Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Fridtjof Storm Flaate

Shining Light on the Black Box

Detection System for Adversarial Attacks on Skin Lesion Classifiers

Master's thesis in Communication Technology and Digital Security
Supervisor: Sule Yildirim Yayilgan
June 2023

NTNU
Norwegian University of
Science and Technology

Fridtjof Storm Flaate

Shining Light on the Black Box

Detection System for Adversarial Attacks on Skin Lesion Classifiers

Master's thesis in Communication Technology and Digital Security
Supervisor: Sule Yildirim Yayilgan
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Shining Light on the Black Box
Detection System for Adversarial Attacks on Skin Lesion Classifiers

Student: Fridtjof Storm Flaate

Problem description:

Artificial intelligence (AI) has the potential to assist doctors in improving various applications within the healthcare industry. With the ability to analyze large amounts of data and identify patterns, AI can support doctors in performing and improving medical diagnosis, drug discovery, personalized treatment, and disease management. AI can also help streamline administrative tasks, reduce costs, and increase efficiency in healthcare systems.

Over the past few years, there has been a significant shift in the field of image recognition with the introduction of Convolutional Neural Networks (CNNs). These advanced deep-learning algorithms have revolutionized the way we process and analyze images. In addition, CNNs have proven to be highly effective at identifying image features, allowing for more accurate object recognition and classification. CNN is beneficial in healthcare, where accurately diagnosing and classifying medical conditions with image recognition can significantly improve patient outcomes.

Despite this, the models used to assist in diagnosis have shown to be vulnerable to attacks in which images are deliberately perturbed. These perturbations are undetectable to the human eye and are mathematically generated to deceive doctors in diagnosing results. Previous research has identified patterns and correlations within the deep models. Particularly low-level kernels. While processing perturbed images, as presented in an earlier master thesis [Orv22]. In this thesis, we aim to investigate further the identified patterns and deep model parameters (the model's internal state) to explore the potential for developing an adversarial detection and mitigation model. By delving deeper into these patterns and parameters, we can gain a comprehensive understanding of how the model behaves in response to adversarial attacks and develop strategies to enhance its robustness and security.

Approved on: 2023-04-18
Main supervisor: Yayilgan, Sule Yildirim, NTNU IIK

Abstract

This Master's thesis addresses the critical cybersecurity challenges associated with implementing Convolutional Neural Networks (CNN) in early skin lesion detection. With the alarming rate of melanoma skin cancer diagnoses and its significant mortality rate, CNN-aided skin lesion classifiers present an innovative solution for early detection and improved patient outcomes. However, their susceptibility to fraudulent manipulation and adversarial attacks can lead to dangerous consequences, from financial exploitation to 'medical hacking'. Our research identifies a gap in the existing literature where a deep-dive exploration into the 'black box' of CNNs facilitates a new approach to detecting adversarial attacks.

Utilizing multiple CNN architectures, including Inception V3 and ResNet-18, trained on datasets such as ISIC 2018 and ISIC 2019. The thesis aims to create a detection model attached to the CNN skin lesion models to detect adversarial attacks. We found that certain CNN components, namely, feature maps (outputs of convolutional layers), activated feature maps (non-linear activation function applied to feature maps), and dense layer weights, are key to identifying adversarial attacks. The detection models we developed demonstrated broad generalizability across various adversarial attacks, including those they were not initially designed to detect. This suggests an advancement in tackling adversarial attacks in CNNs. Our technique is fully automated and suitable for real-time applications where latency for adversarial detection is an important factor.

Sammendrag

Denne masteroppgaven tar sikte på å adressere cybersikkerhetsutfordringene knyttet til implementeringen av dype nevrale nettverk i tidlig påvisning av føflekker og hudlesjoner. Dype nevrale nettverk er en innovativ løsning for å effektivisere klassifiseringen av føflekker og hudlesjoner. En nedside ved disse dype nevrale nettverkene er at de er høyt mottakelig for manipulasjon og angrep. Disse angrepene kan komme med fatale konsekvenser, fra økonomisk forsikrings utnyttelse til 'medisinsk hacking'. I denne masteroppgaven identifiserer vi et gap i den eksisterende litteraturen, der et dypdykk inn i den 'svarte boksen' til de dype nevrale nettverkene, kan brukes til å oppdage disse angrepene.

I oppgaven vil det bli brukt flere dyper nevrale nettverk arkitekturer, inkludert Inception V3 og ResNet-18, trent på forskjellige datasett som ISIC 2018 og ISIC 2019. Oppgaven tar sikte på å lage en deteksjonsmodell som er festet til disse dype nevral nettverkene, og som kan dermed detektere angrepene. Vi fant ut at visse komponenter, nemlig *feature maps*, aktiverte *feature maps* (ikke-lineær aktiveringsfunksjon brukt på *feature maps*), og *dense layers*, er nøkkelen til å identifisere disse angrepene. Deteksjonsmodellene vi utviklet viser bred generaliserbarhet på tvers av en rekke angrep, inkludert de de i utgangspunktet ikke var designet for å oppdage. Dette antyder et fremskritt i å takle angrep i dype nevrale nettverk. Teknikken vår er helautomatisert og egnet for sanntidsapplikasjoner, hvor ventetid for deteksjon er en viktig faktor.

Preface

The Master's thesis is part of my Master of Science in Communication Technology and Digital Security, the thesis is linked to the prelimenary project *Understanding Mitigation Against Adversarial Attacks on Skin Lesion Classifiers*, which was conducted in the autumn of 2022 [Fla22].

I am pleased to present this work, which represents the culmination of intensive research, rigorous analysis, and meticulous attention to detail. This journey has been one of constant learning, filled with challenges that tested my patience, resilience, and intellectual rigor. As with any significant undertaking, I could not have accomplished this alone.

Firstly, I would like to express gratitude to my advisor, Sule Yildirim Yayilgan. I appreciate the guidance and insightful discussions, which have been invaluable throughout this process. I am also grateful to Norwegian University of Science and Technology (NTNU) for providing the environment and resources that I needed to conduct my research.

Furthermore, I would like to thank Leaf Consulting and Start NTNU for their generous support and providing five good years at NTNU.

Contents

List of Figures

List of Tables

List of Acronyms

FGSM Fast Gradient Sign Method

PGD Projected Gradient Descent

BIM Basic Iterative Method

UAP Universal Adversarial Pertubations

CW Carlini & Wagner

I-FGSM Iterative Fast Gradient Sign Method

ART Adversarial Robustness Toolbox

ML Machine Learning

CNN Convolutional Neural Network

DNN Deep Neural Network

AI Artificial Intelligence

CE Cross Entropy Loss

DL Deep Learning

DNN Deep Neural Network

MSE Mean Square Error

ReLU Rectified Linear Unit

SVM Support Vector Machine

SGD Stochastic Gradient Descent

XGBoost Extreme Gradient Boosting

LR Logisitic Regression

SVM Support Vector Machines

NLP Natural Language Processing

TP True Positives

TN True Negatives

FP False Positives

FN False Negatives

NV Melanocytic Nevus

VASC Vascular Lesion

DF Dermatofibroma

BKL Benign Keratosis

MEL Melanoma

BCC Basal Cell Carcinoma

SCC Squamous Cell Carcinoma

AKIEC Acitinic Keratosis Intra Epithelial Carcinoma

ISIC International Skin Imaging Collaboration

API Application Programming Interface

GPU Graphics Processing Unit

Chapter 1
Introduction

The advent of the Internet has undisputedly marked one of the most monumental milestones in human history. When the Internet was first developed, it sparked a technological revolution that fuelled countless advancements and innovations. It was an open field of potential rapidly populated with various functionality and applications, transforming our lives and society in ways we could hardly have imagined. However, in our initial enthusiasm to explore this new frontier, one fundamental aspect was largely overlooked: security. It was only when the vulnerabilities of this cyberspace began to manifest, often with detrimental effects, that the importance of information security became apparent.

Fast-forward to the present day, and we find ourselves at the dawn of another era characterized by the ever-evolving realms of Artificial Intelligence (AI) and Machine Learning (ML). Much like the initial stages of the Internet, we are exploring new applications, finding novel uses, and innovating at an astonishing rate. However, unlike in the past, we are no longer naive about the potential risks. We have learned from our previous oversight and are now more aware of the importance of baking security into the system. With the knowledge and skills we have acquired, we are in a position to build secure applications.

1.1 Motivation

Annually, 160,000 individuals are diagnosed with Melanoma skin cancer, as the International Agency for Research on Cancer (IARC) reported. This form of skin cancer is notoriously challenging to combat in the later stages due to its resilience to traditional chemotherapeutic and radiotherapeutic approaches. Consequently, it is responsible for an alarming 75% of deaths related to skin cancer. However, the key to successful treatment is often early detection; it enhances the likelihood of success. Treatment could often be as straightforward as a simple skin lesion excision. Therefore, early detection is not just critical, but it is life-saving [FFRT21]. In light

of this, Deep Neural Network (DNN) has offered hope by enabling the development of skin lesion classifiers. These AI-powered classifiers could potentially revolutionize early melanoma detection, leading to enhanced survival rates and improved patient outcomes [BR21].

Despite these benefits, the security aspects associated with implementing these classifiers raise significant concerns. With high-value financial transactions involved, especially in health insurance, there is a considerable risk of these models being manipulated to exploit the system falsely. A fraudulent user might adjust the models to misidentify skin lesions as cancerous, leading to unjustifiable financial gains [FBI+19]. More alarmingly, there is a potential for these classifiers to be used for harmful purposes, such as 'medical hacking'. In this scenario, people with ill intentions could misuse these tools to cause harm to others, creating a new cyber threat. This risk could pose significant challenges to individuals and the broader healthcare sector, necessitating the establishment of robust safeguards when utilizing these technologies. This problem is particularly difficult due to the high sophistication of DNN and the complexity of ensuring their security. Creating secure DNN models necessitates a comprehensive understanding of possible attack vectors and the implementation of countermeasures, which can be challenging given the rapidly evolving nature of both AI technologies and cyber threats.

Despite this, we have discovered a potential gap in the research, where using the 'black box' itself can inform us if the skin lesion classifier is currently under an adversarial attack. We will extensively study potential attack vectors and develop corresponding countermeasures [Orv22; Fla22] to accomplish this.

Given the pressing need for early melanoma detection, the clear potential of DNN in this domain, and the increased importance of cybersecurity in healthcare, this work is timely and crucial. We believe our work can pave the way for more secure and efficient use of artificial intelligence in medical diagnostics, making it more trustworthy for all stakeholders, from patients to doctors to insurance companies.

1.2 Research questions

The first research question is central to this thesis as it seeks to ascertain the most crucial components of the CNN that can be effectively integrated as features for adversarial attack identification. Understanding these core elements will provide the foundational building blocks in designing the detection model, which can enhance security.

RQ1: *In the context of enhancing cybersecurity, which essential convolutional neural network components can be effectively integrated as features for the identification of ongoing adversarial attacks?*

Expanding on the findings from the initial research question, this thesis will focus on constructing a generalizable detection model. The success of this model will be assessed in terms of its generalizability across different adversarial attacks targeting skin lesion image classification models, taking us to the second research question.

RQ2: *Considering the strategic employment of key convolutional neural network components outlined in RQ1, to what extent can a detection model be developed, demonstrating generalizability across a diverse range of adversarial attacks targeting skin lesion image classification models?*

The final research question of this thesis acknowledges the practical limitations of computational resources. It aims to evaluate the resources needed to balance resource usage and effectiveness in adversarial attack identification.

RQ3: *What computational resources are necessary to balance resource use and security in the system for effective adversarial attack identification?*

In summary, this thesis proposes exploring the efficacy of CNN components for detecting adversarial attacks on skin lesion classification models. It hopes to contribute to cybersecurity by creating resource-efficient, generalized detection models.

1.3 Thesis Structure

1. **Introduction:** The first chapter provides the context and motivation behind the research, articulates the research questions that the thesis aims to answer, and summarizes the structure of the thesis.
2. **Background:** The second chapter delves into the foundational prerequisites to understand the thesis.
3. **Adversarial Attacks:** The third chapter introduces the concept of adversarial attacks, provides a high-level taxonomy of various adversarial attacks, provides examples of different adversarial attacks, and discusses their transferability.
4. **Related Work:** The fourth chapter presents an overview of prior studies and techniques in the field. Providing an explanation of each method and the following security performance of the method.
5. **Methodology:** The fifth chapter defines the methodology employed during the thesis. Additionally, it explains the rationale behind key decisions made throughout the process.
6. **Experiments and Results:** The sixth chapter presents the experimental design and results of various tests to understand the impact of adversarial attacks and evaluate the security performance of the provided solution.
7. **Discussion:** The final chapter compares the findings of this thesis with related work, addresses the research questions outlined in the first chapter, and discusses the limitations of the thesis and potential areas for future work.

This structure offers a logical flow of information, starting with the necessary background knowledge, proceeding through the specifics of the methodology and experimental results, and culminating in a comprehensive discussion of the findings.

Chapter 2
Background

This chapter is divided into three distinct sections. First, Section 2.1 will explain essential mathematical concepts and functions central to understanding the thesis. Secondly, Section 2.2 will delve into important machine learning concepts relevant to the thesis. Finally, Section 2.3 will provide the prerequisite for skin lesions and applying deep learning in healthcare.

2.1 Mathematical background

2.1.1 Signum Function

The signum function, also known as the sign function, is a mathematical function that returns the sign of a given number. Defined in Equation 2.1.

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \tag{2.1}$$

It is commonly used to define the direction of a vector. Given a vector $\hat{v} = (v_1, v_2, v_3, ..., v_n)$, we can obtain the direction of the vector by taking the signum function of the vector, $sign(\hat{v})$. The signum function can decouple the vector's magnitude from the vector's direction. Given the direction vector, we can then scale the vector using a scalar variable α, $\alpha \cdot sign(\hat{v})$.

2.1.2 Measuring the Magnitude of Vectors

Vectors are mathematical objects that describe quantities possessing both magnitude and direction. As such, the need for a systematic way to measure their magnitudes has led to different norms, essentially functions that assign a non-negative scalar value to a vector. This scalar value corresponds to the 'length' or 'size' of the vector.

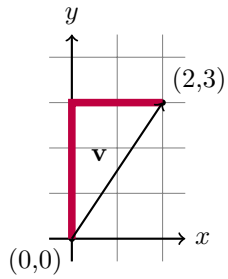


Figure 2.1: Visual representation of $L1$ norm of vector $\mathbf{x} = (2, 3)$, $\|\mathbf{x}\|_1 = |2| + |3| = 5$.

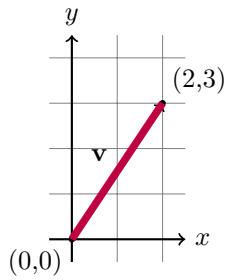


Figure 2.2: Visual representation of $L2$ norm of vector $\mathbf{x} = (2, 3)$, $\|\mathbf{x}\|_2 = \sqrt{2^2 + 3^2} = \sqrt{13}$.

$L1$ Norm

The $L1$ norm is the sum of the absolute values of all the scalar values in the vector, defined in Equation 2.2, visualized in Figure 2.1.

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{2.2}$$

$L2$ Norm

The $L2$ norm, also known as the Euclidean norm, is the square root of the sum of the squares of all the scalar values in the vector, defined in Equation 2.3, visualized in Figure 2.2.

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{2.3}$$

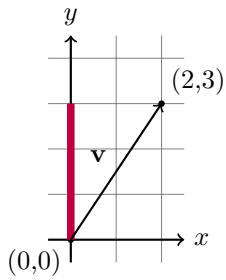


Figure 2.3: Visual representation of L_∞ norm of vector $\mathbf{x} = (2, 3)$, $\|\mathbf{x}\|_\infty = \max(|2|, |3|) = 3$.

L_∞ Norm

The L_∞ norm, also known as the maximum norm, is the maximum absolute value of all the scalar values in the vector, defined in Equation 2.4, visualized in Figure 2.3.

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i| \tag{2.4}$$

2.1.3 Gradient

In vector calculus, the gradient is a concept that helps to determine the direction and rate of the fastest increase for a given function f at a given non-zero point p.

We define the gradient function f as follows:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_i}, \ldots, \frac{\partial f}{\partial x_n} \right) \tag{2.5}$$

Where ∇f is the gradient of the function f, and $\frac{\partial f}{\partial x_i}$ represents the partial derivative of the function with respect to the variable x_i [AAL20].

2.2 Machine Learning Background

ML is a subset of AI that allows systems to learn and improve from experience without being explicitly programmed. ML is achieved using algorithms that iteratively learn from data. The more data the algorithms can access, the more they can learn and adapt their models.

2.2.1 Cost Function

A cost function, alternatively referred to as a loss function, is a mathematical expression that quantifies the difference between the predicted output and the true output. The cost function provides a mathematical construct to optimize in ML [AAL20]. However, paradoxically, it also supplies us with a metric we could potentially degrade.

Several cost functions exist, each being appropriate for different problems and particular instances of those problems. Common examples include the Mean Square Error (MSE) for regression problems, Cross Entropy Loss (CE) for classification tasks, and Hinge Loss for Support Vector Machines (SVM). In the context of this thesis, however, we will primarily focus on the CE function.

Cross Entropy Loss

CE is a cost function frequently employed in classification tasks in Deep Learning (DL), as well as adversarial attack within the realm of adversarial ML. It measures the dissimilarity between the predicted class probabilities and the true class probabilities.

For a given set of input data and corresponding true labels, the cross entropy loss is defined as:

$$\mathrm{CE}(y, \hat{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2.6}$$

Where n is the number of samples, y_i is the true value for the i-th sample, and $\hat{y}_i$ is the predicted value for the i-th sample.[MMZ23].

2.2.2 Supervised vs. Unsupervised Learning

Supervised learning is a type of ML that uses labeled data to learn a function that maps inputs to outputs. Given a training set of n data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $x_i \in X$ are the inputs and $y_i \in Y$ are the outputs, the goal is to learn a function $h : X \to Y$ so that $h(x)$ is a good predictor for the corresponding value of y. For a classification problem, y is a discrete value (class label).

In mathematical terms, supervised learning can be represented as:

$$h = \arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} J(y_i, f(x_i)) \tag{2.7}$$

Where $J(y_i, f(x_i))$ is a cost function (see Section 2.2.1), and $\mathcal{H}$ is a hypothesis space[1] [DFO20].

Unsupervised learning, on the other hand, deals with unlabeled data. The goal is to infer the natural structure within a set of data points. For example, given a set of n data points $x_1, x_2, ..., x_n$ where $x_i \in X$, the goal is to find patterns or relationships in X. Unsupervised learning includes clustering, dimensionality reduction, and density estimation.

2.2.3 Gradient Descent

Gradient Descent is an optimization algorithm commonly used in ML and DL to minimize a given loss function by iteratively updating the model's parameters. The algorithm calculates the gradient (see Section 2.1.3) of the cost function (see Section 2.2.1) with respect to the parameters and then moves the parameters in the direction of the negative gradient.

$$\theta_{i+1} = \theta_i - \alpha \nabla J(\theta_i) \tag{2.8}$$

Where θ represents the model parameters, α is the step size, also known as the learning rate, and ∇J signifies the gradient of the cost function. The gradient descent process repeats until convergence is achieved or a maximum number of iterations is reached. Through this procedure, gradient descent enables the discovery of optimal model parameters that minimize errors in predictions or classifications [AAL20]. Figure 2.4 illustrates the gradient descent process.

2.2.4 Confusion matrix

A confusion matrix, also known as an error matrix, is a tool used in ML to evaluate the performance of an algorithm. The confusion matrix shows the number of correct and incorrect predictions made by a classifier, broken down by each class. The confusion matrix is the basis for many other performance metrics, including accuracy, precision, and recall [FLW+20; GBV20]. Given the four values in a confusion matrix

[1]The hypothesis space $\mathcal{H}$ is the set of all the possible functions that can be chosen by a learning algorithm based on the given data [Blo11].

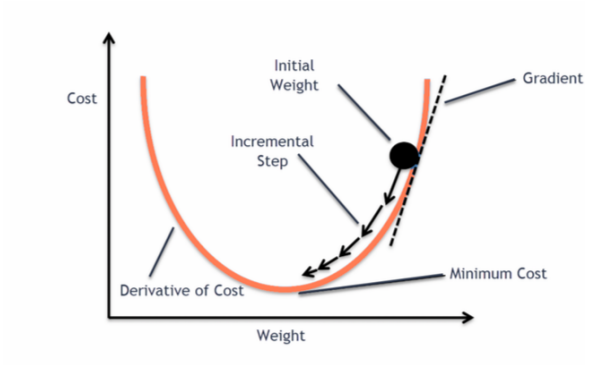


Figure 2.4: Visual representation of gradient descent.

of a binary classifier, the accuracy can be computed using the equation presented as Equation 2.9.

- **True Positives (TP)**: The number of positive instances correctly classified.
- **True Negatives (TN)**: The number of negative instances correctly classified.
- **False Positives (FP)**: The number of negative instances misclassified as positive.
- **False Negatives (FN)**: The number of positive instances misclassified as negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.9}$$

[GBV20]

2.2.5 Convolutional Neural Network

CNNs are a class of DL models that have achieved remarkable success in various computer vision tasks.

Convolution

The core of the CNN is the convolution, which refers to a mathematical operation involving two functions, f and g. This operation generates a third function $(f * g)$ that demonstrates how the shape of one function is influenced by the other. The term convolution pertains to both the resulting function and the computation process

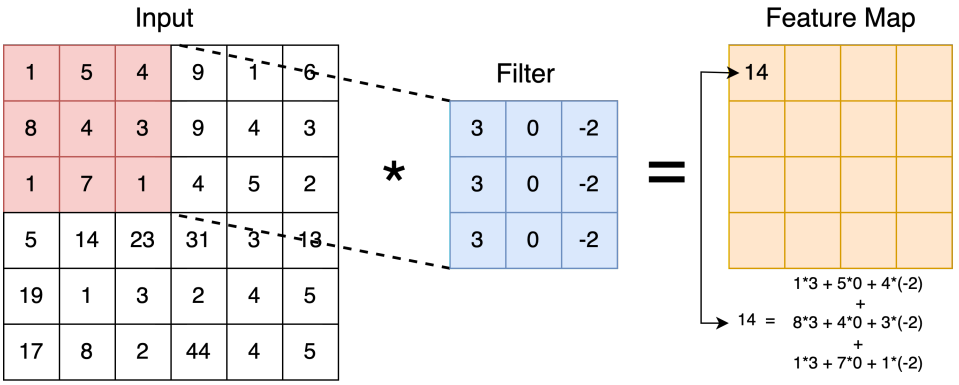


Figure 2.5: Visual representation of the convolution operation between two matrices. The process involves sliding a smaller matrix (filter) over the larger input matrix, computing element-wise multiplication between the overlapping elements, and then summing the results to produce a single value in the output matrix, also called the feature map.

[Con23]. Within a CNN model, function f and g are two matrices calculated as displayed in Figure 2.5.

Convolutional Neural Network Architecture

Convolutional Layer:

The convolutional layer is the primary building block of a CNN. The convolutional layers capture patterns or features in the input data. The first layers in the network typically capture low-level features, such as edges and textures. As we move deeper into the network, the layers progressively capture higher-level and more abstract features, such as corners, shapes, and parts of objects. Eventually, the final layers can capture the most abstract and complex features that help distinguish between different classes, see Figure 2.6. The output of a convolutional layer is called a feature map [AMA17].

Non-linear Activation Function:

After each convolution layer, a non-linear activation function is applied element-wise to the feature maps, as displayed in Figure 2.7. The activation function introduces non-linearity into the model, allowing it to learn complex, non-linear relationships between inputs and outputs. One of the most popular activation functions, the Rectified Linear Unit (ReLU), is illustrated in Equation 2.10.

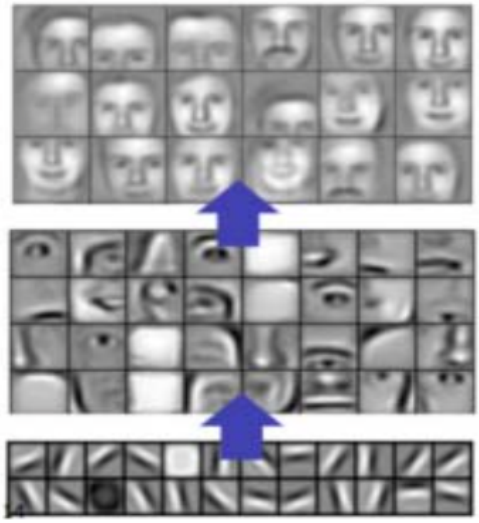


Figure 2.6: The figure illustrates the process of hierarchical feature learning in a CNN. As we move deeper into the network, the layers capture more abstract and complex features. The first layer detects simple features such as edges, while the second layer combines these features to detect more complex features like eyes, noses, eyebrows, and mouths. The third layer captures even higher-level features, such as faces [AMA17].

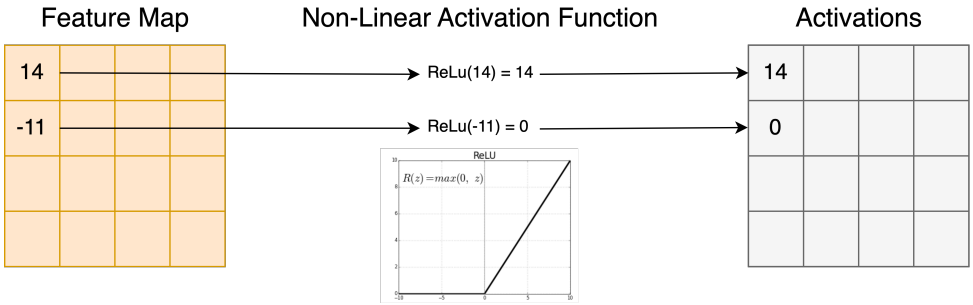


Figure 2.7: Visual representation of the application of a non-linear activation function to the feature map, specifically the application of the ReLU.

$$ReLu(x) = \max(0, x) \tag{2.10}$$

Pooling Layer:

Following the convolutional layers and non-linear activation function, a pooling layer is often added to reduce the spatial dimensions of the activated feature maps. This

downsampling operation reduces the computational complexity of the model and helps prevent overfitting. Standard pooling techniques include average pooling and max pooling, as displayed in Equation 2.12 and 2.13 respectively (based on Equation 2.11).

$$M = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}, \tag{2.11}$$

$$\text{AvgPooling}(M) = \begin{bmatrix} \frac{a+b+e+f}{4} & \frac{c+d+g+h}{4} \\ \frac{i+j+m+n}{4} & \frac{k+l+o+p}{4} \end{bmatrix} \tag{2.12}$$

$$\text{MaxPooling}(M) = \begin{bmatrix} \max(a,b,e,f) & \max(c,d,g,h) \\ \max(i,j,m,n) & \max(k,l,o,p) \end{bmatrix}, \tag{2.13}$$

Fully Connected Layer:

At the end of the CNN comes the fully connected layer, also called the dense layer. Before entering the dense layers, the high-dimensional output from the last convolutional layer is typically flattened into a one-dimensional vector. Each node in a dense layer is connected to every node in the previous layer, hence the term fully connected. The role of the dense layers is to interpret and make decisions based on the features identified by the convolutional layers. They function as a classifier on top of the features extracted by the convolutional layers.

2.2.6 Important definitions:

To keep this thesis clear and precise, we will explicitly define the following:

Feature maps

The matrix output from a convolutional layer **before** the application of a non-linear activation function.

Activations

The matrix output from a convolutional layer **after** the application of a non-linear activation function, also called activated feature maps.

2.3 Deep Learning in Health Care

The advent of DL has revolutionized numerous industries, including health care. In recent years, DL techniques, particularly CNN, have been increasingly applied to medical image analysis tasks, aiding in diagnosing and treating various conditions. One specific area where DL has shown great promise is in the classification of skin lesions, where timely identification is essential for the patient's survival, should the skin lesion turn out to be cancerous [HEA+22; YCD+16; MBG+21]. In this section, we will discuss the various types of skin lesions and their characteristics, as well as the role of deep learning in identifying and classifying these lesions to assist medical professionals in making more accurate and timely diagnoses.

2.3.1 Skin Lesions

Skin lesions are changes in the appearance of the skin that can be caused by various factors, including infections, genetics, environmental factors, or diseases. These skin lesions can vary from being harmless (benign) to cancerous (malignant). This thesis will discuss eight skin lesion categories with distinct features and characteristics.

Melanocytic Nevus (NV) - A mole-like lesion caused by an overgrowth of melanocytes, typically appearing as a spot or bump on the skin. They can be present at birth or develop later in life and are often harmless. However, in some cases, they can develop into melanoma [REGT15].

Vascular Lesion (VASC) - A lesion of the blood vessels, typically appearing as red or purple spots or stretches on the skin. Vascular lesions usually form during embryologic development and are benign (noncancerous) tumors that line the blood vessels [AA05].

Dermatofibroma (DF) - A benign skin lesion, typically appearing as a firm, dome-shaped bump on the skin and is often found on the lower legs. Treatment is typically not necessary [ZZB04].

Benign Keratosis (BKL) - A type of benign skin lesion that typically appears as a scaly, rough patch on the skin. BKL is usually caused by sun damage and is typically found on sun-exposed areas such as the face, neck, and hands. Since a BKL is benign, it is not mandatory to remove it [HV08].

Melanoma (MEL) - A type of skin cancer that arises from the uncontrolled growth of melanocytes, which produce the pigment melanin. MEL can develop in areas of skin that are exposed to the sun, such as the face, arms, and legs, but can also occur in areas that are not usually exposed to sunlight, such as the soles of the feet, under the nails, and in the mouth. If left untreated, MEL can spread to

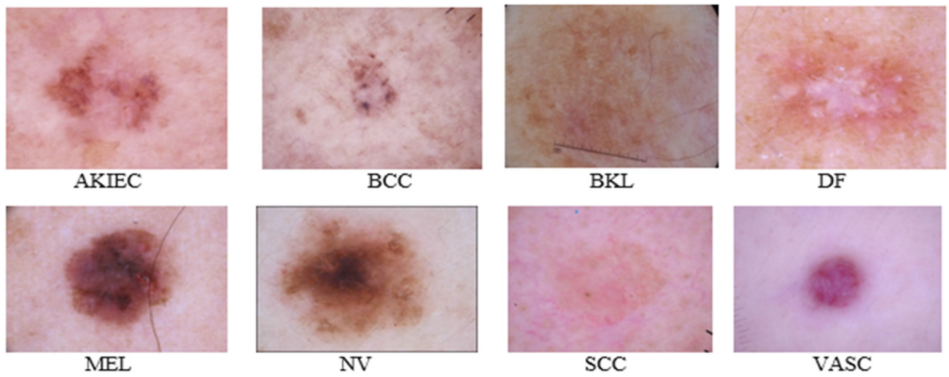


Figure 2.8: Visual representation of the eight skin lesions, illustrating their distinct appearances and characteristics, which can aid in the identification and understanding of various skin conditions [KHF20].

other body parts through the bloodstream, making it more challenging to treat and potentially deadly. Therefore, early diagnosis and treatment of MEL are crucial to improving outcomes [CCP+06].

Basal Cell Carcinoma (BCC) - A type of skin cancer that is the most common form of all skin cancers, typically appearing as a pearly or waxy bump. It is most common to find on the head and neck, where we find 80% of the cases. BCC is usually caused by exposure to ultraviolet radiation from the sun or other sources and is generally slow-growing and non-invasive. Therefore, it is rarely fatal, even though it is cancerous [RCR05].

Squamous Cell Carcinoma (SCC) is a type of skin cancer that can appear as a scaly, red patch on the skin. It is usually caused by exposure to ultraviolet radiation from the sun or other sources and can be found on sun-exposed areas such as the face, neck, and hands. SCC is typically slow-growing and can be treated with surgery, radiation therapy, or topical medications. Regular monitoring of SCC is important to reduce the risk of progression to a more advanced stage [Mar96].

Acitinic Keratosis Intra Epithelial Carcinoma (AKIEC) - is a precancerous skin condition characterized by rough, scaly patches or lesions that can develop into SCC if left untreated. These lesions are typically found on sun-exposed areas of the skin, such as the face, scalp, ears, and hands. Therefore, regular monitoring and sun protection measures are important to reduce the risk of developing AKIEC [Fer17].

2.3.2 Assisting Medical Professionals with Deep Learning

DL based skin lesion classification has the potential to significantly aid medical professionals in making more accurate and timely diagnoses. By providing additional support, these models can enhance the decision-making process for dermatologists and general practitioners, reducing the risk of misdiagnosis and improving patient outcomes. Some of the key benefits of incorporating DL models in the diagnosis of skin lesions include:

Efficient Screening: Given the high prevalence of skin lesions, it can be potentially challenging for medical professionals to examine and evaluate every case on time. DL models can help prioritize cases that require immediate attention, ensuring that patients with potentially malignant lesions receive prompt evaluation and treatment [HJW+14].

Accessibility and early detection: DL models for skin lesion classification can be integrated into mobile applications, making them accessible to individuals who may not have immediate access to dermatologists or other specialists. In addition, by empowering patients to monitor their skin conditions and seek medical advice when needed, DL models can facilitate early detection and intervention, potentially saving lives [FFRT21].

Second Opinion: In the absence of a second opinion, diagnostic accuracy in the United States is estimated to be approximately 83.2%. However, the integration of a second opinion in every diagnostic process presents a substantial improvement, elevating the diagnostic accuracy rate to 87.4%. Utilization of DL models represents an innovative and cost-efficient methodology for offering second opinions for doctors, leading to more accurate patient care and treatment outcomes [TTT+21].

It is essential to note that DL models should not be considered a replacement for the expertise of medical professionals. Instead, they should be viewed as a complementary tool that can augment the diagnostic process, providing additional insights and support to clinicians. By harnessing the power of DL, we can work towards a future where skin lesions are detected and treated more effectively, leading to improved patient outcomes and a reduced burden on healthcare systems.

However, DL models for skin lesion classification are not without challenges. For example, adversarial attacks, where small perturbations are intentionally added to input images to fool the model, can lead to incorrect classifications. Therefore, understanding and detecting these attacks is crucial for ensuring the reliability and robustness of skin lesion classifiers in clinical applications.

Chapter 3
Adversarial attacks

This chapter will discuss different adversarial attacks and details about the implementations of these attacks. First, Section 3.1 will introduce the concept. Section 3.2 will provide a high-level taxonomy of adversarial attacks. Then, Section 3.3 describes different adversarial attack implementations. Lastly, Section 3.4 describes the concept of transferability of adversarial attacks.

3.1 Concept introduction

Classical ML relies on extracting the right features for model training, requiring significant domain expertise to identify which aspects of the data contribute to the desired outcome. In classical image recognition, hard-coded edge detectors were commonly used to detect objects, and these were usually hard to generalize and specific to the task at hand [Bie87; Lin94; ZT+98]. With the introduction of CNN, feature engineering became integrated with DL, allowing the model to choose features independently [Fuk80; LBBH98]. The integrated feature engineering significantly improved image recognition. However, it also outsources the feature selection process, introducing a vulnerability to adversarial attacks [SZS+13; GSS14].

Adversarial attacks refer to deliberate modifications of input data, such as speech, images, or text, intending to cause a ML model to make incorrect predictions [ECC22; MR22; TV16]. Indeed, adversarial attacks become apparent when considering the seemingly paradoxical behavior of high-performance CNN under minor perturbations of the input data. For example, as illustrated in Figure 3.1, an image of a pig, correctly identified by a CNN, can be subtly manipulated – with changes imperceptible to the human eye – causing the same model to misclassify it as an airliner with high confidence.

The adversarial attacks can fool models into making wrong decisions, even when the input data appears unchanged to a human observer [NYC15]. The goal of adversarial attacks can vary widely depending on the context. In some cases, the

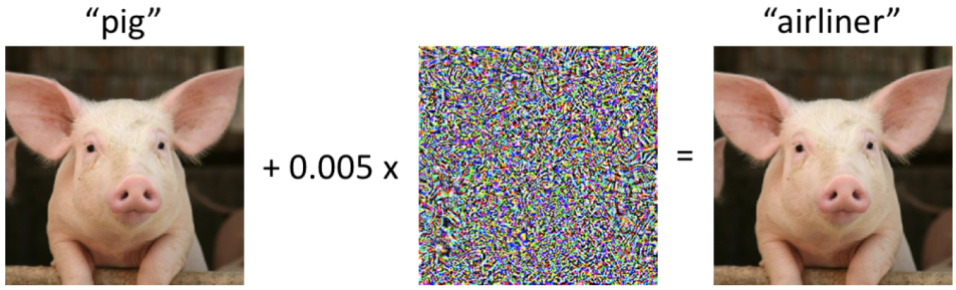


Figure 3.1: Illustration of an adversarial attack on an image classifier. The original image (left) is correctly identified by the CNN as a pig. After adding a subtle perturbation noise (center), the image (right) is mistakenly classified as an airliner by the same CNN. The perturbation exemplifies how minor changes can exploit vulnerabilities in the model, leading to misclassification with high confidence.

goal may be to evade detection or bypass security measures, while in others, the goal may be to manipulate or influence the decisions made by a ML system.

3.1.1 Adversarial Attack Scenario

In the given scenario, the individual, unaware of being afflicted with a skin cancer called MEL, uses a mobile application designed for skin lesion analysis and classification. For further information about MEL, refer to Section 2.3.1. While capturing an image of the skin lesion on her arm, a malicious entity, having successfully compromised the security of the camera, executes an adversarial attack, subtly distorting the image data. There are no discernible alterations to the patient's naked eye; thus, the manipulated image is subsequently fed into a CNN engineered for the specific task of skin lesion classification. Due to the adversarial perturbation, the classifier erroneously identifies the lesion as BKL, a benign type of skin lesion, outlined in Section 2.3.1. Subsequently, the individual, reassured by the benign classification, does not seek further professional medical evaluation, potentially leading to severe health implications, up to and including mortality, due to the undiagnosed MEL.

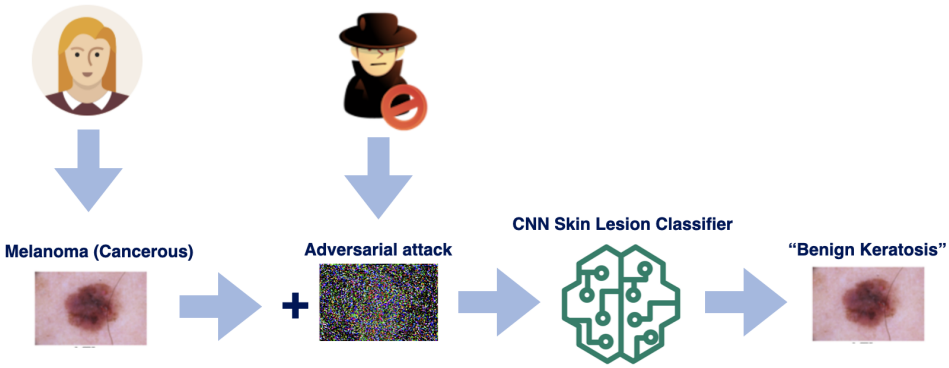


Figure 3.2: Visual illustration of an adversarial attack on a skin lesion classifier, leading to misclassification of MEL as BKL.

3.2 High-Level Taxonomy of Adversarial Attacks

This section presents a high-level taxonomy of adversarial attacks to organize attacks and familiarize readers with key terminology. Adversarial attacks can be broadly classified along four different dimensions: 1) stages of attack implementation, 2) goal of the attacker, 3) attackers capabilities, and 4) knowledge of attacker [OV23; SBG20].

3.2.1 Stage of Attack Implementation

The stage of attack refers to when the attack is implemented and performed in the ML pipeline. Attacks can be implemented at different stages of the pipeline, including:

- **During training**: Implemented in the training stage of the model, the attacker can tamper with the training data or interfere with the training procedure, negatively impacting the functionality of the model.
- **During testing**: Implemented during the evaluation stage of the model, the attacker can tamper with test data or alter the model-generated outcomes, leading to a distorted perception of the performance of the model.
- **After deployment**: These attacks are implemented after the model has been deployed. The attacker may manipulate the input data to insert adversarial examples, resulting in erroneous outputs from the model [OV23; SBG20].

3.2.2 Goal of the Attacker

The goal of an attacker can be categorized along three primary dimensions: 1) availability, 2) integrity, and 3) confidentiality, also called the CIA-triad [SC14].

- **Availability:** An attack on the availability aims to decrement the overall performance of the model, making it unusable or unreliable. In an untargeted availability attack, the attacker wants to make the model reject all legitimate samples. On the other hand, in a targeted attack, the attacker wants to make the model reject specific samples. For instance, it wants a self-driving car not to detect a stop sign.
- **Integrity:** An attack on the integrity is a specific attack toward misclassification of the output of the model. For example, the attacker wants to impersonate another person in a biometric system or trick a self-driving car into detecting a traffic light as a 50 km/h speed sign.

– **Confidentiality:** An attack on confidentiality aims to compromise a model by exposing its components, parameters, or dataset information. For example, an attacker may attempt to steal parameters from a cloud model through continuous querying. The attacker would repeatedly submit queries to the service to gradually extract information about the parameters of the model. By analyzing the model outputs in response to these queries, the attacker may be able to reverse engineer information about the parameters of the model [OV23; SBG20].

3.2.3 Attackers Capabilities

Several competencies could be harnessed by an adversary to manipulate a system. We describe six such capabilities below:

– **Training data:** An attacker could have the capabilities to control or modify a part of the training dataset to perform or set up an attack.

– **Model parameters:** An attacker might be able to disrupt or change model parameters, for instance, by inserting malicious triggers into the model parameters that react to a particular input.

– **Testing data:** An attacker could use the capabilities of controlling parts of the testing data to give a skewed performance evaluation of the model.

– **Labels:** An attacker could control or modify some of the labels within the dataset to manipulate the learning process of the model. The attacker could misguide the model during training by changing these labels, leading to incorrect or biased predictions.

– **Source code:** An attacker could manipulate the source code of an underlying open-source library that performs mathematical calculations.

– **Access to query:** To what extent can the attacker query the model with inputs, and if there are some restrictions on the number of queries an attacker can perform [OV23].

3.2.4 Attackers Knowledge

The last category for which we can classify an adversarial attack is what knowledge an attacker has of the system. The knowledge can be categorized along two dimensions: 1) white-box attacks and 2) black-box attacks.

- **Black-box attacks:** The attacker has limited knowledge of the target model, often only knowing its inputs and outputs. This interaction mirrors a cloud-based ML model, where the extent of engagement permitted with the model is limited to input-output interaction.
- **White-box attacks:** In this scenario, the attacker has complete knowledge of the target model, including its architecture, parameters, and training data. White-box attack is more effective than black-box attacks, as the adversary can use this knowledge to craft targeted attacks that exploit the specific vulnerabilities of the model [OV23; SBG20].

3.3 Examples of adversarial attacks

This section will discuss some of the most well-known adversarial attacks in the context of deep learning models. We will provide an overview of each attack and explain their underlying principles. Table 3.1 presents a summary of the symbols used throughout this section, which will help understand the mathematical formulations of the attacks.

Symbol	Description
θ	Parameters of a model
X	Benign model input
X^{adv}	Pertubated model input constructed from a specific input X
y^{true}	True label associated with a specific input X
y^{adv}	Target label for a specific adversarial attack
$J(\theta, X, y)$	Cost function (see Section 2.2.1).
$\nabla_x f$	Gradient of function f with respect to input X (see Section 2.1.3 and 2.2.3)
$sign(\hat{v})$	Signum function of vector $\hat{v}$ (see Section 2.1.1)
steps	Maximum amount of steps
α	Step size

Table 3.1: Summary of used symbols and their description.

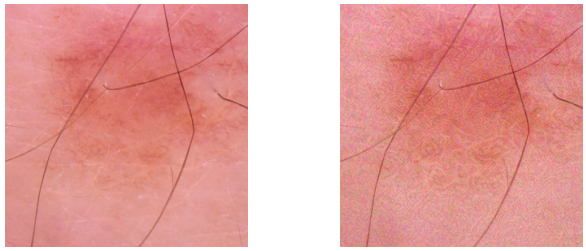
3.3.1 Fast Gradient Sign Method

The FGSM is an adversarial attack that leverages gradient information to craft malicious inputs. By taking the gradient of the cost function ($\nabla_x J(\theta, X, y^{true})$), we obtain the direction towards the maximum of the cost function for parameters θ. Instead of updating the parameters of the model by stepping in the direction of the negative gradient as in gradient descent, see Section 2.8. FGSM will update the parameters of the model by stepping in the direction of the positive gradient; in other words, FGSM will take the step that maximizes the cost function.

To create an attack that can be scaled according to a scalar value ϵ, take the signum function of the gradient (see Section 2.1.1) and multiply it by ϵ. This gives a gradient that has length ϵ and which has the direction towards the maximum of the cost function [GSS14]. Furthermore, we can apply this perturbation to the input image X and obtain the perturbated image X^{adv} [KGB16]. In summary, FGSM can be stated by Equation 3.1, where the cost function is the CE function introduced in Section 2.2.1.

$$X^{adv} = X + \epsilon \cdot \text{sign}(\nabla_x J(\theta, X, y^{true})) \tag{3.1}$$

The symbols used are explained in Table 3.1. The effectiveness of the FGSM adversarial attack is visually illustrated in Figure 3.3, where a benign skin lesion image (Figure 3.3a) is perturbed to mislead the model. The perturbed image (Figure 3.3b), which appears almost identical to the human eye, can cause a well-trained model to misclassify the lesion, highlighting the vulnerability of neural networks to such adversarial attacks.



(a) Before attack **(b)** After attack

Figure 3.3: FGSM adversarial attack on a single skin lesion image.

FGSM in the Taxonomy of Adversarial Attacks

The FGSM adversarial attack is implemented after deployment of ML model, and it aims to attack the integrity of the model. The attacker requires the capabilities to have access to the model parameters, access to query, and a single input they can perform the adversarial example on. Furthermore, the attacker needs to compute the gradient of the cost function with respect to the input, which requires knowledge of the model's internal workings, classifying it as a white-box adversarial attack [MGvDN21]. FGSM, as described earlier, is a non-targeted attack. This implies that the perturbation is designed to maximize the cost function for the true label rather than minimizing the cost for a particular alternative target [XCS19].

3.3.2 One-step Target Class Method

One-step Target Class Method is the targeted implementation of the FGSM adversarial attack. In this attack, we first define a target label y^{adv}. Then, given this target label, the attack uses gradient descent to take one step toward minimizing the cost function for the incorrect class. Gradient descent towards y^{adv} will increase the probability that the output of the model prediction will be the specific target class label [KGB16]. In summary, the one-step target class method can be described by Equation 3.2, where the cost function is the CE function introduced in Section 2.2.1.

$$X^{adv} = X - \epsilon \cdot \text{sign}(\nabla_x J(\theta, X, y^{adv})) \tag{3.2}$$

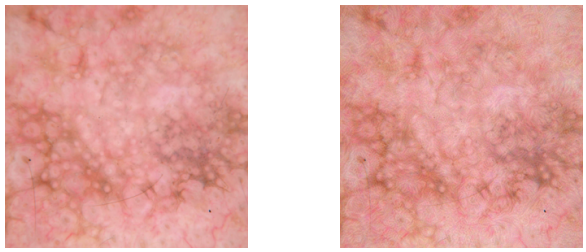
symbols are explained in Table 3.1.

One-step Target Class Method in the Taxonomy of Adversarial Attacks

For classifying One-step Target Class Method adversarial attack following the high-level taxonomy described in Section 3.2, it is equal to the FGSM attack described in Section 3.3.1 [MGvDN21; KGB16].

3.3.3 Iterative Fast Gradient Sign Method

I-FGSM also called Basic Iterative Method (BIM), is an iterative application of the FGSM adversarial attack, using a small step size. In other words, we want to perform FGSM over multiple iterations, gradually generating an undetectable adversarial example. At each iteration, we calculate the gradient of the cost function with respect to the current perturbed input and use this gradient to generate a new perturbation. The perturbations are added to the original input X and clipped to ensure that the pixel values remain within an acceptable range. The process repeats until the desired level of adversarial perturbation is reached [KGB16]. The pseudocode implementation is displayed in Algorithm 3.1, where the cost function is the CE function introduced in Section 2.2.1. The effectiveness of the I-FGSM adversarial attack is visually illustrated in Figure 3.4



(a) Before attack **(b)** After attack

Figure 3.4: I-FGSM adversarial attack on a single skin lesion image.

I-FGSM in the Taxonomy of Adversarial Attacks

Concerning the stage of attack implementation, I-FGSM is performed after the deployment of the model. The goal of the adversarial attack is to degrade the integrity of the model output. It requires the capabilities to have query access, model parameters, and single or multiple inputs they can perform the adversarial attack on. I-FGSM is also a white-box adversarial attack, requiring knowledge of the model's internal workings to compute the gradients [MGvDN21]. I-FGSM often exhibits superior effectiveness compared to FGSM and the One-step Target Class Method. This superiority is primarily due to its ability to generate highly specialized

adversarial perturbations due to its iterative application of the gradient. However, it requires more computational time and resources [XCS19].

Algorithm 3.1 I-FGSM Algorithm

Require: X, θ, y^{true}, α, *steps*, maximum perturbation ϵ,
1: $J(\theta) \leftarrow -\sum_{i=1}^{n} y_i^{true} log(\hat{y}_i)$ ▷ Initialize cost function
2: $X^{adv} \leftarrow X$ ▷ Initialize adversarial input
3: **for** $i = 1$ to *steps* **do**
4: $\nabla_x J(X^{adv}, y^{true}, \theta)$ ▷ Compute gradient
5: $X^{adv} \leftarrow X + \alpha \cdot \text{sign}(\nabla_x J(X^{adv}, y^{true}, \theta))$
6: $X^{adv} \leftarrow \text{clip}(X^{adv}, X - \epsilon, X + \epsilon)$ ▷ Ensure ϵ perturbation size
7: **end for**
8: **return** adversarial example X^{adv}

3.3.4 Universal Adversarial Perturbation

Universal Adversarial Pertubations (UAP) is an image-agnostic perturbation. Unlike the FGSM and I-FGSM, UAP does not generate a perturbation for a single image at a time. UAP solves an optimization problem over a given dataset to generate a universal perturbation. Despite being created based on a specific dataset, this perturbation can be applied universally to new images that were not part of the original dataset used during its training [MFFF17].

UAP in the Taxonomy of Adversarial Attacks

UAP represents an adversarial attack strategy initiated post-deployment, intending to compromise the integrity of the machine learning model. The adversary capabilities required for UAP include access to query and the ability to access training or testing data along with their associated labels. The dependency on access to model parameters is contingent on the specific implementation of UAP. The conventional UAP method operates as a white-box adversarial attack, necessitating access to the model's parameters. Conversely, the unconventional variant pioneered by the Takemoto lab circumvents the need for parameter access, qualifying as a black-box adversarial attack [KT22]. Consequently, UAP showcases the flexibility to function as a black-box or a white-box adversarial attack. Moreover, UAP can be distinguished by two distinct modes of implementation. The default UAP strategy is identified as a non-targeted adversarial attack. However, an alternative approach enables the deployment of UAP as a targeted adversarial attack [HMT21].

3.3.5 DeepFool

DeepFool is an iterative adversarial attack designed to find the minimum amount of perturbation necessary to cause a model to misclassify a given input. In essence,

DeepFool linearizes the decision boundary of the target model around the input and then finds the nearest point on the decision boundary. This process is repeated until the perturbed input crosses the decision boundary, indicating a misclassification. DeepFool starts by taking a benign input X and then iteratively adjusts this input, moving it closer to the classifier's decision boundary. The adjustment at each iteration is performed along the direction of the gradient of the classifier's output with respect to the input. The procedure stops when the adjusted input crosses the decision boundary and the classifier misclassifies the input. The pseudocode implementation is displayed in Algorithm 3.2, where the classifier f is used [MFF16].

Algorithm 3.2 DeepFool Algorithm

Require: X, θ, classifier function f, *steps*
1: $X_0 \leftarrow X$ ▷ Initialize adversarial input
2: **for** $i = 1$ to *steps* **do**
3: $\delta_i \leftarrow \frac{f(X_i,\theta)}{||\nabla_x f(X_i,\theta)||_2^2} \cdot \nabla_x f(X_i,\theta)$ ▷ Compute perturbation
4: $X_{i+1} \leftarrow X_i + \delta$
5: **end for**
6: **return** $X^{adv} \leftarrow \sum_{i=0}^{steps} \delta_i$

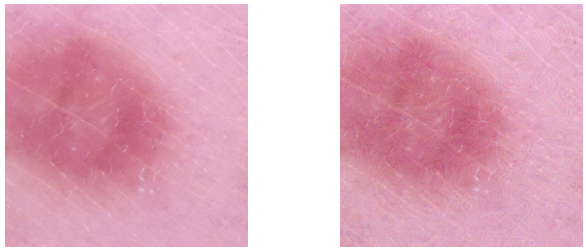
DeepFool in the Taxonomy of Adversarial Attacks

DeepFool is an adversarial attack that is typically executed after the deployment of the machine learning model. Its objective is to compromise the integrity of the model by inducing a misclassification. The adversary requires access to the model parameters and query access to execute DeepFool. DeepFool requires knowledge of the gradient of the classifier output with respect to the input, marking it as a white-box adversarial attack [MGvDN21]. DeepFool is designed as a non-targeted adversarial attack, as its goal is to find the smallest perturbation that causes a misclassification, regardless of the class to which the perturbed input is assigned. It should be noted, however, that DeepFool can be modified to create targeted adversarial attacks, but the original formulation is non-targeted [MFF16].

3.3.6 Projected Gradient Descent

The PGD adversarial algorithm, proposed by Madry et al. in 2017, is an iterative optimization-based method for generating adversarial examples [MMS+17]. The PGD attack is quite similar to the I-FGSM attack, but some key differences exist. First, the PGD attack bases it is starting input on a uniformly random point within epsilon distance away from the starting input point. Secondly, on each iteration, PGD will add the sign of the gradient with respect to the cost function to the last adversarial example. In I-FGSM, each iteration adds the sign of the gradient to the

original input image. The effectiveness of the PGD adversarial attack is visually illustrated in Figure 3.5



(a) Before attack **(b)** After attack

Figure 3.5: PGD adversarial attack on a single skin lesion image.

PGD in the Taxonomy of Adversarial Attacks

The PGD adversarial attack is implemented after the model is deployed and aims to damage the integrity of the model. The PGD attack is a white-box attack that requires the adversary to have complete knowledge of the target model. In addition, the adversary also needs the capability to access the model's parameters, query access, and access to the input data on which the adversary will perform the attack. The general pseudocode outline of the PGD adversarial attack is provided in Algorithm 3.3.

Algorithm 3.3 Projected Gradient Descent Algorithm

Require: X, θ, y^{true}, α, *steps*, maximum perturbation ϵ,

$J(\theta) \leftarrow -\sum_{i=1}^{n} y_i^{true} log(\hat{y}_i)$ ▷ Initialize cost function

$X^{adv} \leftarrow X + U(-\epsilon, \epsilon)$ ▷ Initialize at uniform random point

$X^{adv} \leftarrow \text{clip}(X^{adv}, X - \epsilon, X + \epsilon)$ ▷ Ensure ϵ pertubation size

for $i = 1$ to *steps* **do**

  $\nabla_x J(X^{adv}, y^{true}, \theta)$ ▷ Compute gradient of loss

  $X^{adv} \leftarrow X^{adv} + \alpha \cdot \text{sign}(\nabla_x J(X^{adv}, y^{true}, \theta))$

  $X^{adv} \leftarrow \text{clip}(X^{adv}, X - \epsilon, X + \epsilon)$ ▷ Ensure ϵ pertubation size

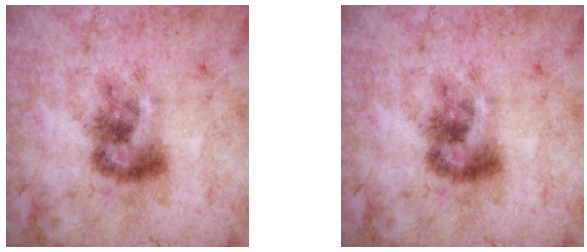
end for

return adversarial example X^{adv}

3.3.7 Carlini & Wagner

CW adversarial attack algorithm, proposed by Nicholas Carlini and David Wagner in 2017, is an optimization-based method for generating adversarial examples. The distinctive feature of CW attack, as compared to both the I-FGSM and PGD, lies in its optimization process. CW uniquely employs two cost functions, one for the misclassification cost and one for the perturbation cost. This optimization is one of

the defining characteristics that differentiate CW from both I-FGSM and PGD. The CW attack is highly effective in finding adversarial examples with small perturbations [CW17]. The effectiveness of the CW adversarial attack is visually illustrated in Figure 3.6



(a) Before attack **(b)** After attack

Figure 3.6: CW adversarial attack on a single skin lesion image.

CW in the Taxonomy of Adversarial Attacks

The CW adversarial attack is implemented after deployment and targets the integrity of a model system. To execute this attack, the adversary needs the capability to access model parameters, query access, and the input data on which the adversary will perform the attack. The CW attack is a white-box attack, as the adversary requires complete knowledge of the target model to perform the attack. The general outline of the CW adversarial attack is defined in Algorithm 3.4.

Algorithm 3.4 Carlini & Wagner Algorithm

Require: X, θ, y^{true}, *steps*, regularization parameter c

Define misclassification cost $J_{misclass}$ and perturbation cost $J_{perturb}$
Initialize perturbation variable w in tanh space.
Generate adversarial example: $X^{adv} \leftarrow x + \text{transform}(w)$
for $i = 1$ to *steps* **do**
    Optimize w to minimize $c \cdot J_{misclass}(X^{adv}, y^{true}) + J_{perturb}(X, X^{adv})$
    Update X^{adv} using the new w: $X^{adv} \leftarrow x + \text{transform}(w)$
end for
return adversarial example X^{adv}

3.4 Transferability of adversarial attacks

Transferability of adversarial attacks refers to the ability of adversarial examples crafted for one model to successfully mislead another model, even if the target model has a different architecture or is trained on a different dataset. The transferability property enables an attacker to craft adversarial examples on a substitute model and subsequently use the adversarial attacks to attack a victim model without the capabilities or knowledge which would constitute a white-box adversarial attack, see Section 3.2. Instead, the adversary will only need access to the input and output of the model. This phenomenon has been observed in various DNN models, and raises concerns about the security of ML systems in real-world applications [PMG16; PMSW16].

In the paper by Papernot, McDaniel, and Goodfellow (2016), titled "Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples", they demonstrated the effectiveness of the transferability attacks on two commercial ML classification systems from Amazon and Google. To quantify the effectiveness of the transferability of adversarial attacks, they defined the misclassification rate. The misclassification rate is calculated by dividing the number of misclassified adversarial examples by the total number generated. A higher misclassification rate indicates a more effective transferability attack. For the two commercial ML classification systems from Amazon and Google, they managed a 96.19% and 88.94% misclassification rate, respectively [PMG16].

Chapter 4
Related work

In this chapter, we present a comprehensive overview of the current state-of-the-art methods employed for defending against adversarial attacks in the field of adversarial machine learning. We elaborate upon the specific implementation of these defenses, delineating their operational methodologies and algorithmic complexities. Further, we describe the corresponding security performance associated with each defense, providing quantifiable measures of their efficacy and robustness against adversarial perturbations.

4.1 Gradient Masking

Many adversarial attacks aimed at fooling a DNN will use the gradients of the model outputs with respect to its inputs. An adversary can reverse these gradients and reapply them to the input to fool the model as observed in Section 3.3; this is the basis for the FGSM attack, the One-step Target Class Method, and the I-FGSM attack. Gradient masking is a defense strategy that aims to make these attacks less effective by removing the possibility of the adversary utilizing the gradients [PMSW16; PMG+17; TKP+17].

4.1.1 Method implementation

Gradient masking techniques are employed during the learning phase of the model. The technique incorporates non-differentiable functions at certain points in the model, obstructing the computation of meaningful gradients. One such operation commonly used in this context is quantization. Quantization involves converting continuous values into a finite set of discrete values. By creating discontinuities in these values, the function becomes non-differentiable at these points, obfuscating the gradient and limiting an adversary's ability to calculate meaningful gradients. As we can observe in Figure 4.1, the gradient of the model's output with respect to its input X is zero, meaning that the slope of the tangent to the function h at point x is equal to zero. In a situation where an adversary computes the gradient h at point x, they would be

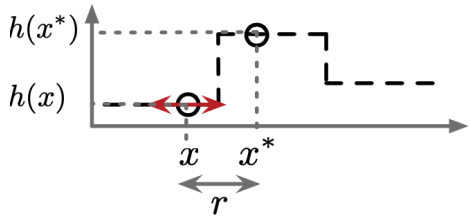


Figure 4.1: Representation of the defended model which uses gradient masking, x is model input, $h(x)$ is model output and x^* is model input which will give a missclassified model output $h(x^*)$ [PMSW16].

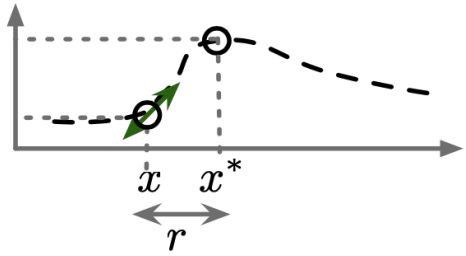


Figure 4.2: Representation of the substitute model, which creates the adversarial examples for the defended model. Point x is the original model input, and x^* is the model input, giving a misclassified model output for the substitute model [PMSW16].

unable to leverage it to find the direction towards x^*, leading to the model output $h(x^*)$, which is the misclassified result [PMSW16].

4.1.2 Security performance

Gradient masking is not an effective defense strategy due to the transferability property described in Section 3.4. Adversaries can bypass gradient masking by crafting adversarial examples using a substitute model, see Figure 4.2. The defense mechanism does not impact the substitute model, and the adversary has the necessary gradients to generate adversarial examples. Interestingly, even under black-box conditions where the adversary lacks complete knowledge about the model, they can still successfully bypass gradient masking techniques [PMG+17].

4.2 Input transformations

One effective method of protecting a DNN from adversarial attacks involves applying some transformation to the original image to create a new input that is more robust to adversarial attacks. By doing so, the model becomes more resistant to perturbations in the input, making it harder for attackers to craft adversarial examples that can fool the model. This approach is effective against various types of attacks. In both instances below, we assess the security performance of the adversarial attacks by employing the classification accuracy metric. The baseline reference classification accuracy is based on benign images that were initially accurately classified by the trained models [DSC+17].

4.2.1 Method Implementations

JPEG compression

In their work, Das, Nilaksh, et al. proposed a defense method based on input transformation. The proposed method comprises a preprocessing step that employs JPEG compression to mitigate the effects of adversarial perturbations before the input undergoes analysis by the classification model. JPEG compression can reduce the impact of small adversarial changes while keeping a decent visual quality, even though it might create block-like patterns due to the compression process. The JPEG algorithm, which is inherently lossy, results in the diminution of high-frequency details in the image. This property aids in lessening the effect of adversarial perturbations [DSC+17].

Flip + WebP compression

Yin, Zhaoxia, et al. present a new input transformation-based defense approach. The input transformation consists of a preprocessing module, which incorporates two low-level image transformations, WebP compression, and a flip operation, to counteract the adversarial perturbations before the classification model processes the input. WebP compression efficiently mitigates minor adversarial interferences while preserving a higher visual quality than JPEG compression, mainly due to the lack of block-like artifacts often associated with JPEG. The flip operation, which involves mirroring the image along one axis, disrupts the distinctive structure of adversarial perturbations, thereby augmenting the resilience of the defense mechanism. The step-by-step process is visualized in Figure 4.3 [YWW+20].

4.2.2 Security performance

The input transformation as a defense method against adversarial attacks was performed on three different adversarial attacks, I-FGSM (see Section 3.3.3), DeepFool

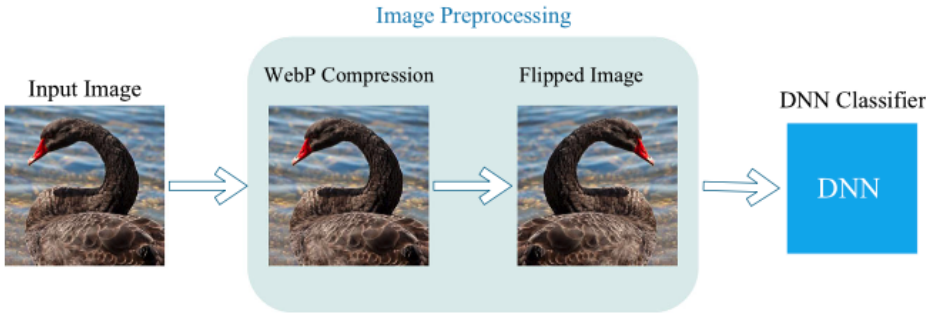


Figure 4.3: Illustration of the two-step input transformation process - applying WebP compression and flip operation on the image before sending it to the classifier [YWW+20].

(see Section 3.3.5), and CW (see Section 3.3.7). The results are in Table 4.1.

Security Performance of Input Transformations				
Evaluation Metric: Classification Accuracy				
Defense	**Benign Images**	**I-FGSM**	**DeepFool**	**CW**
No Defense	100%	9.68%	2.58%	0%
JPEG	93.14%	71.87%	79.29%	80.10%
WebP + Flip	95.84%	80.12%	85.23%	86.59%

Table 4.1: Security performance of two input transformation methods, on white-box adversarial attacks. The rows represent the different methods used, and the columns represent the adversarial attack used, with their respective classification accuracy. Results are from a ResNet-101 architecture on a subset of the ImageNet dataset, collected from [DSC+17; YWW+20].
Note: Only benign images classified correctly by the underlying model are included.

4.3 Adversarial training

Adversarial training involves supplementing the training dataset with adversarially perturbed images while retaining the original labels. This enables the model to enhance its ability to accurately predict labels even when images have been manipulated [TDL20].

4.3.1 Method implementation

There is no single specific implementation of adversarial training, as its realization may vary depending on the particularities of the model and the chosen adversarial attacks. However, a general algorithm can be outlined to demonstrate the typical adversarial training process. The pseudocode provided in Algorithm 4.1 represents a common approach to performing adversarial training, where the model is updated iteratively using batches of both original and adversarially perturbed images.

Algorithm 4.1 Adversarial Training

Require: training data $\{X_1, \ldots, X_n\}$, model parameters θ, iterations I

1: $f_{adv} \leftarrow initializeAdversarialAttack(\theta)$
2: **for** $i = 1$ to I **do**
3: $B \leftarrow \{X_1, \ldots, X_m\}$ ▷ Batch, B, from a subset of the training set
4: $\{X_1^{adv}, \ldots, X_k^{adv}\} \leftarrow f_{adv}(\theta, B)$ ▷ Apply adversarial attack
5: $B' = \{X_1^{adv}, \ldots, X_k^{adv}, X_{k+1}, \ldots, X_m\}$ ▷ New batch
6: Update θ using B' ▷ Update model paramters
7: **end for**

4.3.2 Security performance

Adversarial training has proven effective against black-box adversarial attacks, where the attacker has limited access to the model's parameters and can only interact through input and output. However, this approach has limited efficacy against white-box adversarial attacks, as these attackers have full access to the model's parameters, architecture, weights, and coefficients. This enables them to adapt their adversarial attacks accordingly [TDL20; KGB16]. Table 4.2 shows an overview of the security performance.

While adversarial training can increase a model's robustness against black-box adversarial attacks, it can be computationally intensive. This technique requires perturbing images based on specific attack types and then including them in the training dataset, which increases training time. Furthermore, it has been shown to slightly decrease the accuracy of clean images on large datasets like ImageNet [DDS+09]. Nevertheless, the accuracy on clean images when using smaller datasets, such as MNIST and CIFAR-10, does not change [KH+09; LBBH98]. Adversarial

training remains a valuable method for improving a model's overall robustness, even if it has limited impact on white-box models [TDL20; KGB16].

Security Performance of Adversarial Training			
Evaluation Metric: Fooling Rate			
Defense	**FGSM**	**DeepFool**	**CW**
No Defense	98%	99%	99%
Adversarial Training	49.2%	99%	99%

Table 4.2: Security performance of adversarial training on various white-box adversarial attacks. The reported values are the respective fooling rates after adversarial training. Results are from a ResNet architecture on the TinyImageNet dataset, collected from [TDL20].

4.4 Adversarial Detector Subnetwork

Efforts to counter adversarial attacks through input data manipulation often result in compromised performance on benign images, as detailed in Section 4.2. Additionally, employing adversarial training strategies modifies the model's internal parameters, thereby influencing its performance accuracy on benign images, as discussed in Section 4.3. An alternative approach is to maintain the model's internal structure and address the adversarial attack externally without modifying the model's parameters. One such method is presented in the paper *On Detecting Adversarial Perturbations*, wherein the authors propose a novel technique for detecting adversarial examples by attaching a binary classification subnetwork to the primary model. This method demonstrates potential in mitigating the impact of adversarial attacks and enhancing the robustness of ML models [MGFB17].

4.4.1 Method Implementation

The authors first trained a multi-class image classification model to detect adversarial attacks without altering the model parameters. They then froze all the model weights and attached the detector model, a separate, smaller CNN. Subsequent training of the detector model can be conducted by employing the multi-class image classification model to classify images. Features can then be extracted from the multi-class image classification during classification and provide the detection model with a labeled dataset consisting of extracted features for benign and adversarial images [MGFB17].

The independent binary detector model connects to the multi-class classification model at five distinct attachment points. The detector model was tested for detectability accuracy at each attachment point. Figure 4.4 displays these five attachment points as arrows pointing to AD(0), AD(1), AD(2), AD(3), AD(4), and AD(5). The first attachment point is at the input of the multi-class classification model, meaning that the detector model receives a copy of the input image directly [MGFB17].

4.4.2 Security Performance

For the security performance evaluation of the system, five attacks were implemented, FGSM, I-FGSM ($L2$ and L_∞) and DeepFool ($L2$ and L_∞), see Section 3.3 for details about the attacks. The five attacks were performed on two different datasets CIFAR10 [KH+09] and a subset of ImageNet [DDS+09]. Concerning attachment points, the results indicate that AD(0) had the lowest detectability accuracy, while AD(2) and AD(4) provided the highest detectability accuracy across the five different adversarial attacks. Furthermore, the security performance results for the detection accuracy on the CIFAR 10 dataset and ImageNet dataset can be seen in Table 4.3 [MGFB17].

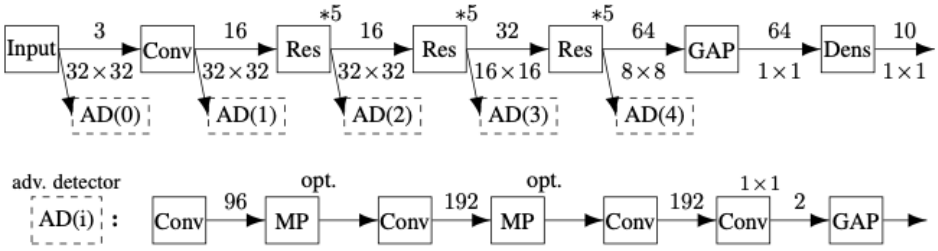


Figure 4.4: Adversarial detector subnetwork attached to a ResNet. The adversarial detector subnetwork is attached at five places. The adversarial detector subnetwork is a CNN [MGFB17].

Security Performance of CNN Detection Network					
Fooling rates: >70%					
Evaluation Metric: Detection Accuracy					
Dataset	**FGSM**	**IFGSM** L_∞	**IFGSM** $L2$	**DeepFool** L_∞	**DeepFool** $L2$
CIFAR10	97%	89%	87%	82%	79%
ImageNet	89%	87%	90%	85%	91%

Table 4.3: Performance of the detector network on white-box adversarial attacks. The perturbation ϵ is minimized to ensure the classification accuracy of the underlying ResNet32 classifier falls below 30%. Results are obtained from [MGFB17].

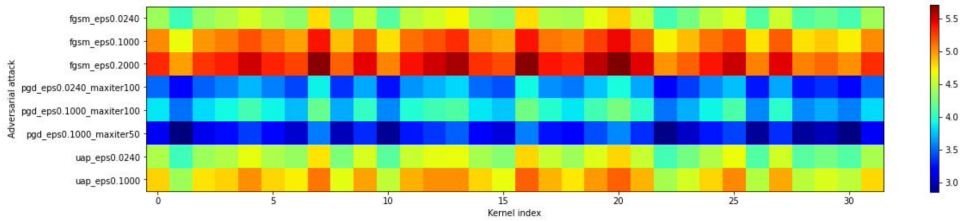


Figure 4.5: Illustration of the logarithmic distance between adversarial and benign activations for each filter (kernel) output in the initial convolution layer. Each row represents a specific instance of an adversarial attack with its corresponding parameters, while the columns represent the filter indexes in the first convolution in the Inception V3 architecture [Orv22].

4.5 Investigating Adversarial Impact on CNN Activations

A recent Master's thesis by Orvedal at NTNU studied how adversarial attacks influence activated feature maps, also known as activations, in a CNN. Through experiments, Orvedal examined the differences in activations by classifying benign and adversarial images. By conducting these tests with various adversarial attacks, he could visually observe the difference in activations and identify specific patterns in the data. He conducted his experiments using three different adversarial attacks: FGSM, PGD, and UAP. More information can be found in Section 3.3 [Orv22].

From Figure 4.5, it can be seen that certain indexes show a consistent pattern, displaying similar cooler or warmer colors across different adversarial attacks. These include but are not excluded to index one, seven, ten, and sixteen. Orvedal further visually examined the entire Inception V3 architecture. He found these patterns easier to detect in the shallower layers [Orv22].

However, it is essential to underscore that Orvedal's methodology of extracting and analyzing the data was a manual process. Therefore, its practical applicability in real-time operational systems is inherently limited. This signifies a crucial aspect of his research, indicating that while the insights derived are informative and beneficial, they may not directly translate into immediate solutions for live adversarial attack detection in CNNs, and therefore is hard to display the security performance of his work. This recognition underscores the necessity for further development to enable automatic detection and response within real-time systems [Orv22].

Chapter 5
Methodology

This chapter describes the methodology used in the thesis. First, Section 5.1 describes the overall research approach. Further, Section 5.2 defines the research design of the methodology and describes the guiding requirements for each step of the research process. The subsequent sections detail each step of the research methodology, providing assumptions and considerations conducted along the way aligned with the guiding requirements.

5.1 Research Approach

For this master's thesis, we will adhere to a quantitative research approach. The selection of a quantitative research approach is driven by the goal of yielding quantifiable and reproducible results from experiments. Furthermore, we want to generate insights that extend beyond the specific context of our investigation, thereby making an objective contribution to the field of adversarial machine learning [Str19].

It is important to note that quantitative research has its potential pitfalls. Risks of quantitative research biases, such as sampling bias[1], and selection bias[2], are inherent to this research approach. To ensure the validity of our results, we will implement measures to minimize these biases and maintain the integrity of our findings.

5.1.1 Generalizibility

In order to minimize the potential for sampling and selection bias, we will focus on the generalizability of our results. Therefore methods to enhance generalizability will play a crucial role in our quantitative research approach. Generalizability refers to how our research findings can be applied to a broader context beyond the specific

[1] Sampling bias is a systematic error that occurs when specific data points are disproportionately likely to be included in a sample. This bias can distort the representation of the data, potentially leading to erroneous conclusions [Hec90].

[2] Selection bias refers to the distortion that occurs when the data chosen is not fully representative of the set or group of data that should have been considered [HS95].

instances or cases studied. The goal is to develop models and draw conclusions not limited to our specific dataset or experimental setup but can be extrapolated to other similar contexts. Moreover, we aim to avoid coincidental results, which could arise from conducting a particular experiment that only holds in a narrow context and fails to yield consistent outcomes upon minor alterations. This is to ensure the robustness and reproducibility of our findings, critical factors in maintaining the integrity of our research.

To ensure generalizability and reduce coincidental results, we design our experiments with various adversarial attacks, using multiple CNN architectures and employing more than one dataset. Taking these precautions allows us to simulate a wide array of scenarios [Hec90; HS95].

5.1.2 Iterative Process

Given our focus on generalizability, we anticipate unexpected findings as we conduct our experiments. Therefore, we commit to an ongoing iterative process to rigorously incorporate these emergent insights. Each data collection and analysis cycle will be systematically followed by a review and refinement phase, informed by the new knowledge gathered. However, to maintain the integrity of our research and minimize selection and sampling bias, it remains crucial that we continue to conduct our experiments within a broad spectrum of contexts; despite the iterative nature of our process, we need to make sure we are testing in a variety of situations to ensure the integrity of our work [Smi12].

5.2 Research Design

Following our quantitative research approach, which aims to produce generalizable results through an iterative process described in Section 5.1, our research design is centered around experimental research.

5.2.1 Experimental Research

To answer our research questions, we will conduct experimental research. The experimental research involves building different CNN models, exposing them to various adversarial attacks, and evaluating their performance. First, we aim to identify the essential components of CNN that can be effectively integrated to identify ongoing adversarial attacks (RQ1). Subsequently, we will develop a detection model and evaluate its robustness and generalizability (RQ2). Furthermore, we will evaluate the resource usage of the detection model compared to its security performance (RQ3).

5.2.2 Overview

To ensure a thorough understanding of our proposed methodology, we outline the complete system overview in Figure 5.1. Each number in this figure signifies a detailed diagram that will be explored further in the corresponding sections of the methodology chapter.

Within Figure 5.1, the rounded boxes with gear icons signify processes, denoting dynamic operations or actions carried out throughout our research. Conversely, the sharp-edged boxes represent static elements in our research methodology. These static elements are either collected or created through a dynamic process.

5.2.3 Guiding Requirements

To effectively carry out the experimental research, we will adopt the following research design process described below. Please note that in this section, we define what needs to be achieved in each step of the process, defining the guiding requirements of the overall system. The methodology chapter will detail the specific implementation details employed to adhere to these guiding requirements.

Software and Hardware

The guiding requirements for software and hardware toolkit will be their accessibility, ease of use for the task at hand, and usability based on previous knowledge.

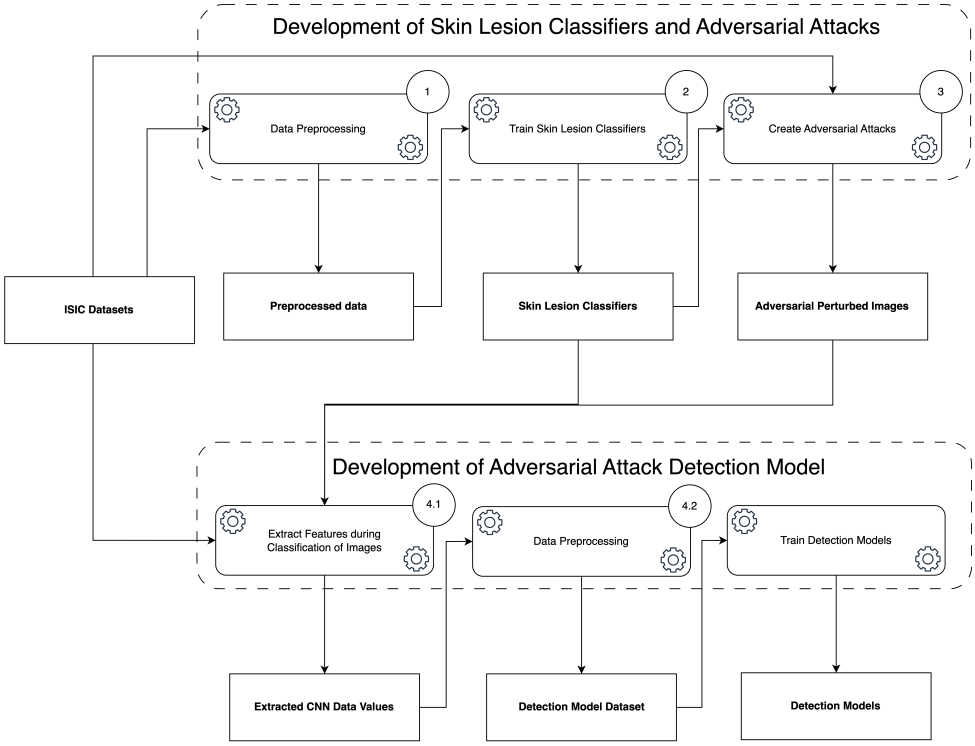


Figure 5.1: Schematic illustration of the complete system overview.

Dataset Selection

- **Size:** Balance the need for large datasets with the practical constraints of time and resources inherent to a master's thesis. This implies selecting datasets of sufficient size to minimize coincidental results while ensuring manageability within our scope.

- **Accessibility:** Prioritize datasets that are readily accessible to ensure that other researchers in the field can replicate our study.

- **Diversity:** Consider the comprehensiveness and diversity of the datasets. This ensures that they encompass a wide range of scenarios and variables, further enhancing the robustness of our research, given the generalizability in our approach outlined in Section 5.1.1

Arhictecture Selection for Skin Lesion Classifiers

- **Accessiblity:** The model architectures must be openly accessible, either via an Application Programming Interface (API) or a detailed description of implementation in literature, to ensure the reproducibility of our study.
- **Prevalence in existing literature:** The prevalence of the architecture's usage in the existing literature is important. Providing a benchmark for model performance, enabling comparison and evaluation of our results to external research.
- **Architectural differences:** The chosen model architectures must have some architectural differences in how they are built. The diversity in architecture will contribute to a more robust evaluation of our methods and ensure we can test for the generalizability outlined in Section 5.1.1.

Data Preprocessing

- **Data augmentation:** The requirement for data augmentation will be determined by the size and variety of our datasets and our models' tendency to overfit. For smaller or less varied datasets, augmentation might be advantageous to boost the model's ability to generalize to new data. Moreover, if overfitting - the model's excessive adaptation to the training data at the expense of its performance on unseen data - appears to be a significant problem, we may employ augmentation as a potential countermeasure.
- **Architectural constraints:** Our preprocessing methodologies must respect the specific needs and restrictions of the chosen CNN architectures. This means the data may need to be reshaped or reformatted based on the requirements of the selected models.
- **Data partitioning:** Ensure that the partitioned data set sizes are representative and balanced, allowing for robust training and validation. Furthermore, the independence of each data subset should be maintained to avoid interdependencies that could lead to biased results and overfitting.

Development of Skin Lesion Classification Model

- **Mitigating Overfitting:** Develop models that generalize well, meaning they perform comparably on the training, validation, and test datasets.
- **Performance Benchmarking:** Create competitive models compared to the current state of the art. To ensure that our skin lesion classifiers meet the

state-of-the-art standard, we should benchmark these classifiers against others with similar architectures in the field

Adversarial Attacks

- **Variety of Attacks:** A variety of adversarial attacks should be used; the diversity should reflect both different techniques and strengths. This will help determine whether the detection technique is consistent across different attacks.
- **Consistent Implementation:** To ensure fair and comparable results, implementing these attacks should be consistent across all tested models. This may require standardizing parameters and applying attacks under the same conditions.
- **Prevalance in existing literature:** The attacks chosen for the evaluation should be represented in the existing literature. The selection should consider the most commonly used and researched attacks, ensuring the relevance and practicality of the evaluation. This will also allow for a better comparison with other models tested under similar adversarial conditions.

Feature Extraction

- **Diversity of Components:** To capture a broad and nuanced understanding of the model's operations, it is vital that the features we extract originate from a diverse range of components within the CNN. This ensures that the extracted features provide a holistic representation of the model's internal operations.
- **Manageability of Data:** Extract as many meaningful features as possible while keeping the volume of data manageable. Balance obtaining comprehensive data and avoiding an overwhelming dataset that could complicate our analysis and extend processing times.
- **Simplicity of Extraction:** The complexity of the feature extraction process should be kept as minimal as possible. This does not imply a sacrifice in the quality or completeness of the data but instead underscores the need for efficiency and straightforward methods. Extracting features should not be an overly intricate process that becomes a project in itself. Instead, features should be designed and selected to be simple to pull from the CNN.

Detection Model Selection

- **Publicly available:** The models must be publicly available to ensure that someone can easily replicate our work.

- **Distribution-agnostic performance:** The detection model should perform optimally without a prerequisite for specific data distributions, given the unpredictability of potential data distribution.
- **Efficient training design:** The detection model should be designed for efficient training. Ideally, it should have a low training time without compromising its performance. This principle aligns with the need for practicality in real-world applications where quick deployment and updates are essential. In addition, an efficient model minimizes computational resources and time, lowering operational costs and fostering sustainable scalability.

Throughout our research process, we will ensure that our research approach is iterative, revisiting and refining our models and experiments based on the results obtained at each stage. Further, to guarantee the replicability of our work, we will keep clear records of our methodologies, including their underlying motivations and implementation details, all following the defined guiding requirements.

5.3 Selection of Deep Learning Framework

This section delves into the selection process for a deep learning framework for our project, focusing on PyTorch and contrasting it with another well-known framework, TensorFlow. The rationale behind the final choice is also presented.

5.3.1 Advantages of Pytorch

PyTorch has earned a strong reputation in the academic and industrial world due to its simplicity, flexibility, and wide range of features, including multi-platform deployment and distributed training [PGM+19]. In addition, the framework is equipped with an intuitive interface for harnessing the computational power of GPUs, leading to more efficient training processes. Moreover, PyTorch comes with various pre-trained models [WGZ+20].

5.3.2 Comparison of PyTorch and TensorFlow

PyTorch and TensorFlow, both open-source libraries, are widely used for creating and training deep learning models. TensorFlow, developed by Google, was introduced in 2015, while PyTorch was developed a year later in 2016 by Facebook AI Research (now Meta AI) [PGM+19; AAB+16].

One of the main differences between PyTorch and TensorFlow is their approach to building and training models. PyTorch employs a dynamic computational graph, offering greater flexibility and ease of use when constructing and modifying models. In contrast, TensorFlow utilizes a static computational graph, which can result in more efficient execution of large models but at the cost of flexibility [PGM+19].

They also diverge in their adopted programming style. PyTorch's imperative programming offers an intuitive and interactive experience. In contrast, TensorFlow's declarative programming is well-suited for production-level development and deployment but might be less intuitive for researchers and developers [RG16].

5.3.3 Rationale for Choosing PyTorch

We opted for PyTorch as our deep learning framework. Its flexibility, ease of use, and dynamic computational graph make it suitable for a research-oriented master's thesis. In addition, the availability of pre-trained models and GPU support further strengthens our choice, allowing us to develop, train, and evaluate our models efficiently. It is also the primary framework from which we derive our previous knowledge, therefore aligning the choice with the guiding requirements for software and hardware outlined in Section 5.2.3.

5.4 Hardware Setup and OS

We leveraged the computational acceleration offered by Graphics Processing Unit (GPU) to train our deep learning models efficiently. Our university had provided us with a virtual machine, which included a quarter of a Tesla A100. Essentially giving us a GPU equipped with 10GB of virtual GPU-RAM. This GPU was accompanied by 60GB of RAM and 12 vCPUs, enabling us to handle large datasets and run multiple tasks simultaneously. In addition, our virtual machine was hosted on Ubuntu 20.04.5 LTS, a stable and widely adopted operating system. This setup allowed us to fully utilize the capabilities of the PyTorch framework, ensuring rapid and efficient training of our models.

The highest quantity of virtual GPU-RAM accessible via the educational institution guided the selection of the given hardware configuration. In addition, Ubuntu Linux was preferred due to its terminal-based controllability, which offers significant advantages when interfacing through SSH (Secure Shell). Additionally, Ubuntu Linux was selected as it represented the primary platform we were familiar with, adhering to the guiding principle of previous knowledge discussed in our research design in Section 5.2.3.

5.5 Selection of Attack Software Framework

This section compares different adversarial attack software frameworks and discusses the rationale for selecting the final framework.

5.5.1 Attack Libraries: Torchattacks vs. Foolbox vs. Adversarial Robustness Toolbox

We considered three main libraries for implementing adversarial attacks: Torchattacks, Foolbox, and Adversarial Robustness Toolbox (ART) [Kim20; RBB17; NST+18]. While all three libraries support adversarial attacks, ART and Foolbox are more general-purpose libraries that can also be used with TensorFlow. Torchattacks, on the other hand, is designed explicitly for PyTorch, offering a PyTorch-like interface and functions that simplify the process of creating adversarial attacks.

Torchattacks provides a range of adversarial attacks, including but not limited to FGSM, CW, and DeepFool [Kim20]. Performance-wise, Torchattacks outperforms ART and Foolbox on FGSM, PGD, and CW, as demonstrated in a fair comparison using Robustbench [CAS+20].

5.5.2 Rationale for Choosing Torchattacks

Considering the selection of PyTorch as the primary deep learning framework, Torchattacks was chosen as the attack software framework. This decision was motivated by Torchattacks' compatibility with PyTorch, user-friendly interface, and ability to generate many unique adversarial attacks with high performance. Moreover, given the absence of familiarity with adversarial attack software frameworks, opting for a Pytorch-specific framework would greatly facilitate the learning process, as Pytorch is a familiar environment.

5.6 Selection of Datasets

In keeping with our aim of generalization as detailed in Section 5.1.1, we have selected two distinct but related datasets. We have chosen two datasets from the ISIC challenges, ISIC 2018 and ISIC 2019.

5.6.1 International Skin Imaging Collaboration Datasets

The ISIC has made available five datasets, each generated annually from 2016 to 2020. The datasets were created for the ISIC competition to enhance the diagnosis of melanoma detection. As a result, these datasets have undergone rigorous experimentation and analysis. [KHF20; HKF20; SAH+22; GNS+20].

5.6.2 Rationale for Choosing ISIC 2018 and ISIC 2019

Firstly, these datasets are substantial in size, which ensures minimalization of coincidental results while simultaneously being manageable within the time and resource constraints of a master's thesis. Secondly, the datasets are publicly accessible via the ISIC challenge archive, promoting the reproducibility of our study. Lastly, the diversity and comprehensiveness of the ISIC 2018 and ISIC 2019 datasets are further enhanced by their inclusion of varied image sizes and multiple skin lesion types. Section 2.3.1 outlines the different skin lesion types.

ISIC 2018 Dataset

The first dataset we will utilize originates from the ISIC 2018 challenge, composed of two sub-datasets. The first part of the dataset is constructed from the HAM10000 Dataset, comprising 10,015 dermoscopic images of skin lesions. The second dataset, from which the ISIC 2018 challenge is constructed—also known as the MSK Dataset—will not be employed for training, validation, or testing purposes [TRK18; NRT+18].

ISIC 2019 Dataset

The second dataset we will utilize originates from the ISIC 2019 challenge, consisting of 25,331 dermoscopic images. The dataset is assembled from three independent datasets: the MSK Dataset, the HAM10000 Dataset, and a new dataset called the BCN_20000 Dataset. The BCN_20000 Dataset comprises 19,424 dermoscopic images of skin lesions.

Dataset Overlap

The ISIC 2018 and 2019 challenge datasets exhibit a degree of overlap, as they both incorporate the HAM10000 and the MSK datasets. However, we have excluded the

MSK Dataset from our analysis of the ISIC 2018 dataset. Despite the HAM10000 dataset's presence in both the ISIC 2018 and ISIC 2019 challenges, we maintain that validating our methodology across datasets of varying dimensions is crucial. Given the limited availability of public skin lesion datasets, we conclude that proceeding with experiments on both datasets is justified. This decision will allow us to extract more comprehensive insights, enhancing the robustness and applicability of our findings.

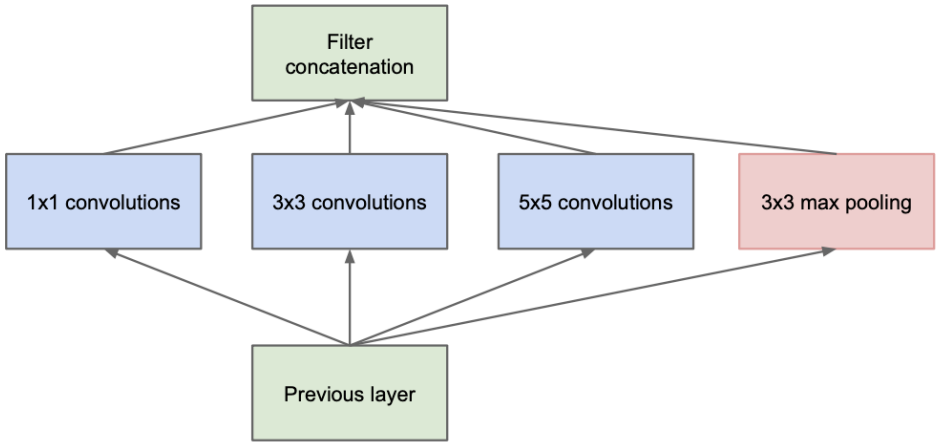


Figure 5.2: Illustration of a simple inception module, naive version [SLJ+15].

5.7 Selection of Skin Lesion Classification Architechtures

In this project, as part of our generalizability approach described in Section 5.1.1, we have selected two distinct CNN architectures for our skin lesion classifiers: Inception V3 and ResNet-18.

5.7.1 Inception V3 Architechture

Inception V3 is a CNN architecture, which has demonstrated strong performance in various computer vision tasks [AZH+21; PFH+18]. Specifically, it has been used to classify skin lesions and is often used as a test benchmark architecture for adversarial attacks in the category of inception architectures [NP21; KT22; XCS19]. The architecture employs a series of inception modules, improving the model's ability to learn complex representations at different levels of abstraction. Furthermore, Inception V3 has about 24 million trainable parameters [SVI+16]. See Table 5.1 for the Inception V3 architecture used in the thesis.

Inception modules

Inception modules are used in CNN to allow for more efficient computation and reduction of overfitting. Instead of only stacking more convolutional layers of different types and sizes sequentially, we let, for example, 1x1 convolution, 3x3 convolution, 5x5 convolution, and a max pooling layer operate on the same level, see Figure 5.2. The Inception V3 model architecture incorporates three distinct inception modules. Each module features its unique mixed architecture, differentiating it from the others [SLJ+15].

Inception V3 Architecture		
Type	**Filter size - Stride**	**Input Size**
conv layer	3x3 - 2	299x299x3
conv layer	3x3 - 1	149x149x32
conv layer padded	3x3 - 1	147x147x32
max pooling	3x3 - 2	147x147x64
conv layer	1x1 - 1	73x73x64
conv layer	3x3 - 1	71x71x80
max pooling	3x3 - 2	35x35x192
3xInception A	*	35x35x192
Reduction A	*	17x17x768
4xInception B	*	17x17x768
Reduction B	*	17x17x768
2xInception C	*	8x8x1280
average pooling	8x8	8x8x2048
linear layer		1x1x2048
dense layer	**	1x1xN

Table 5.1: Summary of the Inception V3 architecture used for the experiments. Each row of the table represents a different layer (or set of layers) in the architecture.
(*) Inception modules and reduction modules as described in [SVI+16]
(**) The size N in the final dense layer represents the number of classes in the output and will be 7 (ISIC 2018) or 8 (ISIC 2019).

5.7.2 ResNet-18 Architechture

ResNet-18 is a member of the Residual Network (ResNet) family of DNN. The primary innovation of the ResNet architecture is the use of residual connections (or skip connections), which makes it easy to optimize even very deep residual networks while still achieving increased accuracy gains from a wide network. This approach allows us to increase accuracy and make it easier to optimize while reducing the overall complexity of the model. Furthermore, ResNet-18 has about 11 million trainable parameters [HZRS16]. See Table 5.2 for the ResNet-18 architecture used in the thesis.

Residual blocks

Residual blocks play a crucial role in ResNets by enabling the transfer of information from earlier layers to later layers of a neural network. A usual method to increase the accuracy of a CNN is to make the model architecture deeper, but this only works up to a certain level before the accuracy starts to degrade rapidly [GB10; BSF94]. Residual blocks allow information in the network to bypass certain layers and retain important information, ultimately improving the accuracy and efficiency of the model. In other words, residual blocks help to ensure that the neural network can learn complex representations of data by allowing an additional channel of information to pass through the network [HZRS16].

ResNet-18 Architecture		
Type	**ResNet-18**	**Input Size**
conv layer	7×7, stride=2, padding=3	$224 \times 224 \times 3$
max pool	3×3, stride=2, padding=1	$112 \times 112 \times 64$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$56 \times 56 \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$56 \times 56 \times 64$
conv4_x	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$28 \times 28 \times 128$
conv5_x	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$14 \times 14 \times 256$
avg pool	7×7	$7 \times 7 \times 512$
dense layer	N* neurons	1x1xN*

Table 5.2: Summary of the ResNet-18 architecture used for the experiments. Each row of the table represents a different layer (or set of layers) in the architecture [NPS18].
(*) The size N in the final dense layer represents the number of classes in the output and will be 7 (ISIC 2018) or 8 (ISIC 2019).

5.7.3 Rationale for choosing ResNet-18 and Inception V3

Both architectures are readily available via the Pytorch API, and both architectures come with a multitude of pre-trained weights. The architectures are, in addition to this, explained in depth in their respective research papers [SVI+16; HZRS16]. Secondly, residual networks like ResNet-18 and inception-based networks like Inception V3 are frequently utilized in the literature. In a meta-analysis of 416 peer-reviewed

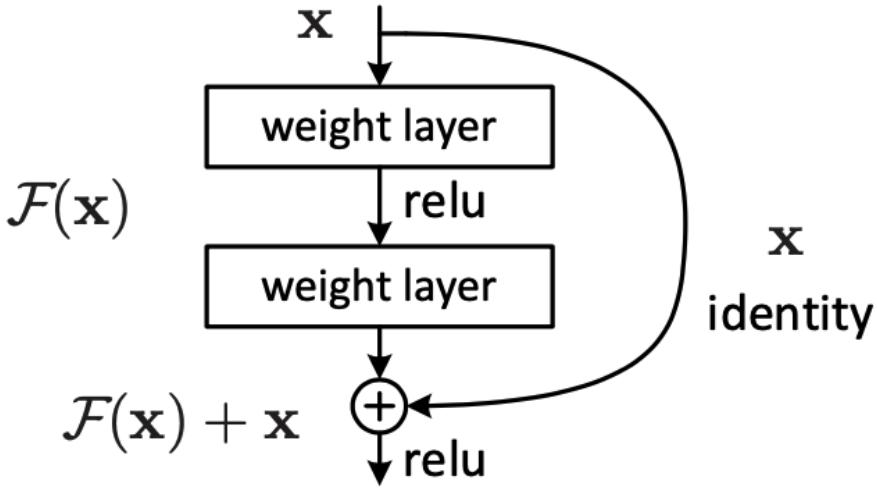


Figure 5.3: Illustration of a single residual learning building block [HZRS16].

journal articles, the residual network family of networks stood for 33% of the 416 papers, and the Inception V3 stood for 5% [GMHM21]. Lastly, the two architectures are unique, characterized by components such as residual blocks in ResNet-18 and inception modules in Inception V3. The uniqueness of model architectures is further emphasized by variations in model sizes, as reflected in the number of layers and trainable parameters. This selection aligns with the guiding requirements for architecture selection defined in Section 5.2.3.

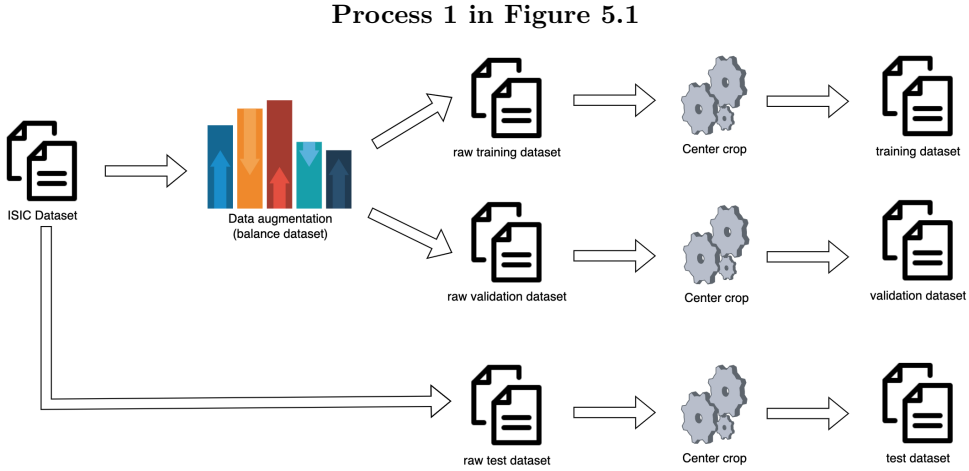


Figure 5.4: Schematic illustration of the preprocessing pipeline for the skin lesion classifiers, originating from the ISIC dataset.
Note: Primary ISIC 2018 dataset includes a distinct testing dataset, the primary ISIC 2018 dataset will therefore only be subdivided into training and validation.

5.8 Dataset Preproccessing

This section will detail the steps to prepare the ISIC 2018 and ISIC 2019 datasets for model training. We will cover a range of preprocessing procedures that were fundamental to ensuring the datasets were well-structured, balanced, and suitable for use in our selected CNN models. Figure 5.4 shows a schematic illustration of the preprocessing pipeline.

5.8.1 Addressing Dataset Imbalance

The ISIC 2018 and ISIC 2019 datasets, selected for developing our skin lesion classifiers, initially exhibited significant class imbalance. Before these datasets were deployed to train our classifiers, it was imperative to discuss if data augmentation techniques were required. The need for data augmentation was based on the considerations outlined in the guiding requirements for data preprocessing in Section 5.2.3. The diversity of the different classes in the dataset, particularly the smaller representation of some skin lesion types, could limit the model's ability to learn effectively. By amplifying the size of under-represented classes, augmentation could improve the classifier's ability to discern between different skin lesion types.

Furthermore, as the dataset was highly imbalanced, it was prone to overfitting. Given these challenges – dataset imbalance, limited diversity, and potential for overfitting – we planned to use data augmentation techniques before initiating the

model training phase to address them. In implementing data augmentation, we applied a variety of transformations to the skin lesion images. For each transformation, we randomly selected an image from a specific category. As a result, we ensured that each skin lesion category in the ISIC 2018 dataset was sampled a minimum of 3000 times and in the ISIC 2019 dataset, a minimum of 6000 times. The series of transformations applied included a random horizontal flip (25% probability), a random vertical flip (25% probability), and a random rotation between -90 and 90 degrees. The effect of these augmentations on the data distributions for the ISIC 2018 and ISIC 2019 datasets can be seen in Figures 5.5 and 5.6, respectively.

5.8.2 Data Partitioning

To train our models and evaluate their performance, we divided our datasets into distinct sets. Specifically, we had the training, validation, and test datasets. The training dataset was utilized to develop and adapt the models. Meanwhile, the validation set was used during training to tune hyperparameters, adjust learning rates, and make other model modifications without risking overfitting. Finally, the test set was used to assess the final model's performance in a controlled and unbiased manner, providing a reliable measure of the model's effectiveness and generalizability to unseen data. The ISIC 2018 dataset conveniently came prepackaged with a dedicated test dataset. As such, we planned to utilize this dataset for our testing needs. We strategically split the default ISIC 2018 dataset for training and validation purposes.

5.8.3 Image Preprocessing

Before utilizing the images for model training, they underwent specific preprocessing. The images in the ISIC 2018 dataset shared dimensions of 3x450x600, where the three channels represented red, green, and blue. 450 is the number of pixels in height, and 600 is the width. However, the images in the ISIC 2019 dataset came in various sizes. The Inception V3 model requires an input size of 3x299x299, while the ResNet-18 model requires an input size of 3x224x224. Therefore, we resized the images to the appropriate dimensions by cropping them based on the center of the image.

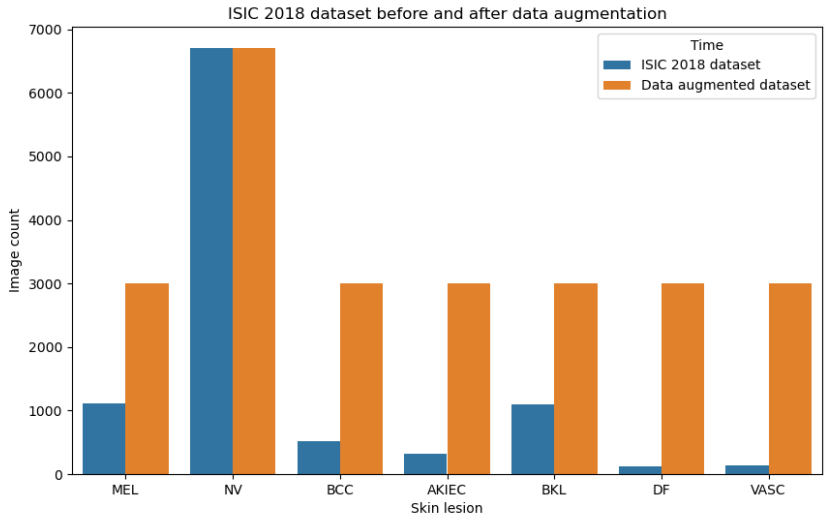


Figure 5.5: Distribution of Skin Lesion Categories in the ISIC 2018 Dataset before and after the data augmentation.

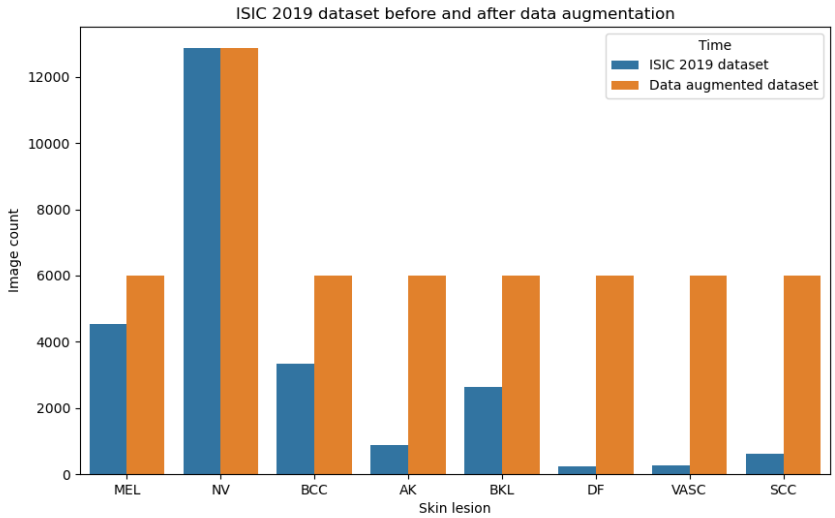


Figure 5.6: Distribution of Skin Lesion Categories in the ISIC 2019 Dataset before and after the data augmentation.

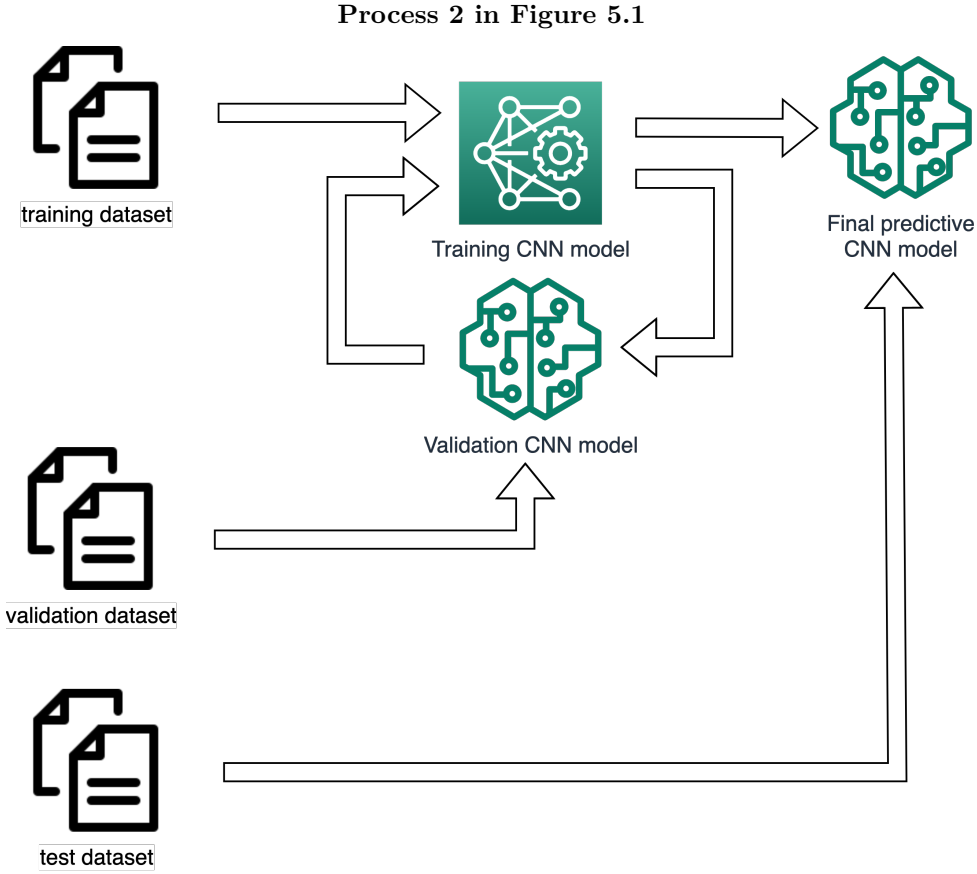


Figure 5.7: Schematic illustration of the skin lesion classifier CNN model training.

5.9 Development of Skin Lesion Classification Model

This section describes the methodology for developing the four individual skin lesion classifiers. To provide a comprehensive overview of our research methodology, we also document those experimental methodologies that did not yield the desired hypothesized outcomes. The motive behind presenting this is to contribute to future research efforts by illuminating effective and less successful strategies. Figure 5.7 shows an overview of the model training process.

5.9.1 Architechture Implementation

PyTorch provides an API for importing model architectures. We employed this API to implement the Inception V3 architecture [SVI+16] and the ResNet-18 architecture [HZRS16], both of which come with pre-trained weights, which are trained on

ImageNet [DDS+09]. We will construct four skin lesion classifiers, two of which will utilize the ResNet-18 architecture, and two will be based on the Inception V3 architecture. For both architectural configurations, we will train one model on the ISIC 2018 dataset and the other on the ISIC 2019 dataset.

5.9.2 Challenges and Shortcomings in Model Fine-tuning

Our initial approach was to fine-tune the models by only training a linear layer, keeping the pre-trained ImageNet weights unchanged. This method is commonly used when leveraging transfer learning, as it allows the model to retain generic features learned from a large-scale dataset like ImageNet while adapting to the specific task at hand. However, this approach could have yielded more satisfactory results, specifically in accuracy. Consequently, we revised our approach and performed full model training, utilizing the ImageNet model weights as a starting point. This decision was motivated by our intention to ensure that our skin lesion classifiers remain competitive concerning state-of-the-art models in terms of overall performance. Unfortunately, with fine-tuning, initial results fell short of this benchmark. As such, a more comprehensive training approach, encompassing the entire model and not merely a linear layer, was necessary to enhance performance and align more closely with the state-of-the-art potential.

5.9.3 Model Training

Given the shortcomings in the initial fine-tuning approach, we strategically decided to perform full model training for our classifiers. We were utilizing the pre-trained weights from ImageNet as the starting point. To begin with, we configured the parameters specific to each of the four models, as illustrated in Table 5.3. Then, with these parameters in place, the training process was initiated. Throughout the training process, at the end of each epoch, we assessed the model's performance using the validation dataset.

Rationale for Parameter Selection

Our choice of parameters was first and foremost governed by the hardware limitations detailed in Section 5.4. We set the number of workers at 10, the highest possible value under our constraints. The batch size selection underwent a similar consideration. Initially, we considered a batch size of 64, but our training GPU could not accommodate such a large size due to memory constraints, necessitating a reduction to a batch size of 32. We conducted a coarse grid search for the learning rate, gamma, momentum, and step size to identify the parameters that would deliver suitable performance. While the primary objective of our master's thesis is not to create the best-performing skin lesion classifier, having a solid and high-performing base model for our experimental investigations is still important. The coarse grid

search approach allowed us to balance performance and computational efficiency while ensuring the parameters selected were fit-for-purpose and capable of fostering an adequately robust model. Initially, we experimented with the Adam optimizer but later transitioned to the Stochastic Gradient Descent (SGD) following its proven effectiveness with CNN architectures as indicated in [PP19]. We adjusted the epoch count for the ResNet-18 model on the ISIC 2018 dataset. We observed that running 50 epochs led to overfitting for this specific model. To mitigate this, we reduced the epoch count to 25, which helped alleviate the overfitting issue.

5.9.4 Model Evaluation

The outcomes of our model training process are presented in Table 5.4. These results are contextualized by reference benchmarks on ISIC 2018 and ISIC 2019 datasets. In previous studies, a ResNet-50 model achieved 89.28% classification accuracy, and an Inception-V3 model attained 88.05% classification accuracy on the ISIC 2018 test dataset [AKK20]. Furthermore, a hybrid architecture combining residual and inception elements reached a classification accuracy of 90.35% on the ISIC 2019 dataset (without segmentation) [SAA22]. These established benchmarks serve as essential reference points for assessing the model's performance concerning state-of-the-art. While our results with the ResNet-18 model are slightly inferior, registering a few percentage points lower for both datasets compared to state-of-the-art, we attributed this to ResNet-18 having half the trainable parameters compared to Inception V3 and ResNet-50.

Consequently, its capacity for learning complex representations is somewhat constrained, which may account for the observed performance difference. Nevertheless, our Inception V3 models demonstrate performance in line with state-of-the-art models, a considerable achievement. Based on our analysis, we confidently assume that both models possess a similar capacity for generalization compared to comparable models, thus fulfilling the performance benchmarking criterion as detailed in Section 5.2.3. Moreover, the models exhibit respectable performance on unseen data, satisfying the second critical factor for our skin lesion classification model's development. Consequently, we believe these models have successfully met the standards we initially set for this research.

	ResNet-18		Inception V3	
	ISIC 2019	**ISIC 2018**	**ISIC 2018**	**ISIC 2019**
Step Size	10	10	10	10
Gamma	0.1	0.1	0.1	0.1
Learning Rate	0.01	0.01	0.01	0.01
Momentum	0.9	0.9	0.9	0.9
Batch Size	32	32	32	32
Number of Workers	10	10	10	10
Dataset Split (%)	80,20*	80,20*	70,20,10	70,20,10
Epoch Count	50	25	50	50
Cost function	CE	CE	CE	CE
Optimization algorithm	SGD	SGD	SGD	SGD

Table 5.3: Comparison of model parameters used for training four skin lesion classifiers. The 'Dataset Split' row describes dataset partitioning into training, validation, and testing.
Note: Cross Entropy Loss (CE), Stochastic Gradient Descent (SGD)
(*): For the ISIC 2018 dataset, a dedicated test dataset is already available, so the dataset is only split into training and validation sets.

	ResNet-18		Inception V3	
	ISIC 2019	**ISIC 2018**	**ISIC 2018**	**ISIC 2019**
Training time	59 min	20 min	68 min	107 min
Classification Accuracy	84.96%	86.01%	88.08%	90.88%
Recall	84.96%	86.01%	88.08%	90.88%
Precision	85.29%	87.17%	88.62%	90.91%
F1 Score	84.84%	85.93%	88.17%	90.85%

Table 5.4: Performance outcomes of the four skin lesion classification models on their respective test datasets.

5.10 Adversarial Attacks

5.10.1 Selection of Adversarial Attacks

To select the adversarial attacks, we will adhere to our guiding requirements for adversarial attacks outlined in Section 5.2.3. We have selected four unique adversarial attacks to ensure various attacks with a diversity of strengths and techniques.

FGSM is selected because of its strong prevalence in existing literature, as it was the first implementation of an adversarial attack [SZS+13]. As we can see from Chapter 4, it gives us a strong comparison basis to compare our method to multiple protection methods. It also provides an example of a less strong attack, giving us a starting point for the whole spectrum of attacks in terms of strength. See Section 3.3.1 for further details about the attack.

CW is selected because it represents the stronger adversarial attacks, as noted in Orvedals master's thesis future work, it would be one of the most interesting adversarial attacks to put into the test, as it makes many protective mechanisms obsolete [Orv22]. It also is highly represented in prevailing literature, giving us a good basis for comparison. See Section 3.3.7 for further details about the attack.

I-FGSM is selected because it significantly escalates the FGSM attack's strength. It is an iterative approach that applies the FGSM attack multiple times with a small step size, providing a more refined adversarial example. This attack is a stepping stone between FGSM and CW in terms of attack strength and sophistication; thus, it serves a crucial role in testing the robustness of our models. I-FGSM has been discussed extensively in the literature, providing a wealth of comparative data [KGB16]. See Section 3.3.3 for further details about the attack.

PGD is selected as it is a variant of the I-FGSM attack, offering further sophistication and potency. This attack assists us in investigating how our models react to two closely related but distinct adversarial attacks. The inclusion of the PGD attack contributes to the breadth of our investigation [MMS+17]. See Section 3.3.6 for further details about the attack.

5.10.2 Implementation of Adversarial Attacks

We have developed an iterative process to maintain uniformity in the application of adversarial attacks. This method is designed to apply these attacks with consistency systematically. A detailed representation of the application process for each adversarial attack can be found in Figure 5.8.

Process 3 in Figure 5.1

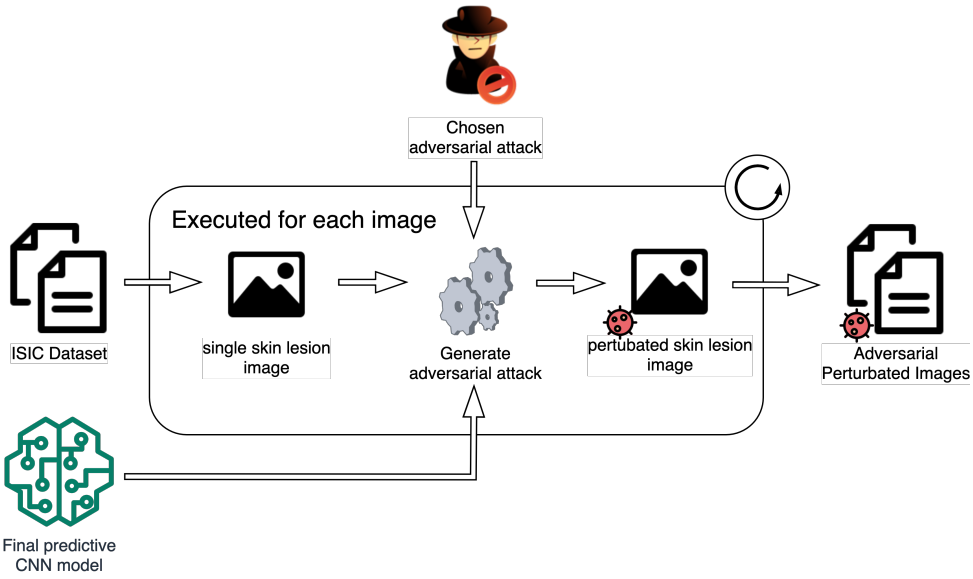


Figure 5.8: Schematic illustration of creating the adversarial attacks.

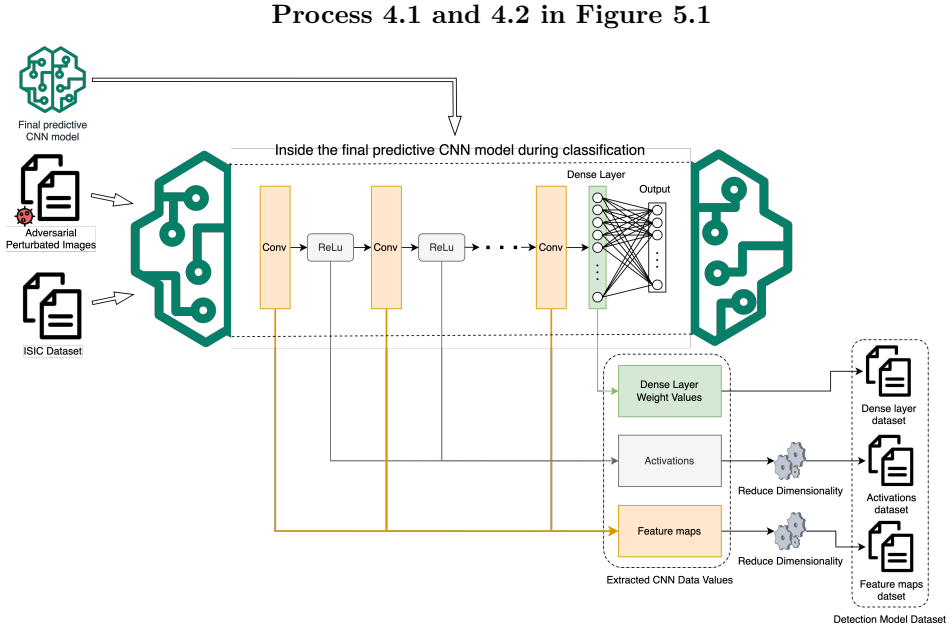


Figure 5.9: Schematic illustration of extracting features from the CNN and preprocessing these features to create a dataset.

5.11 Extraction of Features

This section described our methodology for extracting features from the inner structures of our skin lesion classifiers. For a complete overview of this part of the methodology, see Figure 5.9.

5.11.1 Diversity of Components

In line with the guiding requirements for feature extraction outlined in Section 5.2.3, we highly emphasized the diversity of components when selecting which elements to extract as features. Our approach is rooted in understanding the architecture of a typical CNN, as described in Section 2.2.5. We identified four key components within the CNN architecture: convolutional layers, non-linear activation functions, pooling layers, and fully connected layers. Each of these components plays a unique role in the model's operation, and thus, extracting features from each of these components ensures a comprehensive understanding of the model's internal workings.

While we recognize the importance of pooling layers in reducing the spatial size of the representation and controlling overfitting, we have chosen not to focus on these layers individually for feature extraction. This decision is based on their primary role being to sub-sample the input, which is inherently a part of the feature maps

generated by the convolutional layers and activation functions [AZH+21]. Instead, we focus our feature extraction on the convolutional layers with or without non-linear activation functions and the fully connected layers.

5.11.2 Feature Extraction Simplicity

Our strategy for feature extraction is grounded in the principle of simplicity and focuses on key layers of the respective architectures.

ResNet-18: We targeted the conv1, conv2_x, conv3_x, conv4_x, and conv5_x layers for feature extraction. An overview of the architecture is outlined in Section 5.7.2 and Table 5.2. We extracted features both pre and post-application of the non-linear activation function. Furthermore, we extracted the weight values from the dense layers.

Inception V3: Based on the architecture denoted in Table 5.1, we focused on the first five convolutional layers, excluding the two max pooling layers. We further extracted features from each inception and reduction module, giving us 16 layers. Like ResNet-18, we extracted features before and after applying the non-linear activation function. Furthermore, we extracted the weight values from the dense layers.

5.11.3 Manageability of Data

As we extract features from various layers and components of the CNN models, we confront the challenge of high dimensionality. The direct use of the extracted feature set would lead to an unwieldy volume of data, significantly complicating further analysis and potentially leading to issues like the curse of dimensionality [AK18]. To manage this, we implement a dimensionality reduction strategy.

Our strategy involves applying statistical and mathematical measures that encapsulate the salient properties of the extracted features, thereby reducing the data's dimensionality while retaining crucial information. Specifically, we employ the mean, $L1$-distance, $L2$-distance, and L_∞-distance as our primary measures, see Section 2.1.2.

In essence, these measures reduce the dimensionality of the extracted features while ensuring the preservation of the essential characteristics of the data. As a result, this approach provides easier data handling and more efficient downstream analysis. Simultaneously, it reduces the risk of overfitting and computational complexity.

5.12 Selection of Detection Model

This section systematically evaluates various machine learning models in light of the guiding requirements for detection model selection outlined in Section 5.2.3.

5.12.1 Comparative Evaluation of Machine Learning Models

Initially, we consider several popular models for the task at hand: SVM, Logisitic Regression (LR), Extreme Gradient Boosting (XGBoost) and DNN [Sar21]. Each model's potential is assessed against our established requirements: public availability, requirements about data distribution, and efficient training.

SVM are widely available through open-source libraries, but they can struggle with large datasets and require data to be well-preprocessed and normalized. They work best with small to medium-sized datasets with a clear margin of separation between classes but can be adapted for non-linear problems using the kernel trick. However, due to their high computational complexity, SVM have longer training times than other machine learning algorithms, particularly for large datasets [Nob06].

LR is a popular algorithm for classification and is widely available through libraries in multiple programming languages. It assumes a linear relationship between the independent variables and requires independent observations. It is computationally efficient and relatively fast to train, though performance and training time may vary based on the size and nature of the dataset [HLS13].

XGBoost is an open-source tool, freely accessible and capable of handling diverse data types, including numerical, categorical, sparse, or missing data, without requiring extensive preprocessing. It is specifically designed for efficiency, leveraging parallel computing capabilities for faster training time and supporting distributed computing for large datasets. In addition, XGBoost flexibility and built-in techniques to handle overfitting make it a potentially good choice as a detection model [CG16].

DNN offer high performance but require large and diverse datasets, as well as significant computational resources and time for training. Using open-source neural network frameworks and hardware accelerators can alleviate some of these challenges. However, traditional machine learning methods may be more efficient and practical for smaller datasets or more straightforward tasks.

5.12.2 Rationale for Choosing Extreme Gradient Boosting

Choosing XGBoost as our detection model is driven by its computational efficiency and scalability, handling large datasets effectively through parallel and distributed computing. XGBoost's ability to manage diverse data types, including numerical,

categorical, or missing data, reduces preprocessing effort compared to other models like SVMs. In addition, its built-in techniques to control overfitting make it more reliable and robust. Unlike Logistic Regression, it can model complex, non-linear relationships, and compared to Neural Networks, it is less resource-intensive and more flexible. Overall, XGBoost aligns well with our requirements, offering versatility and practicality for our detection model [CG16].

Chapter 6
Experiments and Results

This chapter delves into a series of carefully designed experiments and their corresponding results aimed at providing us with crucial insights to address our three research questions. Each experiment is crafted with a well-defined objective complemented by an experimental design. Following each experiment, we will present the results obtained. Section 6.1 is related to answering RQ1, Section 6.2 is related to answering RQ2, and lastly, Section 6.3 is related to answering RQ3.

6.1 Experiment 1: Identifying Essential CNN Components for Ongoing Adversarial Attack Detection

In the initial experiment of this master thesis, we will continue to build on the knowledge from Orvedal's research, see Section 4.5. His research demonstrated that specific detectable changes occur within a network during an adversarial attack [Orv22].

6.1.1 Objective

Our first experiment aims to pinpoint the components and metrics within the CNN that will provide valuable information about the model's security state, particularly detecting whether it is currently under an adversarial attack.

6.1.2 Experimental Design

Evaluation Metrics

Five key evaluation metrics are collected: true positives, true negatives, false positives, false negatives, and detection accuracy. See Section 2.2.4 for a more detailed explanation of the confusion matrix. These evaluation metrics will give us an understanding of the contribution for a given *component-metric* combination. Additionally, the confusion matrix is chosen because it allows us to compare the results with various

findings and results from other research studies within the field of adversarial machine learning.

Experiment Setup and Implementation

In the experiment, all four skin lesion classifiers are employed. Detailed information about each specific model can be found in Section 5.9. Four unique white-box adversarial attacks are applied for each image in the dataset. Each attack and its corresponding parameters are detailed in Table 6.1. The four skin lesion classifiers are sequentially supplied with benign and adversarial images during the experiment. During classification, features are extracted; see Section 5.11. The resulting *component-metric* pairs which are extracted are detailed in Table 6.2.

The extracted *component-metric* pairs will be used to generate a labeled dataset. The labels in this dataset will reflect the state of the skin lesion classifier - precisely, whether it is under an adversarial attack or not. Supervised machine learning serves as an assessment instrument to determine which *component-metric* pair yields the most significant insights into the security status of the model. We will train and test a total of 176[1] distinct supervised ML models. After training each supervised ML model on 80% of the available labeled data, we will test it on the remaining 20%. Evaluation metrics will be recorded during testing to assess the different *component-metric* pairs. The supervised machine learning algorithm that will be used is XGBoost; see Section 5.12. An overview of experiment 1 can be viewed in Figure 6.1, while a comprehensive overview can be found in Appendix A.

Attacks	**Parameters**
Fast Gradient Sign Method	$\epsilon = 8/255$
Iterative Fast Gradient Sign Method	$\epsilon = 8/255$ $\alpha = 2/255$ $steps = 10$
Carlini & Wagner	$c = 1$ $\kappa = 0$ $steps = 50$ $lr = 0.01$
Projected Gradient Descent L_∞	$\epsilon = 8/255$ $\alpha = 1/255$ $steps = 10$

Table 6.1: Adversarial attack methods used in the experiments and their respective implementation parameters.

[1]Four skin lesion classifiers, on four adversarial attacks on eleven different *component-metric* pairs (4x4x11=176)

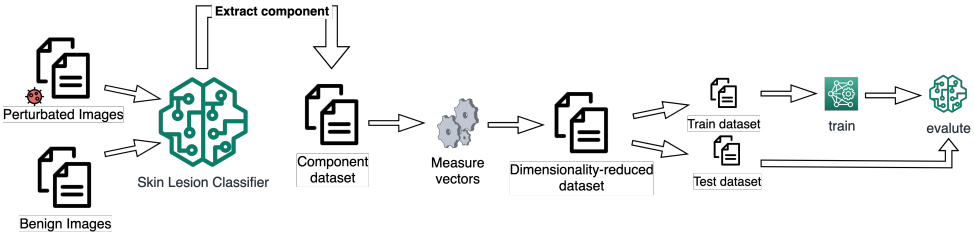


Figure 6.1: Schematic illustration of the process of assessing a single *component-metric* utilizing supervised ML techniques.

Component	**Metric**
Feature maps	Average value $L1$ distance $L2$ distance L_∞ distance
Activations	Average value $L1$ distance $L2$ distance L_∞ distance
Dense layers	Weight values
Combinations*	Activations $L2$ and Feature maps L_∞ Activations $L2$, Feature maps L_∞, and dense layer weight values

Table 6.2: CNN components and the corresponding metrics used for reducing the matrices into multiple scalar values.
(*) Incorporates a combination of components and metrics.

6.1.3 Results

All *component-metric* pairs for each attack on every model, along with the corresponding evaluation metrics, can be found in the tables presented in Appendix B. As evident from the data presented, certain *component-metric* combinations demonstrated higher detection accuracy across all models and attacks, while others underperformed in comparison. An overview of the average detection accuracy over all models and datasets for each attack can be observed in Table 6.3.

We observe that two *component-metric* pairs outperformed others. The first is the combination of Activations $L2$ distance, Feature maps L_∞ distance, and dense layer weight values, achieving an average detection accuracy of 96.14% across all adversarial attacks. The second-best performing pair utilizes a combination of Activations $L2$ and Feature maps L_∞ distance, resulting in a slightly lower average

detection accuracy of 95.67%.

The Feature maps L_∞ distance stood out as the *component-metric* pair, performing best against the CW attack, registering a detection accuracy of 88.84%. This detection accuracy surpasses that of other *component-metric* pairs in identifying the same attack. For instance, the Feature maps $L1$ and $L2$ distances achieved accuracies of 70.93% and 70.27%, respectively, which are considerably lower. The same pattern is observed for the Activations and Dense layer *component-metric* pairs. The second highest accuracy for the CW attack, aside from the combinations that include Feature maps L_∞, is achieved with the Dense layers weight values, standing at 79.07%. Hence, the Feature maps L_∞ distance metric substantially improves the detection of CW adversarial attacks. It is also worth noting that the average detection accuracy of the Feature maps L_∞ distance for all attacks is 90.46%, which is somewhat lower compared to other *component-metric* combinations. However, its strength is clearly in its ability to detect the CW attack, which outperforms other metrics.

	Average over all models and datasets					
Component	**Metric**	**FGSM**	**I-FGSM**	**CW**	**PGD**	**Average**
Feature maps	Average value	98.54%	99.28%	70.12%	99.01%	91.74%
	$L1$ distance	97.38%	99.15%	70.93%	98.94%	91.60%
	$L2$ distance	98.55%	99.28%	70.27%	99.02%	91.78%
	L_∞ distance	89.03%	92.59%	88.84%	91.38%	90.46%
Activations	Average value	97.52%	99.99%	76.10%	99.99%	93.40%
	$L1$ distance	97.45%	99.99%	75.91%	99.99%	93.34%
	$L2$ distance	96.92%	100.00%	76.51%	99.98%	93.35%
	L_∞ distance	88.68%	99.77%	68.11%	99.68%	89.06%
Dense layers	Weight values	90.27%	99.96%	79.07%	99.91%	92.30%
Combinations	Combination*	98.06%	100.00%	84.63%	99.98%	95.67%
	Combination**	98.99%	100.00%	85.60%	99.98%	96.14%

Table 6.3: Average detection accuracy for different *component-metric* pairs for various adversarial attacks across all models and datasets. The accuracies are presented as percentages, indicating the effectiveness of each component-metric combination in detecting the respective adversarial attack.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞, and dense layer weight values.

6.2 Experiment 2: Evaluating the Generalizability of a Detection Model for Adversarial Attacks

6.2.1 Objective

The objective of the second experiment is to evaluate to what extent it is possible to develop a detection model that demonstrates generalizability across a diverse range of adversarial attacks targeting skin lesion classification models.

6.2.2 Experimental Design

Evaluation Metrics

In order to evaluate the generalizability of the detection model for adversarial attacks, we will assess the extent to which the model can effectively detect various adversarial attacks beyond its original design scope. The primary evaluation metric employed will be detection accuracy.

Experiment Setup and Implementation

Before we can start the second experiment, we will begin by selecting the two *component-metric* pairs that yielded the best performance in Experiment 1. Referring to the results presented in Section 6.1.3, the first *component-metric* pair with the best performance is the combination**[2]. The second *component-metric* pair is the combination*[3]. These two *component-metric* pairs will be utilized as the features for our detection model in Experiment 2.

In the second experiment, we employ all four skin lesion classifiers and apply four unique white-box adversarial attacks. The specifics of each attack, including their parameters, are described in Table 6.1, and the specifics of each skin lesion classifier, including their parameters, are described in Section 5.9. Each combination of an adversarial attack, a skin lesion classifier, and the specific features (combination* or combination**) will have a dedicated detection model. We will create, train and test 16 different detection models.

We will provide features extracted from the CNN skin lesion classifier during classification, 50% benign images, and 50% perturbed images to train all detection models. Following the training process, each detection model will be tested against all four adversarial attacks.

[2]combination** is compromised of activations with $L2$ distance, feature maps with L_∞ distance and dense layer weight values

[3]combination* is compromised of activations with $L2$ distance and feature maps with L_∞ distance

6.2.3 Results

A detailed presentation of figures explaining the results of the second experiment can be found in Appendix C. This section will summarize these findings by showcasing the average values computed over all datasets and models. The average results for Combination* and Combination** are illustrated in Figure 6.2 and 6.3.

The experimental findings provide a rich perspective on the transferability of various adversarial attacks. For example, observing the results of the detector models trained on the CW attack, we notice a robust performance when they are subjected to I-FGSM, FGSM, and PGD for both Combination* and Combination** features.

The performance of a detector model trained on I-FGSM and then tested on PGD has a 100% detection accuracy on both the Combination* and Combination** features. The reverse scenario—where a model trained on PGD is tested on I-FGSM has a 99.98% detection rate for both Combination* and Combination** features. However, it is noteworthy that both I-FGSM and PGD do not demonstrate effective transferability to either CW or FGSM adversarial attacks. Since the detection model operates on binary classification, their performance of transferring to CW or FGSM adversarial attacks is no better than random.

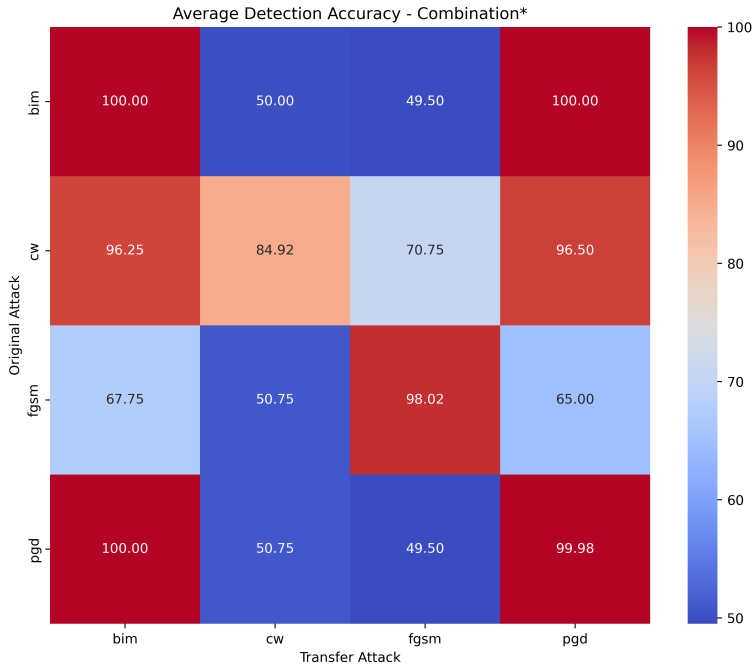


Figure 6.2: Heatmap visualization on the transferability of adversarial attacks for the Combination* features (activations $L2$ distance and feature maps L_∞ distance). The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

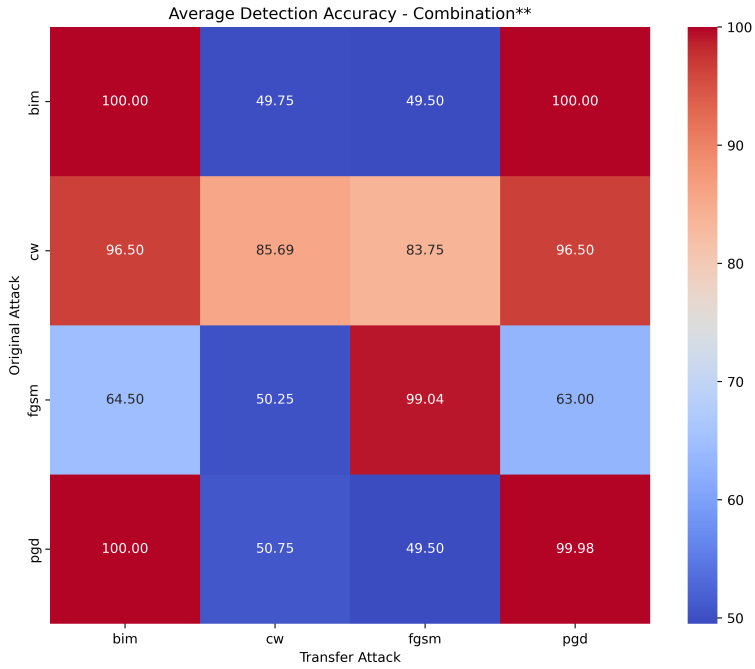


Figure 6.3: Heatmap visualization on transferability of adversarial attacks for the Combination** features (activations $L2$ distance, feature maps L_∞ distance, and dense layer weights). The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

6.3 Experiment 3: Exploring the trade-off between Resource usage and Security

6.3.1 Objective

The third experiment's primary goal is to thoroughly investigate the relationship between computational resources and the system's security. In addition, this investigation seeks to understand how variations in resource allocation can impact the robustness of protective mechanisms.

6.3.2 Experimental Design

Evaluation Metrics

In order to assess the trade-off between resource utilization and security, we will employ the following metrics: computational time and image count for resource evaluation, as well as detection accuracy for security evaluation.

Experiment Setup and Implementation

The third experiment utilizes the most successful features identified from the first and second experiment. According to the observed outcomes, the model employing a combination of $L2$ distance, L_∞ distance, and dense layer weight values demonstrated the highest overall security performance regarding detection accuracy. Further, this experiment is divided into three stages: the adversarial attack phase, the detection model training phase, and the detection model testing phase.

Adversarial attack phase: We will compute and apply adversarial attacks while recording the computational time. Each unique adversarial attack will be evaluated individually, with the average time to perform a single attack being determined from these measurements. The experiment will conduct two evaluations, focusing on the ResNet-18 architecture and the Inception V3 architecture. Given the white-box nature of the attacks, results may vary concerning the size of each architecture.

Detection model training phase: Our assessment will focus on the computation time and the number of image samples utilized during the training process. Additionally, we will evaluate the subsequent performance of the detection model, particularly its ability to ensure security.

Detection model testing phase: We will focus our investigation on the computational demands tied to feature extraction and predictive operations of the detection model. This involves an examination of the model's efficiency in handling real-time predictive tasks on incoming data. By closely tracking the computational

resources and time expended in these procedures, we aim to evaluate the operational viability of the security model.

It is important to highlight that the measurements in the adversarial attack and testing phase are contingent upon the specific hardware configuration utilized in our experiment, as detailed in Section 5.4. Consequently, actual computational times may exhibit some variance across different hardware setups. However, despite this potential variability, these results can still provide valuable insights into the relative computational demands of the different models. Furthermore, this comparison is instrumental in understanding the trade-off between security and resource allocation across different architectures, irrespective of the exact hardware configuration.

6.3.3 Results

Phase: Adversarial Attacks

From Table 6.4, we can observe that the time required to generate and apply adversarial attacks significantly varied according to the employed attack method and the complexity of the targeted skin lesion classifier. As anticipated, the more complex Inception V3 required more computational time to execute each attack when compared to the relatively smaller ResNet-18 architecture. The CW attack consumed the most time on the ResNet-18 and Inception V3, averaging 0.125 seconds and 0.303 per attack on their respective model architectures. Among the three phases, the adversarial attack phase was found to be the most demanding in terms of resource utilization, taking the longest time to complete.

Average Computation Time for Adversarial Attacks		
Attack	**ResNet-18**	**Inception V3**
FGSM	0.006 sec	0.025 sec
I-FGSM	0.068 sec	0.249 sec
C&W	0.125 sec	0.303 sec
PDG	0.067 sec	0.249 sec

Table 6.4: The table presents the average computation time required to generate and apply a single instance of an adversarial attack. The presented values are derived by taking the average from 1,000 iterations of the attack process and are denoted in seconds.

Phase: Detection Model Training

The relation between the number of samples and detection accuracy between different adversarial attacks on two architecture models, ResNet-18 and Inception V3, can be observed through the plots presented in Figure 6.4 and Figure 6.5. Notably, the detection accuracy on both I-FGSM and PGD adversarial attacks exhibit the same pattern, performing precisely the same if the detection model is trained on 100 samples or 10 000 samples. Regarding the FGSM attack, as the size of the training dataset increases, there is a slight improvement in the detection accuracy.

On the other hand, the detection accuracy when the detection model is exposed to the CW adversarial attack demonstrates a stronger correlation with the number of training samples. For example, for the Inception V3 model, the detection accuracy reaches a saturation point of around 2500 samples, indicating that further exposure to additional samples does not yield significant improvements in accuracy. However, for the ResNet-18 model, the detection accuracy continues to trend upward as the number of samples increases.

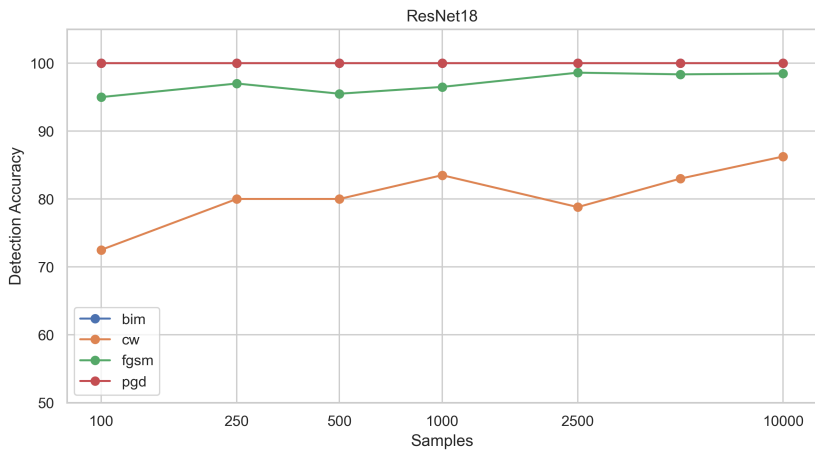


Figure 6.4: This graph illustrates the relationship between the number of training samples and the corresponding detection accuracy of the detection model. The detection model is connected to the ResNet-18 model. The x-axis is presented in a logarithmic scale, while the y-axis represents the detection accuracy, starting from 50%, corresponding to random guessing.
Note: I-FGSM, also called BIM, can not be seen in the plot since it is directly under PGD.

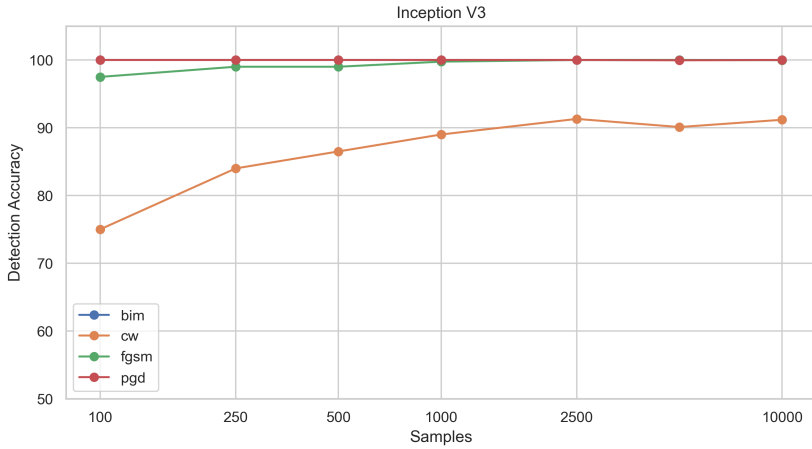


Figure 6.5: This graph illustrates the relationship between the number of training samples and the detection accuracy of the detection model. The detection model is connected to the Inception V3 model. The x-axis is presented in a logarithmic scale, while the y-axis represents the detection accuracy, starting from 50%, corresponding to random guessing.
Note: I-FGSM, also called BIM, can not be seen in the plot since it is directly under PGD.

Phase: Detection Model Testing

Our observations from this phase showed that feature extraction time was subject to variations based on the complexity of the model architecture. Specifically, we noticed that the extraction time for features from the more complex and bigger Inception V3 model was higher, averaging about 0.102 seconds. In contrast, the smaller ResNet-18 model required significantly less time, with an average feature extraction time of just 0.007 seconds. These results help illustrate the influence of model complexity on computational resource demands in the context of adversarial attack detection.

These findings suggest that feature extraction and adversarial attack detection processes can be performed with minimal delays, enabling real-time operations. Furthermore, despite the variations in computational time observed due to the complexity of different model architectures, the extraction times for both the Inception V3 and ResNet-18 models were within an acceptable range, remaining below a tenth of a second. Therefore, in practical applications, these delays would likely not significantly impede the real-time detection of adversarial attacks.

Chapter 7
Discussion

In this chapter, we will discuss the presented solution by comparing it against state-of-the-art solutions in Section 7.1. Subsequently, we address the three research questions posed at the beginning of this study in Section 7.2, facilitating a comprehensive understanding of the implications of our findings. Then, Section 7.3 will explore the constraints and limitations of our research. Finally, Section 7.4 will discuss potential paths for future work.

7.1 Comparison to Related Work

In this section, we will undertake an in-depth comparison and analysis of our outcomes in relation to those achieved by the state-of-the-art solutions that have been previously discussed in Chapter 4. This comparative evaluation serves to situate our work within the broader scientific context and highlight our contributions to the field. By drawing parallels and contrasts with the existing body of knowledge, we aim to demonstrate the novelty and utility of our solution. We will discuss both its innovative elements and practical usefulness.

7.1.1 Comparison to Input Transformations and Detector Subnetwork

In this section, we will examine the security performance of our proposed detection model against the detector model outlined in Section 4.4. Furthermore, we will also compare our detection model against the input transformation method, outlined in Section 4.2. We have chosen Flip + WebP compression as the benchmark for comparison due to its outstanding security performance. However, our comparison will not include gradient masking or adversarial training. The exclusion of gradient masking from our comparative study is based on its lack of substantial security performance, rendering it an insignificant element for comparison. As for adversarial training, we have chosen to exclude it since it was only somewhat effective on FGSM, while for

CW and DeepFool, it did not change the fooling rate after retraining, which means that the effectiveness is limited to black-box adversarial attacks. Given our testing is exclusively focused on white-box adversarial attacks, which are fundamentally stronger, the comparison with adversarial training would not yield valuable insights as these methods are inherently vulnerable to such attacks.

Performance on Benign Images

Input transformations inherently modify the input to the DNN, as the compression operation removes specific features from the image. This invasive technique compromises the classification accuracy of benign images. The accuracy of correctly classified images (100%) dropped to 95.84% following the application of input transformation.

In contrast, the detection model developed in this thesis and the current state-of-the-art detector model leave the structure of the DNN untouched; it is merely appended to the network, observing its operation. Consequently, there is no deterioration in the performance of benign images correctly classified before the application.

7.1.2 Performance on Adversarial Attacked Images

The comparative investigation for input transformations encompassed three adversarial attacks - I-FGSM, DeepFool, and CW - applied to a ResNet-101 architecture. The ResNet-101 architecture contains approximately 44.5 million trainable parameters and is best contrasted with our Inception V3 architecture. While input transformations function primarily as a mitigation technique, not a detection method, it is essential to note that a detector model can preemptively counter an attack, thus providing a form of mitigation. Therefore, we find it meaningful to compare the classification accuracy of the input transformations under attack and the detection accuracy of our detection model.

Focusing on the I-FGSM adversarial attack, the input transformation demonstrated a classification accuracy of 80.12%. However, our generalized detection model, trained on the CW adversarial attack, presented a notably superior average detection accuracy of 97.00%[1]. Remarkably, our detection model specifically trained on I-FGSM exhibited a perfect detection accuracy of 100.00%. Furthermore, considering the CW adversarial attack, the input transformation-based model exhibited a classification accuracy of 86.59%, whereas our detection model achieved an average detection accuracy of 88.29%.

[1]The 97.00% detection accuracy is calculated from the average of two detection models attached to two Inception V3 architecture models, where one is trained on ISIC 2018 and another is trained on ISIC 2019 dataset

For the comparison with the state-of-the-art detector model, which is integrated into a ResNet-32 architecture, we evaluated it against the averaged security performance of our detection model, attached to the ResNet-18 model architecture. Given the CIFAR10 dataset's limitation to images of size 32x32 pixels, we benchmarked our results with the ImageNet security performance. Regarding the FGSM adversarial attack, the present state-of-the-art detector model, specifically trained on the FGSM adversarial attack, attained a detection accuracy of 89%. In contrast, when specifically trained on the same FGSM adversarial attack, our detection model achieved a significantly higher detection accuracy of 98.22%. Thus, our detection model surpasses the current state-of-the-art detector model regarding FGSM security based on detection accuracy. Moreover, when considering the I-FGSM L_∞ adversarial attack, the state-of-the-art detector model yielded a detection accuracy of 87% after being specifically trained on I-FGSM. Conversely, our detection model, under equivalent training conditions on I-FGSM, achieved a remarkably higher detection accuracy of 99.99%. Hence, our detection model methodology proves superior to the existing state-of-the-art detector model even in this scenario.

7.1.3 Comparison of Methods

Input Transformations to our Detection Model

A comparative analysis of our proposed solution with the input transformation methods reveals distinct differences. Firstly, input transformation methods do not necessitate training for their security measures; they are applied once and continue to safeguard the model indefinitely. This no-training approach presents both strengths and drawbacks. On the one hand, it offers advantages like reduced computational cost and simplicity, removing overfitting risks. However, these input transformation methods could be less accurate and robust due to their inability to learn from data and adapt to new scenarios. Hence, while the lack of training requirement simplifies their implementation, it could potentially limit their long-term performance and adaptability compared to our method, which can learn and improve over time with the availability of stronger adversarial attacks.

In contrast, our approach needs initial training of the detection model. This training phase of the detection model has a unique advantage: the ready availability of training data, given that we can self-conduct the adversarial attacks and utilize the data generated for detection model training. One of the benefits of employing a trainable detection model is its ability to be iteratively updated in response to evolving adversarial attacks, thereby improving its effectiveness over time. However, these benefits come with their trade-offs. There is an increased computational cost and complexity since detection model training requires computational resources and involves multiple stages, such as choosing a ML model, creating an attachment data pipeline, and creating adversarial attacks. Furthermore, while our approach mitigates

the risk of overfitting through rigorous training procedures, it remains a potential challenge intrinsic to the training process of ML.

The second distinguishing factor lies in the functionality of our model to signal potential threats, indicating when the underlying skin lesion classifier might be under attack. The signaling enhances security by allowing real-time threat detection and enables proactive measures, such as temporarily shutting down the skin lesion classifier or implementing additional security measures. This could lead to reduced long-term costs associated with damage rectification. However, while these features offer clear advantages, they also bring particular challenges. There can be instances of false positives causing unnecessary disruption. Additionally, the increased complexity in system design requires more development, testing, and upkeep resources and may make the system harder to maintain and troubleshoot.

Contrarily, input transformations have no mechanism to alert when the skin lesion classifier is under attack. This simplifies the system design and operation and leaves the model more vulnerable to unforeseen threats. In conclusion, the decision to implement a threat signaling mechanism requires a careful balance between enhancing security and handling its possible drawbacks.

State-of-the-art Detector Model to our Detection Model

Shifting the focus to comparing our proposed solution and the current state-of-the-art detection model reveals a set of intriguing contrasts. In our approach, we employ a classical ML algorithm for the detection model, a striking deviation from the current state-of-the-art method, which deploys a separate CNN as a detection model.

The biggest difference between the state-of-the-art detector model and our solution is its capability to extract features from the entire underlying CNN model, thus offering a comprehensive perspective of the network's behavior when under attack. As discussed in Section 4.5, based on Orvedal's studies, we understand that adversarial attacks result in significant alterations at particular points within the model. Our detection model is designed to capture and analyze components across the entire CNN. We believe that by examining the complete architecture, we can identify the most indicative features of adversarial activity and consequently improve the accuracy of our detection mechanism. This is a marked departure from the state-of-the-art detector model, which only employs a single attachment point, which is the most intuitive way to tackle this problem, but, as discussed, does not provide the same security protection. As a result, it strives to detect adversarial attacks by analyzing a single layer, whereas our solution summarizes the complete network's performance during an attack. Further, relying on a single attachment point might not be optimal, as adversarial attacks could be constrained to strategically reduce their noise proximate in the detection region while not needing to constrain their noise

elsewhere in the network. In contrast, our method oversees the entire model network, limiting the adversarial attack's ability to hide its manipulations. This holistic view increases the robustness of our solution in the face of intricate adversarial attacks.

Furthermore, our choice to employ a classical ML algorithm in our detection model offers several distinctive benefits. Firstly, the classical ML algorithm provides high interpretability. Unlike the state-of-the-art detector model, which operates as a black box due to the inherent complexities of CNN, our model allows for a transparent understanding of its decision-making process. This interpretability is particularly crucial in cybersecurity applications, where the rationale behind detection decisions often needs to be explained. Furthermore, our model enjoys a superior computational efficiency over the CNN-based detection model. Classical ML algorithms are generally less computationally intensive than a CNN, so our model can be trained and run more quickly, making it more practical for real-time detection scenarios. Additionally, the lower data requirement of our classical ML algorithm offers another stark advantage. While CNNs typically demand large volumes of labeled data to perform optimally, our model can deliver satisfactory performance with smaller datasets. This feature is particularly beneficial when generating training data is computationally costly. As outlined in Section 6.3, generating the adversarial attacks is the most resource-intensive part of creating the detection model. Another advantage lies in the model's operation. Unlike a CNN, which necessitates higher-end hardware, our classical ML model can run on standard computing hardware without compromising its performance. This hardware independence significantly expands the potential deployment environments for our model.

7.2 Research Questions

7.2.1 Research Question 1

RQ1: *In the context of enhancing cybersecurity, which essential CNN components can be effectively integrated as features for identifying ongoing adversarial attacks?*

Based on the results provided in experiment 1, outlined in Section 6.1. We can confirm that specific CNN components are crucial in identifying ongoing adversarial attacks, which, when effectively integrated, can significantly boost the cybersecurity posture of a system. We specifically identified three components: the feature maps generated by the convolutional layers, the activations resulting from the non-linear activation function on the feature maps from the convolutional layers, and the weight values of the dense layers. By reducing the high dimensionality of the feature maps and activations through specific metrics - precisely the $L2$ distance for activations and the L_∞ distance for feature maps - we were able to derive valuable features for the detection of adversarial attacks. The combination of the $L2$ distance for activations, the L_∞ distance for feature maps, and the weight values of dense layers, in particular, yielded an impressive average detection accuracy of 96.14% across all adversarial attacks. This demonstrates the substantial potential of these components when used jointly. Interestingly, we also found that the effectiveness of specific *component-metric* pairs can vary depending on the adversarial attack being executed. For example, the L_∞ distance metric applied on the feature maps was exceptionally adept at detecting the CW attack, exhibiting a significantly higher detection accuracy compared to other *component-metric* combinations.

In conclusion, our findings suggest that to enhance cybersecurity, the internal components of a CNN model, specifically a combination of feature maps, activations, and dense layer weights, can be effectively harnessed to identify ongoing adversarial attacks. While our research has made considerable progress in demonstrating the utility of these CNN components in enhancing cybersecurity and yielding higher performance than current state-of-the-art methods, as described in 7.1, the results also uncover a new area of exploration in adversarial machine learning, as per our knowledge and understanding of the field, this is the first instance where an entire CNN architecture is leveraged for the purpose of detecting adversarial attacks.

7.2.2 Research Question 2

RQ2: *Considering the strategic employment of key convolutional neural network components outlined in RQ1, to what extent can a detection model be developed, demonstrating generalizability across a diverse range of adversarial attacks targeting skin lesion image classification models?*

The results of the second experiment were particularly insightful and provided a deep understanding of the transferability of various adversarial attacks. More specifically, we found that the detection models trained on the CW attack exhibited strong performance when subsequently tested with I-FGSM, FGSM, and PGD attacks. The detection models trained using I-FGSM and PGD exhibited impressive generalizability to each other. Remarkably, when tested interchangeably - the model trained on I-FGSM tested on PGD and vice versa - the detection rates were near perfect. This outstanding performance underscores the strength of the features that the detection model utilizes. However, it is worth noting that the model's generalizability did not hold across all adversarial attacks. Particularly, the detection models that were trained on either I-FGSM or PGD demonstrated no better than random performance when put to the test against CW and FGSM adversarial attacks.

In conclusion, our findings indicate that creating a detection model with high generalizability across adversarial attacks is possible if the detection model is trained on the highest-performing adversarial attack. The detection models trained on CW adversarial attack had an average detection accuracy of 90.61%, calculated by taking the average across all adversarial attacks on all four models.

7.2.3 Research Question 3

RQ3: *What computational resources are necessary to balance resource use and security in the system for effective adversarial attack identification?*

The results from Experiment 3 provide insight into the trade-off between computational resources and security. The investigation began by noting significant variations in computational time associated with the generation and application of adversarial attacks, with these differences being contingent on the complexity of the model architecture and the complexity of the adversarial attack. This revealed the primary bottleneck in resource utilization: the generation of adversarial attacks. Further, it was identified that the CW adversarial attack emerged as the most computationally demanding, especially on the more complex architecture, such as the Inception V3 architecture.

A key finding in training the detection models was that the detection accuracy does not improve for all adversarial attacks with an increased number of training samples. Specifically, the detection accuracy for both I-FGSM and PGD adversarial attacks remained relatively stable irrespective of the number of training samples used. On the other hand, the detection accuracy of the CW attack was more strongly correlated with the number of training samples, indicating that a more extensive training dataset could enhance detection performance. However, it is important to note that this trend was more pronounced when the detection model was connected to the ResNet-18 architecture than the Inception V3 architecture. The testing phase

showed that the feature extraction process, a crucial step in detecting adversarial attacks, is subject to variation based on architecture complexity. Nevertheless, the average extraction times for both models were short, 0.102 seconds for the Inception V3 architecture and 0.007 seconds for the ResNet-18 architecture, indicating that the feature extraction process is not a resource bottleneck for the system.

In conclusion, an effective balance between resource usage and system security necessitates careful consideration of the size of the skin lesion classifier model architecture. As exemplified by ResNet-18, smaller architectures tend to require a higher quantity of training samples to effectively detect certain types of adversarial attacks. For example, the CW adversarial attack, where even with 10,000 training samples, the detection model fails to reach a detection accuracy saturation point, indicates a challenge in pinpointing an ideal trade-off. Conversely, detection models tied to larger architectures, such as Inception V3, exhibit different performances. These models reach detection accuracy saturation with 2,500 training samples. These findings suggest that the computational resources necessary to balance resource use and security in the system for effective adversarial attack identification heavily depend on the complexity and the architecture of the model being used.

The variation in the performance of detection models applied to ResNet-18 and Inception V3 architectures may also be explained by the differing number of attachment points within these models, suggesting a more complex influence than model size alone. For instance, ResNet-18's detection model utilizes five attachment points, while in the case of Inception V3, this number is 16, as outlined in Section 5.11. This suggests an alternative interpretation of the results; the performance difference could be influenced by the volume of information extracted from each architecture. The Inception V3 architecture, with its greater number of attachment points, may allow for a more comprehensive extraction of features, which could potentially enhance its performance in detecting adversarial attacks. Consequently, it seems the intricate link between a model's architecture and the accuracy of the detection model is not merely tied to the model's size. Instead, it may also be influenced by the extent of information accessibility within the model. In that case, the resource usage necessary for the trade-off between resource usage and security would be that the more attachment points one has to the CNN model, the fewer resources are needed.

7.3 Limitations

7.3.1 Parameter Tuning for Skin Lesion Classifiers

The primary objective of this thesis is to delve into the security aspects associated with skin lesion classifiers. Consequently, the classifiers underwent a relatively basic coarse grid search for hyperparameters tuning, as outlined in Section 5.9.3. This was carried out to maintain the research's concentrated attention on evaluating the security rather than the extensive optimization of these classifiers.

7.3.2 Domain-Specific Knowledge in the Medical Field

A limitation of this study lies in its lack of in-depth, domain-specific medical expertise, specifically within dermatology. Only a short introduction to the skin lesion types was provided in Section 2.3.1. While the thesis focuses primarily on the technical aspects of skin lesion classifiers and their associated security vulnerabilities, understanding dermatology and the broader medical context in which these classifiers operate is undoubtedly valuable. These classifiers are developed to work in a particular domain - the medical field, specifically in diagnosing skin lesions. The nuances of medical imaging, the varied appearances of skin lesions, and the dynamic nature of diseases could all influence the performance of the classifiers and their susceptibility to adversarial attacks. Additionally, our lack of understanding could affect the interpretation of results and the subsequent clinical decisions based on these classifiers.

7.3.3 Number of Attachment Points for Detection Model

One limitation in this study relates to the different number of attachment points for the detection model connected to the ResNet-18 and the Inception V3 architectures. The ResNet-18 model had fewer attachment points compared to the Inception V3. While this difference was intentionally implemented to assess the impact of varying attachment point numbers on performance, it could have skewed the performance outcome for the ResNet-18 models. Consequently, it is essential to bear in mind that any performance disparities between the two architectures might not solely be attributed to their architectural differences but also to the varying number of attachment points.

7.3.4 Selection of Feature Combinations

In experiment 1 outlined in Section 6.1, the selection process for creating the combinations of *component-metric* pairs was largely experimental. While this iterative approach, as highlighted in Section 5.1.2, facilitated the discovery of feature combinations that yielded robust performance, it lacked a systematic and structured foundation. This lack of a formalized selection process for feature combinations repre-

sents a limitation in our experiments. The current approach was primarily dependent on experimental outcomes and iterative processes. This may have introduced the risk of overlooking potentially significant combinations.

7.3.5 Resource Bottleneck

A significant resource bottleneck in the present study is centered around the generation of adversarial attacks. This step is resource-intensive and consumes a considerable portion of the capabilities of the system. Although these attacks are only necessary during the training phase of the detection models, they represent a substantial constraint within the overall security framework, significantly impacting the adoptability of the system.

7.3.6 Experimental Constraints

Despite the high degree of automation within the experimental system, there are some limitations to consider regarding time efficiency. A single iteration of Experiment 2 - encompassing the generation of all adversarial attacks, extraction of features, training of detection models, and the evaluation of the generalizability of these models - necessitates approximately 12 hours of processing time, utilizing the hardware outlined in Section 5.4. Experiment 2, in particular, requires a comprehensive production of adversarial attacked images in order to create enough data points. The constraint could potentially affect the overall momentum and effectiveness of the research activities, warranting an exploration of non-invasive alternatives or methodological modifications for streamlining the process. The time commitment underscores a potential limitation concerning the capacity to execute numerous iterative cycles. Particularly given our research approach that emphasizes an iterative process, as discussed in Section 5.1.2.

7.4 Future Work

7.4.1 Standardization and Exploration of Attachment Points

A critical aspect that requires further attention, as pointed out in the limitations discussed in Section 7.3.3, relates to the potential variation in the number of attachment points associated with detection models. In order to make a more objective comparison, we need to standardize the number of attachment points across different model architectures. Additionally, future research endeavors should investigate the implications of manipulating the number of attachment points to recognize its influence on the overall model performance. This exploration could provide critical insights into these detection models' optimal number of attachment points.

7.4.2 Expanding the Architectural Exploration

Future investigations should broaden the spectrum of the analysis by encompassing a wide variety of architectural models. It would be intriguing to explore architectures that represent significant differences in complexity and size. For instance, testing across smaller, simpler architectures and larger, more complex ones could be beneficial. Adopting this approach could extrapolate more general insights into the relationship between architectural design and performance. These insights surpass the specific scenarios examined within ResNet-18 and Inception V3 frameworks. Furthermore, an expanded evaluation incorporating diverse architectures might reveal unobserved patterns or trends within the initial limited sample, which could be instrumental in refining architectural design choices and enhancing performance outcomes.

7.4.3 Broaden Scope Beyond Skin Lesions

While the current research presented here is concentrated on skin lesion datasets, as outlined in Section 5.6. Further research should indeed strive to explore the applicability of these findings across a wider spectrum of image datasets, both in size and domain. Furthermore, considering tasks outside of image-based processing can offer an even more encompassing perspective on the security performance of the method. Extensions to areas like Natural Language Processing (NLP) and audio signal processing could further enhance the applicability and versatility of the study's findings.

7.4.4 Anamoly Detection

The primary resource bottleneck in the proposed solution, as outlined in Section 7.3.5, is centered around generating adversarial attacks. This limitation prompts us to contemplate alternative methodologies to alleviate this constraint while achieving the same performance. One such approach that merits further exploration is

anomaly detection. Unlike the current methodology, which necessitates the creation of adversarial examples for supervised learning, anomaly detection could provide a viable alternative. This technique would pivot the detection model's training to focus solely on benign (non-adversarial) images. The detection model would be trained to understand the baseline characteristics of benign images. Consequently, any input sufficiently deviating from this learned norm could be flagged as adversarial. This model hinges on the premise that adversarial attacks introduce changes that make these images significantly different from benign ones, allowing them to be detected as outliers or anomalies. Anomaly detection could offer a more efficient, scalable solution by bypassing the need to generate many adversarial attacks. This promising alternative approach could be a significant area for future exploration and research.

References

[AK18] N. Altman and M. Krzywinski, «The curse (s) of dimensionality», *Nat Methods*, vol. 15, no. 6, pp. 399–400, 2018.

[AKK20] M. A. Al-Masni, D.-H. Kim, and T.-S. Kim, «Multiple skin lesions diagnostics via integrated deep convolutional networks for segmentation and classification», *Computer methods and programs in biomedicine*, vol. 190, p. 105 351, 2020.

[AMA17] S. Albawi, T. A. Mohammed, and S. Al-Zawi, «Understanding of a convolutional neural network», in *2017 international conference on engineering and technology (ICET)*, Ieee, 2017, pp. 1–6.

[AZH+21] L. Alzubaidi, J. Zhang, *et al.*, «Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions», *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[Bie87] I. Biederman, «Recognition-by-components: A theory of human image understanding.», *Psychological review*, vol. 94, no. 2, p. 115, 1987.

[Blo11] H. Blockeel, «Hypothesis space», *Encyclopedia of Machine Learning*, vol. 1, pp. 511–513, 2011.

[BR21] A. Bhardwaj and P. P. Rege, «Skin lesion classification using deep learning», in *Advances in Signal and Data Processing: Select Proceedings of ICSDP 2019*, Springer, 2021, pp. 575–589.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi, «Learning long-term dependencies with gradient descent is difficult», *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[CAS+20] F. Croce, M. Andriushchenko, *et al.*, «Robustbench: A standardized adversarial robustness benchmark», *arXiv preprint arXiv:2010.09670*, 2020.

[CCP+06] D. L. Cummins, J. M. Cummins, *et al.*, «Cutaneous malignant melanoma», in *Mayo clinic proceedings*, Elsevier, vol. 81, 2006, pp. 500–507.

[CG16] T. Chen and C. Guestrin, «Xgboost: A scalable tree boosting system», in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[Con23] W. Contributors, *Convolution*, https://en.wikipedia.org/wiki/Convolution, Online; accessed 21-April-2023, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Convolution.

[CW17] N. Carlini and D. Wagner, «Towards evaluating the robustness of neural networks», in *2017 ieee symposium on security and privacy (sp)*, Ieee, 2017, pp. 39–57.

[DDS+09] J. Deng, W. Dong, *et al.*, «Imagenet: A large-scale hierarchical image database», in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[DFO20] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*. Cambridge University Press, 2020.

[DSC+17] N. Das, M. Shanbhogue, *et al.*, «Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression», *arXiv preprint arXiv:1705.02900*, 2017.

[ECC22] M. Esmaeilpour, N. Chaalia, and P. Cardinal, «Rsd-gan: Regularized sobolev defense gan against speech-to-text adversarial attacks», *IEEE Signal Processing Letters*, vol. 29, pp. 1998–2002, 2022.

[FBI+19] S. G. Finlayson, J. D. Bowers, *et al.*, «Adversarial attacks on medical machine learning», *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[Fer17] M. Fernandez Figueras, «From actinic keratosis to squamous cell carcinoma: Pathophysiology revisited», *Journal of the European Academy of Dermatology and Venereology*, vol. 31, pp. 5–7, 2017.

[FFRT21] R. Francese, M. Frasca, *et al.*, «A mobile augmented reality application for supporting real-time skin lesion analysis based on deep learning», *Journal of Real-Time Image Processing*, vol. 18, pp. 1247–1259, 2021.

[Fla22] F. S. Flaate, «Understanding mitigation against adversarial attacks on skin lesion classifiers», 2022.

[FLW+20] L. Faes, X. Liu, *et al.*, «A clinician's guide to artificial intelligence: How to critically appraise machine learning studies», *Translational vision science & technology*, vol. 9, no. 2, pp. 7–7, 2020.

[Fuk80] K. Fukushima, «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position», *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[GB10] X. Glorot and Y. Bengio, «Understanding the difficulty of training deep feedforward neural networks», in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[GBV20] M. Grandini, E. Bagli, and G. Visani, «Metrics for multi-class classification: An overview», *arXiv preprint arXiv:2008.05756*, 2020.

[GMHM21] H. Ghanbari, M. Mahdianpari, *et al.*, «A meta-analysis of convolutional neural networks for remote sensing applications», *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 3602–3613, 2021.

[GNS+20] N. Gessert, M. Nielsen, *et al.*, «Skin lesion classification using ensembles of multi-resolution efficientnets with meta data», *MethodsX*, vol. 7, p. 100 864, 2020.

[GSS14] I. J. Goodfellow, J. Shlens, and C. Szegedy, «Explaining and harnessing adversarial examples», *arXiv preprint arXiv:1412.6572*, 2014.

[HEA+22] M. K. Hasan, M. T. E. Elahi, *et al.*, «Dermoexpert: Skin lesion classification using a hybrid convolutional neural network through segmentation, transfer learning, and augmentation», *Informatics in Medicine Unlocked*, vol. 28, p. 100 819, 2022.

[Hec90] J. Heckman, «Varieties of selection bias», *The American Economic Review*, vol. 80, no. 2, pp. 313–318, 1990.

[HJW+14] R. J. Hay, N. E. Johns, *et al.*, «The global burden of skin disease in 2010: An analysis of the prevalence and impact of skin conditions», *Journal of Investigative Dermatology*, vol. 134, no. 6, pp. 1527–1534, 2014.

[HKF20] K. M. Hosny, M. A. Kassem, and M. M. Fouad, «Classification of skin lesions into seven classes using transfer learning with alexnet», *Journal of digital imaging*, vol. 33, pp. 1325–1334, 2020.

[HLS13] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[HMT21] H. Hirano, A. Minagi, and K. Takemoto, «Universal adversarial attacks on deep neural networks for medical image classification», *BMC medical imaging*, vol. 21, pp. 1–13, 2021.

[HS95] J. J. Heckman and J. A. Smith, «Assessing the case for social experiments», *Journal of economic perspectives*, vol. 9, no. 2, pp. 85–110, 1995.

[HV08] C. Hafner and T. Vogt, «Seborrheic keratosis», *JDDG: Journal der Deutschen Dermatologischen Gesellschaft*, vol. 6, no. 8, pp. 664–677, 2008.

[HZRS16] K. He, X. Zhang, *et al.*, «Deep residual learning for image recognition», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[KGB16] A. Kurakin, I. Goodfellow, and S. Bengio, «Adversarial machine learning at scale», *arXiv preprint arXiv:1611.01236*, 2016.

[KH+09] A. Krizhevsky, G. Hinton, *et al.*, «Learning multiple layers of features from tiny images», 2009.

[KHF20] M. A. Kassem, K. M. Hosny, and M. M. Fouad, «Skin lesions classification into eight classes for isic 2019 using deep convolutional neural network and transfer learning», *IEEE Access*, vol. 8, pp. 114 822–114 832, 2020.

[Kim20] H. Kim, «Torchattacks: A pytorch repository for adversarial attacks», *arXiv preprint arXiv:2010.01950*, 2020.

[KT22] K. Koga and K. Takemoto, «Simple black-box universal adversarial attacks on deep neural networks for medical image classification», *Algorithms*, vol. 15, no. 5, p. 144, 2022.

[LBBH98] Y. LeCun, L. Bottou, *et al.*, «Gradient-based learning applied to document recognition», *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[Lin94] T. Lindeberg, «Scale-space theory: A basic tool for analyzing structures at different scales», *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[Mar96] R. Marks, «Squamous cell carcinoma.», *The Lancet*, vol. 347, no. 9003, pp. 735–738, 1996.

[MBG+21] C. Metta, A. Beretta, *et al.*, «Explainable deep image classifiers for skin lesion diagnosis», *arXiv preprint arXiv:2111.11863*, 2021.

[MFF16] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, «Deepfool: A simple and accurate method to fool deep neural networks», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[MFFF17] S.-M. Moosavi-Dezfooli, A. Fawzi, *et al.*, «Universal adversarial perturbations», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[MGFB17] J. H. Metzen, T. Genewein, *et al.*, «On detecting adversarial perturbations», *arXiv preprint arXiv:1702.04267*, 2017.

[MGvDN21] K. Mahmood, D. Gurevin, *et al.*, «Beware the black-box: On the robustness of recent defenses to adversarial examples», *Entropy*, vol. 23, no. 10, p. 1359, 2021.

[MMS+17] A. Madry, A. Makelov, *et al.*, «Towards deep learning models resistant to adversarial attacks», *arXiv preprint arXiv:1706.06083*, 2017.

[MMZ23] A. Mao, M. Mohri, and Y. Zhong, «Cross-entropy loss functions: Theoretical analysis and applications», *arXiv preprint arXiv:2304.07288*, 2023.

[MR22] G. Maimon and L. Rokach, «A universal adversarial policy for text classifiers», *Neural Networks*, vol. 153, pp. 282–291, 2022.

[Nob06] W. S. Noble, «What is a support vector machine?», *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.

[NP21] B. Nandini and R. Puviarasi, «Detection of skin cancer using inception v3 and inception v4 convolutional neural network (cnn) for accuracy improvement», *REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS*, vol. 11, no. 4, pp. 1138–1148, 2021.

[NPS18] P. Napoletano, F. Piccoli, and R. Schettini, «Anomaly detection in nanofibrous materials by cnn-based self-similarity», *Sensors*, vol. 18, no. 1, p. 209, 2018.

[NRT+18] V. Noel Codella, P. Rotemberg, *et al.*, «Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic)», *arXiv preprint arXiv:1902.03368*, 2018.

[NST+18] M.-I. Nicolae, M. Sinn, *et al.*, «Adversarial robustness toolbox v1. 0.0», *arXiv preprint arXiv:1807.01069*, 2018.

[NYC15] A. Nguyen, J. Yosinski, and J. Clune, «Deep neural networks are easily fooled: High confidence predictions for unrecognizable images», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[Orv22] A. O. Orvedal, «Deep learning in health systems: An analysis of adversarial attacks on convolutional neural networks», M.S. thesis, Norwegian University of Science and Technology, 2022.

[OV23] A. Oprea and A. Vassilev, «Adversarial machine learning: A taxonomy and terminology of attacks and mitigations (draft)», National Institute of Standards and Technology, Tech. Rep., 2023.

[PFH+18] S. Palacio, J. Folz, *et al.*, «What do deep networks like to see?», in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3108–3117.

[PGM+19] A. Paszke, S. Gross, *et al.*, «Pytorch: An imperative style, high-performance deep learning library», in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[PMG+17] N. Papernot, P. McDaniel, *et al.*, «Practical black-box attacks against machine learning», in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[PMG16] N. Papernot, P. McDaniel, and I. Goodfellow, «Transferability in machine learning: From phenomena to black-box attacks using adversarial samples», *arXiv preprint arXiv:1605.07277*, 2016.

[PMSW16] N. Papernot, P. McDaniel, *et al.*, «Towards the science of security and privacy in machine learning», *arXiv preprint arXiv:1611.03814*, 2016.

[PP19] R. Poojary and A. Pai, «Comparative study of model optimization techniques in fine-tuned cnn models», in *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, IEEE, 2019, pp. 1–4.

[RBB17] J. Rauber, W. Brendel, and M. Bethge, «Foolbox: A python toolbox to benchmark the robustness of machine learning models», *arXiv preprint arXiv:1707.04131*, 2017.

[RCR05] A. I. Rubin, E. H. Chen, and D. Ratner, «Basal-cell carcinoma», *New England Journal of Medicine*, vol. 353, no. 21, pp. 2262–2269, 2005.

[REGT15] M. R. Roh, P. Eliades, *et al.*, «Genetics of melanocytic nevi», *Pigment cell & melanoma research*, vol. 28, no. 6, pp. 661–672, 2015.

[RG16] L. Rampasek and A. Goldenberg, «Tensorflow: Biology's gateway to deep learning?», *Cell systems*, vol. 2, no. 1, pp. 12–14, 2016.

[SAH+22] S. S. Samsudin, H. Arof, *et al.*, «Skin lesion classification using multi-resolution empirical mode decomposition and local binary pattern», *Plos one*, vol. 17, no. 9, e0274896, 2022.

[Sar21] I. H. Sarker, «Machine learning: Algorithms, real-world applications and research directions», *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[SBG20] K. Sadeghi, A. Banerjee, and S. K. Gupta, «A system-driven taxonomy of attacks and defenses in adversarial machine learning», *IEEE transactions on emerging topics in computational intelligence*, vol. 4, no. 4, pp. 450–467, 2020.

[SC14] S. Samonas and D. Coss, «The cia strikes back: Redefining confidentiality, integrity and availability in security.», *Journal of Information System Security*, vol. 10, no. 3, 2014.

[SLJ+15] C. Szegedy, W. Liu, *et al.*, «Going deeper with convolutions», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[Smi12] D. Smith, «Planning as an iterative process», in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, 2012, pp. 2180–2185.

[Str19] R. Streefkerk, *Qualitative vs. quantitative research*, en, https://www.scribbr.com/methodology/qualitative-quantitative-research/, Accessed: 2023-4-01, Apr. 2019.

[SVI+16] C. Szegedy, V. Vanhoucke, *et al.*, «Rethinking the inception architecture for computer vision», in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[SZS+13] C. Szegedy, W. Zaremba, *et al.*, «Intriguing properties of neural networks», *arXiv preprint arXiv:1312.6199*, 2013.

[SAA22] S. K. Singh, V. Abolghasemi, and M. H. Anisi, «Skin cancer diagnosis based on neutrosophic features with a deep neural network», eng, *Sensors (Basel, Switzerland)*, vol. 22, no. 16, p. 6261, 2022.

[TDL20] N. Thakur, Y. Ding, and B. Li, «Evaluating a simple retraining strategy as a defense against adversarial attacks», *arXiv preprint arXiv:2007.09916*, 2020.

[TKP+17] F. Tramèr, A. Kurakin, *et al.*, «Ensemble adversarial training: Attacks and defenses», *arXiv preprint arXiv:1705.07204*, 2017.

[TRK18] P. Tschandl, C. Rosendahl, and H. Kittler, «The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions», *Scientific data*, vol. 5, no. 1, pp. 1–9, 2018.

[TTT+21] A. N. Tosteson, S. Tapp, *et al.*, «Association of second-opinion strategies in the histopathologic diagnosis of cutaneous melanocytic lesions with diagnostic accuracy and population-level costs», *JAMA dermatology*, vol. 157, no. 9, pp. 1102–1106, 2021.

[TV16] P. Tabacof and E. Valle, «Exploring the space of adversarial images», in *2016 international joint conference on neural networks (IJCNN)*, IEEE, 2016, pp. 426–433.

[WGZ+20] K. Wang, X. Gao, *et al.*, «Pay attention to features, transfer learn faster cnns», in *International conference on learning representations*, 2020.

[XCS19] J. Xu, Z. Cai, and W. Shen, «Using fgsm targeted attack to improve the transferability of adversarial example», in *2019 IEEE 2nd International Conference on Electronics and Communication Engineering (ICECE)*, IEEE, 2019, pp. 20–25.

[YCD+16] L. Yu, H. Chen, *et al.*, «Automated melanoma recognition in dermoscopy images via very deep residual networks», *IEEE transactions on medical imaging*, vol. 36, no. 4, pp. 994–1004, 2016.

[YWW+20] Z. Yin, H. Wang, *et al.*, «Defense against adversarial attacks by low-level image transformations», *International Journal of Intelligent Systems*, vol. 35, no. 10, pp. 1453–1466, 2020.

[ZT+98] D. Ziou, S. Tabbone, *et al.*, «Edge detection techniques-an overview», *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, vol. 8, pp. 537–559, 1998.

[ZZB04] B. Zelger, B. G. Zelger, and W. H. Burgdorf, «Dermatofibroma—a critical evaluation», *International Journal of Surgical Pathology*, vol. 12, no. 4, pp. 333–344, 2004.

[AA05] S. Astner and R. R. Anderson, «Treating vascular lesions», *Dermatologic therapy*, vol. 18, no. 3, pp. 267–281, 2005.

[AAB+16] M. Abadi, A. Agarwal, *et al.*, «Tensorflow: Large-scale machine learning on heterogeneous distributed systems», *arXiv preprint arXiv:1603.04467*, 2016.

[AAL20] C. C. Aggarwal, L.-F. Aggarwal, and Lagerstrom-Fife, *Linear algebra and optimization for machine learning*. Springer, 2020, vol. 156.

Appendix A

Experiment 1: Comprehensive Overview

The following appendix contains the schematic illustration of performing the first experiment on a single adversarial attack for a single skin lesion classifier, depicted in Figure A.1.

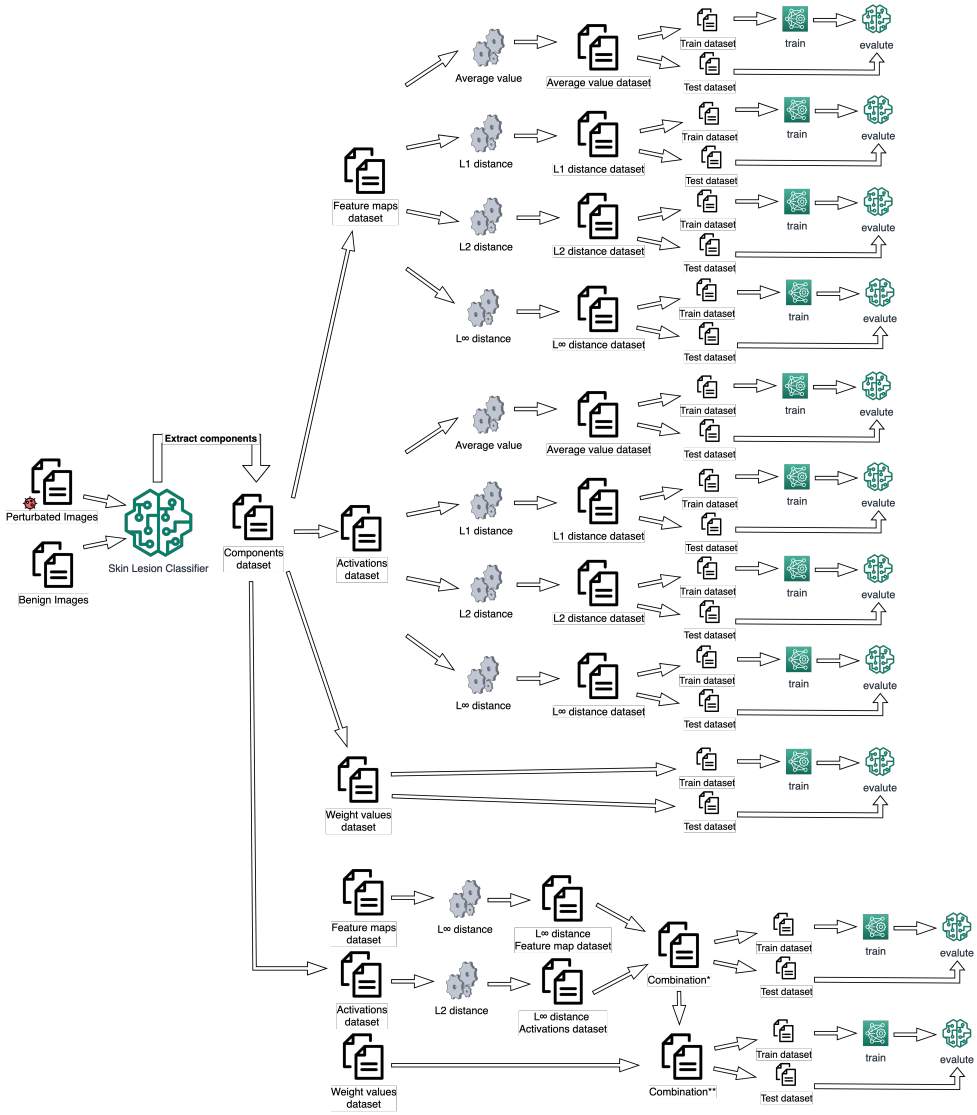


Figure A.1: Schematic illustration of Experiment 1.

Appendix B
Experiment 1: Table results

In Section 6.1, a multitude of different adversarial attacks were performed on the Inception-V3 and the ResNet18 model. The tables show the outcome of experiment 1. The first in the table columns specifies the component of the CNN from which the matrices are collected; the second column explains what metric was used to reduce the matrices into a few scalar values. The third and fourth column represents the detection results from the attached detection model, where the detection model is attached to two different CNN skin lesion classification models trained on two different datasets: ISIC 2018 and ISIC 2019 [NRT+18; KHF20]. The resulting values are TP, TN, FP, FN and Accuracy, outlined in Section 2.2.4. Note: The feature maps and activations include one or more pooling layers in the component.

ResNet18											
Fooling rates: ISIC 2018: 91.38% & ISIC 2019: 84.19%											
Adversarial attack: FGSM											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	1980	1974	7	40	98.83%	1904	1952	29	116	96.38%
	$L1$ distance	1848	1939	42	172	94.65%	1861	1937	44	159	94.93%
	$L2$ distance	1980	1974	7	40	98.83%	1904	1952	29	116	96.38%
	L_∞ distance	1654	1685	296	366	83.45%	1507	1696	285	513	80.05%
Activations	Average value	1904	1922	59	116	95.63%	1881	1915	66	139	94.88%
	$L1$ distance	1891	1921	60	129	95.28%	1883	1912	69	137	94.85%
	$L2$ distance	1806	1908	73	214	92.83%	1847	1934	47	173	94.50%
	L_∞ distance	1660	1729	252	360	84.70%	1496	1662	319	524	78.93%
Dense layers	Weight values	1882	1871	110	138	93.80%	1638	1636	345	382	81.83%
Combinations	Combination*	1907	1930	51	113	95.90%	1913	1948	33	107	96.50%
	Combination**	1981	1962	19	39	98.55%	1950	1966	15	70	97.88%

Table B.1: Results of the Fast Gradient Sign Method (FGSM) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

ResNet18											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: I-FGSM											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	2006	1967	14	14	99.30%	1981	1950	31	39	98.25%
	$L1$ distance	2000	1964	17	20	99.08%	1971	1950	31	49	98.00%
	$L2$ distance	2006	1967	14	14	99.30%	1981	1950	31	39	98.25%
	L_∞ distance	1778	1834	147	242	90.28%	1602	1769	212	418	84.25%
Activations	Average value	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	$L1$ distance	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	$L2$ distance	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	L_∞ distance	2016	1975	6	4	99.75%	2019	1980	1	1	99.95%
Dense layers	Weight values	2019	1980	1	1	99.95%	2019	1980	1	1	99.95%
Combinations	Combination*	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	Combination**	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%

Table B.2: Results of the Iterative Fast Gradient Sign Method (I-FGSM) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Model arhcitecture: ResNet18											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: C&W											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	1236	1266	715	784	62.53%	1159	1138	843	861	57.41%
	$L1$ distance	1229	1364	617	791	64.81%	1218	1200	781	802	60.43%
	$L2$ distance	1223	1272	709	797	62.36%	1172	1148	833	848	57.99%
	L_∞ distance	1952	1786	195	68	93.43%	1867	1637	344	153	87.58%
Activations	Average value	1597	1394	587	423	74.76%	1479	1203	778	541	67.03%
	$L1$ distance	1574	1370	611	446	73.58%	1502	1200	781	518	67.53%
	$L2$ distance	1536	1450	531	484	74.63%	1534	1242	739	486	69.38%
	L_∞ distance	1388	1447	534	632	70.86%	1414	1198	783	606	65.28%
Dense layers	Weight values	1611	1633	348	409	81.08%	1597	1379	602	423	74.38%
Combinations	Combination*	1830	1733	248	190	89.05%	1731	1519	462	289	81.23%
	Combination**	1756	1709	272	264	86.60%	1671	1513	468	349	79.58%

Table B.3: Results of the Carlini & Wagner (C&W) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Model arhcitecture: ResNet18											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: PGD											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	2005	1962	19	15	99.15%	1962	1944	37	58	97.63%
	$L1$ distance	1993	1961	20	27	98.83%	1954	1949	32	66	97.55%
	$L2$ distance	2005	1962	19	15	99.15%	1962	1944	37	58	97.63%
	L_∞ distance	1742	1793	188	278	88.35%	1563	1720	261	457	82.05%
Activations	Average value	2020	1981	0	0	100.00%	2020	1981	0	0	100.00%
	$L1$ distance	2020	1981	0	0	100.00%	2020	1981	0	0	100.00%
	$L2$ distance	2018	1981	0	2	99.95%	2020	1981	0	0	100.00%
	L_∞ distance	2013	1979	2	7	99.78%	2018	1980	1	2	99.93%
Dense layers	Weight values	2018	1980	1	2	99.93%	2015	1981	0	5	99.88%
Combinations	Combination*	2018	1981	0	2	99.95%	2020	1981	0	0	100.00%
	Combination**	2018	1981	0	2	99.95%	2020	1981	0	0	100.00%

Table B.4: Results of the Projected Gradient Descent (PGD) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Inception-V3											
Fooling rates: ISIC 2018: 80.52% & ISIC 2019: 69.90%											
Adversarial attack: FGSM											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	2018	1979	2	2	99.90%	2017	1978	3	3	99.85%
	$L1$ distance	2019	1980	1	1	99.95%	2017	1979	2	3	99.88%
	$L2$ distance	2019	1979	2	1	99.93%	2017	1978	3	3	99.85%
	L_∞ distance	1955	1932	49	65	97.15%	1944	1910	71	76	96.33%
Activations	Average value	2019	1981	0	1	99.98%	2015	1976	5	5	99.75%
	$L1$ distance	2019	1981	0	1	99.98%	2015	1976	5	5	99.75%
	$L2$ distance	2017	1979	2	3	99.88%	2017	1980	1	3	99.90%
	L_∞ distance	1948	1917	64	72	96.60%	1934	1919	62	86	96.30%
Dense layers	Weight values	1999	1954	27	21	98.80%	1742	1721	260	278	86.55%
Combinations	Combination*	2014	1979	2	6	99.80%	2015	1981	0	5	99.88%
	Combination**	2019	1977	4	1	99.88%	2015	1980	1	5	99.85%

Table B.5: Results of the Fast Gradient Sign method (FGSM) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Inception-V3											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: I-FGSM											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	2020	1980	1	0	99.98%	2018	1979	2	2	99.90%
	$L1$ distance	2020	1979	2	0	99.95%	2017	1981	0	3	99.93%
	$L2$ distance	2020	1980	1	0	99.98%	2018	1980	1	2	99.93%
	L_∞ distance	2003	1957	24	17	98.98%	1991	1961	20	29	98.78%
Activations	Average value	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	$L1$ distance	2020	1980	1	0	99.98%	2020	1981	0	0	100.00%
	$L2$ distance	2020	1981	0	0	100.00%	2020	1981	0	0	100.00%
	L_∞ distance	2015	1975	6	5	99.73%	2009	1971	10	11	99.48%
Dense layers	Weight values	2020	1981	0	0	100.00%	2018	1980	1	2	99.93%
Combinations	Combination*	2020	1981	0	0	100.00%	2020	1981	0	0	100.00%
	Combination**	2020	1981	0	0	100.00%	2020	1981	0	0	100.00%

Table B.6: Results of the Iterative Fast Gradient Sign method (I-FGSM) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Inception-V3											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: C&W											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	1709	1560	421	311	81.70%	1699	1501	480	321	79.98%
	$L1$ distance	1644	1541	440	376	79.61%	1717	1511	470	303	80.68%
	$L2$ distance	1705	1555	426	315	81.48%	1709	1504	477	311	80.30%
	L_∞ distance	1846	1767	214	174	90.30%	1748	1606	375	272	83.83%
Activations	Average value	1726	1604	377	294	83.23%	1726	1528	453	294	81.33%
	$L1$ distance	1716	1602	379	304	82.93%	1723	1517	464	297	80.98%
	$L2$ distance	1706	1585	396	314	82.25%	1732	1516	465	288	81.18%
	L_∞ distance	1398	1317	664	622	67.86%	1527	1335	646	493	71.53%
Dense layers	Weight values	1696	1639	342	324	83.35%	1582	1533	448	438	77.86%
Combinations	Combination*	1783	1694	287	237	86.90%	1726	1574	407	294	82.48%
	Combination**	1832	1818	163	188	91.23%	1727	1688	293	293	85.35%

Table B.7: Results of the Carlini & Wagner (CW) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Inception-V3											
Fooling rates: ISIC 2018: 100% & ISIC 2019: 100%											
Adversarial attack: PGD (L_∞)											
Component	**Metric**	**ISIC 2018**					**ISIC 2019**				
		TP	**TN**	**FP**	**FN**	**Acc**	**TP**	**TN**	**FP**	**FN**	**Acc**
Feature maps	Average value	2020	1977	4	0	99.90%	2018	1978	3	2	99.88%
	$L1$ distance	2020	1978	3	0	99.93%	2018	1977	4	2	99.85%
	$L2$ distance	2020	1977	4	0	99.90%	2018	1978	3	2	99.88%
	L_∞ distance	1995	1952	29	25	98.65%	1991	1952	29	29	98.55%
Activations	Average value	2019	1981	0	1	99.98%	2020	1980	1	0	99.98%
	$L1$ distance	2019	1981	0	1	99.98%	2020	1980	1	0	99.98%
	$L2$ distance	2019	1981	0	1	99.98%	2019	1981	0	1	99.98%
	L_∞ distance	2009	1968	13	11	99.40%	2013	1973	8	7	99.63%
Dense layers	Weight values	2018	1981	0	2	99.95%	2017	1979	2	3	99.88%
Combinations	Combination*	2019	1981	0	1	99.98%	2019	1981	0	1	99.98%
	Combination**	2019	1981	0	1	99.98%	2019	1981	0	1	99.98%

Table B.8: Results of Projected Gradient Descent (PGD) adversarial attack from experiment 1.
(*) Uses a combination of Activations $L2$ and Feature maps L_∞.
(**) Uses a combination of Activations $L2$, Feature maps L_∞ and dense layer weigth values.

Appendix C

Experiment 2: Heatmaps

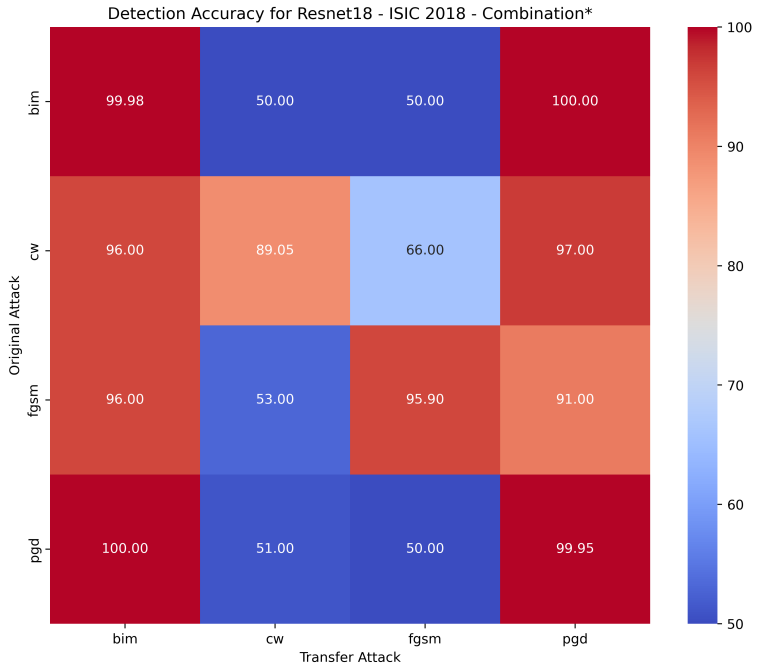


Figure C.1: Heatmap visualization on the transferability of adversarial attacks for the Combination* features (activations $L2$ distance and feature maps L_∞ distance) on ResNet-18 architecture trained on ISIC 2018. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

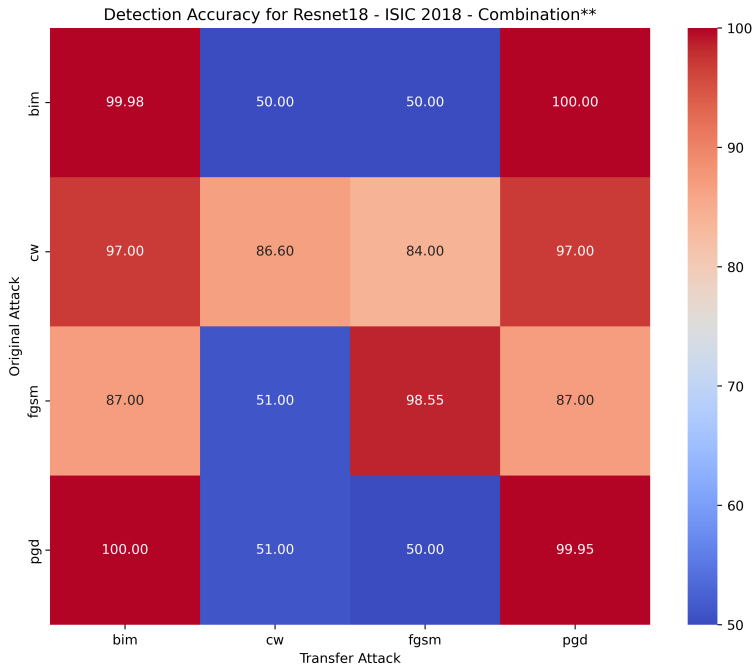


Figure C.2: Heatmap visualization on transferability of adversarial attacks for the Combination** features (activations $L2$ distance, feature maps L_∞ distance, and dense layer weights) on ResNet-18 architecture trained on ISIC 2018. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

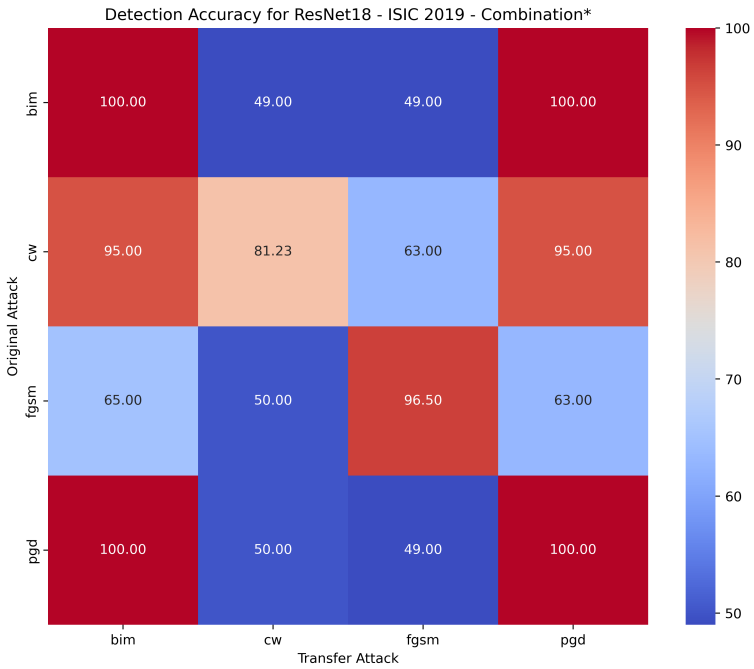


Figure C.3: Heatmap visualization on the transferability of adversarial attacks for the Combination* features (activations $L2$ distance and feature maps L_∞ distance) on ResNet-18 architecture trained on ISIC 2019. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

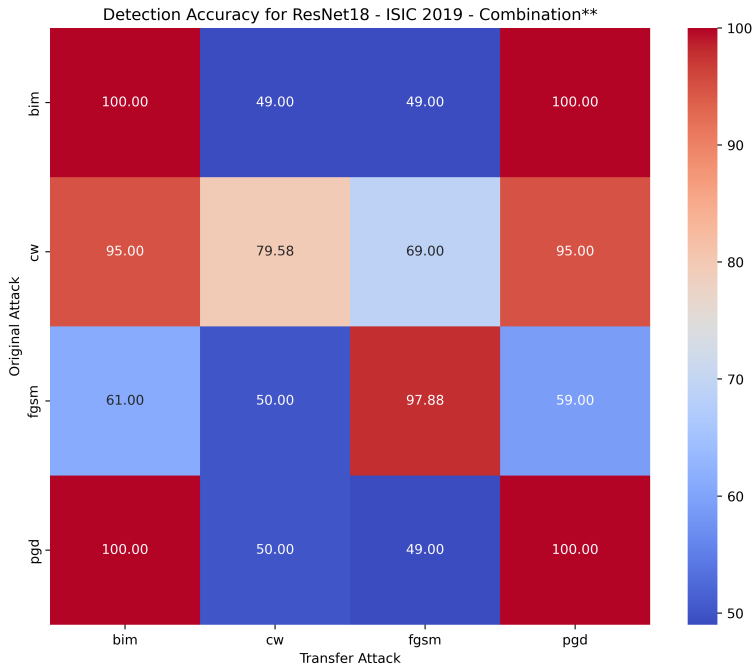


Figure C.4: Heatmap visualization on transferability of adversarial attacks for the Combination** features (activations $L2$ distance, feature maps L_∞ distance, and dense layer weights) on ResNet-18 architecture trained on ISIC 2019. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

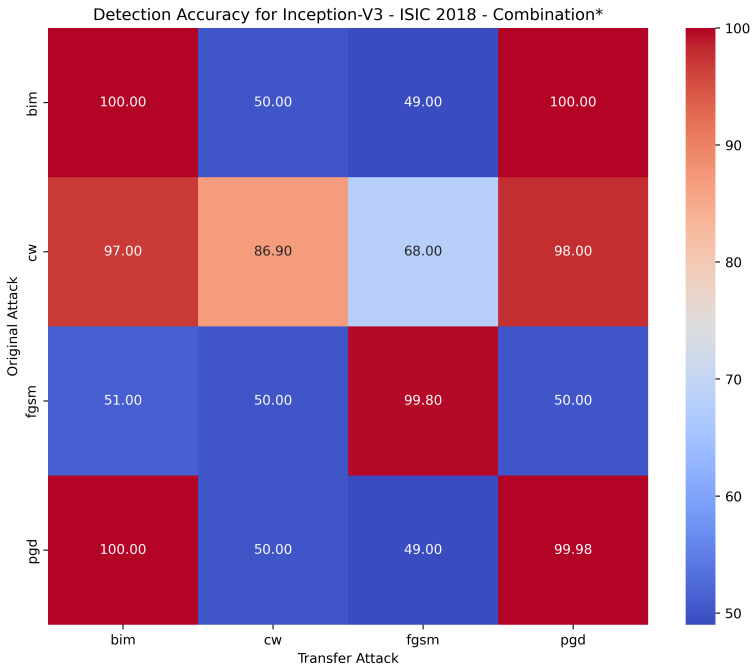


Figure C.5: Heatmap visualization on the transferability of adversarial attacks for the Combination* features (activations $L2$ distance and feature maps L_∞ distance) on Inception V3 architecture trained on ISIC 2018. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

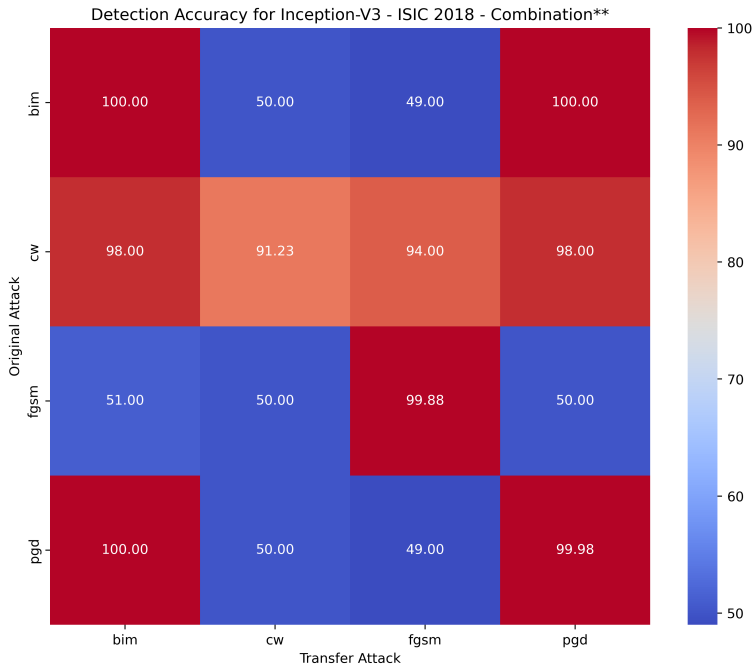


Figure C.6: Heatmap visualization on transferability of adversarial attacks for the Combination** features (activations $L2$ distance, feature maps L_∞ distance, and dense layer weights) on Inception V3 architecture trained on ISIC 2018. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

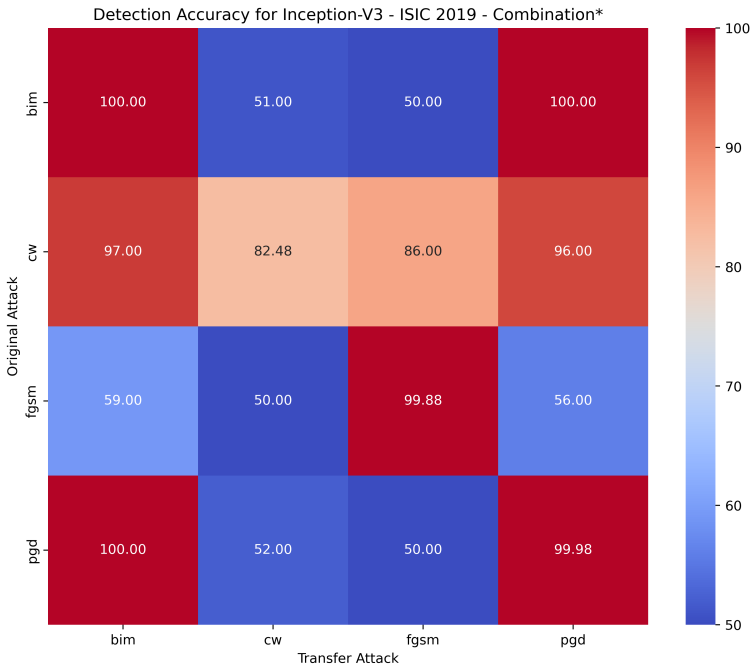


Figure C.7: Heatmap visualization on the transferability of adversarial attacks for the Combination* features (activations $L2$ distance and feature maps L_∞ distance) on Inception V3 architecture trained on ISIC 2019. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

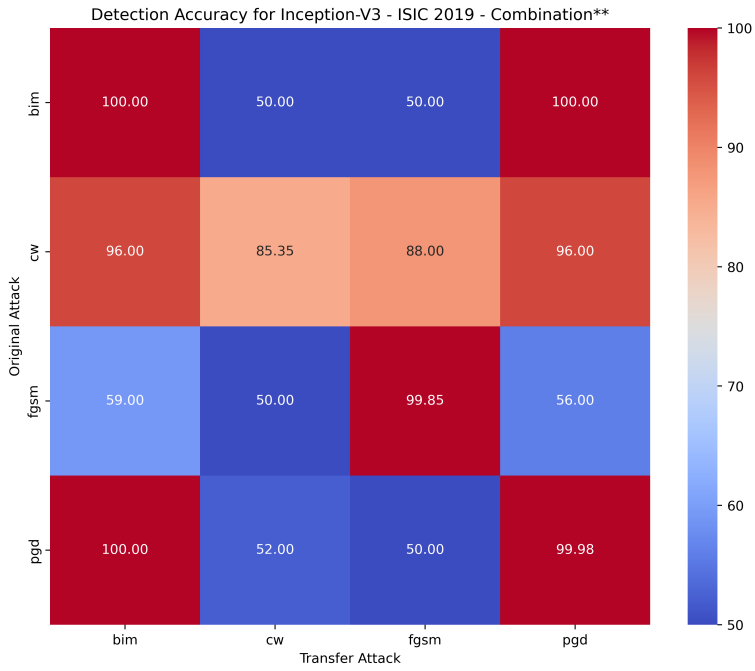


Figure C.8: Heatmap visualization on transferability of adversarial attacks for the Combination** features (activations $L2$ distance, feature maps L_∞ distance, and dense layer weights) on Inception V3 architecture trained on ISIC 2019. The adversarial attack type used to train the detector model is indicated on the vertical axis, while the adversarial attack it was tested on is shown on the horizontal axis. The color gradation within each cell symbolizes the degree of transferability, with warmer colors corresponding to higher detection rates.

NTNU

Norwegian University of
Science and Technology