention class. Then considered models were trained on the training dataset with abstention class. Later adversarial patch attack was performed on these trained models to check whether adding abstention class (subset images of other classes) during the training process provides any defense. The observation results of this experiment are shown in the following section.

6.3.3 Observation

In this experimentation, after training MobileNetV2, ResNet50 and VGG16 model with respective datasets (Beans dataset, Imagenette dataset and RoboCup@Work dataset) containing respective abstention class, an adversarial patch attack is performed on all three models w.r.t the unseen testing set of Beans dataset, Imagenette dataset and RoboCup@Work dataset considering a patch size of 20%, 25%, 35% and 40%. Here the abstention class added to the respective datasets were created based on the process mentioned in 6.3.2 i.e., adding a subset of all classes present in dataset to the abstention class.

Here, the defense mechanism is successful if the adversarial images are correctly classified to their respective original classes or if adversarial images are identified as adversaries and classified as hot class

(abstention class).

Beans Dataset

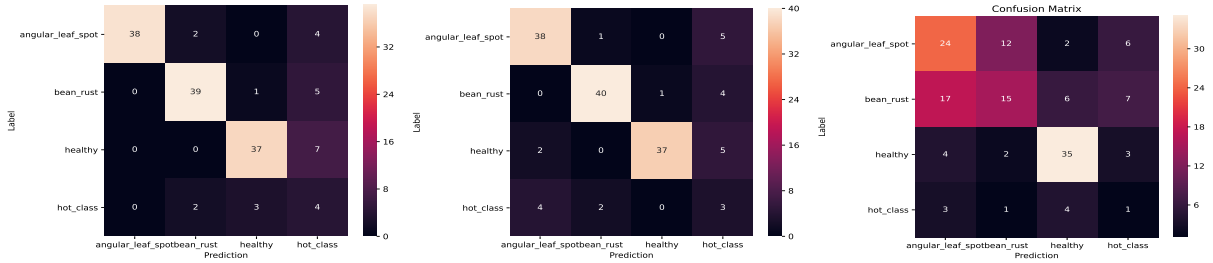
The confusion matrix of the all models trained on the Beans dataset before the attack is shown in Figure 6.31. As shown in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting angular leaf spot class w.r.t each model is shown in Figure 6.32. Then after the attack, firstly, in the case of the MobileNetV2 model, Figure 6.32(d) depicts that when targeting angular leaf spot class, a large amount of images are classified as hot class (abstention class). In ResNet50 model's case (refer Figure 6.32(e)), all adversarial images are classified as hot class. But in the case of the VGG16 model (refer Figure 6.32(c)), the targeted attack happens on the healthy class instead of the intended angular leaf spot class.

Secondly, the patch generated to perform adversarial patch attack targeting bean rust class is shown in the Figure 6.33 w.r.t each model. Then when adversarial patch attack targets bean rust class, in case of MobileNetV2 model (refer Figure 6.33(d)), as depicted, a high amount of images are classified as a hot class, and even intensity of attack is also less. In the case of the ResNet50 model, most of the images are classified as a hot class, as depicted in Figure 6.33(e). But in the case of the VGG16 model (refer Figure 6.33(f)), a patch attack takes place and is targeting the healthy class instead of the bean rust class.

Finally, the patch generated to perform adversarial patch attack targeting healthy class is shown in the Figure 6.34 w.r.t each model. Then when adversarial patch attack targets healthy class, as depicted in 6.34(d), some of the images are correctly classified, and some others are classified as a hot class in the case of the MobileNetV2 model. Similarly, in the case of the ResNet50 model (refer Figure 6.34(e)), targeted attack occurs on a hot class instead of a healthy class, which means defense is successful in classifying adversarial examples to the hot class. But in the case of VGG16 model (refer Figure 6.34(f)), targeted attack occurs on healthy class.

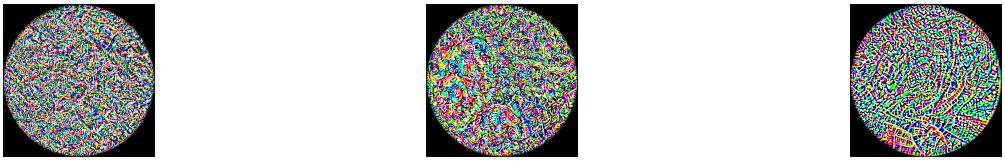
Figure 6.35 provides confidence levels of final predictions of DNNs before the attack and after the attack with varied patch sizes considered. As depicted in Figure 6.35, the confidence of the trained models is high before the attack in the case of the MobileNetV2 model and ResNet50 model. Once adversarial attacks occur targeting beans rust class, the confidence of these two models decreases. But as depicted, the confidence level during clean accuracy of VGG16 is less before the attack but increases after the attack and fails to improve the defense. Similar confidence level results are obtained even when an adversarial attack is performed, targeting other classes.

From the above observations, it can be inferred that the defense against adversarial patch attack can be improved using the Beans dataset with abstention class if the confidence level of the considered models during clean accuracy is high. That is if the confidence level of the considered models is high before the attack, then once the attack occurs, the confidence level decreases since adversarial images contains patch and is classified as a hot class, and hence, the defense against adversarial patch attack improves.

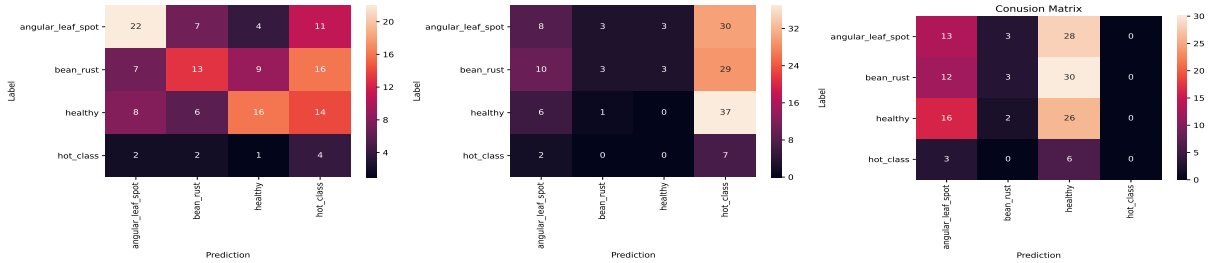


(a) Confusion matrix of MobileNetV2 (b) Confusion matrix of ResNet50 (c) Confusion matrix of VGG16

Figure 6.31: Confusion matrix of adversarial trained models on Beans dataset before adversarial patch attack

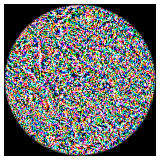


(a) Patch generated by MobileNetV2 model (b) Patch generated by ResNet50 model (c) Patch generated by VGG16 model

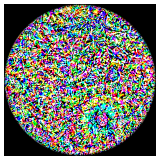


(d) Confusion matrix of MobileNetV2 (e) Confusion matrix of ResNet50 (f) Confusion matrix of VGG16

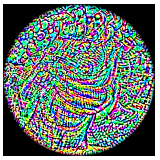
Figure 6.32: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting angular leaf spot class



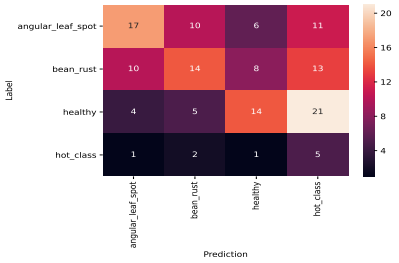
(a) Patch generated by MobileNetV2 model



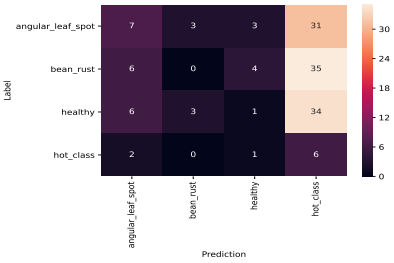
(b) Patch generated by ResNet50 model



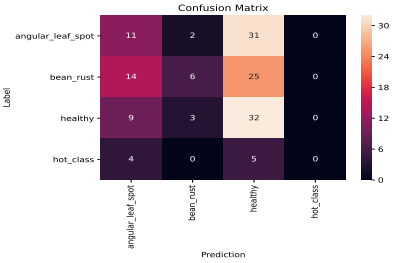
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

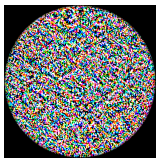


(e) Confusion matrix of ResNet50

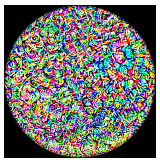


(f) Confusion matrix of VGG16

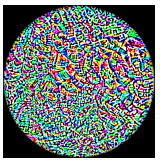
Figure 6.33: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bean rust class



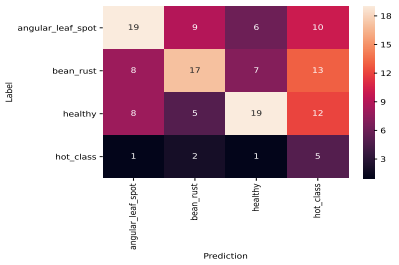
(a) Patch generated by MobileNetV2 model



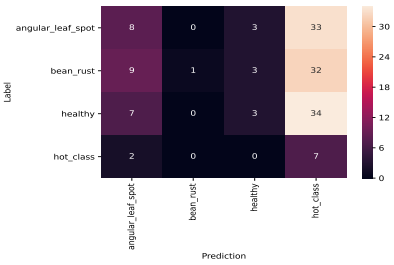
(b) Patch generated by ResNet50 model



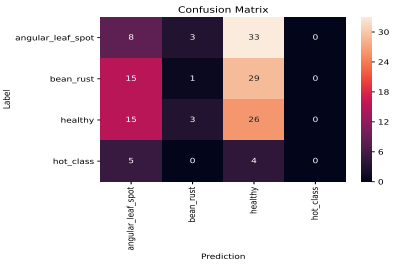
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

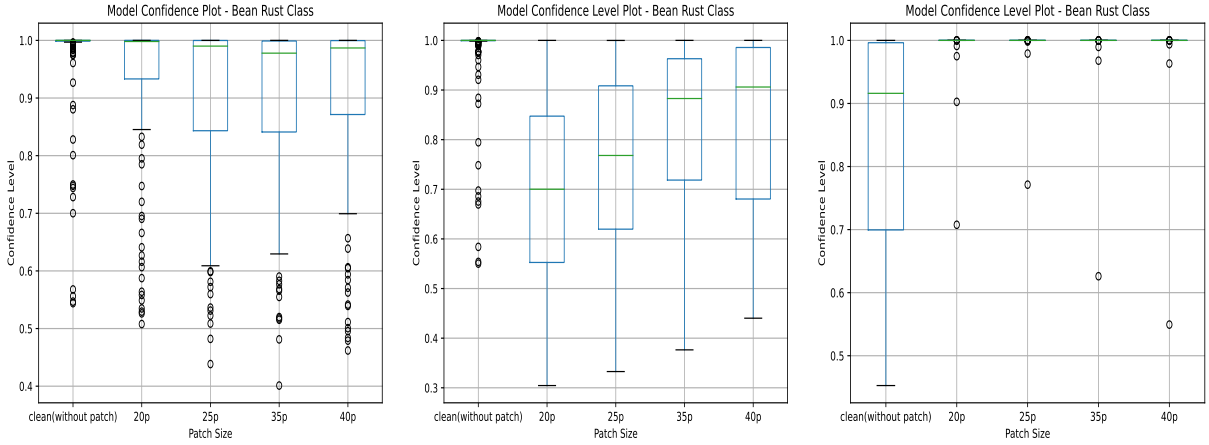


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.34: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting healthy class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 6.35: Confidence of models after adversarial patch attack targeting bean rust class

Imagenette Dataset

The confusion matrix of all trained models on the Imagenette dataset before the attack is shown in Figure 6.36. As illustrated in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting building class w.r.t each model is shown in Figure 6.37. Then after the attack, firstly, in the case of the MobileNetV2 model, Figure 6.37(d) depicts that when targeting building class, almost all images are classified as hot class (abstention class). In the case of ResNet50 model (refer Figure 6.37(e)), most of adversarial images are classified as hot class. But in the case of the VGG16 model (refer Figure 6.37(f)), the targeted attack happens on ball class instead of intended building class.

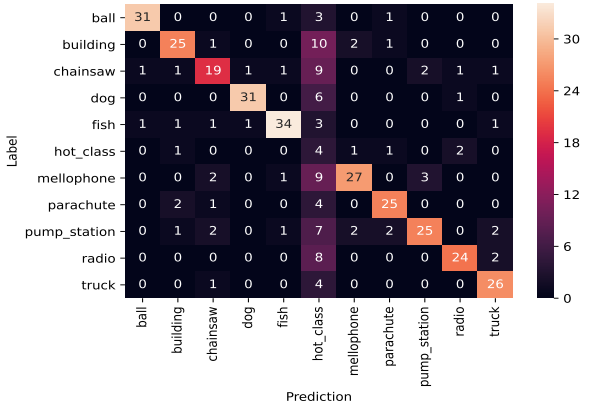
Secondly, the patch generated to perform adversarial patch attack targeting mellophone class is shown in the Figure 6.38 w.r.t each model. Then when adversarial patch attack targets mellophone class, in the case of MobileNetV2 model (refer Figure 6.38(d)), as depicted, almost all images are classified as a hot class. In the case of the ResNet50 model, most of the images are classified as a hot class, as depicted in Figure 6.38(e). But in the case of the VGG16 model (refer Figure 6.38(f)), a patch attack takes place and is targeting ball class instead of mellophone class.

Then, the patch generated to perform adversarial patch attack targeting truck class is shown in the Figure 6.39 w.r.t each model. When adversarial patch attack targets truck class, as depicted in 6.39(d), most of the images are classified as a hot class in the case of the MobileNetV2 model. Similarly, in the case of the ResNet50 model (refer Figure 6.39(e)), targeted attack occurs on a hot class instead of a truck class, i.e., a high number of adversarial images are classified as a hot class which means defense is successful in classifying adversarial samples as a hot class. But in the case of the VGG16 model (refer Figure 6.39(f)), targeted attack occurs on both ball class and mellophone class instead of truck class.

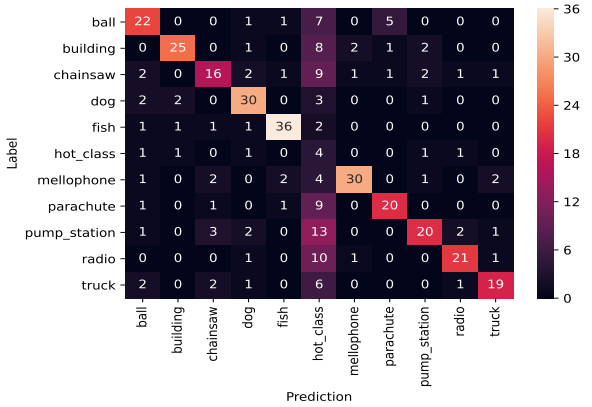
Figure 6.40 provides confidence levels of final predictions of DNNs before the attack and after the

attack with different patch sizes considered. As depicted in Figure 6.40, the confidence of the trained models is high before the attack in the case of the MobileNetV2 model and ResNet50 model. Once adversarial attacks occur targeting the mellophone class, the confidence of the models decreases. But as depicted, the confidence level of VGG16 is less before the attack (clean accuracy), and it increases after the attack. Similar confidence level results are obtained even when an adversarial attack is performed, targeting other classes.

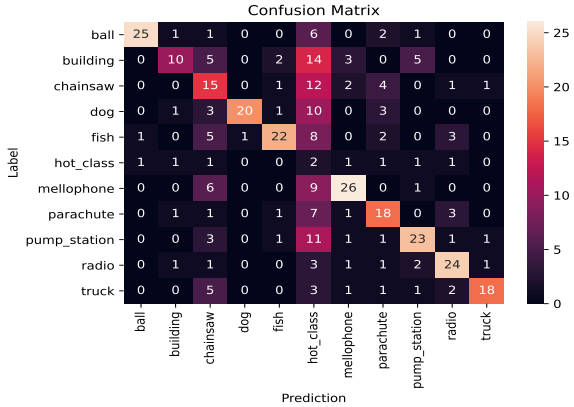
Inference from the experiment is that the defense against adversarial patch attack can be improved using Imagenette dataset with abstention class if the confidence level of the model during clean accuracy is high. That is if the confidence level of the model is high before the attack, then once the attack occurs, the confidence level decreases since adversarial images contains patch and is classified as a hot class, and hence, the defense against adversarial patch attack improves.



(a) Confusion matrix of MobileNetV2

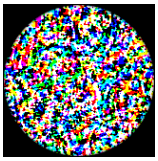


(b) Confusion matrix of ResNet50

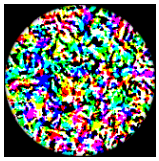


(c) Confusion matrix of VGG16

Figure 6.36: Confusion matrix of models trained on Imagenette dataset before adversarial patch attack

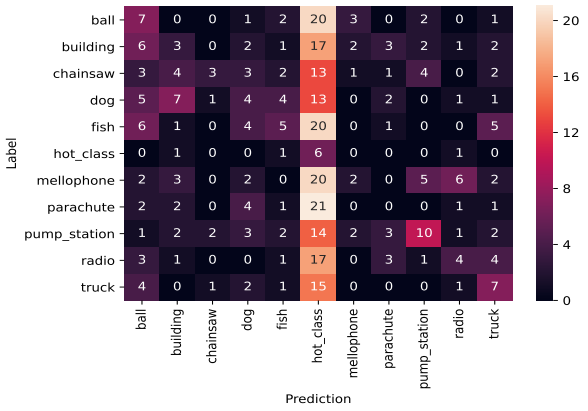


(a) Patch generated by MobileNetV2 model

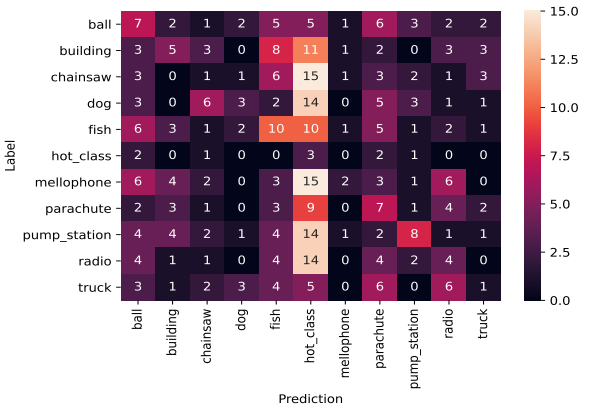


(b) Patch generated by ResNet50 model

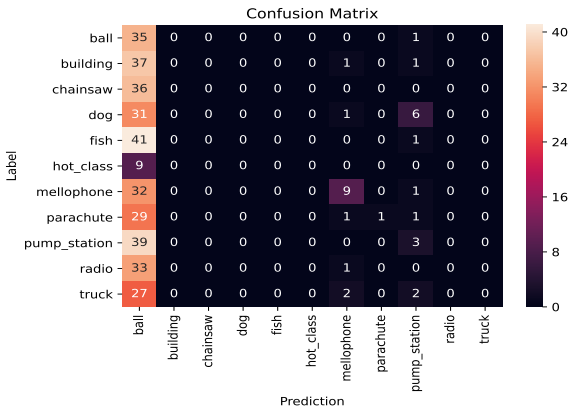
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

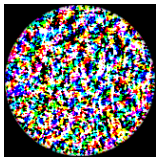


(e) Confusion matrix of ResNet50

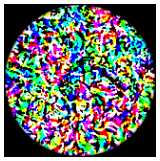


(f) Confusion matrix of VGG16

Figure 6.37: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting building class

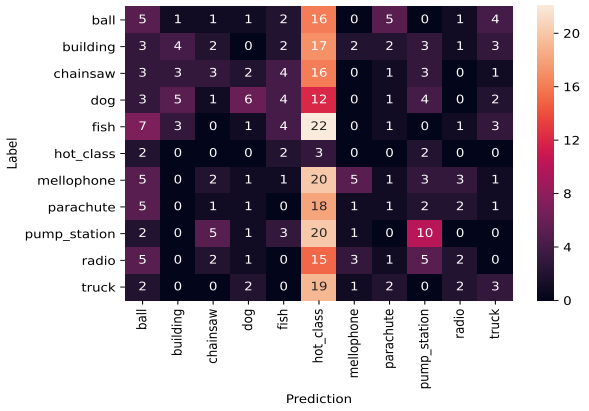


(a) Patch generated by MobileNetV2 model

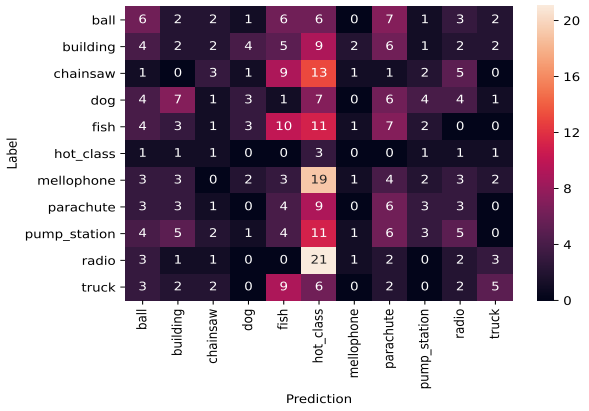


(b) Patch generated by ResNet50 model

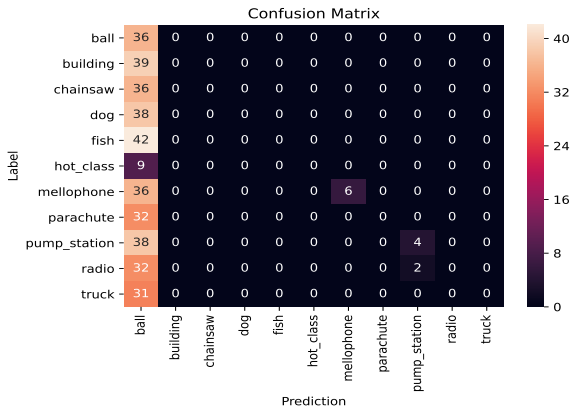
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

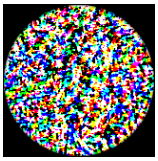


(e) Confusion matrix of ResNet50

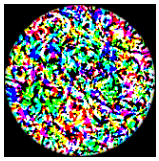


(f) Confusion matrix of VGG16

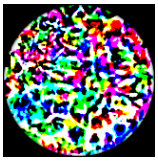
Figure 6.38: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting mellophone class



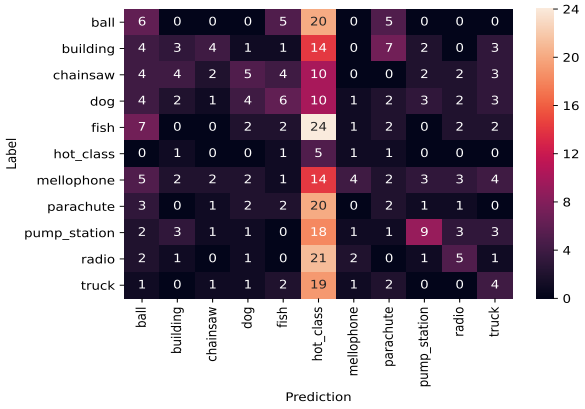
(a) Patch generated by MobileNetV2 model



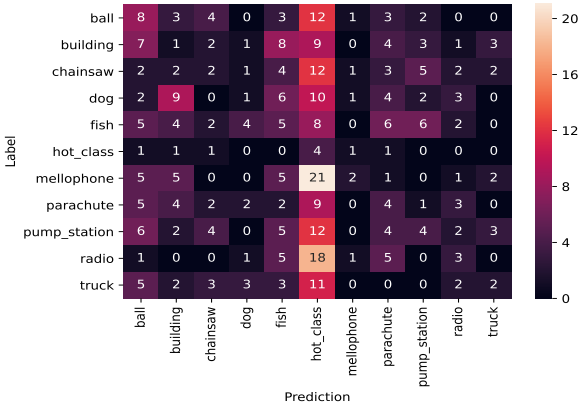
(b) Patch generated by ResNet50 model



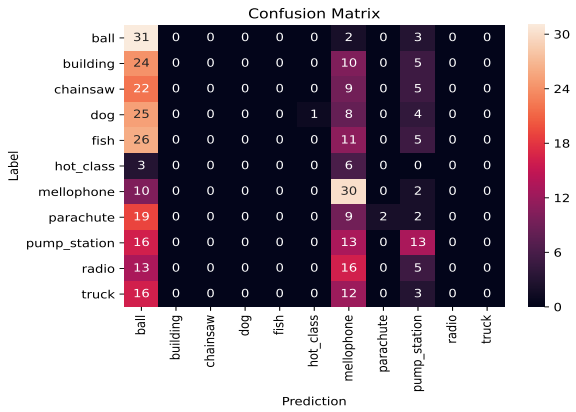
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

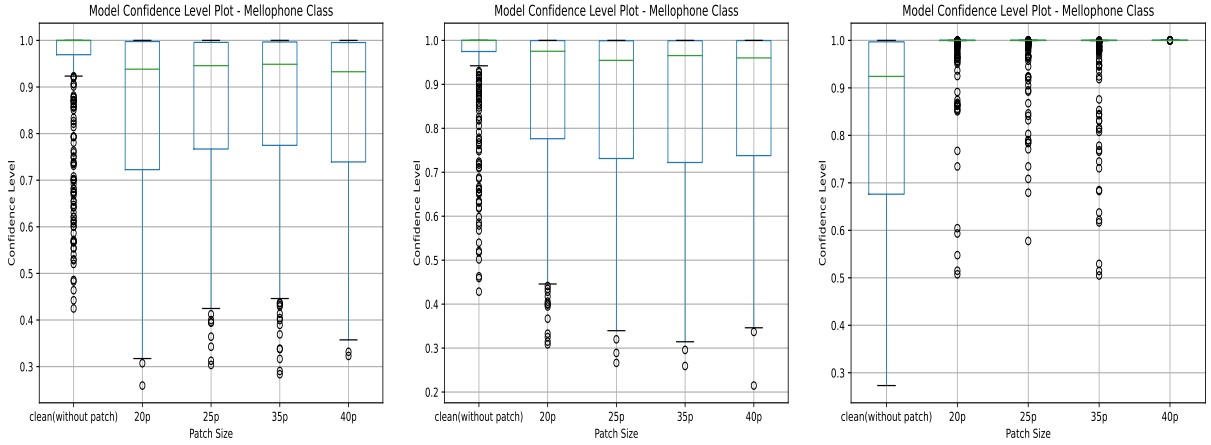


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.39: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting truck class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 6.40: Confidence of adversarial trained models before and after adversarial patch attack targeting mellophone class

RoboCup@Work Dataset

The confusion matrix of models trained on the RoboCup@Work dataset before the attack is shown in Figure 6.41. As illustrated in the confusion matrix, images have been adequately classified. The patch generated to perform the adversarial attack targeting bearing box class w.r.t each model is shown in Figure 6.42. Then after the attack, firstly, when targeting bearing box class, all the three models, MobileNetV2 model (refer Figure 6.42(d)), ResNet50 model (refer Figure 6.42(e)) and VGG16 model (refer Figure 6.42(f)) classifies most of the adversarial images as hot class. Here, adding an abstention class helps to classify adversarial images as a hot class.

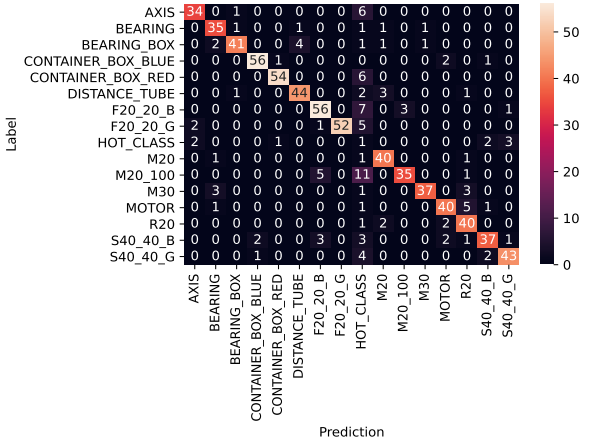
Secondly, the patch generated to perform adversarial patch attack targeting M20_100 class is shown in Figure 6.43 w.r.t each model. Similarly, when an adversarial patch attack targets the M2_100 class, the majority of the adversarial images are classified as a hot class in all three models. In case of MobileNetV2 model (refer Figure 6.43(d)) and ResNet50 model (refer Figure 6.43(e)), the adversarial images which are not classified as hot class is spread over all other classes. But in the VGG16 case, as shown in Figure 6.43(f), the adversarial images which are not classified as a hot class are classified as F20_20_B or F20_20_G and not as any other classes.

Then, the patch generated to perform adversarial patch attack targeting the S40_40_G class is shown in Figure 6.44 w.r.t each model. Then when adversarial patch attack targets S40_40_G class, as depicted in both MobileNetV2 model (refer Figure 6.44(d)) and ResNet50 model (refer Figure 6.44(e)), most of the adversarial images are classified as a hot class and the rest are spread over other classes. But in the case of the VGG16 model (refer Figure 6.44(f)), almost all adversarial examples are classified as a hot class.

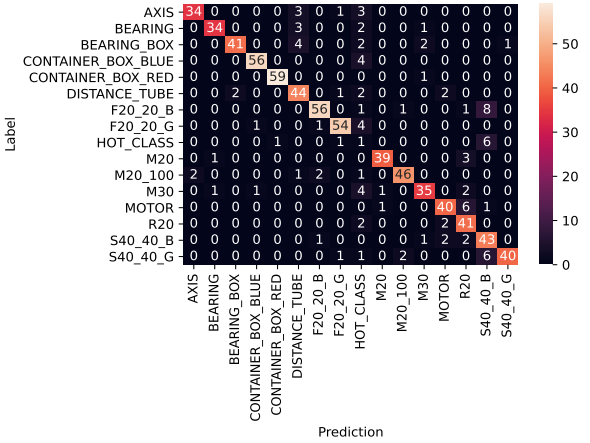
Figure 6.45 provides confidence levels of final predictions of DNNs before the attack and after the attack with different patch sizes considered. As depicted in Figure 6.45, the confidence level of all the

trained models is high before the attack. Once adversarial attacks occur targeting M20_100 class, the confidence of the models decreases. Similar confidence level results are obtained even when an adversarial attack is performed targeting other classes.

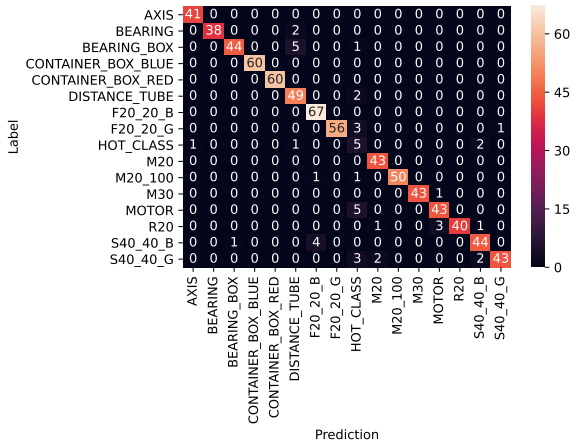
From the above observations, inference can be made that defense against adversarial patch attack can be improved using RoboCup dataset with addition of abstention class if the confidence level of the model during clean accuracy is high. That is, before the attack, if the confidence level of the model is high. Then defense against adversarial patch attacks improves by classifying adversarial examples as a hot class (abstention class).



(a) Confusion matrix of MobileNetV2

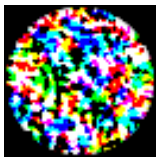


(b) Confusion matrix of ResNet50

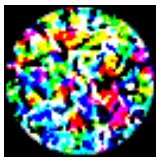


(c) Confusion matrix of VGG16

Figure 6.41: Confusion matrix of models trained on RoboCup@Work dataset before adversarial patch attack

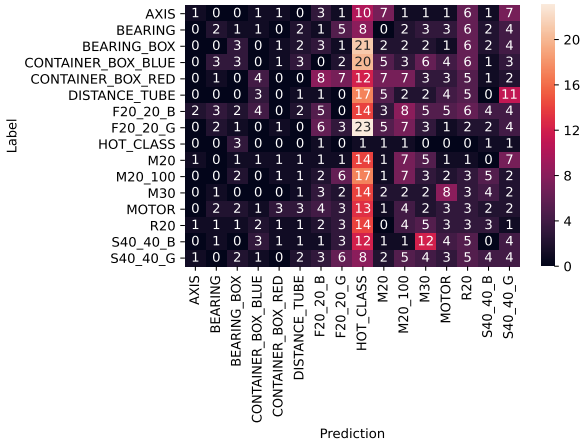


(a) Patch generated by MobileNetV2 model

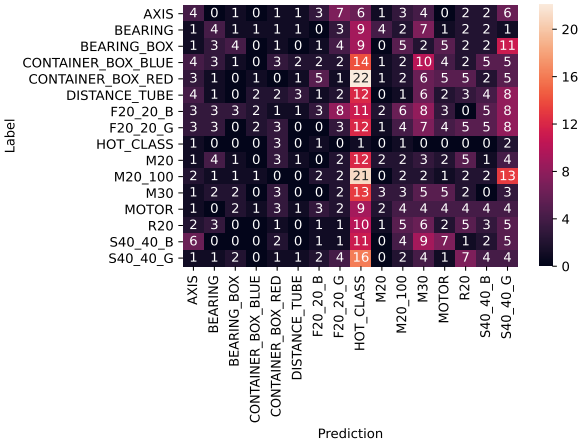


(b) Patch generated by ResNet50 model

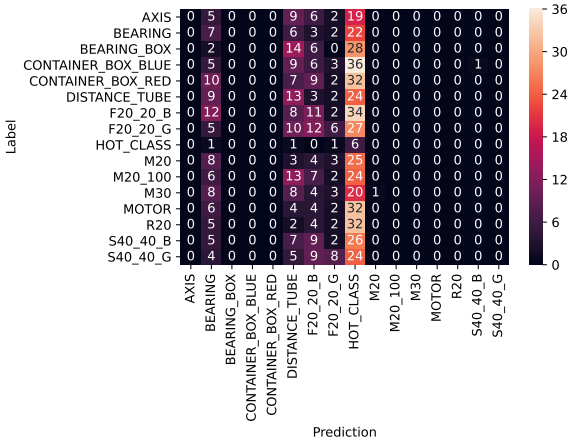
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

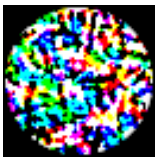


(e) Confusion matrix of ResNet50

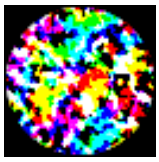


(f) Confusion matrix of VGG16

Figure 6.42: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bearing box class

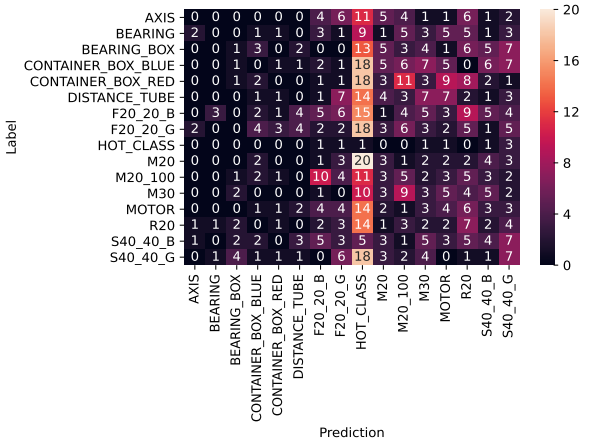


(a) Patch generated by MobileNetV2 model

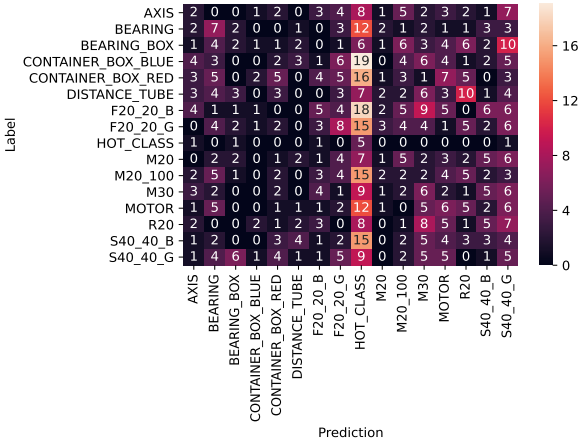


(b) Patch generated by ResNet50 model

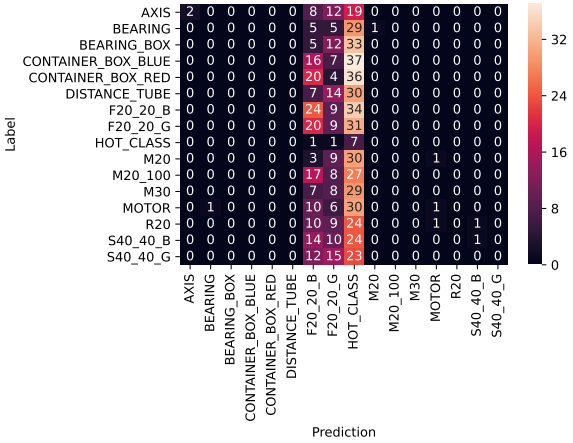
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

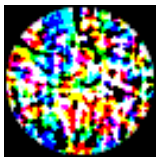


(e) Confusion matrix of ResNet50

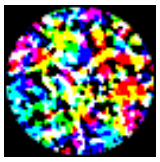


(f) Confusion matrix of VGG16

Figure 6.43: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting M20_100 class

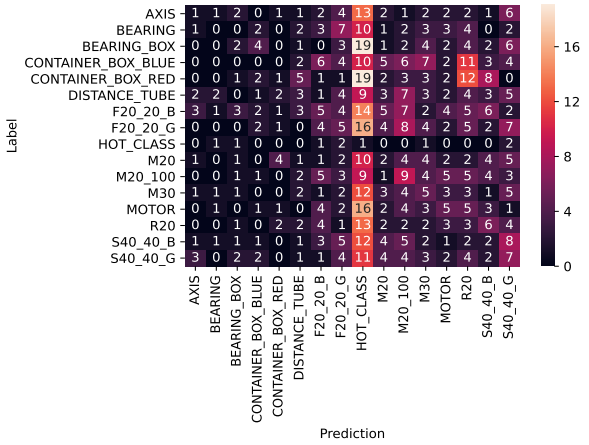


(a) Patch generated by MobileNetV2 model

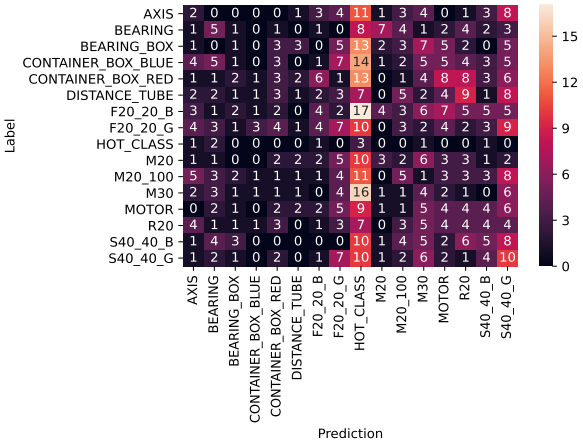


(b) Patch generated by ResNet50 model

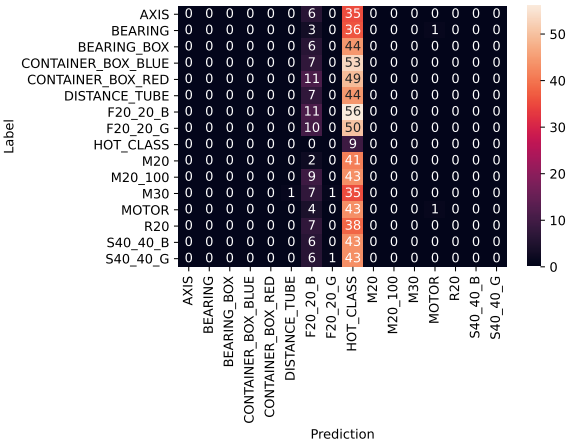
(c) Patch generated by VGG16 model



(d) Confusion matrix of MobileNetV2

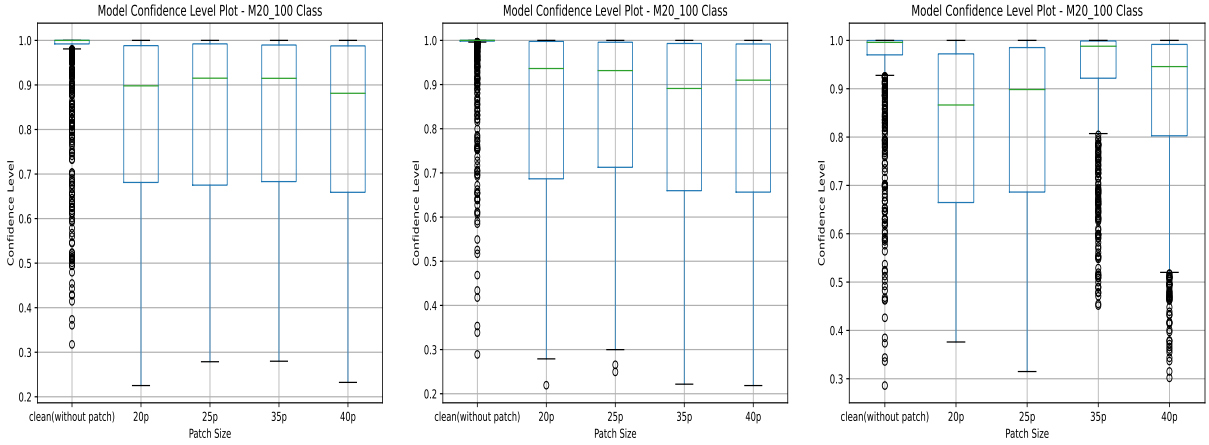


(e) Confusion matrix of ResNet50



(f) Confusion matrix of VGG16

Figure 6.44: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting S40_40_G class



(a) MobileNetV2 model (b) ResNet50 model (c) VGG16 model

Figure 6.45: Confidence of adversarial trained models before and after adversarial patch attack targeting M20_100 class

6.3.4 Verdict

Based on the considered datasets and models, the results obtained from the experiments adhere to the hypothesis considered in Section 6.3.1. Hence, hypothesis considered is valid and can be accepted. From this, it can be concluded that by adding an abstention class to the dataset, defense against adversarial patch attack can be improved. But the model trained with dataset and abstention class must have a high confidence level before the attack. That is, if the confidence level of the model before the attack is high then it provides better defense against adversarial patch attacks.

6.4 Evidential Uncertainty Estimation Defense against Adversarial Patch Attack

RQ5. Can the evidential uncertainty estimation method be used to defend against adversarial patch attacks?

The main idea behind the evidential uncertainty estimation defense method is to check how confident a neural network is in its prediction. As neural networks are trained to minimize the loss in the prediction result, they are blind to their final prediction's confidence level. This method was introduced in the paper "*Evidential Deep Learning to Quantify Classification Uncertainty*" [56]. In this research, the evidential uncertainty estimation method is used against adversarial patch attacks to check the uncertainty level of a neural network after the attack i.e., to find how certain and uncertain the neural network is about its result. "*For instance, is the neural net able to identify data points belonging to an unrelated data distribution? Can it simply say I do not know if we feed in a cat picture after training the net on a set of handwritten digits?*" [56]. The intuition behind this defense mechanism is to check the uncertainty level of a neural network's prediction to determine whether an adversarial patch attack has happened or not.

In this research work, this evidential uncertainty estimation method is used to check whether an adversarial attack has occurred or not, based on the level of uncertainty present in the predictions of neural networks. To predict categorical distribution, softmax is used in standard neural network. When an unknown/unseen or highly different example is presented to a network, softmax in the neural network helps make overconfident wrong predictions. Softmax estimates the class probabilities of a given sample without indicating the uncertainty associated with it. Hence, this method makes use of Dirichlet distribution to estimate class probability distribution which helps to quantify the uncertainty associated with the prediction of the network.

6.4.1 Hypothesis

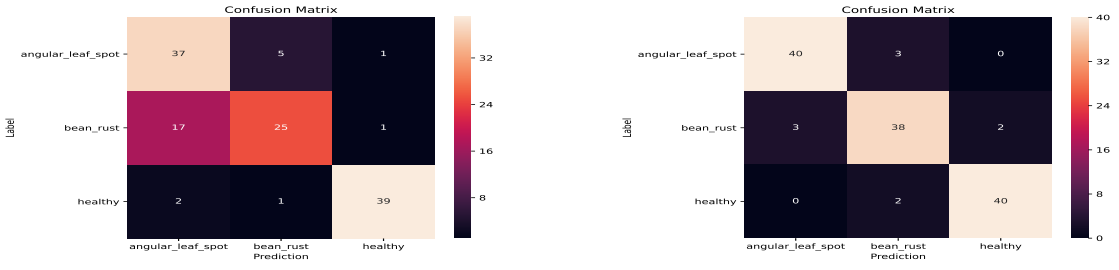
Hypothesis: evidential uncertainty estimation can be used to provide defense against adversarial patch attack

To perform this experiment, a hypothesis is considered. The hypothesis states it is possible to use the evidential uncertainty estimation technique as a defense against an adversarial patch attack.

6.4.2 Observation

In this experimentation, an adversarial patch attack is performed on the MobileNetV2 network and ResNet50 network w.r.t the Beans dataset. Then with respect to the Imagenette dataset, a patch attack is performed on the MobileNetV2 model. The considered patch size for this experiment is 20%, 25%, 35% and 40%.

Beans Dataset



(a) MobileNetV2 model (b) ResNet50 model

Figure 6.46: Confusion matrix of the models before adversarial patch attack

The confusion matrix of Beans dataset trained using evidential uncertainty estimation method before the attack is depicted in Figure 6.46. After the attack, firstly, in the case of MobileNetV2 model, Figure 6.47(c) depicts that when the attack has targeted intended angular leaf spot class with low intensity, and it can also be noticed that the model is uncertain, as some adversarial images are classified as bean rust and healthy class. In the ResNet50 model's case (refer Figure 6.47(d)), a high percentage of images are

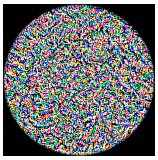
classified as bean rust, and the rest are classified as a healthy class. In this case, target attack fails and is uncertain about its prediction.

Then when adversarial patch attack targets bean rust class, in case of MobileNetV2 model (refer Figure 6.48(c)), as illustrated, a high amount of images are classified as an angular class instead of targeted bean class. In the case of the ResNet50 model, most of the images are classified as intended bean rust class as depicted in Figure 6.48(d), and due to prediction uncertainty, some images are classified as a healthy class.

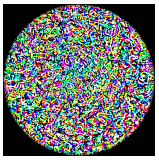
Then when adversarial patch attack targets healthy class, as depicted in 6.49(c), most of the images are correctly classified, in the rest, some images have classified as angular leaf spot and other are classified randomly due to uncertainty in the case of MobileNetV2 model. Even targeted adversarial attacks did not occur. But, in the case of the ResNet50 model (refer Figure 6.49(d)), targeted attack occurs on bean rust class with high intensity instead of healthy class which model is not confident in its prediction.

As depicted in Figure 6.50, the confidence of the pre-trained models is high before the attack. Once adversarial attacks occur targeting beans rust class, the confidence of models decreases significantly. For MobileNetV2 model (refer Figure 6.50(a)), before the attack, the confidence level is between 50-90% but after the attack, confidence drops to 40-55%. This shows that the model trained on the uncertainty estimation method is not confident while an unknown (adversarial) image is given as input. Similarly, the ResNet50 model has a high confidence level between 90-100% before the attack but significantly decreases to 50-90% after the attack depending on the scale of the adversarial patch as shown in Figure 6.50(b). Similar confidence level results were obtained even when an adversarial attack is performed, targeting other classes. These results say that when an extremely different image (adversarial image) is presented to the model trained on the evidential uncertainty estimation method, the confidence level of the output will be too low, which helps distinguish between adversarial images and non-adversarial images.

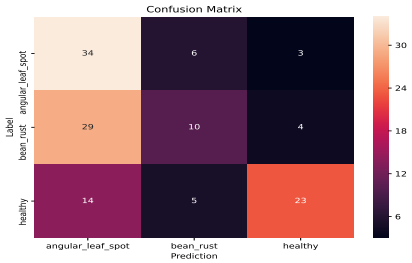
From the above observations, inference can be made that defense against adversarial patch attacks can be improved using the evidential uncertainty estimation method. The results show that the confidence level of the model's prediction will be high when similar class images are presented as input. But when adversarial images are presented as input, the model's confidence level will be low during its prediction, which makes it easy to identify adversarial images when attacked.



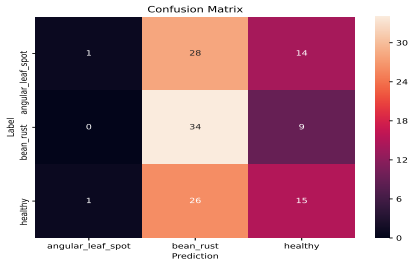
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

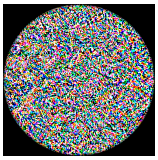


(c) Confusion matrix of MobileNetV2

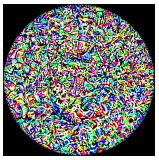


(d) Confusion matrix of ResNet50

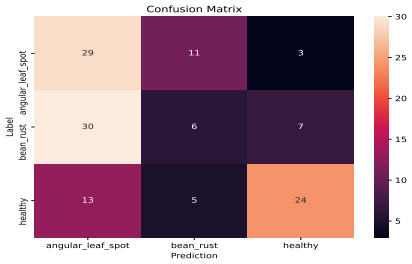
Figure 6.47: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting angular leaf spot class



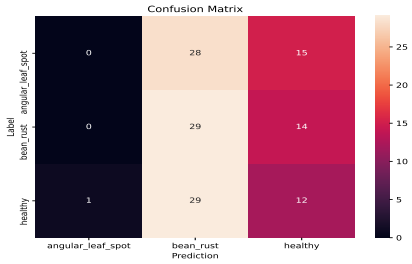
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

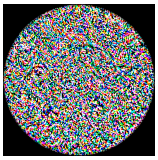


(c) Confusion matrix of MobileNetV2

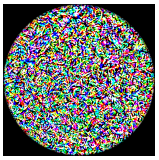


(d) Confusion matrix of ResNet50

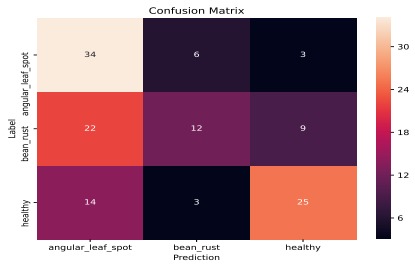
Figure 6.48: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting bean rust class



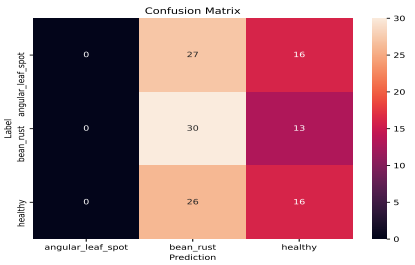
(a) Patch generated by MobileNetV2 model



(b) Patch generated by ResNet50 model

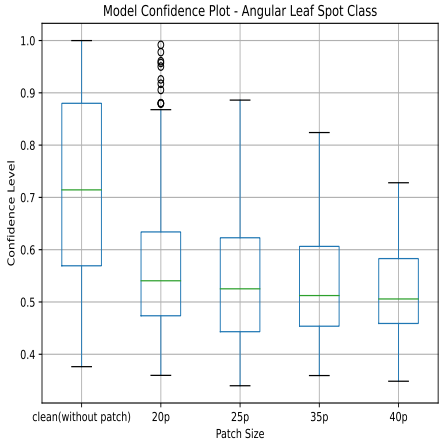


(c) Confusion matrix of MobileNetV2

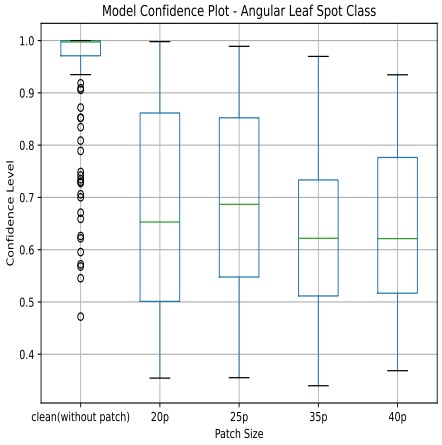


(d) Confusion matrix of ResNet50

Figure 6.49: Patch generated to perform adversarial patch attack and confusion matrix after adversarial patch attack targeting healthy class



(a) MobileNetV2 model



(b) ResNet50 model

Figure 6.50: Confidence of models after adversarial patch attack targeting bean rust class

Imagenette Dataset

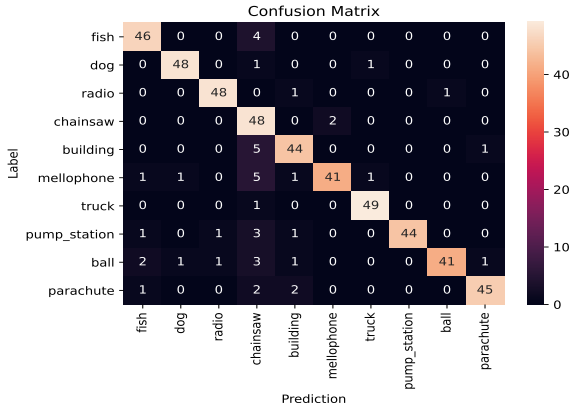
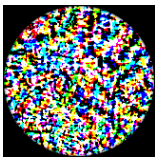


Figure 6.51: Confusion matrix of MobileNetV2 model before attack

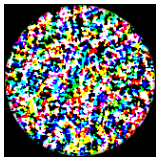
The confusion matrix of Imagenette dataset trained using evidential uncertainty estimation method before the attack is depicted in Figure 6.51. The patches generated for adversarial patch attack is depicted in Figure 6.52. After the attack on MobileNetV2 model, when attack targeted radio class (refer Figure 6.52(d)), mellophone class (refer Figure 6.52(e)) and truck class (refer Figure 6.52(f)), results shows that in all three instance targeted attack happened on chainsaw class with high intensity instead of intended classes.

As depicted in Figure 6.53, the confidence of the trained models is high before the attack. Once adversarial attacks occur, the confidence of the models decreases remarkably. As shown, the confidence level of the MobileNetV2 model before the attack is nearly 100%, but after the attack, in the case of radio class (refer Figure 6.53(a)), the confidence level drops to 0-100% depending upon the patch size of the attack. Similarly, both in the case of mellophone class (refer Figure 6.53(b)) and truck class (refer Figure 6.53(c)), accuracy drop to 0-100% after the attack. One more interesting observation made from the result is that as the patch size increases, the model's confidence level keeps on decreasing. As shown, for the patch size of 35% and 40%, the confidence level is between 0-20%. This shows that the model is not confident in predicting when adversarial images with large patch sizes are presented as inputs. Similar confidence level results were obtained even when an adversarial attack is performed, targeting other classes. These results show that when an extremely different image (adversarial image) is presented to the model trained on the evidential uncertainty estimation method, the confidence level of the output will be too low, which helps distinguish between adversarial images and non-adversarial images.

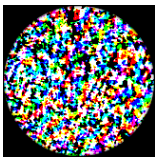
From the above observations, inference can be made that defense against adversarial patch attacks can be improved using the evidential uncertainty estimation technique. The results show that the confidence level of the model's prediction will be high when similar class images are presented as input. But when adversarial images are presented as input, the model's confidence level will be low during its prediction, which makes it easy to identify adversarial images when attacked.



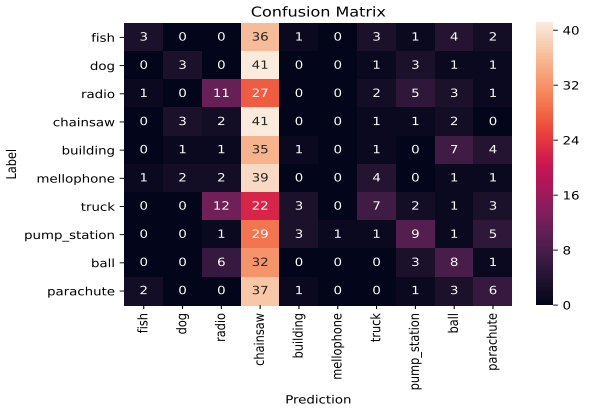
(a) Patch generated for radio class



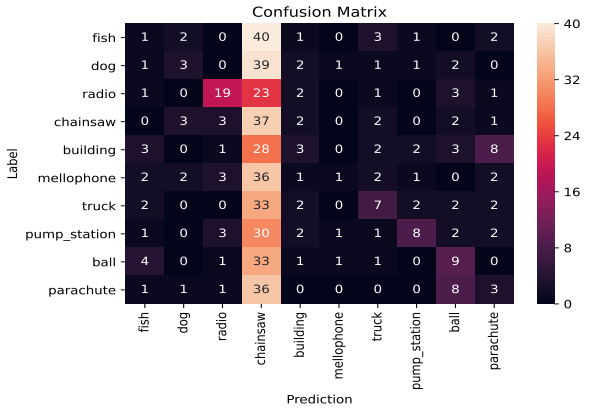
(b) Patch generated for mellophone class



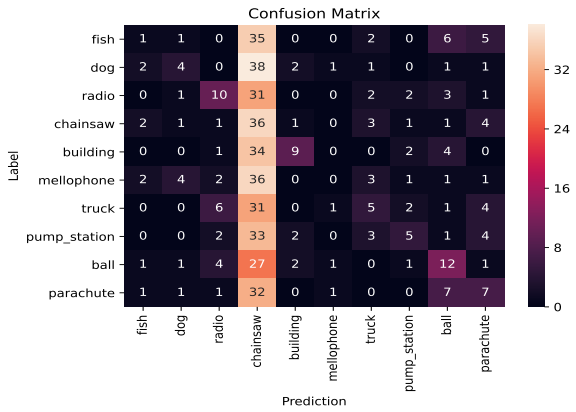
(c) Patch generated for truck class



(d) Radio class

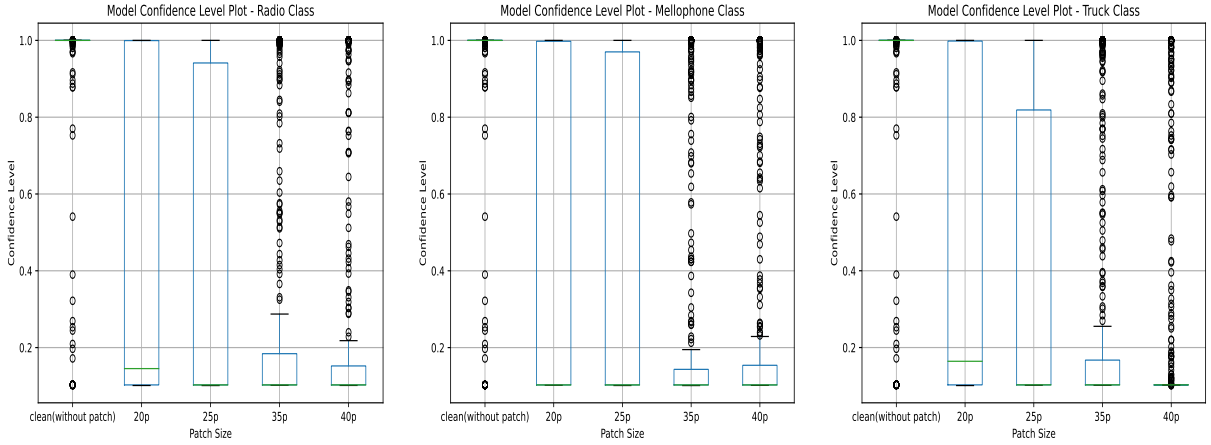


(e) Mellophone class



(f) Truck class

Figure 6.52: Patch generated to perform adversarial patch attack on MobileNetV2 model and confusion matrix after adversarial patch attack targeting respective classes are shown



(a) Radio class (b) Mellophone class (c) Truck class

Figure 6.53: Confidence of MobileNetV2 model after patch attack targeting respective classes

6.4.3 Verdict

Based on the considered datasets and models, the results obtained from the experiments adhere to the hypothesis considered in Section 6.4.1. Hence, hypothesis considered is valid and can be accepted. From this, it can be concluded that by using evidential uncertainty estimation method, defense against adversarial patch attack can be improved.

6.5 Evaluation Summary

All the adversarial patch attack experiments conducted in this research work is summarised in Table 6.1. The experiments which gave negative results are represented as “Not worked” and the experiments with positive results are represented as “Worked” highlighting with green colour. A targeted adversarial patch attack experiment is performed to check whether is it possible to achieve a targeted patch attack every time. The experiments provide negative results, which show it is impossible to perform an adversarial patch attack every time.

All the defense experiments conducted against adversarial patch attack in this research work is summarised in Table 6.2. The experiments which gave negative results are represented as “Not effective” and the experiments with positive results are expressed as “Effective” highlighting with green colour and - indicates unsuccessful experiments. As depicted in the Table 6.2, the results from the first defense experiment tell adversarial training defense is unsuccessful against adversarial patch attacks. The second experiment, abstention class defense, proves that it is feasible to deflect all adversarial images towards an abstention class, which tells abstention class defense can be employed against adversarial patch attacks if confidence level of predictions of the models is high before the attack occurs. This abstention class defense is not effective when confidence of the model is low before the attack. Lastly, the final experiment

is evidential uncertainty estimation defense, whose results show it is feasible to provide defense against the adversarial images.

Experiment	Model	Dataset			Metric
		Beans	Imagenette	RoboCup@Work	
Targeted adversarial patch attack	MobileNetV2	Not worked	Not worked	Not worked	Confusion matrix
	ResNet50	Not worked	Not worked	Not worked	Confusion matrix
	VGG16	Not worked	Not worked	Not worked	Confusion matrix

Table 6.1: List of all adversarial patch attack experiments performed in this research

Experiment	Model	Datasets			Metrics
		Beans	Imagenette	RoboCup@Work	
Adversarial training defense	MobileNetV2	Not effective	Not effective	Not effective	Confusion matrix & Model confidence level
	ResNet50	Not effective	Not effective	Not effective	Confusion matrix & Model confidence level
	VGG16	-	Not effective	Not effective	Confusion matrix & Model confidence level
Abstention class defense	MobileNetV2	Effective	Effective	Effective	Confusion matrix & Model confidence level
	ResNet50	Effective	Effective	Effective	Confusion matrix & Model confidence level
	VGG16	Not effective	Not effective	Effective	Confusion matrix & Model confidence level
Evidential uncertainty estimation defense	MobileNetV2	Effective	Effective	-	Confusion matrix & Model confidence level
	ResNet50	Effective	-	-	Confusion matrix & Model confidence level

Table 6.2: List of all defense experiments performed in this research against adversarial patch attack

7

Conclusions

This research work focuses on a comparative evaluation of defense mechanisms against adversarial patch attack performed on DNNs in the field of image classification. The purpose of this research work is to perform an adversarial patch attack on DNNs and evaluate various defense techniques against it as a means to provide potential defense techniques against physical real-world adversarial patch attack for DL researchers for future research and development. A survey was conducted on adversarial attacks. Considering real-world scenarios, an adversarial patch attack was chosen for this research work as the base attack with a small patch size of 20%, 25%, 35% and 40%. In this research study, an inspection was performed on adversarial patch attacks to check whether is it possible to perform a targeted adversarial patch attack at all times. A survey was conducted to identify defense techniques present against adversarial attacks. One of the state-of-the-art methods against digital-world adversarial attacks was chosen, which is adversarial training defense. Then a state-of-the-art Out-Of-Distribution (OOD) detection mechanism was selected and evaluated on adversarial patch attack. In this study, the evidential uncertainty estimation method was also tried against adversarial patch attack. The attack and defense techniques were performed on three TensorFlow DNNs, namely MobileNetV2, ResNet50 and VGG16 using publicly available datasets, namely Beans and Imagenette datasets, and the RoboCup@Work dataset provided by the RoboCup team of Hochschule Bonn-Rhein-Sieg.

A hypothesis was assumed for each research question (refer Figure 1.3)and validity of hypothesis was verified using suitable experiments. The inspection performed on adversarial patch attack reveals that it is not feasible possible to achieve a targeted attack. The adversarial training, which is the state-of-the-art defense against digital-world adversarial attacks, is not useful against physical real-world adversarial patch attacks. However, abstention class defense and evidential uncertainty defense techniques provide positive results and can be employed against adversarial patch attacks.

The following section 7.1 reports about the experiments and the work done in this research. In section 7.2, results obtained are presented and in section 7.3, future explorations are presented.

7.1 Contributions

Contributions of this research are as follows:

- **Adversarial patch attack:** This research works provides detailed explanation about adversarial

patch attack and patch generation process. In addition to this, an evaluation of adversarial attack is performed on three DNN using three datasets.

- **Literature:** This research work provides a literature review on defense mechanisms used against adversarial patch attacks.
- **Targeted adversarial patch attack:** An experiment is conducted to explore the possibility of always performing a targeted adversarial patch attack.
- **Adversarial training defense:** In this research work, evaluation of adversarial training defense is performed against adversarial patch attack.
- **Abstention class defense:** In this research work, abstention class defense is evaluated against adversarial patch attack. An abstention class is created and added to the training dataset to increase the protection of the network against adversarial patch attacks.
- **Evidential uncertainty estimation:** In this research work, the evidential uncertainty estimation technique is performed against adversarial patch attack.

7.2 Lessons Learned

Lessons learned during this research are as follows:

- A study was made to identify and learn about various defense techniques that can be deployed against adversarial patch attack.
- A targeted adversarial patch attack is not always possible.
- Adversarial defense training, which is one of the state-of-the-art defense against digital-world adversarial attacks, does not provide defense against real-world adversarial patch attacks.
- Abstention class can provide the defense against adversarial patch attack. Elements added to the abstention class must be chosen carefully; choosing the wrong items for the abstention class will affect the performance of its defense against patch attacks. It is possible to deflect adversarial images towards added hot class (abstention class) by adding the abstention class to training set.
- Evidential uncertainty estimation technique can be used to detect the adversarial images based on the confidence of networks prediction. When a network gets an adversarial image, the evidential uncertainty estimation method outputs the prediction with very low confidence, which helps to distinguish between normal samples and adversarial samples.

7.3 Future Work

The potential future direction of this research effort is as follows.

- Transferability of adversarial training can be made to check its defense efficiency towards adversarial patch attack.

- Research about abstention class could be done to find the exact amount of items needed to provide a maximum defense. What different items can be used as abstention class? An underlying pattern of elements belonging to the abstention class is to be researched in future.
- Research can also be done on the right number of elements needed for the abstention class. As the size of the dataset differs, the number of elements necessary for abstention class also differs.
- Evidential uncertainty estimation defense can be extended to enhance its performance.

A

Details of elements added to abstention class

The correct elements need to be added to the new class (abstention/hot class) in abstention class defence to protect against an adversarial patch attack. In this research, the number of elements added to the abstention class w.r.t each dataset is given below,

- Beans dataset - 70 random images from each three classes, a total of 210 images
- Imagenette dataset - 25 random images from each ten classes, a total of 250 images
- RoboCup@Work dataset - 25 random images from each fifteen classes, a total of 375 images

The images added to the abstention class are obtained from training data. That is, a random part of training data is removed and added to the abstention class. Hence, after adding the abstention class, both the training and testing set contains unique elements.

References

[1] Mohiuddin Ahmed and AKM Najmul Islam. Deep learning: hope or hype. *Annals of Data Science*, 7 (3):427–432, 2020.

[2] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/athalye18b.html`.

[4] Andreas Bar, Jonas Lohdefink, Nikhil Kapoor, Serin John Varghese, Fabian Huger, Peter Schlicht, and Tim Fingscheidt. The vulnerability of semantic segmentation networks to adversarial attacks in autonomous driving: Enhancing extensive environment sensing. *IEEE Signal Processing Magazine*, 38(1):42–52, Jan 2021. ISSN 1558-0792. doi: 10.1109/MSP.2020.2983666.

[5] Daniel S Berman, Anna L Buczak, Jeffrey S Chavis, and Cherita L Corbett. A survey of deep learning methods for cyber security. *Information*, 10(4):122, 2019.

[6] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[7] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial Patch. *arXiv e-prints*, art. arXiv:1712.09665, December 2017.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[9] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7. IEEE, 2018.

[10] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[11] Tong Chen, Jiqiang Liu, Yingxiao Xiang, Wenjia Niu, Endong Tong, and Zhen Han. Adversarial attack and defense in reinforcement learning-from ai security view. *Cybersecurity*, 2(1):1–22, 2019.

[12] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693*, 2020.

[13] Li Deng and Yang Liu. *Deep learning in natural language processing*. Springer, 2018.

[14] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and trends in signal processing*, 7(3–4):197–387, 2014.

[15] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*, 2016.

[16] Sid Ahmed Fezza, Yassine Bakhti, Wassim Hamidouche, and Olivier Déforges. Perceptual evaluation of adversarial attacks for cnn-based image classification. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2019.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[18] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[19] Samuel Harding, Prashanth Rajivan, Bennett I Bertenthal, and Cleotilde Gonzalez. Human decisions on targeted and non-targeted adversarial sample. In *CogSci*, 2018.

[20] Mohammed Hassanin, Nour Moustafa, and Murat Tahtali. A deep marginal-contrastive defense against adversarial attacks on 1d models. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1499–1505, Dec 2020. doi: 10.1109/SSCI47803.2020.9308330.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Jeff Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning, 2018.

[23] Shahar Hoory, Tzvika Shapira, Asaf Shabtai, and Yuval Elovici. Dynamic adversarial patch for evading object detection models. *arXiv preprint arXiv:2010.13070*, 2020.

[24] Jeremy Howard. imagenette. URL `https://github.com/fastai/imagenette/`.

[25] Anil K Jain, Debayan Deb, and Joshua J Engelsma. Biometrics: Trust, but verify. *arXiv preprint arXiv:2105.06625*, 2021.

[26] Branislav Kisačanin. Deep learning for autonomous vehicles. In *2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 142–142. IEEE, 2017.

[27] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[28] Hyun Kwon, Yongchul Kim, Ki-Woong Park, Hyunsoo Yoon, and Daeseon Choi. Multi-targeted adversarial example in evasion attack on deep neural network. *IEEE Access*, 6:46084–46096, 2018.

[29] Makerere AI Lab. Bean disease dataset, January 2020. URL `https://github.com/AI-Lab-Makerere/ibean/`.

[30] Jiao Li, Yang Liu, Tao Chen, Zhen Xiao, Zhenjiang Li, and Jianping Wang. Adversarial attacks and defenses on cyber–physical systems: A survey. *IEEE Internet of Things Journal*, 7(6):5103–5115, 2020.

[31] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018.

[32] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017.

[33] Yan Luo, Xavier Boix, Gemma Roig, Tomaso Poggio, and Qi Zhao. Foveation-based mechanisms alleviate adversarial examples. *arXiv preprint arXiv:1511.06292*, 2015.

[34] Samaneh Mahdavifar and Ali A Ghorbani. Application of deep learning to cybersecurity: A survey. *Neurocomputing*, 347:149–176, 2019.

[35] Michael J McCoyd. *Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches*. University of California, Berkeley, 2020.

[36] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147, 2017.

[37] Lubin Meng, Chin-Teng Lin, Tzyy-Ping Jung, and Dongrui Wu. White-box target attack for eeg-based bci regression problems. In *International conference on neural information processing*, pages 476–488. Springer, 2019.

[38] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

[39] Gautam Raj Mode and Khaza Anuarul Hoque. Adversarial examples in deep learning for multivariate time series regression. In *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–10. IEEE, 2020.

[40] Sina Mohseni, Mandar Pitale, JBS Yadawa, and Zhangyang Wang. Self-supervised learning for generalizable out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5216–5223, 2020.

[41] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1300–1307. IEEE, 2019.

[42] Elior Nehemya, Yael Mathov, Asaf Shabtai, and Yuval Elovici. Taking over the stock market: Adversarial perturbations against algorithmic traders.

[43] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molloy, and Ben Edwards. Adversarial robustness toolbox v1.2.0. *CoRR*, 1807.01069, 2018. URL `https://arxiv.org/pdf/1807.01069`.

[44] Mesut Ozdag. Adversarial attacks and defenses against deep neural networks: a survey. *Procedia Computer Science*, 140:152–161, 2018.

[45] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[46] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[47] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrami. Adaptive adversarial videos on roadside billboards: Dynamically modifying trajectories of autonomous vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5916–5921. IEEE, 2019.

[48] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431, 2018.

[49] Huali Ren, Teng Huang, and Hongyang Yan. Adversarial examples: attacks and defenses in the physical world. *International Journal of Machine Learning and Cybernetics*, pages 1–12, 2021.

[50] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[51] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[52] Meysam Sadeghi and Erik G. Larsson. Adversarial attacks on deep-learning based radio signal classification. *IEEE Wireless Communications Letters*, 8(1):213–216, Feb 2019. ISSN 2162-2345. doi: 10.1109/LWC.2018.2867459.

[53] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[54] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[55] Swami Sankaranarayanan, Arpit Jain, Rama Chellappa, and Ser Nam Lim. Regularizing deep networks using efficient layerwise adversarial training. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[56] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *arXiv preprint arXiv:1806.01768*, 2018.

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] Kalaivani Sundararajan and Damon L Woodard. Deep learning for biometrics: A survey. *ACM Computing Surveys (CSUR)*, 51(3):1–34, 2018.

[59] Saeid Asgari Taghanaki, Arkadeep Das, and Ghassan Hamarneh. Vulnerability analysis of chest x-ray image classification against adversarial attacks. In *Understanding and interpreting machine learning in medical image computing applications*, pages 87–94. Springer, 2018.

[60] Rohan Taori, Amog Kamsetty, Brenton Chu, and Nikita Vemuri. Targeted adversarial examples for black box audio systems. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 15–20. IEEE, 2019.

[61] Sunil Thulasidasan, Sushil Thapa, Sayera Dhaubhadel, Gopinath Chennupati, Tanmoy Bhattacharya, and Jeff Bilmes. A simple and effective baseline for out-of-distribution detection using abstention. 2020.

[62] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[63] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[64] www.alamy.com. Background or wallpaper with colorful geometric elements with random colors. `https://bit.ly/3gn6asf`, 2021.

[65] www.colourbox.com. Color fluidism — abstract random colors 16. `https://bit.ly/3z1nXwp`, 2021.

[66] www.pinterest.com. Random color mosaic tiles. abstract background, stock image. `https://bit.ly/3yNrkGN`, 2021.

[67] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwag, and Prateek Mittal. Patchguard: Provable defense against adversarial patches using masks on small receptive fields. *arXiv e-prints*, pages arXiv–2005, 2020.

[68] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1369–1378, 2017.

[69] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[70] Zhanyuan Zhang, Benson Yuan, Michael McCoyd, and David Wagner. Clipped bagnet: Defending against sticker attacks with clipped bag-of-features. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 55–61. IEEE, 2020.