ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF ELECTRONIC ENGINEERING

SECOND CYCLE DEGREE

Master's Degree in Electronic Engineering

Statistical Characterization of Deep Neural Network Intermediate Activations for Adversarial Attacks Detection

Supervisor

Prof. Mauro Mangia

Defended by

MohammadHossein Rezaei

Co-Supervisors

Dott. Lorenzo Capelli
Prof. Riccardo Rovatti
Dott. Leandro De Souza Rosa

First Session / July / 2026

Academic Year 2025/2026

Contents

Abstract

Deep neural networks have achieved high performance in image classification tasks, yet they remain vulnerable to adversarial examples, where small and often imperceptible input perturbations can lead to incorrect predictions. Conventional robustness evaluation usually focuses on the final model output, such as predicted labels or confidence scores. However, these quantities provide limited information about how adversarial inputs are transformed inside the network before classification. This thesis investigates whether intermediate activations of convolutional neural networks can be statistically characterized and used for adversarial attack detection.

The proposed approach first extracts layer activations and flattens them into activation vectors. For each selected layer, a layer-wise SVD projection basis is obtained from the corresponding layer transformation. For convolutional layers, this transformation is represented through an equivalent operator form, while linear layers are handled through their weight transformation. Sample activation vectors are then projected onto the corresponding SVD basis to obtain corevectors. Based on these projected representations, a tail-energy statistic is introduced to measure the energy of a sample in the lower singular-value components of the SVD basis. The central hypothesis is that adversarial examples may follow abnormal internal trajectories and therefore produce distinctive energy patterns in selected intermediate layers.

The method is evaluated using standard and adversarially trained Wide Residual Networks under an adversarial robustness framework involving AutoAttack components. The experimental analysis includes output-level confidence, calibration, risk-coverage behavior, corevector evaluation, and single-layer and ensemble tail-energy detectors. The results show that intermediate representations contain useful statistical information for detecting certain adversarial attacks, particularly APGD-based attacks. However, detection remains more difficult for FAB and Square Attack, whose energy distributions overlap more strongly with clean samples. Overall, the study demonstrates that internal activation geometry provides a meaningful direction for adversarial detection, while also highlighting the need for broader attack-aware and layer-wise analysis in future robustness research.

Acknowledgments

My sincere thanks go to Professor Mauro Mangia, Professor Riccardo Rovatti, Professor Leandro De Souza Rosa, and Professor Lorenzo Capelli for presenting me with different challenges on a daily basis but refusing to leave me alone with them. Their guidance and insight were invaluable throughout this work.

I am deeply grateful to my family for their continuous support throughout the years, especially during this academic journey. This achievement would not have been possible without their encouragement and help.

Finally, I would like to thank my friends for their encouragement and companionship throughout these years. A special thanks goes to my partner for her patience, understanding, and continuous support throughout my studies.

Last but not least, I also thank myself for somehow surviving this thesis and not giving up along the way. Obviously, without me, this thesis would not have been finished.

This achievement would not have been possible without all of us.

Introduction

Deep neural networks have become one of the most successful tools in modern artificial intelligence. They can classify images, recognize speech, detect patterns, and solve complex tasks with a level of accuracy that earlier machine learning methods could not reach. Their success comes from their ability to build internal representations of data across many layers. In early layers, a network may detect simple visual patterns such as edges, colors, and textures. In deeper layers, these patterns are gradually transformed into more abstract features that support the final classification decision.

However, this success also hides an important weakness. A neural network can classify an image correctly under normal conditions, yet fail completely when the same image is changed by a very small perturbation. These modified inputs are known as adversarial examples. To a human observer, an adversarial image may look almost identical to the clean image. To the model, however, it may become something entirely different. This creates a serious problem for the reliability of deep learning systems, especially in applications where incorrect predictions can have practical consequences.

This gap led to the development of adversarial robustness studies to examine the accuracy of classifying models within a network. Specifically, adversarial robustness studies whether the predicted label changes, the confidence score decreases, or the attack succeeds under a particular threat model. Although this information is useful, it gives only a limited view of the network's behavior. The final output tells us what decision the model made, but it does not explain how the input moved through the internal layers before reaching that decision. In other words, it shows the result of the computation, but not the path taken inside the model.

This thesis is based on the idea that adversarial behavior should also be studied inside the network. If clean and adversarial images can appear similar in pixel space but produce different predictions, then the difference may be visible in the hidden representations of the model. The internal activations of a deep neural network may therefore contain information that is not available from the final output alone. Studying these activations can help reveal whether adversarial inputs follow abnormal internal trajectories through the network.

The main objective of this thesis is to investigate whether intermediate activations of deep neural networks can be statistically characterized and used for adversarial attack detection. The work focuses on convolutional neural networks, where layer activations can be extracted as feature maps and flattened into activation vectors. For each selected layer, a layer-wise SVD basis is obtained from the corresponding layer transformation. Sample activations are projected onto this basis to obtain corevectors, which describe the coordinates of the internal representation in the SVD space. This makes it possible to study the geometry of the representation space associated with each selected layer.

A central method used in this thesis is Singular Value Decomposition. SVD provides a way to identify dominant and lower singular-value directions associated with the selected layer transformation. The dominant directions correspond to the largest singular values, while the

lower singular-value directions form the tail of the spectrum. This thesis examines whether adversarial examples produce unusual energy in these tail directions. The resulting statistic, called tail energy, measures how much of a sample representation lies in the lower singular-value part of the SVD basis.

The motivation behind this approach is simple but important. Clean samples are used to define the reference behavior of the tail-energy score. Adversarial samples, by contrast, may produce different internal activation patterns after projection onto the layer-wise SVD basis. These abnormal changes may not always appear clearly in the final output or in the dominant representation directions. They may instead become more visible in the tail of the internal representation space. Tail energy is therefore used as a representation-level signal for detecting suspicious inputs.

The experimental part of the thesis evaluates this idea under established adversarial robustness settings. The study uses standard and adversarially trained Wide Residual Networks, together with attacks from the AutoAttack framework. The analysis first examines model behavior using output-level indicators such as confidence, calibration, and risk coverage. It then moves to corevector evaluation, where intermediate activations are flattened, projected onto layer-wise SVD bases, and analyzed through their energy in selected singular directions. This progression allows the thesis to compare what can be learned from the final prediction with what can be learned from the internal representation of the model.

The detector developed in this thesis is based on the tail energy of selected intermediate layers. Instead of modifying the classifier itself, the detector observes the internal representation produced by the model and decides whether the input should be accepted or rejected. A sample is treated as suspicious when its tail-energy score exceeds a fixed threshold. This creates a simple and interpretable detection rule. The approach is then evaluated in single-layer, accuracy-matched, and ensemble settings in order to study both its strengths and its limitations.

This thesis makes three main contributions. First, it provides a structured mathematical description of CNN internal representations using activations, flattened activation vectors, layer-wise SVD projection bases, SVD-projected corevectors, and tail energy. Second, it investigates how adversarial samples can differ from clean samples in the lower singular-value directions of intermediate SVD-projected representations. Third, it develops and evaluates a tail-energy detector that uses this representation-level behavior to identify adversarial inputs.

The thesis is organized as follows. Chapter 1 reviews the development of neural networks, from early logical and biological models to deep learning architectures, and explains how adversarial examples revealed the fragility of modern neural networks. Chapter 2 presents the main adversarial attack and defense methods, including L-BFGS, FGSM, DeepFool, JSMA, defensive distillation, Carlini and Wagner attacks, and Projected Gradient Descent. It also introduces the robustness-evaluation framework used later in the thesis, including reliable robustness evaluation, AutoAttack, RobustBench, and Wide Residual Networks. Chapter 3 introduces the mathematical model of CNN internal representations, including layer activations, flattened activation vectors, SVD-projected corevectors, and tail energy. The following four chapters present the experimental setup, model analysis, corevector evaluation, tail-energy detector, and ensemble detector.

Overall, this thesis argues that the hidden layers of a neural network should not be treated as a black box. They are not merely intermediate steps between input and output. They contain

structured information about how the model represents the world, and this structure can reveal when an input is behaving abnormally. By looking inside the network rather than only at its final answer, this work aims to contribute to the broader goal of building deep learning systems that are not only accurate, but also more reliable, interpretable, and resistant to adversarial manipulation.

Chapter 1: From Neural Networks to Adversarial Vulnerability

Artificial intelligence began as an effort to translate human reasoning into computation. Early researchers believed that thought could be expressed through logic and that the brain might be modeled as a network of simple decision units. From this idea came the first artificial neurons, designed to reproduce how biological neurons process signals. What started as a mathematical exploration gradually became a search for systems that could adjust their behavior through experience. The field's development from logical machines to adaptive networks reflects a broader attempt to understand intelligence as a process rather than a fixed set of rules.

The history of neural networks shows a continuous exchange between biology and engineering. Findings on how real neurons strengthen their connections inspired algorithms that allowed machines to adjust internal weights over time. Models such as the Perceptron and later multi-layered architectures demonstrated that computers could identify patterns and improve through feedback, although within clear limits. Each step built on earlier insights: logic led to memory, memory to learning, and learning to recognition. The link between neuroscience and computation remained central, guiding efforts to design systems that reflect both biological organization and computational efficiency.

As computing resources expanded, so did the scale of these systems. Deep neural networks introduced multiple layers of processing, allowing data to be represented at different levels of abstraction. This structure improved performance in areas such as image and speech recognition, but it also raised new questions. The same depth that increased their effectiveness made them less transparent, and their reliance on statistical patterns rather than understanding revealed important constraints. The contrast between accuracy and interpretability became a key issue for research in artificial intelligence.

This chapter follows that development in three main parts. The first part reviews the logical and biological origins of neural networks, from McCulloch and Pitts to Hebb, Rosenblatt, and early work on pattern recognition. The second examines the rise of deeper architectures, focusing on backpropagation, convolutional models, and the transition to large-scale applications. The third discusses the limitations of these systems, showing how adversarial examples and structural sensitivity expose their weaknesses. Together, these sections trace how artificial intelligence evolved from early logical formulations to complex models that still face fundamental questions about stability and understanding.

Part I — Neural Networks: From Logic to Learning

Introduction

The roots of artificial intelligence lie in an effort to model the brain as a system of logic and computation. Before machines could learn or adapt, researchers needed to understand how thought itself could be represented mathematically. Early pioneers believed that the brain could be described as a network of logical elements that processed information through simple binary operations. This idea sparked the creation of artificial neurons, which eventually evolved into full neural networks. The first generation of models explored logic, memory, and the possibility of learning, gradually shifting from static representations of intelligence to systems that could adapt through experience. This section traces that evolution, showing the early development of single-layer neural networks.

Early Logical Models of the Brain

The first serious attempt to model artificial intelligence drew inspiration from the structure of the human brain. In 1943, Warren McCulloch and Walter Pitts proposed a model of the brain as a logical network [1]. They argued that neurons could be represented as binary devices that either fired or remained inactive depending on whether the input reached a threshold. Each neuron carried out a logical function, such as "and" or "or," and the connections between them determined how information flowed through the system. This made the brain appear, at least in principle, as a system that could be described using the tools of mathematics and logic. The model was groundbreaking because it suggested that cognition itself could be reduced to a set of formal rules.

This binary representation of the brain closely mirrored the architecture of digital computers, which operate on sequences of ones and zeros. The appeal of this similarity was clear. If neurons and their connections could be represented as logic gates, then building a machine that mimicked the brain might simply require arranging enough gates in the right pattern. McCulloch and Pitts introduced the idea that thinking was essentially a matter of computation. Their model reduced the complexity of the brain to a manageable set of equations and functions. While oversimplified, it provided a starting point for researchers to imagine how machines could process information in ways that resembled human cognition.

To replicate the full function of the brain, three core components were deemed necessary. The first was logical processing, which relied on the binary model of neural activity as true or false, one or zero. The second was memory, modeled through cyclic networks where information could loop back on itself, enabling repetition and reinforcement of patterns. The third, and most elusive, was learning. At this stage, learning was beyond the reach of computers. Machines could only follow predetermined rules without deviation, making their behavior entirely predictable. They could simulate logic and store memory, but they could not adapt or generate new knowledge. This limitation underscored the gap between human intelligence and machine capabilities, pushing researchers to search for mechanisms that could give machines the power to learn from their environment.

These early logical models established the foundation for the next wave of discoveries. They

demonstrated that it was possible to map brain activity into formal structures, but they also highlighted what was missing. Memory and logic alone could not explain how humans improved over time or adapted to new challenges. The rigid predictability of these systems made them poor candidates for real intelligence. This realization paved the way for Hebb and others to investigate how experience might change the connections between neurons, introducing the concept of learning as a fundamental requirement for artificial intelligence.

Hebbian Learning and the Concept of Memory

Donald Hebb's work in 1949 marked a turning point in the study of intelligence because it directly addressed the issue of learning [2]. While earlier models focused on logic and memory as static processes, Hebb argued that experience actively changed the physical structure of the brain. He proposed that when one neuron repeatedly excites another, the connection between them strengthens. This process, which later became known as Hebbian learning, provided a biological explanation for how memories form. Instead of treating the brain as a fixed logical circuit, Hebb described it as a dynamic system that reorganizes itself based on activity. His theory suggested that the brain could adapt to new information, laying the foundation for models that aimed to replicate this adaptability in machines.

The key idea in Hebb's model was the concept of cell assembly. A cell assembly forms when groups of neurons repeatedly fire together in response to a stimulus, eventually creating a closed system capable of acting as a unit. This assembly represents the formation of a memory, as the network of neurons becomes tuned to recognize and respond to specific inputs. When multiple cell assemblies interact, they create what Hebb called a phase sequence, a chain of activity that represents more complex thought patterns. Phase sequences allow the brain to combine simple memories into more elaborate structures, such as problem solving or reasoning. Unlike the rigid binary models of McCulloch and Pitts, Hebb's framework explained how the brain could store knowledge in a flexible way that changed over time.

What made Hebb's theory revolutionary was its potential application to machines. If researchers could design systems where connections between artificial units strengthened with repeated activation, they could simulate learning. Instead of preprogramming every possible response, the machine could adapt based on experience. This would transform computers from passive rule-followers into active learners. Although Hebb's work remained largely theoretical in his time, it inspired future generations to design artificial neural networks that incorporated the principle of connection strengthening. The idea that *neurons that fire together wire together* became one of the most influential concepts in neuroscience and artificial intelligence, bridging biology and computer science in a way that continues to shape the field.

Rosenblatt and the Perceptron

Frank Rosenblatt advanced the study of artificial intelligence by transforming Hebb's theoretical ideas into a working model that could be tested on machines [3]. He focused on how information was stored and used for recognition, emphasizing that learning was not just about static assemblies but about probabilities of activation. According to Rosenblatt, memory resided in the connections between neurons, and learning was the process of adjusting these connections

over time. This made his model more dynamic than earlier representations of the brain, as it captured both the randomness and variability of neural activity. Instead of relying purely on logic, Rosenblatt proposed a system that could respond differently based on experience, moving closer to how biological brains actually function. His work marked an important shift from rigid models to adaptable systems capable of improvement.

The Perceptron, which Rosenblatt introduced in 1958, was the first trainable neural network. It was designed with three main layers of units: sensory units, association units, and response units. The sensory units detected simple patterns from input data, such as light and dark spots, and transmitted these signals to the association units. Each association unit combined inputs from multiple sensory units and compared the total signal to a threshold value. If the signal exceeded the threshold, it was passed on to the response units. The response units then generated an output, which represented the system's decision. In this way, the Perceptron could simulate a very basic form of recognition, turning input patterns into outputs that resembled behavioral responses Figure 1.

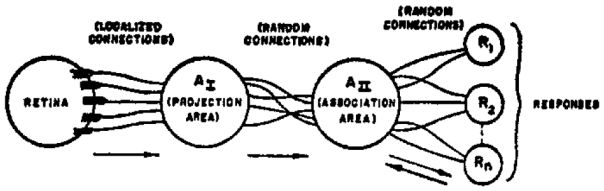


Figure 1: Organization of a perceptron

What made the Perceptron remarkable was its ability to learn through reinforcement. When the machine produced an incorrect output, the weights of the connections between units were adjusted to reduce the error. Over time, these adjustments allowed the system to improve performance and adapt to new patterns, even though the same input could lead to different weight distributions across different machines. This showed that the Perceptron could memorize training data, but it also revealed a limitation: it struggled to generalize beyond what it had learned. Despite this, the Perceptron became a landmark achievement. It demonstrated that artificial systems could be trained through trial and error rather than being fully programmed in advance. Rosenblatt's work inspired decades of research, and even though later critiques exposed its shortcomings, the Perceptron remains a foundational model in the history of machine learning.

Samuel and Machine Learning through Games

Arthur Samuel was one of the first researchers to show that a computer could improve its performance through experience rather than by being reprogrammed [4]. His work in the late 1950s introduced the concept of machine learning in practice, long before the term became widely used. Samuel chose the game of checkers as his testing ground, since the rules were well defined yet the strategies required for strong play were complex. In 1959, he developed a checkers-playing program that could simulate potential moves, evaluate the likely outcomes, and select the most advantageous choice. The program represented a shift from theoretical discussions of intelligence to a concrete example where a machine could learn to perform a task better over time. Unlike earlier models, which remained static, Samuel's machine demonstrated progress through repeated play, moving closer to the human capacity for skill development.

At the core of Samuel's system was the idea of rote learning, in which the machine stored past board positions along with their evaluations. When the same or similar positions appeared in later games, the machine could reuse this stored knowledge to make quicker and more effective decisions. However, memory limitations posed a serious challenge. The computer could not store every possible board configuration, so it had to prioritize valuable positions and discard redundant or less useful ones. To address this, Samuel introduced a method of pruning the stored positions, ensuring that memory was used efficiently. This feature made the program more practical while still allowing it to learn from past experience. The idea that a computer could remember and reuse past encounters was crucial for later machine learning systems, which rely heavily on data storage and retrieval.

Samuel also developed an evaluation function to enhance the system's learning capacity. This function acted as a mathematical formula that judged how strong or weak a given board position was. After each game, the program adjusted its evaluation function based on the results, essentially refining its strategy through trial and error. This created a feedback loop in which the machine's future choices were shaped by its past successes and failures. The evaluation function enabled the computer to generalize from experience rather than simply repeating memorized moves. When tested, the version with the evaluation function significantly outperformed the base version that relied only on rote learning. Samuel's work demonstrated that a computer could learn to play checkers at a level that challenged, and in some cases even surpassed, human players. More broadly, it proved that machines could adapt and improve autonomously, opening the door for future research into general learning algorithms.

Pattern Recognition and the Work of Bledsoe and Selfridge

While Samuel focused on teaching machines to play games, other researchers in the late 1950s turned to the problem of recognition. The ability to identify symbols, letters, and images was seen as essential if machines were to eventually interact with the real world. In 1959, Bledsoe and Browning designed a system aimed at recognizing visual patterns such as handwritten and printed numbers [5]. Their innovation lay in using a photocell mosaic that converted visual input into data a computer could process. Each photocell measured brightness and encoded light and dark regions as binary values, creating a memory matrix that represented the image. This matrix was then compared to stored patterns, and each possible match received a score. By analyzing these scores, the machine selected the most likely match, even when the input was imperfect. This marked one of the first practical applications of machine learning to visual recognition, a step that foreshadowed modern optical character recognition.

What made Bledsoe and Browning's work significant was its departure from earlier rigid methods. Before their system, recognition relied on exact matching techniques such as magnetic ink characters or precise template alignment. These methods worked only when inputs were clean and uniform, which limited their usefulness in real environments where handwriting or printing was often irregular. By contrast, their memory-based recognition system allowed for flexibility. It could process imperfect or noisy inputs, much like humans can recognize familiar words even when written messily. This attempt to model generalization made their system more human-like and revealed how machines might eventually move beyond rigid symbol matching. Although still limited in scope, the project showed that computers could be trained to reflect the way people interpret ambiguous or unclear information.

At the same time, Oliver Selfridge introduced a different but complementary approach to recognition known as the Pandemonium model [6]. Rather than focusing on photocells and stored memory, Selfridge theorized a hierarchical system inspired by parallel processing. In his model, multiple "demons" worked together in layers to recognize patterns. Feature demons in the first layer detected simple elements such as lines, angles, or curves. These signals were passed upward to cognitive demons, each of which represented a possible letter or symbol and responded according to how well the detected features matched its template. The cognitive demons competed for attention, and their collective outputs were assessed by a final decision demon, which selected the strongest candidate as the recognized symbol. This layered structure allowed the system to break down complex recognition tasks into smaller, manageable steps. By simulating how features combine into more complex patterns, the Pandemonium model became one of the earliest representations of hierarchical feature extraction, an idea that would later become central to computer vision and neural networks.

Biological Foundations in Hubel and Wiesel's Discoveries

By the early 1960s, the theoretical models of recognition developed by Bledsoe and Selfridge gained new credibility when experimental neuroscience produced similar findings. In 1962, David Hubel and Torsten Wiesel conducted groundbreaking studies on the visual cortex of cats [7]. They discovered that neurons in the cat's brain responded selectively to specific visual features, such as edges, orientations, and movement directions. These responses were not random but organized hierarchically, with simpler cells detecting basic features and more complex cells combining them into higher-order structures. This organization suggested that the brain itself relied on a layered system of feature extraction, closely resembling the architecture of Selfridge's Pandemonium model. The discovery confirmed that hierarchical recognition was not just a clever engineering trick but a biological reality.

Hubel and Wiesel's findings carried deep implications for both neuroscience and artificial intelligence. Their experiments showed that perception did not occur all at once but was instead built up step by step, with information processed at different levels of complexity. A line detected by one group of neurons might be combined with other lines to form a shape, which could then be recognized as an object by higher layers. This process mirrored the way machines could combine features to recognize symbols or images. The idea that the brain itself worked through layers of processing gave researchers in computer science confidence that their layered recognition models were moving in the right direction. Biological validation encouraged further exploration of artificial systems that attempted to mimic these mechanisms.

The influence of Hubel and Wiesel extended far beyond their immediate findings. Their work laid the foundation for computer vision and, decades later, for convolutional neural networks, which are now central to image recognition in artificial intelligence. By demonstrating that real neurons operate hierarchically, they bridged the gap between biological research and computational modeling. This connection gave engineers a biological justification for designing artificial systems that used layered architectures. It also emphasized the importance of feature extraction, which remains one of the most critical concepts in modern machine learning. Hubel and Wiesel thus provided both inspiration and scientific grounding for the next generation of artificial neural networks, ensuring that AI research stayed closely tied to the study of human and animal cognition.

Minsky, Papert, and the Limitations of the Perceptron

Despite the excitement surrounding Rosenblatt's Perceptron, by the late 1960s doubts began to surface about how powerful this model really was. Marvin Minsky and Seymour Papert, two leading figures in artificial intelligence research, set out to evaluate the Perceptron more systematically. In their influential 1969 book, they provided rigorous mathematical proofs that revealed the Perceptron's limitations [8]. They showed that while the model could solve linear classification problems, it could not handle more complex tasks that required non-linear boundaries. For example, a Perceptron could classify inputs separated by a straight line but failed to manage problems such as determining whether a number of inputs was even or odd. This critique challenged the optimistic claims surrounding Rosenblatt's model and forced the research community to reconsider its assumptions.

The essence of Minsky and Papert's argument was structural rather than computational. They demonstrated that the single-layer architecture of the Perceptron imposed inherent restrictions on what it could achieve. Linear problems, like distinguishing between two simple categories, fit neatly within its capabilities. But non-linear problems, such as parity or connectedness, demanded a more sophisticated system. These limitations were not the result of insufficient training or poor implementation but were built into the design itself. By exposing this weakness, Minsky and Papert highlighted the need for more complex, multi-layered systems. However, at the time, no practical alternatives had been developed, which left the field at an impasse.

Part II — Deep Neural Networks: The Age of Depth and Hierarchies

Introduction

Deep neural networks (DNNs) are systems with multiple layers that process data through increasing levels of abstraction. This depth allows them to learn non-linear relationships, capture complex patterns, and build hierarchical representations. Moving from shallow to deep architectures transformed AI by vastly expanding what machines could learn. The rediscovery of backpropagation in the 1980s revived neural networks, while Fukushima's Neocognitron and LeCun's convolutional network showed how layers could extract visual features automatically. Later breakthroughs like deep belief networks and AlexNet proved that depth delivers exceptional performance. This section explores that evolution and how deep learning became the foundation of modern AI.

Backpropagation and the Revival of Neural Networks

By the mid-1980s, the limitations of single-layer networks identified by Minsky and Papert still posed a serious challenge. However, this situation began to change with the rediscovery and popularization of the backpropagation algorithm and the development of deep neural networks. In 1986, David Rumelhart, Geoffrey Hinton, and Ronald Williams published work that demonstrated how multi-layer neural networks could be effectively trained using backpropagation [9]. Their breakthrough provided a method for adjusting weights in networks with hidden layers, which allowed the system to capture complex, non-linear relationships in data.

The backpropagation process worked by introducing hidden layers between the input and output layers. When data entered the network, it was processed through these layers, which transformed the inputs into intermediate representations before producing an output. The network then compared this output with the correct answer, calculating the difference as an error. This error was propagated backward through the network, allowing each weight to be adjusted in proportion to its contribution to the mistake. The method relied on gradient descent, an optimization technique that measured how much the error changed relative to each weight. By iteratively updating the weights, the network gradually improved its ability to generate correct outputs. This process was not biologically identical to how neurons function, but it provided a powerful computational tool that gave artificial networks far greater flexibility.

The success of backpropagation revived the study of neural networks and brought new momentum to artificial intelligence research. It showed that multi-layer networks, once considered impractical, could indeed solve problems that single-layer systems could not. Tasks that required non-linear decision boundaries, such as image recognition or speech processing, now became achievable. Researchers began experimenting with deeper architectures, adding more layers to capture increasingly complex features of the data. Although computing power at the time limited how far these experiments could go, the principle was established. Backpropagation became the cornerstone of modern machine learning and is still used in training deep neural networks today. Its reintroduction is often credited with ending the first AI winter, proving that neural networks were not a dead end but a technology with enormous potential.

Fukushima and the Neocognitron

While backpropagation offered a powerful tool for supervised learning, another line of research in the early 1980s explored how machines might learn in an unsupervised way. Kunihiko Fukushima introduced the Neocognitron in 1980, a model that sought to replicate the hierarchical feature detection found in biological vision systems [10]. Inspired by the discoveries of Hubel and Wiesel, Fukushima designed a network with multiple layers of processing units that could recognize visual patterns even when they were distorted or shifted in position. Unlike earlier models that required labeled data for training, the Neocognitron could organize itself based on repeated exposure to patterns. This made it an important step toward self-organizing neural networks and paved the way for future breakthroughs in computer vision.

The architecture of the Neocognitron consisted of layers of *simple cells* and *complex cells*, mirroring the structure observed in the mammalian visual cortex. Simple cells in the early layers detected local features, such as edges or corners. These outputs were pooled by complex cells in the middle layers, which combined signals from nearby simple cells. This pooling reduced the network's sensitivity to shifts in position or size, allowing the system to recognize a pattern even when it appeared in a different place or with slight distortions. Over time and through multiple exposures, the higher layers learned to represent increasingly complex patterns by integrating the information passed upward from lower layers. The result was a system that achieved a form of tolerance to variability in visual input, something earlier models struggled with Figure 2.

Experiments with the Neocognitron showed promising results. Each input pattern consistently activated a specific output cell, even when the size or location of the input changed. This ability to generalize across variations was a major achievement in machine vision. However,

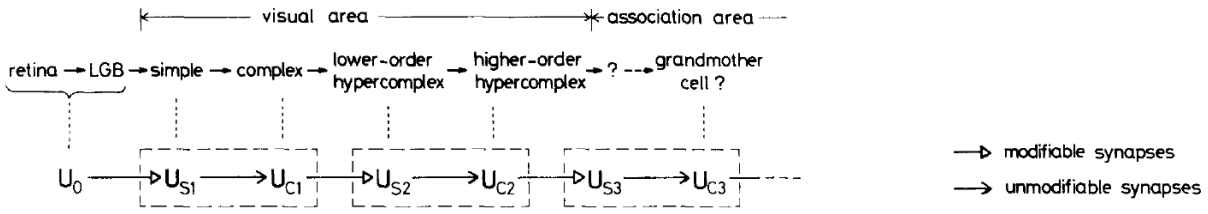


Figure 2: Correspondence between the visual hierarchy and the Neocognitron layers

the model also had limitations. As the number of categories increased, the network struggled with capacity, and performance degraded. Despite these challenges, Fukushima's work marked a turning point. It provided a concrete demonstration that machines could approximate the hierarchical and self-organizing mechanisms of biological vision. The Neocognitron directly influenced the development of convolutional neural networks, which now power many of the most advanced image recognition systems in use today.

LeCun's Backpropagation Network and the Birth of Convolutional Neural Networks

Building on Fukushima's Neocognitron and the introduction of backpropagation, LeCun *et al.* [11] attempted to apply neural networks to the recognition of handwritten digits from U.S. zip codes. This was a demanding real-world task since digits varied greatly in style, size, and clarity, often appearing noisy or ambiguous. To manage this, the authors introduced several innovations. Firstly, local receptive fields limited each neuron's connections to a small neighborhood of pixels, allowing the system to capture spatial relationships without an unmanageable number of parameters. Secondly, weight sharing enabled neurons that detected the same feature, such as a curve or line, to use identical weights. These weights were numbers assigned to inputs to signify their importance, across the image, ensuring the system could recognize patterns regardless of position. Together, these features reduced complexity while making the network more robust to translation and variability in handwriting.

A further improvement came through subsampling, or pooling, which preserved essential information while reducing sensitivity to small distortions. This design helped the network generalize better across the wide variety of digit shapes. The architecture also relied on stochastic gradient descent, a method that updates the network's weights after each training example instead of waiting to process the entire dataset. This process makes learning both faster and more adaptive. The network itself contained three hidden layers and learned all ten digit classes directly through backpropagation, without the need for preprocessing or handcrafted feature extraction. With only a five percent error rate on test data, proved that neural networks could handle complex, noisy inputs. While the term *convolutional neural network* (CNNs) was not used at the time, LeCun's model introduced the core components of CNNs, laying the foundation for deep learning's later success in computer vision.

While LeCun's work demonstrated that backpropagation could successfully handle complex vision classification tasks, training deeper networks still posed significant challenges. The main issue was the problem of vanishing gradients. In backpropagation, each weight is updated based on the gradient of the loss, which effectively measures how much that weight contributed to

the overall error. In shallow networks, this process works well, but in deeper architectures the backward signal must pass through many layers, each involving derivatives of activation functions. As these derivatives are often small, the gradients for early layers shrink rapidly as they propagate backward. The result is that the earliest layers barely receive any updates, leaving them effectively frozen during training. This made it difficult for the network to learn useful low-level features, limiting the effectiveness of deeper models.

A related issue was slow convergence. Because the gradients became so small, the network required many iterations before weights reached meaningful values, making training inefficient and unstable. Together, vanishing gradients and slow convergence discouraged researchers from pursuing multilayered networks during this period. Despite the early promise shown by LeCun's work, the field largely shifted focus to other machine learning approaches, and interest in deep neural networks declined for nearly a decade [12].

The Revival of Deep Learning through Deep Belief Nets

It was not until 2006, with the publication of Hinton *et al.* [13], that interest in multilayered neural networks was substantially revived. This paper marked a major turning point in the history of deep learning because it demonstrated how networks with many hidden layers could be trained effectively, after a long period in which such models had been considered difficult to optimize. Hinton and colleagues proposed a fast, greedy, layer-wise training method for deep generative models, which they called Deep Belief Networks (DBNs). These models are generative because they do more than classify input data into predefined categories. Once trained, they are able to learn the statistical structure of the training data and generate new samples that resemble the original examples. For instance, a DBN trained on images of handwritten digits could produce new digit-like images that were not part of the original dataset. This ability to model the underlying distribution of the data itself was a major conceptual step forward and clearly distinguished DBNs from purely discriminative models.

The approach was built on three central ideas. The first was the introduction of suitable priors to overcome the problem of *explaining away*. Explaining away occurs in probabilistic models when several hidden variables can explain the same observed data. In such cases, the hidden variables become dependent on one another, which makes inference computationally intractable. For example, if two hidden causes can both explain why a digit appears to be a three, knowledge of one cause reduces the likelihood that the other cause is also responsible. Hinton and colleagues addressed this difficulty by using appropriate priors that encouraged independence between hidden layers. This design helped keep inference computationally manageable, even when the number of layers in the network increased.

The second key idea was to simplify deep networks by relating them to a more tractable structure known as the Restricted Boltzmann Machine (RBM). An RBM is a two-layer undirected graphical model composed of a visible layer and a hidden layer. The visible layer represents the observed data, while the hidden layer represents latent features learned from the data. Unlike more complex Boltzmann machines, RBMs impose an important restriction: there are no connections between units within the visible layer and no connections between units within the hidden layer. Connections exist only between visible and hidden units. This structural restriction makes the model easier to train and analyse. Training an RBM involves adjusting its weights so that the

probability distribution generated by the model becomes closer to the real data distribution. This is typically achieved using an approximate learning procedure called Contrastive Divergence. In this method, the data reconstructed by the RBM are compared with the original training data, and the weights are gradually updated to reduce the difference between them.

The third innovation was greedy, layer-wise training. Instead of training the whole deep network at once, which often failed because of problems such as vanishing gradients, Hinton and colleagues proposed training one RBM at a time. The process began with the first RBM, which was trained directly on the raw input data. After this first RBM had learned useful features, its hidden layer activations were used as the input data for the next RBM. The same procedure was then repeated for additional layers. In this way, each new layer learned to represent the statistical patterns present in the layer below it. After several RBMs had been stacked in this greedy manner, the resulting multilayer network already contained useful feature representations at different levels of abstraction. A final fine-tuning stage was then applied. At this point, all layers were connected into a single network, and the entire model was trained jointly using backpropagation. This final stage slightly adjusted the weights throughout the network and improved overall performance.

This training method solved two critical problems in the development of deep neural networks. First, it addressed the vanishing gradient issue by giving the lower layers a good initialization before backpropagation was applied. As a result, these lower layers had already learned meaningful features and were not left with negligible weight updates during supervised training. Second, the method made training computationally feasible by breaking the learning process into smaller and more manageable steps. Rather than requiring the optimization of a deep network from random initial weights, the model was built progressively, layer by layer. The result was a practical approach to constructing deep neural networks that could capture the statistical structure of data in an unsupervised and scalable way. The principle of unsupervised pretraining established in this work laid the foundation for many subsequent developments in deep learning. It contributed to the rapid progress of deep architectures that followed, including the renewed success of convolutional networks and the later development of more sophisticated models such as recurrent neural networks and transformers.

Despite their importance DBNs still had limitations that prevented them from becoming the dominant architecture for large-scale image recognition. Their training procedure relied on complex unsupervised pretraining using stacked RBM, followed by a separate fine-tuning stage with backpropagation, making the overall process computationally expensive and difficult to scale. In addition, DBNs were not specifically designed to exploit the spatial structure of images. DBNs therefore could not effectively utilize the strong relationship between neighboring pixels that forms meaningful visual patterns such as edges, textures, and object parts. This had important implications for image recognition performance because the network was forced to learn these spatial relationships indirectly, resulting in less efficient feature extraction, a larger number of parameters, and poorer scalability to complex visual tasks compared to convolutional neural networks.

AlexNet and the Breakthrough of Modern Deep Learning

It was not until 2012, with the publication of the famous AlexNet paper, that modern deep learning began to gain wide recognition [14]. The paper is widely credited with launching the modern deep learning era because it demonstrated that deep convolutional neural networks could achieve a level of performance that traditional computer vision methods had not reached. AlexNet consisted of five convolutional layers, which automatically learned hierarchical feature representations from images, followed by three fully connected layers used for classification. This architecture allowed the model to move from simple visual patterns in the early layers, such as edges and textures, to more abstract and class-specific representations in deeper layers Figure 3.

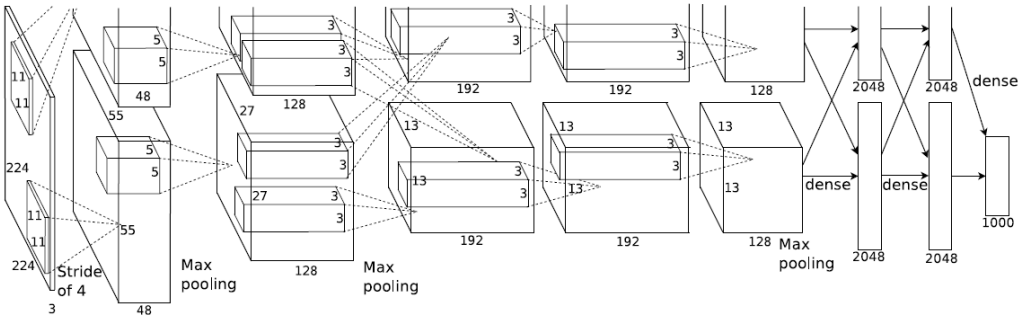


Figure 3: AlexNet architecture for large-scale image classification

Before this breakthrough, image recognition was strongly constrained by the limited size of available datasets. The release of ImageNet changed this situation by providing approximately 15 million labelled images across more than 22,000 categories. This large-scale dataset offered the diversity and volume needed to train deep models on complex visual data. However, deep architectures still faced two major challenges. The first was their high computational demand, since training large neural networks required substantial processing power. The second was overfitting, which occurs when a model learns noise or accidental patterns in the training data instead of general patterns that transfer to unseen data. AlexNet showed that these obstacles could be overcome and that deep convolutional networks, when trained properly, could outperform earlier computer vision approaches.

Several design innovations made AlexNet possible. The first was the use of Rectified Linear Units (ReLU) as the activation function. Earlier networks often relied on sigmoid or tanh functions, which squashed values into a limited small range. During backpropagation, these small values would be multiplied and become even smaller, leading to a phenomenon called the vanishing gradient issue. By contrast, ReLU simply outputs the input directly if it is positive and zero if it is negative. This simple function helped the gradients to remain stronger during backpropagation, making training faster and more effective.

Another key contribution was Local Response Normalization (LRN), which was inspired by biological neural networks. After applying ReLU, neurons with particularly high activation values tended to suppress their neighbors, causing the network to focus too narrowly on a small subset of features. LRN addressed this by normalizing the activations within a local neighborhood of neurons. This created a form of competition among neurons that prevented a few dominant signals from overwhelming the representation and encouraged a richer set of features.

Pooling was also a crucial element of AlexNet's architecture. Instead of relying on the non-overlapping pooling operations commonly used before, the authors employed overlapping pooling, in which neighboring pooling regions shared some pixels. Although this introduced a degree of redundancy, it helped the model generalize more effectively by reducing its sensitivity to small spatial shifts in the input image. Pooling also reduced the dimensionality of the feature maps, which lowered computational costs while preserving the most salient visual information. This made the network both more efficient and more robust to minor variations in object position.

To further improve generalization, the authors used data augmentation. This technique artificially increased the size and diversity of the training set by applying random transformations to the images. These transformations included cropping, horizontal flipping, and changes in color intensity. Data augmentation helped the model learn invariances to changes in orientation, position, and lighting conditions. As a result, AlexNet became less dependent on the exact appearance of the training images and more capable of recognizing objects under varied visual conditions.

A final major innovation was the use of dropout, a regularization method designed to reduce overfitting. During each training step, dropout randomly "dropped" or ignored a subset of neurons. This prevented the network from relying too heavily on particular neurons and forced it to distribute learning across multiple pathways. In practice, dropout functioned similarly to training an ensemble of smaller networks within the larger architecture. This improved the robustness of the model and made it less likely to memorize the training data. The authors also used powerful graphics processing units (GPUs), which had recently become more accessible for scientific computing. GPUs enabled the parallelization of training and made it computationally feasible to train such a deep model on the large ImageNet dataset.

The impact of these choices was remarkable. AlexNet achieved a top-five error rate of 17 percent in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), compared with the previous best result of 26.2 percent. This improvement was not merely incremental; it represented a major leap in large-scale image classification. The study also demonstrated the importance of network depth. Removing any one of the convolutional layers degraded performance by almost two percent, showing that each layer contributed meaningfully to the model's representational capacity. This finding reinforced the idea that hierarchical feature learning, in which successive layers capture increasingly abstract representations, was central to the success of deep neural networks.

Despite its success, AlexNet also had several important limitations. One major drawback was that training the network required multiple GPUs and several days of computation, making it impractical for many researchers at the time. The model also contained around 60 million parameters, resulting in high memory usage and an increased risk of overfitting. In addition, some techniques introduced in AlexNet, such as Local Response Normalization (LRN), were later found to provide limited benefits and were largely replaced by more effective methods such as batch normalization in later architectures. The network also relied on relatively large convolutional filters in the early layers, which increased computational complexity compared to later models that used smaller and deeper filter designs. Furthermore, although AlexNet was deep for its time, it still struggled with optimization difficulties as networks became even deeper, motivating the development of architectures such as VGGNet, GoogLeNet, and ResNet that improved efficiency, scalability, and training stability.

Part III — The Problems of Deep Neural Networks: Vulnerabilities and Fragility

Introduction

Deep neural networks achieved extraordinary success in tasks such as classification, pattern recognition, and large-scale visual recognition. However, this success also revealed a set of important weaknesses. Although these models can reach very high levels of accuracy, they remain fragile when exposed to small changes in their inputs. In particular, almost invisible perturbations can cause dramatic classification errors, showing that high predictive performance does not necessarily imply genuine understanding. This section examines the fragility of deep neural networks from three related perspectives. First, it discusses how these models represent information internally and why meaning is distributed across many neurons rather than stored in single units. Second, it explains the discovery of adversarial examples and the mathematical methods used to generate them. Finally, it considers how researchers have attempted to defend networks against adversarial attacks and why robustness has become a central issue in modern artificial intelligence.

Distributed Meaning & the Discovery of Adversarial Examples

Deep neural networks have achieved remarkable accuracy, yet the way they perceive meaning remains difficult to interpret. Szegedy *et al.* [15] showed that high-level neurons do not correspond to distinct concepts in the simple way that earlier interpretations suggested. Meaning is not localized in individual neurons. Instead, it is distributed across many units, with each neuron contributing only a partial element to the overall representation. The strength of a deep network lies in the coordinated activation of multiple neurons. These units work together to form recognizable patterns across layers. Each neuron may respond to specific aspects of an input, but coherent meaning appears only after several layers of transformation. In this sense, features emerge from collective activity rather than from isolated units. This distributed form of representation makes the internal operation of neural networks less transparent, but it also shows that meaning is an emergent property of the network's overall organization.

The implication of this finding is significant for interpretability. If concepts are not stored in single neurons, then interpretation cannot rely only on identifying individual units and assigning meanings to them. Instead, interpretability must shift toward the analysis of global activation spaces, where the joint contribution of many neurons can be studied. This perspective suggests that neural representations should be understood as patterns distributed across the model rather than as simple one-to-one mappings between neurons and concepts.

Szegedy *et al.* also discovered that deep neural networks can be deceived by slightly perturbed inputs known as adversarial examples. These are images that appear normal to human observers but cause the model to produce incorrect predictions. The process of generating such inputs is called an adversarial attack. A perturbation is a small change made to an input, such as a slight modification of pixel values in an image. In ordinary situations, such changes would not affect how a human understands the image. However, in neural networks, carefully chosen perturbations can push the input across a decision boundary. A decision boundary is the internal

separation the model uses to distinguish one class from another.

Szegedy *et al.* showed that the attack algorithm uses the network's parameters, including its weights and biases, to determine how the input should be modified in order to maximize the probability of misclassification. This finding revealed that even highly accurate networks, including those capable of strong generalization to unseen data, can be extremely fragile when exposed to targeted perturbations. The existence of adversarial examples therefore exposed a crucial distinction between performance and understanding. A model may classify an image correctly under normal conditions, but this does not mean that it has understood the underlying features in a human-like or robust way.

To formally describe the creation of adversarial examples, Szegedy *et al.* used an optimization method known as the box-constrained L-BFGS algorithm. The objective was to minimize the size of the perturbation while also maximizing the classification loss. This can be expressed as follows:

$$\text{Minimize } c\|r\| + \text{loss}_f(x + r, l) \quad \text{subject to } x + r \in [0, 1]^m$$

In this formulation, x represents the original input image, r represents the minimal perturbation, l represents an incorrect label, and f represents the classifier. The term $c\|r\|$ penalizes large perturbations, keeping the modification as small as possible so that the altered image remains visually almost identical to the original. Solving this optimization problem produces an adversarial example that forces the network to misclassify the image even though the input still appears correct to a human observer.

Using the ℓ_2 norm to measure the distance between the original image and the modified image, Szegedy and his colleagues tested several architectures, including AlexNet and models trained on the MNIST dataset. In every case, they succeeded in generating nearly indistinguishable images that confused the models. More importantly, they found that adversarial examples created for one network often transferred to other networks. This showed that the vulnerability was not limited to one specific architecture or training procedure. Rather, it appeared to be a systemic weakness of deep neural networks.

Measuring Perturbations: Norms and Sensitivity

Perturbations can be evaluated using different mathematical norms, each of which captures a different type of change in an image. The ℓ_0 norm measures how many pixels have been changed, regardless of the magnitude of each change. Attacks based on this measure therefore alter only a limited number of pixels. The ℓ_2 or Euclidean norm measures the overall size of the perturbation by distributing small modifications across many pixels. This often produces smoother distortions that are difficult to detect visually. The ℓ_∞ norm limits the maximum change allowed for any single pixel. Under this measure, all pixels may shift slightly, but no pixel is permitted to change beyond a fixed threshold.

These small yet powerful perturbations can also be explained through the Lipschitz constant, which measures how sensitive a function's output is to changes in its input. A deep neural network is composed of many layers, and each layer has its own Lipschitz constant. Because

these constants combine multiplicatively across layers, the overall sensitivity of the network can become very large. As a result, even a minimal disturbance at the input level can be amplified as it moves through the network. By the time the signal reaches the final layers, this small initial change may lead to a large error in prediction.

This observation provided a mathematical basis for understanding why imperceptible perturbations can produce dramatic effects. It also emphasized the importance of measuring and controlling sensitivity in deep learning systems. The discovery of adversarial attacks revealed that deep networks contain intrinsic blind spots. Around nearly every input image, there may exist a small region in which slight changes can cause the model to misclassify the image, even though the altered version appears identical to human observers. These blind spots can occur even in networks that perform well on standard test data. This proves that high accuracy alone does not guarantee robustness. Robustness is therefore not only a practical concern but also a mathematical problem related to how neural networks transform input spaces across multiple layers.

The existence of these vulnerabilities led to the development of adversarial robustness as a major research field. Researchers began to study how models could be defended against subtle but dangerous attacks. They also examined how different architectures, activation functions, regularization methods, and training strategies affect sensitivity to perturbations. Over time, this research improved understanding of neural fragility and contributed to the development of more secure machine learning frameworks. These frameworks aim to handle uncertainty, resist malicious interference, and maintain reliable performance under conditions that differ from ordinary test environments.

The Linearity Hypothesis

Goodfellow *et al.* [16] expanded on these findings and proposed a different explanation for the origin of adversarial examples. They argued that adversarial examples arise not mainly from the nonlinearity of neural networks, but from their linearity in high-dimensional spaces. Modern neural architectures are often designed to behave in approximately linear ways because this makes optimization easier and more stable. Even nonlinear activation functions, such as sigmoid and ReLU, often operate within mostly linear regimes during training.

However, this linearity creates a structural weakness. In high-dimensional input spaces, many small changes can accumulate across dimensions. Although each individual change may be almost imperceptible, their combined effect can significantly alter the model's output. This means that a perturbation does not need to be large in order to be effective. It only needs to be aligned with the direction that increases the model's loss.

To demonstrate this effect, Goodfellow et al. introduced the Fast Gradient Sign Method, a technique for generating adversarial examples efficiently and at low computational cost:

$$\eta = \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y)).$$

In this formula, η represents the perturbation, $J(\theta, x, y)$ represents the loss function with respect to the input, and ϵ controls the magnitude of the change. The method uses the sign of the gradient

to determine the direction in which the input should be modified. Because the calculation is simple, it allows large numbers of adversarial examples to be generated quickly.

Using this approach, Goodfellow and his colleagues caused a classifier to reach an error rate of 99.9 percent with a confidence of 79.3 percent. This result showed that even minimal perturbations can drastically reduce model performance. The efficiency of the method also made it possible to conduct more extensive experiments and comparisons across different models and datasets Figure 4.

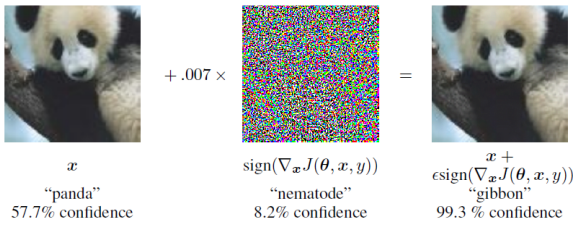


Figure 4: Illustration of an adversarial example generated using the Fast Gradient Sign Method

The authors demonstrated that adversarial vulnerability is a general property of linear systems operating in high-dimensional input spaces. This explanation helped clarify why adversarial examples appear across different datasets, architectures, and training procedures. Their findings also highlighted an important trade-off. The same design choices that make neural networks easier and more efficient to train can also make them structurally vulnerable and unstable during prediction.

From Linearity to Robustness

Goodfellow and his colleagues also introduced adversarial training as a defense mechanism for improving model robustness. This approach involves retraining the model not only on clean images but also on adversarially perturbed images. By exposing the model to adversarial examples during training, the network learns to resist similar attacks in the future.

While Szegedy's initial experiments with the L-BFGS method achieved only limited success, adversarial training based on the Fast Gradient Sign Method proved much more effective. It reduced the error rate of a maxout network regularized with dropout from 0.94 to 0.84 percent. In a larger model, it reduced the error rate from 89.4 to 17.9 percent. These results showed that adversarial training could substantially improve robustness when applied effectively.

The weights of adversarially trained networks also became more localized and interpretable. This suggests that exposure to adversarial examples during training helps the model develop more stable internal representations. Instead of relying on fragile patterns that can easily be disrupted, the model learns features that are more resistant to small input changes. The success of this method encouraged the research community to adopt adversarial training as a standard baseline for robustness research. It also inspired further work on alternative loss functions, noise regularization, and adaptive perturbation bounds.

Another important observation concerns the overconfidence of linear models under adversarial attack. When these models are fooled by adversarial examples, they often assign very high confidence to incorrect predictions. This makes their errors especially dangerous, because the

model does not merely fail; it fails with strong certainty. Nonlinear models, such as Radial Basis Function networks, tend to show lower confidence when exposed to unfamiliar inputs. However, these models generally do not generalize as well to new data, which limits their practical usefulness.

Experiments also showed that ensembles of networks remain vulnerable to adversarial attacks. This reinforced the importance of transferability. Because different models often learn similar decision boundaries, an adversarial example designed for one model can mislead another model, even when the architectures are different. This finding showed that adversarial vulnerability is not only a weakness of individual models but also a systemic feature of deep learning systems. Transferability makes attacks easier to scale and creates serious risks in domains where models are shared, reused, or deployed across multiple applications.

These findings led researchers to conclude that one of the main weaknesses of modern deep networks lies in their linearity. A network may perform well on both training and test data, yet adversarial examples reveal that it may still lack a robust understanding of the task it performs. This realization encouraged researchers to reconsider the assumptions underlying neural architecture design. It also motivated the search for a better balance between efficiency and stability. New approaches to robustness began to emerge, including nonlinear feature mapping, input compression, defensive distillation, and certified robustness methods that attempt to mathematically bound the effect of perturbations. As a result, the field gradually shifted from a narrow focus on predictive performance toward a broader concern with security, reliability, and trustworthiness in artificial intelligence.

Testing and Evaluating Defenses

The discovery of adversarial examples created a technological arms race between attackers and defenders. In this context, a technological arms race refers to a cycle in which each new attack method motivates the development of a stronger defense, while each new defense encourages the design of more adaptive attacks. One research direction therefore focused on producing more powerful attacks that could expose weaknesses in neural networks. The other direction focused on building defenses that could make models more resilient to these attacks.

However, early evaluation methods for adversarial defenses were often inconsistent. This made it difficult to determine whether real progress had been achieved or whether a defense only appeared effective under weak testing conditions. Carlini *et al.* [17] addressed this problem by proposing a standardized framework for testing adversarial robustness. Adversarial robustness refers to a model's ability to maintain correct predictions when its inputs are intentionally modified in small but harmful ways.

A central part of this framework is the threat model. A threat model defines the attacker's capabilities, objectives, and limits before the defense is evaluated. The attacker's capabilities describe what information the attacker can access, while the attacker's objectives specify what the attack is trying to achieve. By clearly defining these conditions, researchers can judge whether a defense truly provides the level of protection it claims.

Carlini and colleagues distinguished between untargeted and targeted attacks. An untargeted attack aims simply to cause misclassification, meaning that any wrong label is considered a

successful attack. A targeted attack is more specific because it forces the model to predict a particular incorrect label chosen by the attacker. This distinction is important because targeted attacks are usually more difficult and therefore provide a stricter test of model robustness.

They also stressed the need to restrict the attacker's power by defining a maximum allowable perturbation. This restriction is usually expressed using an ℓ_p norm, which is a mathematical measure of how large the perturbation is. Although this method does not represent all possible real-world attacks, it provides a useful benchmark for comparing different defenses under controlled conditions.

Carlini et al. also distinguished between white-box and black-box attacks. In a white-box attack, the attacker has full access to the model's architecture, parameters, and sometimes its training procedure. In a black-box attack, the attacker has limited or no internal information about the model and may only observe its outputs. This distinction matters because a defense that works only when the attacker lacks information may fail once the model is studied more closely.

Importantly, Carlini and colleagues argued that defenses should not depend on secrecy. A defense based on secrecy is often described as security through obscurity, which means that the system appears secure mainly because its internal design is hidden. Following Shannon's principle [18], a system should remain secure even when its structure is publicly known. This principle reinforces the importance of transparent defenses that remain robust under open and realistic testing conditions.

A defense is considered effective only if it can resist adaptive attacks. An adaptive attack is designed specifically to bypass a particular defense after the attacker has studied how that defense works. Testing only with existing standard attacks can create a false sense of robustness, because older attacks may not exploit the specific weaknesses of a new method. Therefore, a serious evaluation must ask whether the defense remains effective when the attacker changes strategy in response to it.

Carlini and colleagues also emphasized that robustness should not be achieved by sacrificing normal accuracy. Clean-data accuracy refers to performance on unmodified inputs that have not been adversarially perturbed. A defense that greatly reduces clean-data accuracy has limited practical value because it weakens the model under ordinary conditions. Their framework therefore encouraged researchers to measure both robustness under attack and accuracy on clean data.

The standards introduced by Carlini et al. provided a foundation for more reliable defensive strategies. They encouraged researchers to test defenses more rigorously and to explain the precise conditions under which the defense was evaluated. They also promoted reproducible experiments, meaning experiments that other researchers can repeat using the same assumptions and methods. This made adversarial robustness research more disciplined, transparent, and comparable across studies.

Tramèr *et al.* [19] later clarified another widespread misconception in adversarial robustness research. Many researchers had assumed that detecting adversarial examples was easier than correctly classifying them. Detection means identifying whether an input has been adversarially modified, while classification means assigning the correct label to the input. Tramèr and colleagues showed mathematically that detection and classification share equal difficulty under the

conditions they studied.

Their analysis used the distance parameter ϵ, which represents the maximum allowed size of a perturbation. They proved that a detector capable of identifying adversarial inputs within distance ϵ implies the existence of a classifier that is robust up to $\frac{\epsilon}{2}$, The reverse is also true, meaning that progress in classification robustness also implies progress in detection. This result showed that detection cannot be treated as an easier substitute for robust classification.

Tramèr and colleagues tested fourteen detectors that had claimed high levels of robustness. They found that many of these detectors reported impossible or unrealistic results because they had not been evaluated against adaptive attacks. Once the attack was adjusted to account for the detector, the apparent robustness often disappeared. This demonstrated that detection alone does not guarantee meaningful protection against adversarial examples.

Their findings showed that many apparent improvements in robustness were illusions caused by incomplete testing. Incomplete testing occurs when a defense is evaluated only against weak, outdated, or non-adaptive attacks. This problem can make a model appear secure even though it remains vulnerable under stronger evaluation. For this reason, Tramèr et al. reaffirmed the need for theoretical rigor and practical transparency when testing adversarial defenses.

Adversarial examples exposed deep weaknesses in how neural networks interpret information. They challenged the assumption that high accuracy implies true understanding of the task. A model may classify many images correctly while still relying on fragile statistical patterns that humans do not notice. This distinction made robustness a central concern in evaluating whether artificial intelligence systems can be trusted.

The research that followed produced a wide range of attack and defense methods. These methods showed that robustness is not a fixed quality that a model either possesses or lacks. Instead, robustness is an ongoing pursuit shaped by the interaction between stronger attacks and stronger defenses. Each new defensive method must therefore be tested against adaptive attacks that reflect the current state of adversarial research.

Building secure and trustworthy artificial intelligence depends on balancing learning efficiency with resistance to deception. Learning efficiency refers to the ability of a model to train successfully and perform well using available data and computational resources. Resistance to deception refers to the model's ability to avoid being misled by small, intentional input changes. As models continue to grow in scale and complexity, maintaining interpretability and resilience will become one of the most important challenges for the future of artificial intelligence.

Conclusion

The studies reviewed in this chapter show that the development of artificial intelligence has been a steady process shaped by experimentation and revision. The logical models of McCulloch and Pitts established a way to describe thought in computational terms. Computational terms means that mental activity could be represented as operations performed by formal systems. Hebb's work on learning then introduced adaptability, showing how connections could change through experience rather than remain fixed.

Rosenblatt's Perceptron and Samuel's game-playing programs extended this shift toward adaptive

machines. The Perceptron showed that a machine could learn to classify inputs by adjusting its weights through feedback. Samuel's programs demonstrated that machines could improve their behavior through repeated play and evaluation. Together, these works showed that learning could be treated as a process of gradual adjustment.

The work of Bledsoe, Selfridge, and others extended these principles to recognition tasks. Recognition tasks require a system to identify patterns, such as shapes, characters, or visual objects, from input data. This connected abstract computation with perception, because machines were no longer only manipulating symbols but also interpreting sensory-like information. These contributions created a consistent framework for understanding how information can be processed and improved through experience.

Research in the 1980s and 1990s built directly on these early ideas. Backpropagation provided a method for training multilayer networks by calculating how errors should be distributed backward through the system. The Neocognitron and early convolutional architectures applied hierarchical organization to visual data. Hierarchical organization means that lower layers learn simple features while higher layers combine them into more complex representations.

Deep belief networks and AlexNet refined these designs further. Deep belief networks showed how unsupervised pretraining could help initialize deep architectures, while AlexNet demonstrated the practical power of deep convolutional networks at large scale. These models showed how depth and structured training could increase accuracy in real-world tasks. They also demonstrated that complex learning behavior can emerge from simple mechanisms when sufficient data and computing power are available.

With these advances came a clearer understanding of their limitations. As models became deeper, they also became more sensitive to small variations in input. Work by Szegedy, Goodfellow, and others revealed that slight and carefully designed changes could cause major classification errors. This showed that statistical learning systems can perform well under normal conditions while remaining vulnerable to subtle distortions.

The discussion of adversarial examples and defensive strategies began from this point. It emphasized that model evaluation must consider not only performance but also resilience. Performance refers to how accurately a model behaves under ordinary test conditions, while resilience refers to how reliably it behaves under disturbance or attack. This distinction became essential for understanding the limits of deep learning systems.

Taken together, these papers form a continuous sequence. Each addressed a weakness of the previous stage and raised new questions for the next stage of research. The principles that once guided the creation of learning machines now also inform the study of their vulnerabilities. This continuity shows that attacks and defenses are not separate from the history of artificial intelligence but part of its ongoing development.

Understanding this progression clarifies why the study of deep neural network attacks and defenses is necessary. The field has moved from logical computation to adaptive learning systems and then to robustness research. This movement reflects a broader effort to understand not only how machines learn but also how they make errors. It also shows why future artificial intelligence must be designed to adjust to its environment while remaining secure, interpretable, and resistant to deception.

Chapter 2: Adversarial Attacks, Defense Methods, and Robustness Evaluation

Chapter 1 traced the development of artificial intelligence from early logical models to adaptive neural systems, and finally to the problem of robustness. Deep neural networks achieved their success by learning hierarchical representations from data. However, this success also revealed a central weakness. Models that classify clean inputs with high accuracy can remain highly sensitive to small, carefully designed perturbations. Adversarial examples expose this weakness by showing that accurate prediction does not necessarily imply stable or reliable understanding. A network may perform well while depending on fragile decision boundaries that can be manipulated by changes that are almost imperceptible to humans.

Building on this foundation, the present chapter examines how adversarial vulnerability has been studied through attacks and defenses. Adversarial examples are not treated here as isolated errors, but as part of a broader research dynamic. In this dynamic, stronger attacks reveal weaknesses in existing models, while defensive methods attempt to reduce those weaknesses under specific threat assumptions. This relationship is central to adversarial machine learning because robustness cannot be evaluated independently of the attacker's knowledge, objective, and available perturbation budget. A defense that appears effective against one attack may fail when tested against a stronger or more adaptive method.

This chapter focuses on evasion attacks. In this setting, the model has already been trained, and the attacker modifies the input at inference time in order to cause misclassification. This focus follows the scope of the thesis, which investigates adversarial behavior after training rather than poisoning attacks that corrupt the training data. The chapter begins with the L-BFGS attack, which first formalised the search for small perturbations capable of changing a model's prediction. It then examines early defensive strategies, including denoising autoencoders and Deep Contractive Networks, before turning to FGSM and adversarial training as more efficient approaches to adversarial generation and robustness improvement.

The discussion then moves to methods that refined the attack-defense framework. DeepFool is presented as an iterative method for identifying small untargeted perturbations, while JSMA is examined as an early targeted attack based on saliency maps. Defensive distillation is then discussed as an important attempt to smooth decision boundaries, although later work showed that its robustness was limited. The Carlini and Wagner attacks are considered because they demonstrated that several earlier defenses produced only apparent robustness when evaluated against weaker attacks. PGD is examined as a central method in adversarial robustness because it formulates the problem as a min-max optimization task and links adversarial attack generation directly to robust training. Finally, the chapter covers reliable adversarial robustness evaluation methods through AutoAttack and RobustBench.

The main argument of this chapter is that adversarial vulnerability is not limited to a particular architecture, dataset, or training procedure. The reviewed literature shows that this vulnerability

persists across different attacks, perturbation norms, and evaluation settings. Although defensive methods can improve robustness, their effectiveness often depends on the specific attack used during training or evaluation. Robustness should therefore be understood as an empirical and threat-dependent property, rather than as a fixed attribute of a model.

This discussion also prepares the experimental direction of the thesis. Much of the existing literature evaluates adversarial robustness by measuring input perturbations and final predictions. While this approach is useful, it leaves open the important question of how adversarial examples behave inside the network. In particular, it remains unclear how adversarial inputs affect intermediate activations and feature representations across layers. Starting from chapter 3, this thesis will address this gap by introducing the mathematical framework used to describe internal representations, including layer activations, flattened activation vectors, SVD-projected corevectors, and tail energy. This framework then supports the later experimental analysis of whether adversarial examples produce distinctive internal representation patterns.

Attack and Defense

The discovery of adversarial examples marked an important turning point in the study of deep neural networks. It showed that models with high accuracy on clean data could still be vulnerable to small, carefully designed perturbations. This finding led researchers to investigate two related questions: how adversarial examples can be generated more effectively, and how neural networks can be made more robust against them.

This chapter examines selected adversarial attack and defense methods that are relevant to the aims of this thesis. The discussion begins with early optimization-based attacks and then moves towards more efficient gradient-based methods and defense strategies. This structure is useful because it reflects the development of the field: early methods demonstrated the existence of adversarial vulnerability, while later methods improved the speed, scalability, and practical relevance of adversarial attacks.

Adversarial attacks are commonly divided into poisoning attacks and evasion attacks. Poisoning attacks occur during training, when malicious or adversarial samples are inserted into the training set in order to influence the learned model. Evasion attacks occur after training, at inference time, when the attacker modifies an input sample so that the trained model produces an incorrect prediction. This distinction is important because the two attack types assume different levels of access to the machine learning pipeline.

This thesis focuses on evasion attacks because they directly evaluate the robustness of trained models at test time. In many practical scenarios, an attacker may not be able to modify the training data, but may still be able to manipulate the input given to a deployed model. For this reason, evasion attacks provide a suitable framework for studying how small input perturbations can affect the reliability of neural network predictions.

L-BFGS Attack

Szegedy et al. [15] introduced one of the first systematic methods for generating adversarial examples in deep neural networks. Their work showed that adding a small perturbation r to an input x, could cause the model to misclassify the perturbed input, even when the perturbation

was almost imperceptible. This condition can be expressed as $\|r\| < \varepsilon$, where ε defines a small neighborhood around the original input.

This paper is relevant to this thesis because it established the optimization-based view of adversarial example generation. Before this work, high classification accuracy was often treated as strong evidence of model reliability. Szegedy *et al.* showed that this assumption was incomplete, since a model could perform well on clean data while remaining highly sensitive to small adversarial changes. The method is therefore important not only as an attack, but also as a motivation for studying robustness.

The experiments were conducted on MNIST models, AlexNet, and a large unsupervised QuocNet model trained on YouTube images. The attack was performed in a white-box setting, meaning that the attacker had full knowledge of the model architecture, parameters, and loss function. It was also formulated as a targeted attack, where the objective was to force the model to classify the input as a specific target label rather than simply causing any incorrect prediction.

The L-BFGS method formulates adversarial example generation as a constrained optimization problem under the ℓ_2 norm. The aim is to find the smallest perturbation r that changes the classifier's output to a chosen target label l. This objective can be written as follows:

- Minimize $\|r\|_2$ subject to:
  1. $f(x + r) = l$
  2. $x + r \in [0, 1]^m$

where f denotes the classifier, x is the original input, r is the perturbation, and l is the target misclassification label. The constraint $x + r \in [0, 1]^m$ ensures that the perturbed image remains within the valid pixel range. This constraint is necessary because an adversarial image must still be a valid input to the model.

Since solving this constrained problem exactly is computationally difficult, Szegedy *et al.* proposed an approximate solution using box-constrained L-BFGS. The original constrained problem is transformed into a penalized objective that combines the size of the perturbation with the classification loss. The resulting optimization problem is:

$$\text{Minimize } c\|r\| + \text{loss}_f(x + r, l) \quad \text{subject to } x + r \in [0, 1]^m$$

In this formulation, c is a penalty coefficient that controls the trade-off between perturbation size and successful misclassification. A smaller value of c places more emphasis on keeping the perturbation minimal, whereas a larger value places more emphasis on reaching the target class. The method searches for a suitable value of c and then uses box-constrained L-BFGS to solve the inner optimization problem.

The main advantage of L-BFGS is that it provides a clear and principled formulation of adversarial attacks. It shows that adversarial examples can be generated by solving an optimization problem rather than by adding random noise. This makes the method theoretically important, since it connects adversarial vulnerability to the geometry of the model's loss function and decision boundaries.

Another strength of the method is its ability to produce small perturbations. Because the objective explicitly minimizes the ℓ_2 norm, the generated adversarial examples can remain very close to the original input. This property is particularly important in image classification, where the central problem is that two visually similar images may receive different predictions from the same neural network.

The original paper also showed that adversarial examples can transfer across models. This means that examples generated to fool one model may also reduce the accuracy of another model. This finding is important because it suggests that adversarial vulnerability is not limited to one specific architecture or training set. It also indicates that adversarial attacks may succeed in black-box settings, where the attacker does not have direct access to the target model.

Despite these advantages, L-BFGS has significant limitations. Its main disadvantage is computational cost, because it requires an iterative optimization procedure for each input image. It also requires a search over the penalty coefficient c, which makes the method slower than later attacks such as FGSM, and PGD. For this reason, L-BFGS is not usually the preferred method for large-scale adversarial training or rapid robustness evaluation.

A further limitation is that the method was originally designed for a targeted white-box setting. This makes it useful for demonstrating the existence of adversarial examples, but less representative of all real-world attack scenarios. In practice, an attacker may not know the model parameters and may only aim to cause any incorrect classification rather than a specific target label.

For this thesis, L-BFGS is included as a foundational method rather than as the main experimental attack. Its relevance lies in its historical and theoretical role: it introduced the idea that adversarial examples can be constructed through optimization and showed that deep neural networks are vulnerable to small, structured perturbations. However, its computational cost limits its practical use compared with later, faster methods. Therefore, presenting L-BFGS at the beginning of the attack section helps explain why more efficient adversarial methods were later developed.

Denoising Autoencoders to Remove Adversarial Examples

One of the earliest defense approaches against adversarial examples was proposed by Gu and Rigazio [20]. Building on the adversarial formulation introduced by the L-BFGS attack, they examined whether adversarial perturbations could be removed or reduced before the input reached the classifier. This approach is relevant to this thesis because it represents an early attempt to defend neural networks through input transformation and model regularization. It also shows an important limitation of simple preprocessing methods: removing visible or random noise does not necessarily remove adversarial perturbations, since adversarial examples are specifically optimized to exploit the classifier's decision boundary.

Gu and Rigazio first investigated noise injection as a possible defense. The idea was to add Gaussian noise or blur to the input image in order to move adversarial examples away from the model's vulnerable regions. In principle, this additional corruption could disrupt the adversarial perturbation and help the classifier recover the correct label. Experiments on convolutional networks showed that this method could partially restore correct classification, reaching almost 50 percent accuracy in some cases. However, this improvement came at the cost of reduced accuracy on clean data. This result suggests that adding random noise can weaken some adversarial

effects, but it is not a reliable defense because it also damages useful information in the original input.

The authors then considered preprocessing with an autoencoder. An autoencoder is a neural network designed to reconstruct its input through a compressed internal representation. It consists of an encoder, which maps the input image into a lower-dimensional or more abstract representation, and a decoder, which reconstructs the image from that representation. In the adversarial defense setting, the purpose of the autoencoder is to remove perturbations while preserving the main visual content of the input. This method is relevant because it treats adversarial perturbations as a form of noise that can be filtered before classification.

However, the autoencoder defense was also shown to be limited. Gu and Rigazio used an autoencoder with three hidden layers and stacked it before the classifier. Although this preprocessing step could reduce some distortions, the combined autoencoder-classifier system remained vulnerable to newly generated adversarial examples. In fact, when the attacker adapted to the full defended system, adversarial examples with smaller distortions could still be produced. This finding is important because it shows that a defense may appear effective against existing adversarial examples but fail when the attacker is aware of the defense and optimizes against it.

The third strategy examined by the authors was the denoising autoencoder. Unlike a standard autoencoder, which learns to reproduce its input, a denoising autoencoder is trained to reconstruct a clean input from a corrupted version of that input. In this case, the authors corrupted each input pixel with Gaussian noise and trained the model to recover the original image. This method is conceptually stronger than a standard autoencoder because it explicitly learns a mapping from noisy inputs to clean data. For adversarial robustness, the expectation is that the denoising process will remove small perturbations before classification.

Despite this additional training objective, the denoising autoencoder also failed to provide a complete defense. When it was stacked with the classifier, the resulting system still remained vulnerable to adversarial examples. This limitation is significant because it indicates that adversarial perturbations are not equivalent to ordinary Gaussian noise. A denoising autoencoder may learn to remove random corruption, but adversarial perturbations are structured and model-specific. Therefore, a defense trained only on random noise may not generalize well to adversarially optimized distortions.

After evaluating these preprocessing strategies, Gu and Rigazio proposed a stronger defense called the Deep Contractive Network, or DCN. Instead of only modifying the input, DCN changes the training objective of the network by adding a layer-wise contractive penalty. The purpose of this penalty is to encourage local smoothness in the network's mappings, so that small changes in the input or intermediate representations do not cause large changes in the output. This approach is relevant to this thesis because it shifts the defense strategy from input cleaning to robustness-oriented model training.

The DCN objective adds a regularization term to the standard classification loss. The objective function is defined as follows:

$$J_{\mathrm{DCN}}(\theta) = \sum_{i=1}^{m} \left(L\left(t^{(i)}, y^{(i)}\right) + \sum_{j=1}^{H+1} \lambda_j \left\| \frac{\partial h_j^{(i)}}{\partial h_{j-1}^{(i)}} \right\|_2 \right)$$

In this equation, the first term represents the original classification loss, while the second term is the added contractive penalty. The penalty is applied across the layers of the network and measures how sensitive each layer is to changes in the previous layer. The coefficient λ_j controls the strength of the penalty for layer j. This term is often understood as a Jacobian penalty, since it penalizes large derivatives of layer activations with respect to their inputs.

The purpose of the Jacobian penalty is to make the model less sensitive to small input variations. If a small change in the input causes a large change in the internal representation or output, the model may be vulnerable to adversarial perturbations. By penalizing this sensitivity, DCN encourages smoother decision functions and increases the amount of distortion required to produce a successful adversarial example. In the experiments reported by Gu and Rigazio, DCN increased the minimum adversarial distortion on models trained on the MNIST dataset. Final network's minimum adversarial distortion increased from 0.084 to 0.107, while maintaining almost the same clean-data accuracy.

The main advantage of DCN is that it improves robustness without relying only on external preprocessing. Noise injection, standard autoencoders, and denoising autoencoders attempt to repair the input before classification, but DCN directly modifies the classifier's behavior by encouraging smoother mappings. This makes the method theoretically stronger, since it addresses the model's sensitivity rather than only the appearance of the input. Another advantage is that DCN can increase the required adversarial distortion while preserving clean-data performance, which is a desirable property for any defense method.

However, DCN also has important limitations. Its main disadvantage is computational cost. The contractive penalty requires the derivatives of layer activations to be computed during training, which increases the complexity of the optimization process. This makes the method more expensive than standard training and less practical for larger networks or larger datasets. In addition, the method was evaluated mainly against the ℓ_2 gradient-based white-box L-BFGS attack proposed by Szegedy *et al.*. Therefore, its robustness against stronger or more diverse attacks remains limited in the original evaluation.

Overall, the work of Gu and Rigazio is important because it compares several early defense strategies and shows why simple input preprocessing is insufficient. Noise injection and autoencoder-based preprocessing can partially reduce adversarial effects, but they do not provide reliable robustness when the attacker adapts to the defense. DCN improves on these methods by regularizing the network itself and encouraging local smoothness. For this thesis, the method is relevant because it illustrates the transition from simple adversarial example removal to defense mechanisms that modify the learning objective. At the same time, its computational cost and limited evaluation show why later defense methods became necessary.

FGSM and Adversarial Training

Goodfellow *et al.* [16] proposed a different explanation for the existence of adversarial examples. While earlier work associated adversarial vulnerability with the non-linear nature of deep neural

networks, Goodfellow *et al.* argued that adversarial examples mainly arise from the locally linear behavior of high-dimensional models. This argument is important because it shifts the explanation from model complexity to model geometry. In high-dimensional input spaces, even very small changes in each input feature can accumulate and produce a large change in the model output.

This idea can be illustrated using a linear model. If an input image x is perturbed by a small vector η, the adversarial input can be written as:

$$\tilde{x} = x + \eta$$

The effect of this perturbation on the model activation is then:

$$w^\top \tilde{x} = w^\top x + w^\top \eta$$

Although each individual component of η may be small, the term $w^\top \eta$ can become large in a high-dimensional space. This means that a perturbation that is almost imperceptible to humans can still significantly affect the classifier's decision. This explanation is relevant to this thesis because it provides a simple theoretical basis for understanding why neural networks are vulnerable to adversarial perturbations.

Based on this linearity hypothesis, Goodfellow *et al.* proposed the Fast Gradient Sign Method, or FGSM. FGSM is a white-box attack, since it assumes that the attacker has access to the model parameters and, in particular, to the gradient of the loss function with respect to the input. The method generates an adversarial perturbation by computing the direction that increases the loss most rapidly under an ℓ_∞ constraint. The perturbation is defined as:

$$\eta = \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y)).$$

In this equation, ϵ controls the perturbation budget, $\nabla_x J(\theta, x, y)$ is the gradient of the loss function with respect to the input, and $\operatorname{sign}(\cdot)$ returns the sign of each gradient component. The gradient indicates how the loss changes when each input feature, such as a pixel, is slightly modified. FGSM then takes only the sign of this gradient and multiplies it by the perturbation bound ϵ. The resulting perturbation is added to the original input to produce the adversarial example.

FGSM is usually formulated as an untargeted attack. Its objective is not to force the classifier to predict a specific target label, but to increase the loss so that the model assigns the input to an incorrect class. This makes the attack simple and efficient. Unlike L-BFGS, which requires an iterative optimization process, FGSM generates an adversarial example using only one gradient computation. For this reason, it is much faster and more scalable than earlier optimization-based methods.

The efficiency of FGSM makes it highly relevant to this thesis. It provides a practical baseline for evaluating adversarial vulnerability because it can be applied to large datasets and modern neural networks at relatively low computational cost. It also represents an important transition in adversarial machine learning: instead of solving a complex optimization problem for each

input, FGSM shows that a single gradient step can be sufficient to generate effective adversarial examples. This makes it useful both for attack evaluation and for adversarial training.

Goodfellow *et al.* evaluated FGSM on models trained on MNIST, CIFAR-10, and ImageNet. Their results showed that FGSM could achieve fooling rates comparable to those of L-BFGS, despite being much faster. On MNIST, for example, FGSM caused a shallow SoftMax classifier to misclassify 99.9 percent of adversarial examples, with an average confidence of 79.3 percent. These results demonstrated that adversarial examples were not only a theoretical weakness, but also a practical vulnerability across different datasets and model types.

The main advantage of FGSM is its computational efficiency. Since the method requires only one additional backpropagation step, it can generate adversarial examples quickly. This makes it suitable for large-scale experiments and for use during training. Another advantage is its simplicity. The method has a clear mathematical formulation and is easy to implement, which explains why it became one of the most widely used baseline attacks in adversarial machine learning.

However, FGSM also has limitations. Since it is a single-step method, it may not always find the strongest possible adversarial perturbation. Iterative attacking methods, such as Projected Gradient Descent (PGD), can produce stronger attacks because they refine the perturbation over several steps. FGSM is also based on a white-box assumption, meaning that it requires access to the model's gradients. Although transferability can make FGSM relevant in some black-box scenarios, its original formulation assumes direct knowledge of the target model.

In addition to proposing FGSM as an attack, Goodfellow *et al.* also introduced adversarial training as a defense strategy. Adversarial training improves robustness by augmenting the training process with adversarially perturbed examples. Instead of training only on clean inputs, the model is trained on both clean and adversarial versions of the data. The aim is to teach the model to classify perturbed inputs correctly and to reduce its sensitivity to adversarial directions in the input space.

The adversarial training objective modifies the standard cost function as follows:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J\big(\theta, x + \epsilon \operatorname{sign}(\nabla_x J(\theta, x, y))\big)$$

where α controls the balance between the clean loss and the adversarial loss. In many cases, α is set to 0.5 so that clean and adversarial examples contribute equally to the training objective. The first term encourages correct classification of clean inputs, while the second term encourages correct classification of FGSM-generated adversarial inputs.

Adversarial training is relevant to this thesis because it represents one of the first practical defense methods against adversarial examples. Compared with regularization-based defenses such as DCN, adversarial training is conceptually simpler and more directly connected to the attack process. It treats adversarial examples as informative training data and uses them to improve the model's robustness. This makes it a natural defense to discuss after FGSM, since the same attack used to expose the vulnerability is also used to improve the model during training.

The results reported by Goodfellow *et al.* showed that adversarial training could improve robustness. On MNIST, the test error decreased from 0.94 percent to 0.84 percent, while the adversarial

error on FGSM-generated examples decreased from 89.4 percent to 17.9 percent. These results indicate that adversarial training can substantially reduce vulnerability to the attack used during training. At the same time, the adversarial error remained significant, which shows that the defense did not completely solve the problem.

The main advantage of adversarial training is that it is a practical and effective form of data augmentation. By exposing the model to adversarial examples during training, the classifier can learn more robust decision boundaries. Another advantage is that, when FGSM is used to generate the adversarial examples, the additional computational cost remains manageable. Although training becomes more expensive because each batch requires adversarial example generation, FGSM is sufficiently cheap for the method to remain tractable.

Nevertheless, adversarial training also has several limitations. First, it tends to provide robustness mainly against the type of adversarial examples used during training. A model trained with FGSM examples may still be vulnerable to stronger or iterative attacks. Second, adversarial training increases training cost because it requires one gradient computation to generate the attack and another to update the model parameters. Third, the method does not eliminate high-confidence errors: when the model still misclassifies adversarial examples, it may continue to assign them high confidence.

Overall, Goodfellow *et al.* made two major contributions to adversarial machine learning. First, they explained adversarial examples through the linear behaviour of high-dimensional neural networks. Second, they proposed FGSM as a fast and scalable attack, together with adversarial training as an efficient defense strategy. For this thesis, FGSM is important because it provides a simple and computationally efficient baseline attack, while adversarial training is important because it introduces the idea of improving robustness by training directly on adversarial examples. Although both methods have limitations, they remain central to understanding the development of adversarial attacks and defenses.

DeepFool

DeepFool was proposed by Moosavi-Dezfooli *et al.* [21] as an efficient method for generating adversarial examples by approximating the classifier locally as a linear function. The method builds on the same general observation used by FGSM: neural networks often behave approximately linearly in local regions of the input space. However, DeepFool addresses an important limitation of FGSM. While FGSM takes only one step in the direction of the gradient sign, it does not necessarily find the smallest perturbation needed to cause misclassification. DeepFool was therefore introduced to estimate a smaller and more accurate adversarial perturbation.

The main idea of DeepFool is to iteratively move an input sample towards the closest decision boundary. For a binary affine classifier, the decision boundary is a hyperplane, and the smallest perturbation corresponds to the orthogonal projection of the input onto that hyperplane. In this case, the perturbation can be computed directly. For non-linear classifiers, DeepFool approximates the classifier as linear around the current point and repeatedly updates the perturbation until the classifier changes its prediction. This makes the method more precise than one-step attacks, while still remaining computationally practical Figure 5 .

The method was first developed for binary classifiers and then extended to multi-class classifiers networks. In the multi-class case, DeepFool estimates the closest decision boundary among all

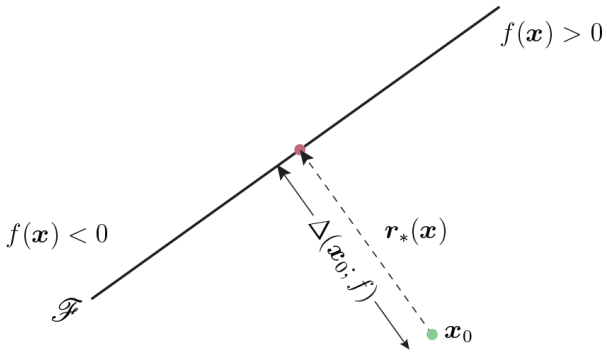


Figure 5: Geometric interpretation of DeepFool for a linear binary classifier

possible class boundaries and moves the input towards it. Although the original work mainly focused on the ℓ_2 norm, the method can also be adapted to other norm constraints. This flexibility is important because adversarial robustness may be evaluated under different distance metrics, depending on the type of perturbation being studied.

DeepFool is relevant to this thesis because it provides a stronger estimate of the minimum perturbation required to fool a classifier. Unlike FGSM, which is mainly valued for speed and simplicity, DeepFool is useful for measuring how close an input is to the decision boundary. This makes it particularly important for robustness evaluation. If a model can be fooled by a very small DeepFool perturbation, this indicates that its decision boundary lies very close to the natural data point.

Moosavi-Dezfooli *et al.* evaluated DeepFool on neural networks trained on MNIST, CIFAR-10, and ImageNet. Their results showed that DeepFool could generate smaller perturbations than several state-of-the-art attacks available at the time, while remaining efficient enough for practical use. This demonstrated that adversarial examples could be produced not only quickly, as in FGSM, but also with relatively low distortion. For this reason, DeepFool became an important baseline for evaluating the geometric robustness of classifiers.

The main advantage of DeepFool is that it usually finds smaller perturbations than FGSM. Since it approximates the closest decision boundary and updates the perturbation iteratively, it provides a more accurate estimate of the model's vulnerability. Another advantage is that it remains relatively efficient compared with more expensive optimization-based methods such as L-BFGS. Therefore, DeepFool offers a useful balance between attack strength and computational cost.

However, DeepFool also has limitations. The algorithm is greedy because it makes locally optimal updates at each iteration. As a result, it may not always find the true global minimum perturbation, especially for highly non-linear decision boundaries. In addition, DeepFool is usually formulated as an untargeted attack. Its goal is to find the smallest perturbation that changes the classifier's prediction to any incorrect class, rather than to a specific target class. This limits its use in settings where targeted misclassification is required.

Overall, DeepFool is important because it improves on one-step gradient attacks by producing smaller and more informative perturbations. For this thesis, it is relevant as a method for understanding the distance between clean inputs and the classifier's decision boundary. Its main strengths are precision and efficiency, while its main limitations are its greedy nature and its

focus on untargeted attacks.

JSMA: A Targeted Saliency-Based Attack

Papernot *et al.* [22] introduced one of the earliest targeted feature-based attacks against deep neural networks. Their method, later known as the Jacobian-based Saliency Map Attack, or JSMA, uses information about how each input feature affects the model's output. Instead of perturbing all pixels at once, as in FGSM, JSMA identifies a small number of influential features and modifies them iteratively until the model predicts a chosen target class.

This work is relevant to this thesis because it introduced a more explicit adversarial threat model for deep neural networks. Papernot *et al.* described different levels of attacker knowledge, ranging from full knowledge of the model to limited access where the attacker can only query the model and observe its outputs. Their experiments were conducted in a setting where the attacker had full knowledge of the network, but no access to the training set. This distinction is important because adversarial attacks depend strongly on what the attacker is assumed to know.

The attack is based on the forward derivative, which measures the sensitivity of the network's outputs with respect to its inputs. For a neural network function $\mathbf{F}$ that maps an input vector $\mathbf{X}$ to an output vector, the forward derivative is the Jacobian matrix:

$$\nabla \mathbf{F}(\mathbf{X}) = \frac{\partial \mathbf{F}(\mathbf{X})}{\partial \mathbf{X}} = \left[\frac{\partial F_j(\mathbf{X})}{\partial x_i} \right]_{i \in 1..M,\, j \in 1..N}$$

where $\mathbf{X}$ represents the input features, M is the number of input features, and N is the number of output classes. Each element of the Jacobian measures how a small change in an input feature affects one output class. This information is then used to decide which features should be modified.

The saliency map is constructed from the forward derivative. Its purpose is to identify input features whose modification will increase the probability of the target class while decreasing the probabilities of the other classes. A simplified form of the saliency score for feature i and target class t can be written as follows:

$$S(\mathbf{X}, t)[i] = \begin{cases} 0, & \text{if } \frac{\partial F_t(\mathbf{X})}{\partial X_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(\mathbf{X})}{\partial X_i} > 0, \\ \left(\frac{\partial F_t(\mathbf{X})}{\partial X_i} \right) \left| \sum_{j \neq t} \frac{\partial F_j(\mathbf{X})}{\partial X_i} \right|, & \text{otherwise.} \end{cases}$$

This score gives priority to features that are likely to increase the target class while reducing the competing classes. The algorithm then modifies the features with the highest saliency values. This process continues until the model predicts the target label or until a maximum distortion threshold is reached.

JSMA is important because it shows that adversarial examples can be generated by modifying only a small fraction of the input features. In experiments on MNIST using the LeNet-5 network, the attack achieved a success rate of 97.1 percent while modifying only 4.02 percent

of input features on average. This result is significant because it demonstrates that adversarial misclassification does not always require widespread perturbation across the whole image.

The main advantage of JSMA is its targeted nature. Unlike untargeted attacks that only aim to cause any incorrect prediction, JSMA attempts to force the model into a specific target class. Another advantage is that it produces sparse perturbations, since it modifies only selected input features. This makes it useful for studying adversarial attacks under an ℓ_0-style perspective, where the number of modified features is more important than the total perturbation magnitude.

However, JSMA is more computationally expensive than FGSM. It requires the construction of saliency maps and the iterative modification of input features. This makes it slower than one-step gradient-based attacks, although it can remain feasible with GPU acceleration. A further limitation is that the original evaluation was mainly performed on MNIST, which is a low-dimensional grayscale dataset. Therefore, its effectiveness and efficiency on larger and more complex datasets, such as ImageNet, are less straightforward.

The authors also evaluated adversarial training using examples generated by JSMA. This reduced the attack success rate and increased the average distortion required for successful attacks, indicating a limited robustness improvement. However, the improvement was not sufficient to eliminate adversarial vulnerability. This result is relevant because it reinforces a recurring theme in adversarial machine learning: defenses often improve robustness against known attacks, but may remain vulnerable to stronger or adaptive attacks.

Overall, JSMA is relevant to this thesis because it introduced a targeted and feature-selective approach to adversarial example generation. Its main strengths are its use of saliency information and its ability to produce sparse targeted perturbations. Its main weaknesses are computational cost and limited scalability to high-dimensional datasets.

Defensive Distillation

Papernot *et al.* [23] proposed defensive distillation as an early defense against adversarial examples. The method was adapted from knowledge distillation, which was originally introduced to transfer knowledge from a larger model to a smaller model. In standard distillation, a large teacher network is first trained on the data, and a smaller student network is then trained to imitate the teacher's output probabilities. In defensive distillation, however, the objective is not model compression. Instead, the goal is to smooth the model's decision boundaries and reduce sensitivity to small input perturbations.

This method is relevant to this thesis because it represents an important early attempt to improve robustness during training rather than by preprocessing the input. Unlike noise injection or autoencoder-based defenses, defensive distillation keeps the architecture largely unchanged at test time. The defense is applied through the training procedure, where the model learns from softened probability distributions instead of hard class labels.

The method uses a softmax function with a temperature parameter T. For logits $z_i(X)$, the softened output probabilities are defined as:

$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

where T controls the smoothness of the output distribution. When $T = 1$, the softmax corresponds to the standard form used in classification. When $T > 1$, the output distribution becomes softer, meaning that probability mass is distributed more evenly across classes. These soft labels contain more information than one-hot labels because they describe not only the correct class, but also the relative similarity between classes according to the teacher model.

Defensive distillation is performed in two stages. First, an initial network is trained using the softmax layer with a high temperature. This network produces probability vectors for the training data. Second, a distilled network is trained using these probability vectors as labels, again using the same temperature during training. At test time, the temperature is set back to one. The intended effect is to produce a model with smoother decision boundaries and reduced sensitivity to small perturbations Figure 6.

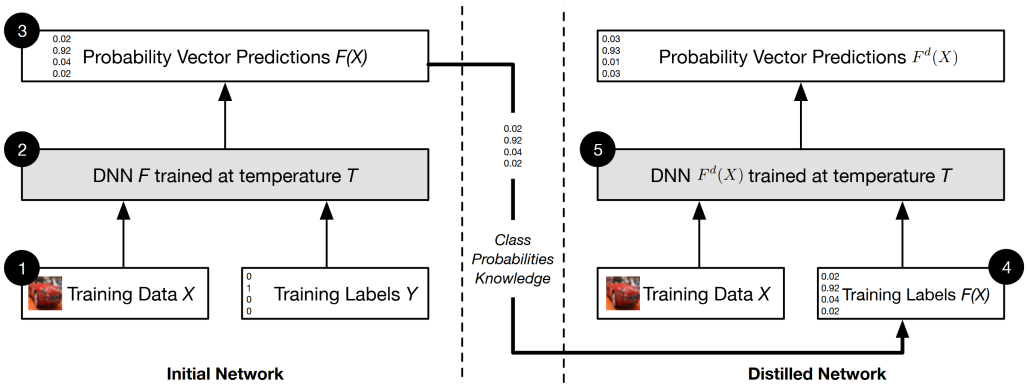


Figure 6: Overview of defensive distillation

The authors argued that higher temperature values reduce the magnitude of the model's input-output derivatives. In this view, smaller derivatives make it more difficult for gradient-based attacks to identify effective adversarial directions. This is why defensive distillation appeared promising against attacks based on saliency maps and gradients. The method aimed to reduce the model's sensitivity while preserving its accuracy on clean data.

Papernot *et al.* evaluated defensive distillation on MNIST and CIFAR-10 using convolutional neural networks. Their results showed little negative effect on clean accuracy. On MNIST, the clean accuracy dropped by only 0.46 percent, while on CIFAR-10 it improved slightly by 0.44 percent. Against the attack used in their earlier work, the attack success rate decreased from 95.89 percent to 0.45 percent on MNIST and from 87.89 percent to 5.11 percent on CIFAR-10. These results suggested a substantial robustness gain under the tested attack conditions.

The main advantage of defensive distillation is that it improves apparent robustness without changing the model architecture at inference time. Once training is completed, the defended model can be used like a standard classifier. Another advantage is that the method preserves clean-data accuracy, which is important because many defense methods reduce performance on non-adversarial inputs. In this sense, defensive distillation appeared to offer a favorable trade-off between robustness and accuracy.

However, defensive distillation has important limitations. First, the training cost is increased because two networks must be trained: the initial teacher network and the distilled student network. Second, very high temperature values may over-smooth the output distribution and reduce the model's discriminative power. Third, and most importantly, later attacks showed that defensive distillation does not provide reliable robustness. Instead, it can create the appearance of robustness by making gradients less useful, while the model remains vulnerable to attacks designed to overcome this effect.

For this thesis, defensive distillation is important because it illustrates both the promise and the weakness of early defense methods. It showed that modifying the training objective could make some attacks less effective, but it also demonstrated that robustness must be evaluated against adaptive and stronger attacks. Its main contribution is historical and conceptual: it introduced a training-based defense that motivated later work, but its limitations show why apparent robustness should not be confused with true adversarial robustness.

Carlini & Wagner Attack

Carlini & Wagner [24] proposed a set of strong optimization-based attacks that challenged the robustness claims of earlier defenses, especially defensive distillation. Their work showed that defensive distillation appeared robust mainly because it made some gradient-based attacks ineffective, not because it removed adversarial vulnerability. This distinction is important for this thesis because it highlights the need to evaluate defenses using adaptive attacks that are designed with knowledge of the defense mechanism.

The Carlini and Wagner attacks, commonly referred to as C&W attacks, were formulated under three different distance metrics: ℓ_0, ℓ_2, and ℓ_∞. This allowed the authors to evaluate adversarial perturbations from different perspectives. The ℓ_0 version focuses on the number of modified input features, the ℓ_2 version focuses on the Euclidean size of the perturbation, and the ℓ_∞ version constrains the maximum change applied to any individual input feature. This multi-norm formulation made the attack more comprehensive than many earlier methods.

The general adversarial objective can be written as follows:

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_p + c \cdot f(x+\delta) \\ \text{s.t.} \quad & x + \delta \in [0,1]^n \end{aligned}$$

where δ is the adversarial perturbation, $|\delta|_p$ measures the perturbation size under a chosen norm, and c controls the balance between minimizing distortion and achieving misclassification. The box constraint ensures that the adversarial example remains within the valid input range. A key contribution of Carlini and Wagner was the use of a loss function based on logits rather than softmax probabilities. For a targeted attack with target class t, the objective can be expressed as:

$$f(x') = \max\left(\max_{i \neq t} Z(x')_i - Z(x')_t,\ -\kappa\right)$$

where $x' = x + \delta$, $Z(x')$ denotes the logits of the classifier, and κ is a confidence parameter.

This objective encourages the target class logit to become larger than the logits of all other classes. Using logits avoids some of the gradient problems associated with softmax probabilities, especially in defensively distilled networks.

The authors used the Adam optimizer to solve the optimization problem because it converged efficiently while producing strong adversarial examples. They also searched for an appropriate value of the constant c, since this value determines the trade-off between small distortion and attack success. This optimization-based structure made C&W attacks more expensive than one-step attacks such as FGSM, but substantially stronger.

Carlini and Wagner evaluated their attacks on MNIST, CIFAR-10, and ImageNet. They used models similar to those evaluated in defensive distillation and showed that their attacks could defeat both distilled and undistilled networks. Their results demonstrated that defenses previously thought to be robust could be broken with carefully designed loss functions and optimization procedures. This made the C&W attack a new benchmark for evaluating adversarial defenses.

The main advantage of the C&W attack is its strength. It can generate adversarial examples with very small distortions and high success rates. Another advantage is its adaptability across different norms, which makes it useful for evaluating robustness under multiple threat models. The attack also showed that relying on weak or non-adaptive attacks can lead to misleading conclusions about defense effectiveness.

However, the C&W attack also has limitations. Its main disadvantage is computational cost. Since it solves an optimization problem for each input, it is slower than FGSM and other simple gradient-based attacks. It also requires careful tuning of parameters such as c and κ. In addition, although it is very strong, it still evaluates robustness under specific optimization objectives and threat models. Therefore, robustness remains empirical and attack-dependent.

For this thesis, the C&W attack is relevant because it marks a turning point in adversarial robustness evaluation. It showed that a defense must be tested against strong adaptive attacks, not only against the attack for which it was designed. Its main contribution is methodological: it raised the standard for evaluating adversarial defenses and demonstrated that apparent robustness can disappear under stronger attacks.

Projected Gradient Descent

Madry *et al.* [25] reformulated adversarial robustness as a min-max optimization problem. This formulation places attacks and defenses within a single framework. The attacker solves an inner maximization problem by searching for the perturbation that maximizes the loss, while the defender solves an outer minimization problem by training model parameters that reduce this worst-case loss. This framework is important because it defines robustness as performance against the strongest adversary within a specified perturbation set Figure 7.

The robust optimization problem can be written as:

$$\min_{\theta} \ \rho(\theta), \quad \text{where} \quad \rho(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} \mathcal{L}(\theta, x+\delta, y) \right]$$

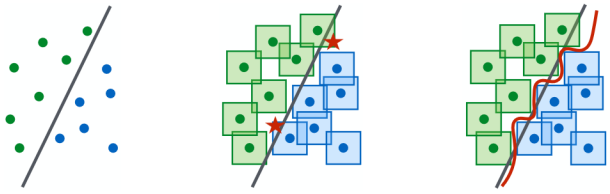


Figure 7: Conceptual illustration of standard and adversarially robust decision boundaries

where θ represents the model parameters, (x, y) is a data-label pair, $\mathcal{L}$ is the loss function, and S is the allowed perturbation set. In many experiments, S is defined as an ℓ_∞ ball around the original input. The inner maximization represents the adversarial attack, while the outer minimization represents adversarial training.

Building on FGSM, Madry *et al.* proposed Projected Gradient Descent, or PGD, as a stronger multi-step adversary. FGSM takes a single step in the direction that increases the loss, whereas PGD takes multiple smaller steps. After each step, the perturbed input is projected back into the allowed perturbation region. The PGD update under an ℓ_∞ constraint can be written as:

$$x^{t+1} = \Pi_{x+S}\left(x^t + \alpha \operatorname{sgn}\left(\nabla_x \mathcal{L}(\theta, x, y)\right)\right)$$

where α is the step size and Π_{x+S} denotes projection back onto the allowed perturbation set around the original input. This projection ensures that the adversarial example remains within the permitted distortion bound.

PGD is relevant to this thesis because it became one of the most important baseline attacks for evaluating adversarial robustness. Unlike FGSM, PGD can explore the perturbation region more effectively because it uses multiple iterations. This makes it a stronger approximation of the inner maximization problem. If a model is robust against PGD, it is generally considered more robust than a model tested only against weaker one-step attacks.

From the defense perspective, PGD can also be used during adversarial training. During training, adversarial examples are generated by solving the inner maximization problem approximately using PGD. The model is then updated using these adversarial examples, usually through stochastic gradient descent. This process trains the classifier to reduce the loss not only on clean inputs, but also on strong adversarial examples. For this reason, PGD adversarial training became a central method for building robust models.

Madry *et al.* evaluated their approach on MNIST and CIFAR-10. Their results showed strong robustness on MNIST, but lower robustness on CIFAR-10. This difference is important because it shows that adversarial robustness becomes more difficult as the dataset and classification task become more complex. Although PGD adversarial training can substantially improve robustness, it does not solve the problem completely, especially on high-dimensional natural images.

The main advantage of PGD is its strength as a first-order adversary. Since it repeatedly uses gradient information and projects the result back into the valid perturbation set, it is much stronger than FGSM while remaining conceptually simple. Another advantage is that it provides a unified framework for both attack and defense. The same method can be used to generate adversarial examples for evaluation and to train models through adversarial training.

However, PGD also has important limitations. Its main disadvantage is computational cost. Because it requires multiple gradient steps for each input, PGD is significantly slower than FGSM. This becomes especially expensive when PGD is used during training. Another limitation is that the min-max problem is non-convex and non-concave for deep neural networks. Therefore, PGD does not guarantee finding the true worst-case adversarial example, even though it often provides a strong empirical approximation.

A further limitation is that robustness may depend on the chosen threat model. For example, a model trained against ℓ_∞ perturbations may not be equally robust against all other types of attacks. Although some experiments suggest that robustness can partially transfer across norms, this is not guaranteed. Therefore, evaluating a model only against PGD under one norm may provide an incomplete picture of its robustness.

Overall, PGD is important because it established a rigorous and practical framework for adversarial training. For this thesis, it is relevant because it connects adversarial attacks and defenses through the min-max formulation. Its main strengths are attack strength, conceptual clarity, and usefulness in robust training. Its main limitations are computational cost, lack of global optimality guarantees, and dependence on the chosen perturbation norm.

Reliable Robustness Evaluation

The discussion of PGD shows that adversarial robustness must be evaluated against strong attacks within a clearly defined threat model. However, evaluating adversarial robustness has proved methodologically difficult because existing studies have often relied on different attacks, threat models, and experimental settings. This inconsistency makes it difficult to compare defenses fairly and can lead to an overestimation of robustness. In such cases, a defense may appear effective not because it genuinely resists adversarial perturbations, but because the attack used in the evaluation is too weak or poorly adapted to the defense.

Athalye, Carlini, and Wagner [26] demonstrated that many proposed defenses achieved apparent robustness by relying on obfuscated gradients, a form of gradient masking. An obfuscated gradient occurs when a defense makes the gradient information unavailable, misleading, or uninformative for the attacker. Since many attacks use gradients to identify how small input changes can alter the model's prediction, distorted gradients can cause these attacks to fail for the wrong reason. Consequently, standard gradient-based attacks may fail even when the underlying model remains vulnerable. By recovering or approximating useful gradient information, the authors developed adaptive attacks that successfully broke seven of the nine defenses previously claimed to be robust.

Similarly, Tramèr *et al.* [27] showed that even defenses evaluated with adaptive attacks can remain vulnerable if the evaluation is not sufficiently strong. An adaptive attack is an attack designed with knowledge of the defense mechanism. Instead of applying a fixed or generic attack, the attacker studies how the defense works and then modifies the attack strategy to bypass it. The authors systematically evaluated thirteen defenses by first studying each defense, then identifying possible weaknesses, and finally designing adaptive attacks that exploited those weaknesses. Their study concluded that a strong adaptive attack should be simple, targeted, and closely matched to the defense under evaluation. In particular, the attack objective must be chosen carefully. The attack objective is the mathematical goal that guides the generation

of adversarial examples. If this objective does not account for the defense, the attack may fail for methodological reasons rather than because the model is truly robust. Therefore, Tramèr *et al.* argue that reliable robustness evaluation requires attacks that are explicitly adapted to the defense being tested, rather than attacks applied in a generic or poorly tuned manner.

These evaluation problems motivate the use of standardized and reproducible robustness frameworks. For this reason, the following sections introduce AutoAttack and RobustBench, which are used later in this thesis to define the adversarial evaluation setting in a controlled and comparable way.

AutoAttack

Croce *et al.* introduced AutoAttack as a standardized method for evaluating adversarial robustness [28]. AutoAttack combines four complementary attacks into a single parameter-free framework. In this context, *parameter-free* means that the evaluator does not need to manually tune important attack hyperparameters. This is important because poorly chosen attack parameters can make a weak defense appear robust simply because the attack was not properly optimized.

The attacks included in AutoAttack are designed to cover different failure modes of robustness evaluation. Some of them are *white-box attacks*, meaning that they use internal information from the model, especially gradients. A gradient indicates how the model's loss changes when the input is slightly modified, and it therefore provides a direction for constructing adversarial perturbations. Other attacks are *black-box attacks*, meaning that they do not use gradients or internal model parameters. Instead, they rely only on the model's outputs, such as predicted labels or confidence scores. By combining white-box and black-box attacks, AutoAttack reduces the risk that robustness is overestimated because one particular attack fails.

One component of AutoAttack is the Fast Adaptive Boundary Attack (FAB). FAB is a white-box attack that aims to find a minimal adversarial perturbation. An adversarial perturbation is a small change added to an input image in order to cause misclassification while remaining within the allowed threat model. FAB works by moving the input toward the model's decision boundary, which is the boundary separating one predicted class from another. This makes FAB especially useful when standard gradient-based attacks fail to identify a successful perturbation, because it directly searches for small changes that cross the classification boundary.

Another component of AutoAttack is the Square Attack, which is a black-box attack. Unlike gradient-based methods, the Square Attack does not require access to the model's gradients. Instead, it changes square-shaped regions of the input image and observes how the model's output changes. Because it only needs model outputs, it is useful for testing whether a defense is genuinely robust or whether it merely disrupts gradient information. This is particularly important in cases of gradient masking, where gradients become misleading or uninformative even though adversarial examples still exist.

A central part of AutoAttack is Auto-PGD (APGD), an improved version of Projected Gradient Descent (PGD). Standard PGD generates adversarial examples by repeatedly moving the input in the direction that increases the attack loss, while projecting the result back into the allowed perturbation region. However, traditional PGD requires manual choices such as the step size and the iteration schedule. If these parameters are poorly selected, the attack may stop too early,

move too aggressively, or fail to exploit the full perturbation budget. As a result, robustness may be overestimated.

APGD addresses this problem by automatically adapting the step size during the attack. Its update first computes an intermediate adversarial point:

$$z^{(k+1)} = P_S\left(x^{(k)} + \eta^{(k)} \nabla f\left(x^{(k)}\right)\right),$$

where $x^{(k)}$ is the adversarial example at iteration k, $\eta^{(k)}$ is the step size at that iteration, $\nabla f(x^{(k)})$ is the gradient of the attack objective with respect to the input, and P_S is the projection operator. The set S represents the allowed perturbation region, such as an ℓ_∞ or ℓ_2 ball around the original input. The projection step ensures that the adversarial example remains within the threat model.

APGD then applies a momentum-based interpolation:

$$x^{(k+1)} = P_S\left(x^{(k)} + \alpha \cdot \left(z^{(k+1)} - x^{(k)}\right) + (1-\alpha) \cdot \left(x^{(k)} - x^{(k-1)}\right)\right),$$

where α is the momentum coefficient. The first term moves the attack toward the newly computed point $z^{(k+1)}$, while the second term incorporates information from the previous update direction. This momentum term helps stabilize the optimization process and prevents the attack from relying only on the most recent gradient step. APGD also monitors the progress of the loss during optimization. If the loss stops improving or begins to oscillate, the step size is reduced. In this way, APGD avoids the fixed-step limitation of standard PGD and makes the attack less dependent on manual tuning.

AutoAttack uses two versions of APGD: APGD with cross-entropy loss, denoted APGD-CE, and APGD with Difference of Logits Ratio loss, denoted APGD-DLR. Cross-entropy loss is the standard loss function used in classification tasks. In an adversarial attack, maximizing cross-entropy usually reduces the model's confidence in the correct class and pushes the prediction toward an incorrect class. APGD-CE is therefore a natural extension of the standard PGD attack. However, cross-entropy can be unreliable in some cases, especially when gradients are masked, saturated, or distorted. In such cases, the attack may fail even though the model is still vulnerable.

To address this limitation, Croce *et al.* introduced the Difference of Logits Ratio (DLR) loss. Logits are the raw output scores produced by a neural network before they are converted into probabilities. The DLR loss compares the logit of the correct class with the strongest incorrect class. Its purpose is to reduce the margin between the correct class and the most competitive wrong class until the wrong class receives a higher score. The untargeted DLR loss is defined as:

$$\mathrm{DLR}(x, y) = -\frac{z_y - \max_{i \neq y} z_i}{z_{\pi_1} - z_{\pi_3}},$$

where z_y is the logit of the correct class, $\max_{i \neq y} z_i$ is the largest logit among all incorrect classes, z_{π_1} is the largest logit overall, and z_{π_3} is the third-largest logit. The denominator normalizes the loss by the spread of the logits. This normalization makes DLR invariant to shifts and rescaling

of the logits. In other words, if all logits are increased by the same amount or multiplied by a constant, the DLR loss remains stable. This makes it more reliable than cross-entropy in cases where the scale of logits affects the strength of the gradient signal.

The authors also introduced a targeted version of DLR:

$$\text{Targeted-DLR}(x, y) = -\frac{z_y - z_t}{z_{\pi_1} - \left(z_{\pi_3} + z_{\pi_4}\right)/2}.$$

Here, z_t is the logit of the target class, meaning the specific incorrect class that the attack tries to force the model to predict. A targeted attack is therefore more restrictive than an untargeted attack: instead of merely causing any wrong prediction, it attempts to move the model toward a chosen wrong label. The denominator again normalizes the logit differences, using the largest, third-largest, and fourth-largest logits to stabilize the objective.

In their experiments, Croce *et al.* found that APGD with DLR loss outperformed standard PGD on most evaluated models. APGD-DLR became one of the strongest components of AutoAttack, particularly on datasets such as CIFAR-10, CIFAR-100, and ImageNet. However, this does not mean that APGD-DLR always replaces APGD-CE. Different defenses fail in different ways. APGD-CE can be effective when confidence reduction provides a useful attack signal, while APGD-DLR is often more reliable when logits are rescaled, gradients are poorly conditioned, or the attack needs to focus more directly on the decision boundary. The inclusion of both losses therefore increases the diversity and reliability of the evaluation.

Overall, AutoAttack shows that reliable robustness evaluation requires diverse, adaptive, and well-calibrated attacks rather than a single manually tuned method. This is important for the present thesis because the experimental attacks used later are taken from the AutoAttack framework. Its components can be analyzed jointly, as a complete robustness evaluation, or separately, in order to understand which types of attacks produce stronger or weaker effects on the selected models. The next section introduces RobustBench, which provides the standardized model and evaluation environment in which these attacks are commonly used.

RobustBench

The evaluation failures discussed above point to a broader methodological problem: the absence of a reliable and unified evaluation framework for adversarial robustness. Although nearly 3,000 papers on adversarial robustness had been published, there was still no consistent benchmark that allowed fair comparison across defenses. Building on this problem, and on the evaluation principles discussed by Carlini *et al.* [17], Croce *et al.* introduced *RobustBench*, a standardized benchmark for evaluating adversarial robustness in deep learning models [29].

RobustBench addresses these evaluation problems by excluding defense mechanisms that are especially likely to produce misleading robustness estimates. In particular, it does not include classifiers with zero gradients with respect to the input, randomized classifiers, or models that depend on an optimization loop at inference time. A classifier has zero gradients when its output does not provide useful derivative information about how small input changes affect the prediction. This can cause gradient-based attacks to fail, not because adversarial examples

are absent, but because the attack cannot obtain a meaningful direction for updating the input. Randomized classifiers introduce stochastic operations into the prediction process, so the same input may produce different outputs or gradients across repeated evaluations. For this reason, RobustBench avoids these cases in its main benchmark, since they can obscure weaknesses rather than demonstrate true robustness [29].

The benchmark mainly focuses on the ℓ_2 and ℓ_∞ threat models. A threat model defines what the attacker is allowed to do, including the size and type of perturbation that may be added to an input. The ℓ_2 threat model constrains the Euclidean magnitude of the perturbation, meaning that the overall amount of change across the image must remain small. By contrast, the ℓ_∞ threat model constrains the maximum change allowed to any individual pixel, ensuring that no pixel is altered beyond a fixed perturbation budget. These two settings are widely used because they provide standard and measurable ways to compare adversarial robustness across models. RobustBench evaluates models under these threat models on established datasets such as CIFAR-100, which contains 60,000 color images divided into 100 object classes.

A central contribution of RobustBench is its model zoo, which is a curated repository of trained models and evaluated defenses. This model zoo allows researchers to compare new defenses against previously submitted models under the same evaluation conditions. It also improves reproducibility, since researchers can directly access existing models instead of relying only on reported results. As a consequence, new attacks can be tested against a broad collection of defenses in a controlled and comparable manner.

RobustBench also adopts AutoAttack as a standard baseline for adversarial evaluation. AutoAttack is an ensemble of four complementary attacks combined into a parameter-free procedure. In this context, parameter-free means that the attack does not require manual tuning of attack-specific hyperparameters, which reduces the risk that results depend on weak or poorly selected settings. The ensemble includes both white-box attacks, which use gradient information from the model, and a black-box attack, which does not require direct access to gradients. By combining different attack strategies, AutoAttack provides a stronger and more reliable estimate of robustness than a single attack used in isolation. Its purpose is therefore to reduce the chance that a defense appears robust merely because it was evaluated with an insufficiently strong or improperly tuned attack [28, 29].

For the present thesis, RobustBench is important because it provides a standardized source of pretrained models and evaluated robustness settings. This allows the later experiments to compare standard and adversarially trained models under a common framework, rather than relying on separately trained models or inconsistent evaluation procedures. In particular, RobustBench makes it possible to select Wide Residual Network models and evaluate them using attacks from the AutoAttack framework.

WideResNets

After introducing the benchmark and attack framework, it is also necessary to clarify the model family used in the experiments. This thesis focuses on Wide Residual Networks because they are widely used in adversarial robustness evaluation and appear frequently in RobustBench-style comparisons. They provide a strong convolutional baseline for CIFAR-scale image classification

and allow standard and adversarially trained models to be compared under the same architectural family.

A major limitation of deep neural networks before the introduction of residual learning through ResNet was the difficult training of deep architectures with multiple layers. In principle, adding more layers should allow the network to learn more complex features. In practice, however, very deep networks often suffered from the degradation problem: after a certain depth, training error increased even when overfitting was not the main cause. This meant that the network was not merely failing to generalize to unseen data; rather, it was becoming difficult to optimize during training. He *et al.* introduced Residual Networks (ResNets) to address this optimization problem and showed that substantially deeper networks could be trained effectively [30].

To address this issue, ResNet introduced residual learning, a training strategy in which a neural network learns the change needed to improve an input representation, rather than learning the entire transformation from scratch. Instead of requiring a group of layers to learn a direct mapping $H(x)$ from an input x to an output, the network learns a residual function:

$$F(x) = H(x) - x,$$

so that the desired mapping becomes:

$$H(x) = F(x) + x.$$

Here, x is the input representation, $H(x)$ is the full transformation that the network aims to learn, and $F(x)$ is the residual correction added to the input. This structure is implemented through shortcut, or skip connections. A skip connection passes the input of a block directly to a later layer, where it is added to the learned residual output Figure 8. This mechanism makes optimization easier because the network can preserve useful information through the identity path while learning only the additional changes needed for better prediction. Using this design, He *et al.* trained networks with up to 152 layers, achieving stronger performance than earlier architectures such as VGG while using lower computational complexity. However, extremely deep ResNets, such as those with more than 1000 layers, still showed limitations, particularly in terms of diminishing performance gains and increased training cost.

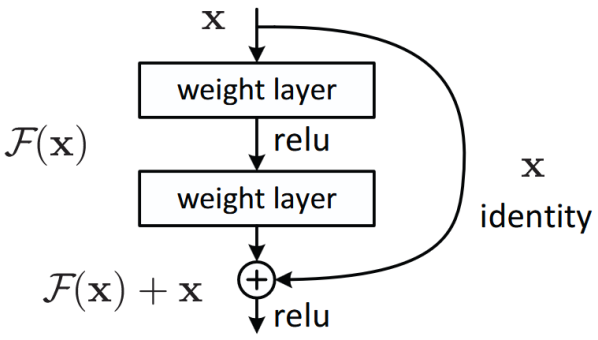


Figure 8: Residual learning block with an identity shortcut connection

Building on residual learning, Zagoruyko and Komodakis [31], challenged the assumption that increasing depth is always the best way to improve performance. They argued that excessively

deep residual networks can suffer from diminishing feature reuse. Feature reuse refers to the extent to which later layers can effectively use representations learned by earlier layers. When a network becomes too deep, many layers may contribute only small or redundant transformations, which increases computational cost and slows training. Instead of making ResNets deeper, the authors proposed making them wider. Width refers to the number of channels, or feature maps, in a convolutional layer. A feature map is a learned representation that captures a particular visual pattern, such as an edge, texture, or object part. Increasing width therefore gives each layer more representational capacity without requiring a very large number of layers.

This design led to Wide Residual Networks (WRNs), which reduce depth while increasing the number of channels inside each residual block. In this architecture, the model can learn richer representations at each layer, while avoiding the optimization and training difficulties associated with extremely deep networks. Zagoruyko and Komodakis showed that WRNs with only 16 to 40 layers could outperform much deeper residual networks, including networks with more than 1000 layers, while also training more efficiently. They also introduced dropout between convolutional layers. Dropout is a regularization method that randomly removes some activations during training, forcing the network not to depend too strongly on specific features. Regularization refers to techniques that reduce overfitting, where a model performs well on training data but poorly on unseen data. In standard narrow ResNets, dropout was often ineffective or even harmful, but in wider residual networks it helped improve generalization by controlling the larger capacity introduced through increased width.

The importance of residual architecture also extends to adversarial robustness. Cazenavette *et al.* [32], argued that architectural choices can influence how vulnerable a network is to adversarial noise. Their analysis is based on the idea that each layer of a feed-forward network can be interpreted as an approximate solution to a sparse coding problem. Sparse coding is a representation method in which an input is described using only a small number of active components from a larger set of possible features. Basis pursuit is an optimization method used to find such sparse representations. The authors note that iterative pursuit methods can be more stable against adversarial noise, but when these methods are applied layer by layer, errors may accumulate as the signal passes through deeper networks.

Deep Pursuit addresses this issue by treating the activations of all layers as part of a single global optimization problem rather than as separate layer-wise computations. This is important because layer-wise processing can create an information bottleneck: each layer must pass information to the next through a limited representation, so small distortions introduced early in the network may become amplified later. Skip connections help reduce this problem by allowing information to bypass some intermediate transformations and flow more directly across the network. As a result, residual architectures can limit the accumulation of errors and provide more stable propagation of useful features. In this sense, the ResNet family is not only important for training deeper models, but also for understanding how architectural design can affect robustness against adversarial perturbations.

For these reasons, Wide Residual Networks provide a suitable model family for the experimental part of this thesis. They combine strong image-classification performance, residual information flow, and sufficient representational capacity for intermediate-layer analysis. In the later experiments, WRN28-10 and WRN70-16 are therefore used as the main architectures for comparing standard and adversarially trained models under the AutoAttack and RobustBench evaluation setting.

Conclusion

This chapter has shown that adversarial vulnerability is a persistent and structurally significant problem in deep neural networks. Across the methods reviewed, from L-BFGS and FGSM to DeepFool, C&W, and PGD, adversarial examples consistently demonstrate that small, carefully constructed perturbations can produce substantial changes in model prediction. This vulnerability is therefore not limited to a specific architecture, dataset, or attack formulation. Rather, it reflects a broader weakness in the geometry of learned decision boundaries, where high clean accuracy can coexist with instability under adversarial perturbation.

Early preprocessing methods, including noise injection and denoising autoencoders, attempted to remove adversarial perturbations before classification, but they were unable to provide reliable protection against adaptive attacks. Regularization-based and training-based methods, such as Deep Contractive Networks, defensive distillation, and adversarial training, improved robustness under specific conditions, but each introduced its own limitations. Some defenses increased computational cost, while others provided robustness only against the attacks used during evaluation. The failure of defensive distillation under the Carlini and Wagner attacks is especially important because it shows that apparent robustness may disappear when a defense is tested against a stronger and better-adapted adversary.

A central conclusion of this chapter is that adversarial robustness must be understood as empirical and threat-dependent. A model cannot be described as robust in a general sense unless the attacker's knowledge, objective, perturbation norm, and computational budget are clearly specified. This is why methods such as PGD became important: they placed attacks and defenses within a common min-max optimization framework and established a stronger standard for robustness evaluation. Nevertheless, even PGD-based adversarial training does not eliminate adversarial vulnerability. It improves resistance within a defined perturbation set, but it does not fully explain how adversarial examples are represented inside the model.

This limitation motivates the experimental direction of the thesis. Most of the literature reviewed in this chapter evaluates adversarial robustness at the input-output level, by measuring perturbation size, attack success rate, or final classification accuracy. While these measures are essential, they do not explain how adversarial inputs move through the internal layers of a network. In particular, they leave open the question of whether adversarial examples form distinct patterns in feature space, whether these patterns change across layers, and whether adversarially trained models organize these representations differently from standard models.

The following chapter addresses this gap by shifting from external prediction behavior to internal representation behavior. It defines the mathematical framework used to describe CNN activations, flattened activation vectors, SVD-projected corevectors, and tail energy. This framework provides the basis for the later experimental analysis of whether adversarial samples behave differently from clean samples inside the network.

Chapter 3: Mathematical Model based on internal representation analysis

Introduction

The previous chapters introduced adversarial examples, attack and defense methods, and the standard robustness-evaluation tools used in this thesis. However, the aim of this work is not limited to determining whether an adversarial input changes the final prediction of a model. The thesis also investigates how clean and adversarial inputs behave inside the network before the final classification decision is produced.

This chapter introduces the mathematical framework used to describe internal representations in convolutional neural networks. It defines layer activations, convolutional feature maps, flattened activation vectors, SVD-projected corevectors, and tail energy. These concepts are necessary because the final output of a neural network provides only a limited view of model behavior. A clean image and an adversarial image may look almost identical in pixel space, but they may produce different activation patterns across the hidden layers of the network.

The framework developed in this chapter provides the basis for the later corevector analysis and tail-energy detector. It explains how internal behavior is represented mathematically and how representation-level statistics can be used to study adversarial samples inside the network.

Mathematical Model of CNN Internal Representations

Layer Activations

Let an input image be denoted by x. A convolutional neural network processes this image through a sequence of layers. The input itself is written as:

$$h^{(0)}(x) = x.$$

The output of layer l is called the activation of that layer and is denoted by:

$$h^{(l)}(x).$$

In general, each layer applies a transformation to the output of the previous layer:

$$h^{(l)}(x) = f^{(l)}\left(h^{(l-1)}(x)\right).$$

Here, $f^{(l)}$ represents the operation performed by layer l. Depending on the type of layer, this operation may include convolution, normalization, a nonlinear activation function, pooling, or a fully connected transformation. Each layer therefore transforms the representation produced by the previous layer into a new representation.

This notation makes it possible to describe the network as a sequence of internal transformations rather than only as a function that maps an image to a final class label. This is important for the present thesis because adversarial perturbations may affect the hidden representations of the network even when the input image appears visually unchanged.

Convolutional Feature Maps

In a convolutional layer, the activation is not represented as a single vector. Instead, it is represented as a three-dimensional tensor:

$$h^{(l)}(x) \in \mathbb{R}^{C_l \times H_l \times W_l},$$

where C_l is the number of channels, and H_l and W_l are the spatial height and width of the activation.

Each channel of this tensor is called a feature map. A feature map contains the response of one learned filter across different spatial positions in the image. In early layers, these feature maps usually capture simple visual patterns, such as edges, corners, textures, or colour changes. In deeper layers, the feature maps usually represent more complex and abstract patterns that are more closely related to the final classification task [33, 34, 14].

For a convolutional layer, the pre-activation value of output channel k at spatial position (i, j) can be written as

$$z^{(l)}_{k,i,j}(x) = b^{(l)}_k + \sum_c \sum_u \sum_v w^{(l)}_{k,c,u,v} h^{(l-1)}_{c,i+u,j+v}.$$

Here, $w^{(l)}_{k,c,u,v}$ are the learned convolutional weights and $b^{(l)}_k$ is the bias term. The index c refers to the input channel, while u and v refer to the spatial coordinates of the convolutional filter. The same filter is applied across different spatial positions. This property is called weight sharing, and it allows the network to detect the same visual pattern in different parts of the image.

After the convolution operation, a nonlinear function is usually applied:

$$h^{(l)}_{k,i,j}(x) = \phi\left(z^{(l)}_{k,i,j}(x)\right).$$

where ϕ may be a function such as ReLU or SiLU. This nonlinearity is important because it allows the network to represent complex patterns that cannot be captured by linear transformations alone. As a result, the network can gradually build more expressive representations from layer to layer.

From Activations to Corevectors

Although convolutional activations are naturally represented as tensors, the analysis in this thesis requires vector representations, since the later SVD-based analysis projects activations onto layer-wise SVD bases and studies the resulting projected coefficients. For layer l, the activation tensor is therefore flattened into a vector

$$a_l(x) = \mathrm{vec}\left(h_{\mathrm{in}}^{(l)}(x)\right) \in \mathbb{R}^{d_l},$$

where, for a convolutional layer with activation shape $C_l \times H_l \times W_l$, the dimension is $d_l = C_l H_l W_l$.

To study the geometry of layer l, an SVD basis is obtained from the corresponding layer transformation. In the implementation used in this work, the SVD basis is obtained from the layer operator. For a fully connected layer, this operator is given by the layer weight transformation. For a convolutional layer, the convolution is represented through an equivalent operator form.

When the selected layer includes a bias term, the bias can be included in the same operator form by concatenating a constant value to the flattened activation vector and concatenating the bias as an additional column of the layer operator:

$$\tilde{a}_l(x) = \begin{bmatrix} a_l(x) \\ 1 \end{bmatrix}, \qquad A_l = \begin{bmatrix} W_l & b_l \end{bmatrix}.$$

In this way, the affine transformation can be written as

$$A_l \tilde{a}_l(x) = W_l a_l(x) + b_l.$$

Singular Value Decomposition

Singular Value Decomposition is applied to the layer operator W_l:

$$W_l = U_l \Sigma_l V_l^T.$$

Here, U_l and V_l are orthogonal matrices containing the left and right singular vectors, respectively, while Σ_l is a diagonal matrix whose entries are the singular values. The matrix V_l contains orthogonal directions associated with the selected layer transformation, and the diagonal entries of Σ_l are the singular values, which indicate the strength of each direction in the layer operator [35, 36]. The singular directions are ordered from largest to smallest singular value: the first directions correspond to the dominant directions of the layer transformation, while the last directions correspond to lower singular-value, or tail, directions.

This distinction is important for the detector developed later in the thesis. Dominant directions describe the main structure of the layer transformation, while tail directions describe weaker components associated with lower singular values. Although these directions are less dominant, they may still be informative: adversarial perturbations may introduce abnormal changes that appear more clearly in these lower singular-value directions than in the dominant representation structure.

Projection onto the SVD Basis

Given the SVD basis of layer l, the flattened activation $a_l(x)$ is projected onto this basis. When a bias term is included in the layer operator, the activation vector is augmented with a constant component before projection. For an input x, this projection defines the corevector,

$$\alpha_l(x) = V_l^T a_l(x).$$

Each coefficient $\alpha_{l,i}(x)$ measures how strongly the representation of x is aligned with singular direction i. Two images may be close to each other in pixel space, but they may still produce different corevectors inside the network. This distinction is especially important for adversarial examples. An adversarial image can look almost identical to a clean image while producing a different internal representation that leads the model toward an incorrect prediction [15, 16].

Using the corevector, the energy of sample x in a set of singular directions S is defined as

$$E_S^{(l)}(x) = \sum_{i \in S} \alpha_{l,i}(x)^2.$$

This quantity measures how much of the layer representation lies inside the subspace spanned by the directions in S. If the energy is large, then the sample has a strong component in that selected subspace. If the energy is small, then the sample is only weakly represented in that subspace.

This definition is useful because it allows different parts of the representation space to be studied separately. For example, one can measure the energy of a sample in the dominant directions, in the intermediate directions, or in the tail directions. This makes it possible to examine where clean and adversarial samples differ most strongly inside the layer representation.

Tail Energy

In this thesis, special attention is given to the tail of the SVD spectrum. Let T_l denote the set of tail directions at layer l. In the experiments, this set corresponds to the last retained SVD components used for the detector.

$$E_{\text{tail}}^{(l)}(x) = \sum_{i \in T_l} \alpha_{l,i}(x)^2.$$

This score measures how much of the sample representation lies in the lower singular-value part of the SVD basis. A high tail-energy score means that the sample has a relatively large component in directions that are not dominant in the selected layer transformation.

The motivation behind this statistic is that clean samples define the reference distribution of tail-energy scores, while adversarial samples may produce abnormal activation patterns in the SVD-projected space. Even when an adversarial image remains visually close to a clean image, its internal representation may produce stronger energy in the lower singular-value directions. If this occurs, then the tail energy of the adversarial sample can become larger than the tail energy of a clean sample.

For this reason, tail energy provides a representation-level statistic for detecting abnormal internal behavior. It does not depend directly on the final predicted class or on the model's softmax confidence. Instead, it measures whether the internal representation of the sample is unusual with respect to the clean reference distribution.

Connection to Adversarial Detection

The final output of a neural network gives only limited information. It indicates which class the model predicts and how confident the model is, but it does not explain how the input moved through the internal layers before reaching that prediction. This limitation motivates the use of intermediate representations for adversarial detection.

Intermediate representations provide additional information because they show how the input is transformed at different depths of the network. This is useful for adversarial detection because adversarial perturbations may affect the internal representation even when the input image still appears normal. By analyzing these representations, it becomes possible to identify changes that are not visible from the final output alone.

Using the notation introduced in this chapter, the detector can be described with the following formula. For a selected layer l, the input is rejected when its tail energy is larger than a threshold τ:

$$D_l(x) = \begin{cases} 1, & E_{\text{tail}}^{(l)}(x) > \tau, \\ 0, & E_{\text{tail}}^{(l)}(x) \leq \tau. \end{cases}$$

Here, $D_l(x) = 1$ means that the sample is detected as suspicious, while $D_l(x) = 0$ means that the sample is accepted. The threshold τ determines how strict the detector is. A lower threshold rejects more samples, while a higher threshold accepts more samples.

Figure 9 provides a conceptual overview of the analysis pipeline used in this thesis. An input image is processed by the convolutional neural network through a sequence of early, intermediate, and final layers. The corresponding internal representations can then be extracted, converted into vector representations, and analysed through SVD-based methods. In principle, this analysis can be applied to any selected layer of the network. The resulting representation-level score is later used to study whether adversarial samples behave differently from clean samples inside the network.

Conclusion

This chapter established the mathematical foundation for the internal representation analysis used in the thesis. The network was described as a sequence of layer activations, and convolutional feature-map tensors were flattened into activation vectors so that they could be analyzed using matrix-based methods.

The chapter then introduced layer-wise SVD projection bases obtained from the selected layer transformations. By projecting activations onto these bases, each sample is represented through a

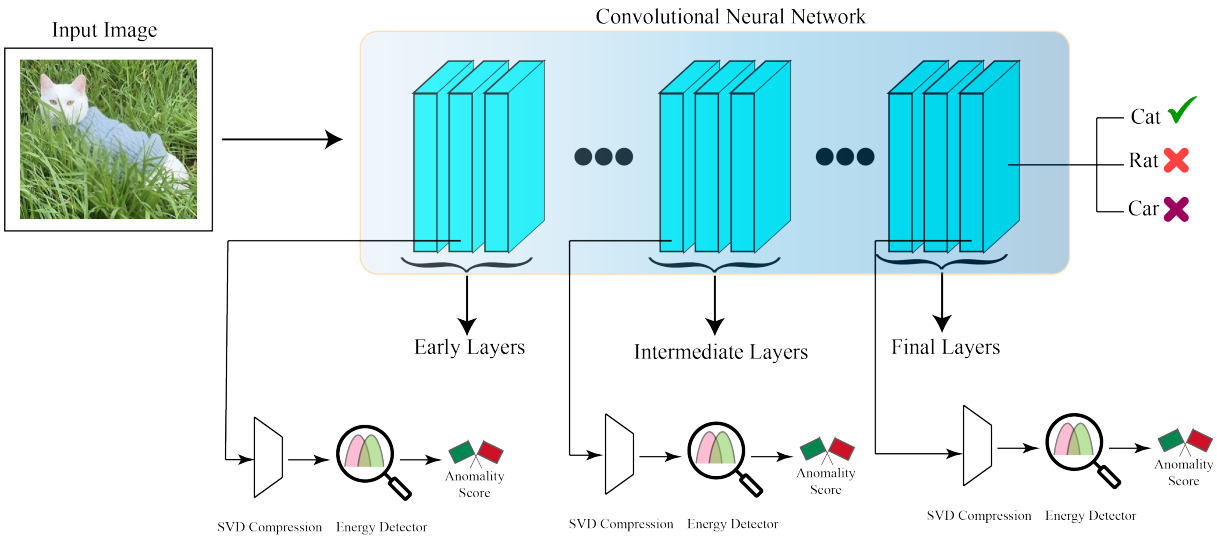


Figure 9: Overview of the layer-wise analysis pipeline

corevector. This makes it possible to study how much of a sample representation lies in different parts of the SVD spectrum.

Finally, tail energy was introduced as the main statistic used in the later detector. The key idea is that clean samples define the reference behavior of the detector, while adversarial samples may produce abnormal energy in lower singular-value tail directions. This representation-level view motivates the experimental analysis developed in the following chapters.

Chapter 4: Experimental Setup and Model Analysis

Introduction

The previous chapters established the foundations required for the experimental analysis. Chapter 2 reviewed the adversarial attacks, defenses, robustness benchmarks, and model architectures used in this work, including AutoAttack, RobustBench, and Wide Residual Networks. Chapter 3 then introduced the mathematical framework for analyzing internal representations through activations, corevectors, SVD bases, and tail energy.

This chapter has two main aims. First, it defines the experimental environment, including the model architectures, training settings, dataset, and adversarial attacks used in the evaluation. Second, it provides a preliminary analysis of the selected models before introducing the proposed detector. This analysis focuses on output-level quantities such as softmax confidence, calibration, prediction margins, and risk-coverage behavior. The purpose is to determine whether the final prediction layer already contains enough information to distinguish reliable from unreliable predictions, and whether this behavior changes under adversarial training.

This chapter therefore serves as a bridge between the robustness evaluation framework and the detector-based analysis developed in the following chapter. If output-level confidence were sufficient to separate clean, incorrect, and adversarial samples, then a more complex detector based on internal activations would be less necessary. However, as the results show, confidence-based indicators are limited and highly dependent on both the model training procedure and the attack objective. This motivates the later use of corevector-based methods, which examine intermediate representations rather than only the final softmax output.

Experimental Setup

As discussed in Chapter 2, Wide Residual Networks are suitable for this study because they combine residual connections with increased representational capacity. Residual connections make deep networks easier to optimize, while widened residual blocks increase the number of feature maps without requiring excessive depth. These properties make Wide Residual Networks a widely used architecture in adversarial robustness evaluation.

For these reasons, this work uses Wide Residual Networks as the main convolutional architecture. In particular, two variants are considered: WRN28-10 and WRN70-16. WRN28-10 denotes a WideResNet with 28 layers and a widening factor of 10, whereas WRN70-16 denotes a deeper and wider variant with 70 layers and a widening factor of 16. Including both architectures makes it possible to examine whether the observed behavior is specific to a single model configuration or remains consistent when model capacity is increased.

The standard WRN28-10 and WRN70-16 models are trained in this work and used as non-

adversarially trained baselines. These models provide the clean reference setting against which the behavior of adversarially trained models and the proposed detector can be compared.

The adversarially trained models are taken from the RobustBench evaluation framework. In particular, this thesis uses the WideResNet models reported by Wang *et al.* [37], where additional generated data is used to strengthen adversarial training. At the time of model selection, the WRN70-16 model from this work represented the leading WideResNet-based model on RobustBench for CIFAR-100 under the ℓ_∞ threat model.

The dataset used in this study is CIFAR-100. CIFAR-100 contains 60,000 color images of size 32×32, divided into 100 object classes. Each class contains 600 images, with 500 images used for training and 100 images used for testing. In total, the dataset contains 50,000 training images and 10,000 test images [38]. CIFAR-100 is selected because it is widely used in adversarial robustness research and is supported by RobustBench, which makes it suitable for comparing standard and adversarially trained models under a common evaluation protocol.

The attacks used in the experiments are the main components of AutoAttack: APGD-CE, APGD-T, FAB-T, and Square Attack. APGD-CE is an untargeted white-box attack based on cross-entropy loss. APGD-T is a targeted version of Auto-PGD, designed to force the model toward specific incorrect classes. FAB-T is a targeted boundary-based attack that searches for small perturbations capable of crossing the model's decision boundary. Square Attack is a black-box attack that does not require gradients and instead modifies square-shaped regions of the input while observing the model's output. Using these attacks together provides a diverse evaluation setting, since the attacks differ in their objectives, access assumptions, and optimization behavior.

The clean accuracy and Attack Success Rate (ASR) under APGD-CE, APGD-T, FAB-T, and Square Attack are reported in Table 1 for both standard and adversarially trained models. Clean accuracy measures the proportion of correctly classified unperturbed test samples. ASR measures the proportion of samples for which the attack successfully changes the model's prediction. A higher ASR therefore indicates that the model is more vulnerable to the corresponding attack.

Table 1: Standard accuracy and attack success rates for standard and adversarially trained WideResNet models.

Model	**Standard Acc.**	**APGD-CE ASR**	**APGD-T ASR**	**FAB-T ASR**	**Square ASR**
WideResNet-28-10 (Standard training)	76.20%	100%	100%	99.96%	99.73%
WideResNet-28-10 (Adversarial training)	72.57%	39.21%	46.56%	60.81%	55.57%
WideResNet-70-16 (Standard training)	76.62%	100%	100%	99.98%	99.73%
WideResNet-70-16 (Adversarial training)	75.22%	51.85%	57.30%	56.92%	51.07%

The results show a clear difference between standard and adversarially trained models. The standard models achieve slightly higher clean accuracy, but they are almost completely vulnerable to all attacks, with attack success rates close to 100%. In contrast, the adversarially trained models show lower attack success rates, confirming that adversarial training improves robustness. However, this improvement does not eliminate vulnerability. The robust models remain attackable, and the level of vulnerability depends on both the architecture and the attack used. These results justify the need for a more detailed analysis of model behavior beyond clean accuracy alone.

Model Analysis

Before introducing the proposed detector, this section analyzes the behavior of the standard and adversarially trained models from the perspective of their final outputs. The analysis focuses on softmax-based quantities, including maximum confidence, calibration, prediction margin, and risk-coverage behavior. These quantities are useful because they describe how certain the model appears to be when making predictions.

The main question is whether the final prediction layer contains enough information to distinguish reliable predictions from unreliable ones. A reliable prediction is one that is likely to be correct, whereas an unreliable prediction may correspond to a misclassified or adversarially perturbed input. If clean and adversarial samples were clearly separable using only output-level confidence, then a detector based on deeper internal activations would be less necessary. However, the following results show that confidence-based behavior is unstable across model types and attack objectives. Therefore, output-level analysis is treated as a diagnostic baseline rather than as a complete detection strategy.

Confidence Analysis in the In-Distribution Setting

The first analysis examines the distribution of softmax confidence on the original CIFAR-100 test set. The samples are divided into correctly and incorrectly classified groups for both the standard and adversarially trained models. The model logits are first transformed into softmax probabilities:

$$p_k(x) = \frac{e^{z_k(x)}}{\sum_{j=1}^{K} e^{z_j(x)}},$$

where $z_k(x)$ is the logit assigned to class k, and K is the total number of classes. A logit is the raw score produced by the network before the softmax transformation. The softmax function converts these logits into probabilities that sum to one.

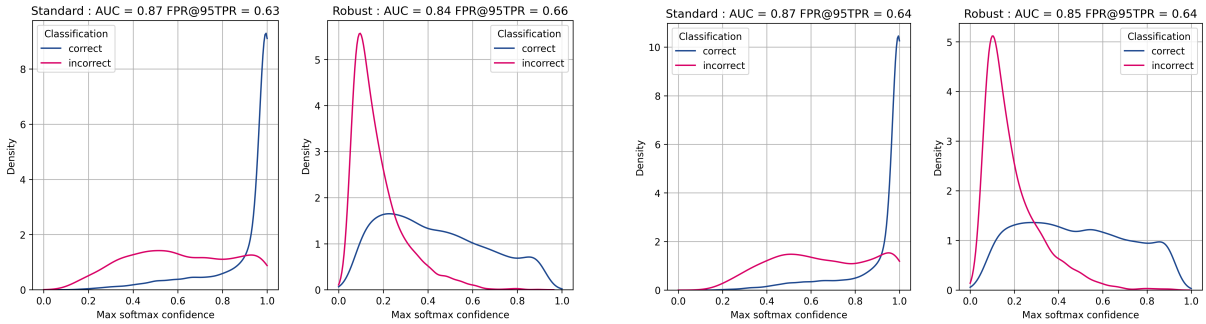
Confidence is defined using the Maximum Softmax Probability (MSP):

$$s(x) = \max_k p_k(x).$$

The MSP score corresponds to the probability assigned to the predicted class. A high MSP value means that the model is highly confident in its prediction, while a low MSP value indicates uncertainty.

Figure 10 shows the confidence distributions for WRN28-10 and WRN70-16.

The results show that the standard models are highly confident when their predictions are correct. However, they are also often highly confident when their predictions are wrong. This indicates overconfidence: the model assigns high probability to its predicted class even when the



(a) WRN28-10 (b) WRN70-16

Figure 10: Confidence analysis for standard and adversarially trained WRN28-10 and WRN70-16 models.

prediction is incorrect. Overconfidence is problematic because it makes confidence less reliable as an indicator of correctness.

The adversarially trained models show a different behavior. Their correct and incorrect predictions are both concentrated at lower confidence values. This means that adversarial training reduces overconfident errors, but it also makes the model underconfident when it is correct. Underconfidence occurs when the model's predicted confidence is lower than its empirical accuracy. As a result, the model becomes less confident overall, and the separation between correct and incorrect predictions becomes weaker.

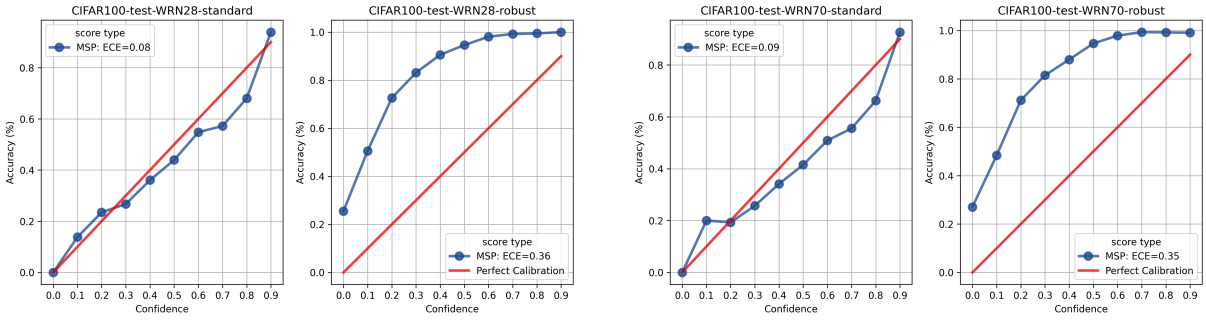
This behavior is consistent across both WRN28-10 and WRN70-16. The AUC values are similar for the two architectures, and the FPR@95TPR remains high, approximately between 0.63 and 0.66 in all cases. AUC measures how well the confidence score separates correct from incorrect predictions across all possible thresholds. FPR@95TPR denotes the false positive rate when the threshold is chosen to achieve a true positive rate of 95%. A high FPR@95TPR means that many incorrect samples are still accepted as reliable when most correct samples are retained. Therefore, although maximum softmax confidence provides some separation, it is not sufficient for reliable distinction between correct and incorrect predictions.

Calibration Analysis

The next analysis examines calibration. Calibration measures whether the confidence assigned by the model matches its empirical accuracy. A well-calibrated model should be correct approximately 80% of the time when it predicts with 80% confidence. Calibration is important because confidence is only meaningful if it reflects the true likelihood of correctness.

Figure 11 compares the calibration behavior of the standard and adversarially trained models.

Expected Calibration Error (ECE) measures the average mismatch between confidence and accuracy:



(a) WRN28-10 (b) WRN70-16

Figure 11: Calibration analysis for standard and adversarially trained WRN28-10 and WRN70-16 models

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right| .$$

Here, n is the total number of samples, and B_m is a bin containing predictions with similar confidence values. The term $\mathrm{acc}(B_m)$ denotes the empirical accuracy within bin B_m, while $\mathrm{conf}(B_m)$ denotes the average confidence in that bin. A lower ECE indicates better calibration.

The standard models are relatively well calibrated, with ECE values of 0.08 for WRN28-10 and 0.09 for WRN70-16. Their calibration curves remain close to the perfect calibration line, which indicates that predicted confidence approximately matches empirical accuracy. In contrast, the adversarially trained models are substantially less calibrated, with ECE values of 0.36 for WRN28-10 and 0.35 for WRN70-16. Their curves lie mostly above the perfect calibration line, meaning that their empirical accuracy is higher than their predicted confidence. This confirms that the adversarially trained models are underconfident.

These results show that adversarial training changes not only robustness but also the reliability of confidence estimates. Although adversarial training improves resistance to attacks, it weakens calibration under clean in-distribution conditions. Therefore, the robust model is not automatically better for confidence-based reliability estimation.

Risk-Coverage Analysis

The third analysis uses the Area Under the Risk-Coverage Curve (AURC) to evaluate selective prediction. In selective prediction, the model is allowed to reject samples with low confidence and make predictions only on the remaining samples. This is useful for evaluating whether a confidence score can rank predictions from most reliable to least reliable.

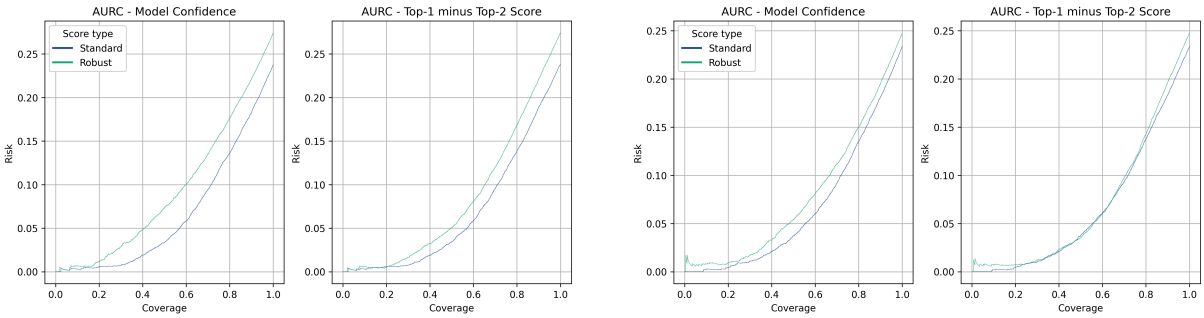
Coverage is the fraction of samples retained by the model. Risk is defined as:

$$\mathrm{Risk} = 1 - \mathrm{Accuracy}.$$

A good confidence score should assign higher scores to correct predictions and lower scores to incorrect ones. Therefore, as coverage decreases, the model should keep the most reliable samples and the risk should decrease.

This analysis compares two confidence-based scoring methods: Maximum Softmax Probability (MSP) and the Top-1 minus Top-2 softmax margin. The Top-1 minus Top-2 margin is the difference between the highest and second-highest softmax probabilities. A large margin indicates that the model strongly prefers one class over the next most likely alternative, while a small margin indicates uncertainty between two competing classes.

Figure 12 reports the AURC results.



(a) WRN28-10

(b) WRN70-16

Figure 12: AURC analysis for standard and adversarially trained WRN28-10 and WRN70-16 models.

For the same coverage level, the standard model curves are generally below the robust model curves. This means that the standard models produce lower risk when predictions are ranked by confidence. Using MSP as the confidence score, the adversarially trained models show higher risk than the standard models across most coverage levels. This indicates that MSP is less effective at ranking correct predictions above incorrect ones in the robust models.

When the Top-1 minus Top-2 margin is used, the gap between standard and robust models becomes smaller, particularly for WRN70-16. However, the robust models still do not show a clear advantage. This suggests that margins provide a slightly more informative ranking than MSP in some cases, but they still do not solve the reliability problem.

Overall, the in-distribution analysis shows that adversarial training substantially changes the confidence behavior of WideResNet models. Although adversarially trained models are more robust under attack, they also show lower or altered clean accuracy, weaker calibration, and less reliable confidence-based separation between correct and incorrect predictions. Standard models perform better on clean data and provide more useful confidence behavior under normal conditions. Therefore, relying exclusively on adversarially trained models is not necessarily optimal for uncertainty estimation or detection.

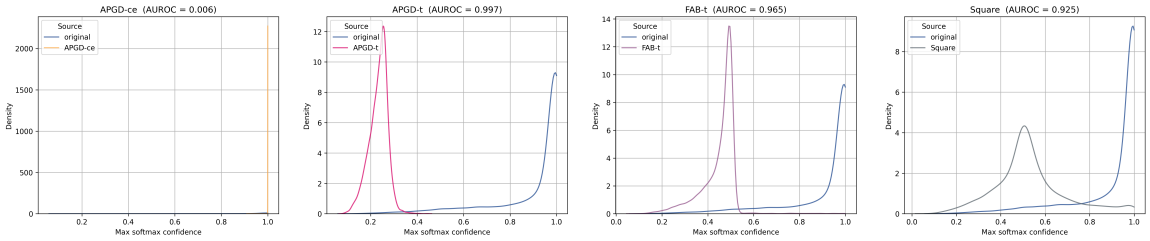
Analysis of Applied Attacks

The previous section examined confidence behavior on clean in-distribution data. This section extends the analysis to adversarial inputs. The aim is to determine whether final-output confidence can distinguish clean samples from attacked samples. This is an important diagnostic step because

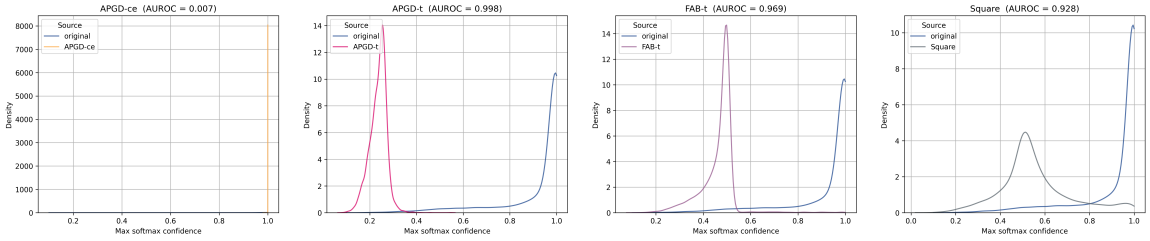
many adversarial detection methods assume that adversarial examples produce lower confidence than clean inputs. However, this assumption does not always hold, since different attacks optimize different objectives and may therefore produce different confidence patterns.

Attack Confidence on Standard Models

Figure 13 shows the confidence distributions of clean and adversarial samples for the standard WRN28-10 and WRN70-16 models.



(a) WRN28-10



(b) WRN70-16

Figure 13: Confidence distribution of adversarial attacks for the standard WRN28-10 and WRN70-16 models.

For the standard models, different attacks produce different confidence behaviors. APGD-CE pushes the model toward extremely confident wrong predictions, resulting in adversarial samples with confidence concentrated near one. This behavior is expected because the cross-entropy objective encourages the attack to increase the loss of the correct class and often produces a confident incorrect prediction.

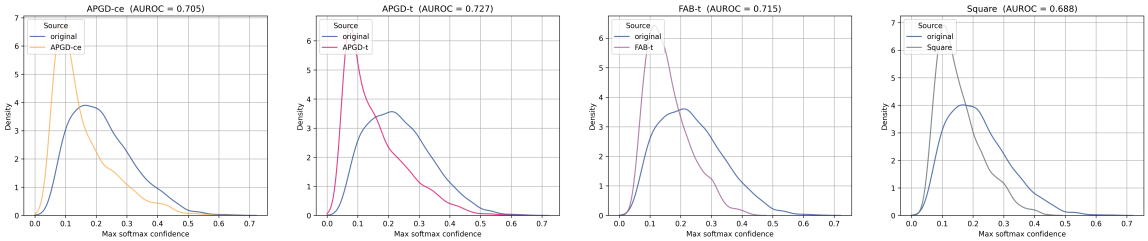
In contrast, APGD-T, FAB-T, and Square Attack generally move samples toward lower-confidence regions. These attacks are therefore more easily separated from clean samples using confidence alone. This pattern is consistent across both WRN28-10 and WRN70-16. However, the difference between attacks also shows a limitation of confidence-based detection. A detector based only on confidence may appear effective against one attack while failing against another attack that produces high-confidence adversarial examples.

Attack Confidence on Adversarially Trained Models

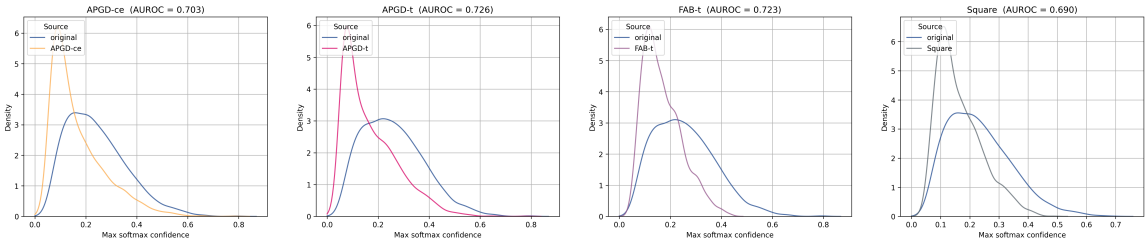
The same analysis is repeated for the adversarially trained models in Figure 14. In this setting, the clean confidence distribution is already shifted toward lower values because of the under-

confident behavior observed in the calibration analysis. Successful adversarial samples are also concentrated in low-confidence regions. As a result, the separation between clean and adversarial samples is weaker than in the standard models.

The AUROC values are approximately between 0.69 and 0.73. AUROC measures the ability of a score to separate two groups across all possible thresholds. Values close to one indicate strong separation, while values closer to 0.5 indicate weak separation. The obtained values suggest that confidence still contains some useful information, but it is not sufficient for reliable adversarial detection in the robust models.



(a) WRN28-10

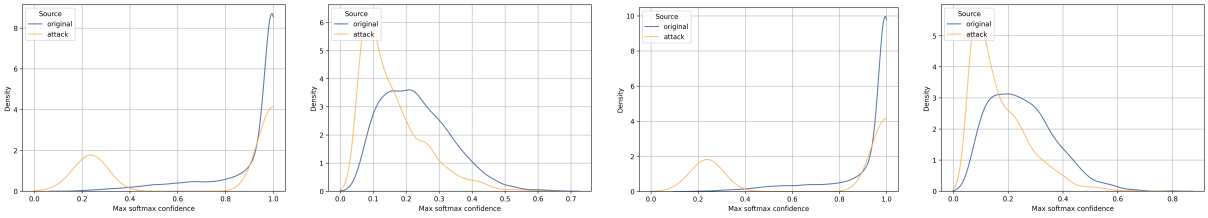


(b) WRN70-16

Figure 14: Confidence distribution of adversarial attacks for the adversarially trained WRN28-10 and WRN70-16 models.

Combined Attack Setting

When all attacks are combined, the confidence distributions become more difficult to separate, as shown in Figure 15. This setting is important because a practical detector should not depend on knowing the exact attack in advance. Instead, it should remain effective across different attack objectives and access assumptions.



(a) WRN28-10 standard (b) WRN28-10 robust (c) WRN70-16 standard (d) WRN70-16 robust

Figure 15: Confidence distributions for combined adversarial attacks on standard and adversarially trained WRN28-10 and WRN70-16 models.

For the standard models, the adversarial confidence distribution becomes bimodal. One group of attacked samples is shifted toward low confidence, while another group remains highly confident and overlaps strongly with the clean distribution near confidence one. This reflects the different behavior of the attack objectives. Some attacks reduce the model's confidence, while others, especially APGD-CE, can produce highly confident adversarial examples. As a result, the performance of confidence-based detection decreases substantially compared with the single-attack cases. For the adversarially trained models, both clean and attacked samples occupy lower confidence regions. The attacked samples are generally shifted further toward lower confidence, but there is still substantial overlap with the clean distribution. Therefore, although the robust models show more stable behavior across combined attacks, softmax confidence alone still does not provide reliable separation between clean and adversarial inputs. The AUROC values remain close to those observed for individual attacks on the robust models, indicating that the combined setting does not fundamentally change the confidence behavior of adversarially trained models.

Conclusion

This chapter defined the experimental environment and provided a preliminary output-level analysis of the selected WideResNet models. The experimental setup used WRN28-10 and WRN70-16 architectures, standard and adversarially trained model variants, the CIFAR-100 dataset, and the main attack components of AutoAttack. The attack success rates showed that standard models are almost completely vulnerable to the evaluated attacks, while adversarially trained models provide stronger but still incomplete robustness.

The model analysis showed that final-output quantities provide useful diagnostic information but are not sufficient for reliable detection. On clean data, standard models are often overconfident, whereas adversarially trained models are generally underconfident and less well calibrated. Risk-coverage analysis further showed that confidence-based scores do not consistently rank reliable predictions above unreliable ones, especially for robust models. Under adversarial attacks, confidence behavior depends strongly on the attack objective. Some attacks produce low-confidence adversarial samples, while others produce highly confident incorrect predictions that overlap with clean samples.

These findings identify the main gap addressed in the next chapter. If confidence, calibration, and softmax margins are unstable at the output level, then adversarial detection should not rely only on the final prediction layer. The next chapter therefore moves to the proposed corevector-based detector, which analyzes intermediate activations inside the network. This shift allows the study to test whether internal representations contain more stable and discriminative information for separating clean and adversarial samples.

Chapter 5: Corevector Evaluation

The previous chapter examined adversarial detection from the perspective of output-level behavior. In particular, it focused on signals that can be observed from the final model prediction, such as softmax confidence. Although these output-level quantities are useful, they describe only the final decision of the network. They do not explain how clean and adversarial samples are represented inside the model before the final classification layer.

This chapter therefore moves inside the network and studies intermediate activations. The main aim is to determine whether clean and adversarial inputs produce different internal representation patterns, and whether these differences provide stronger detection signals.

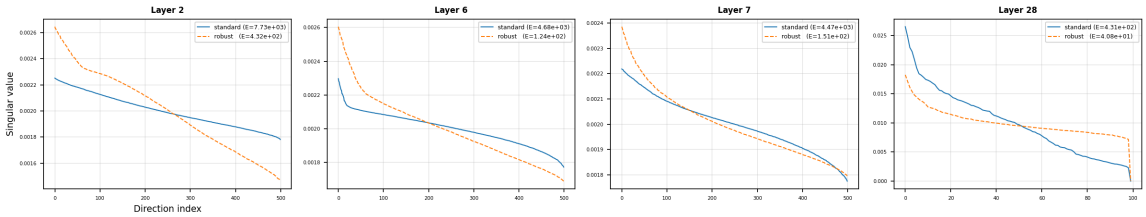
The chapter proceeds in two stages. First, SVD is used to study the geometry of the layer representations. This allows us to examine whether the most informative differences between clean and adversarial samples occur in dominant singular directions or in lower singular-value tail directions. Second, the analysis focuses on corevector tail energy, which measures how much of a sample representation lies in the lower singular-value part of the SVD basis. This statistic is then evaluated as a representation-level detector for adversarial samples.

One architectural point must be clarified. The standard and adversarially trained models are not identical implementations of the same architecture. Both models belong to the WideResNet family, but the standard model uses ReLU nonlinearities, whereas the adversarially trained model uses SiLU nonlinearities. This difference does not undermine the present analysis, because the corevector evaluation is performed within each model separately. The purpose is not to compare individual neurons across the standard and robust models. Instead, the purpose is to determine whether each model contains layer-wise information that can separate clean inputs from adversarial inputs.

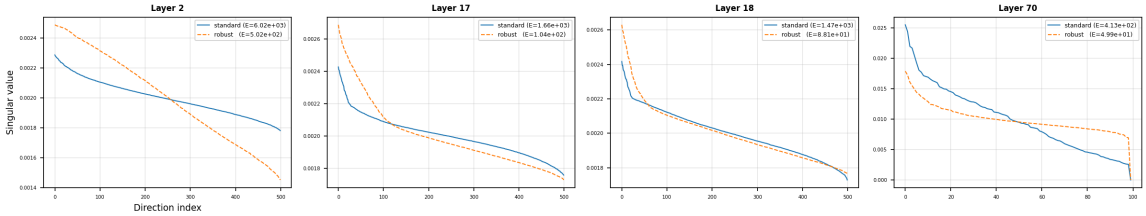
Singular Value Decomposition

Singular Value Decomposition provides a way to analyze the spectral structure associated with selected layer transformations. For each selected layer, the corresponding SVD basis provides orthogonal directions ordered by their singular values. A larger singular value indicates that the corresponding direction is more dominant in the layer-wise transformation. In this setting, the singular-value profile describes how concentrated or distributed the internal representation is across different directions.

We first compare the singular-value profiles of the layer-wise SVD bases under test. Plotting all layers would make the comparison difficult to interpret. Therefore, four representative layers are selected from each architecture: one early layer, two intermediate layers, and the final classification layer. For WRN28-10, layers 2, 6, 7, and 28 are shown. For WRN70-16, layers 2, 17, 18, and 70 are shown. This selection makes it possible to examine how the spectral structure changes across depth without overcrowding the figure.



(a) WRN28-10 standard



(b) WRN70-16 standard

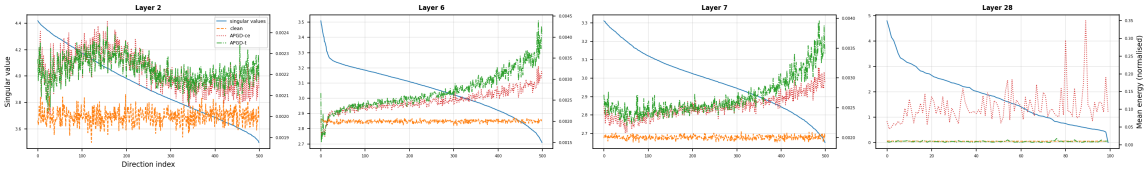
Figure 16: Normalized SVD profiles for selected layers of WRN28-10 and WRN70-16, comparing standard and adversarially trained models

Figure 16 presents the normalized singular-value spectra for the selected layers. The normalized spectra are used instead of raw singular values because raw spectra are strongly affected by scale differences between layers and models. Normalization therefore makes the comparison more meaningful, since it emphasizes the shape of the spectrum rather than its absolute magnitude. Specifically, each singular value is divided by the sum of all singular values for that layer, so that the resulting profile reflects the relative contribution of each direction rather than its absolute scale. To preserve this information, the total spectral energy of each layer, defined as the sum of the squared singular values, is also reported alongside the corresponding curve. This value reflects the overall magnitude of the layer's representation, independently of how that magnitude is distributed across directions.

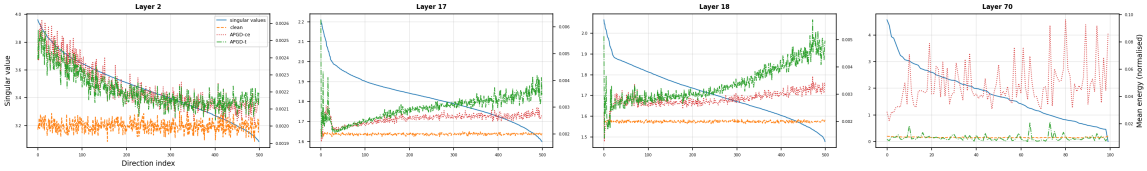
Across both architectures, the standard and adversarially trained models show similar broad spectral patterns, but the relationship between them depends on layer depth. In the early and intermediate layers, the adversarially trained model exhibits a steeper decay than the standard model: although it starts from a higher initial singular value, it falls below the standard model's curve within the first quarter of directions. In the final classification layer in Figure 16, however, this relationship reverses: the standard model decays sharply toward zero after only a small number of directions, while the adversarially trained model retains a comparatively flat profile across most of the spectrum. This indicates that the standard model concentrates most of its final-layer spectral structure in a small number of dominant directions, whereas the adversarially trained model distributes it more evenly at this stage, even though its early and intermediate layers show the opposite tendency.

This distinction is important for the subsequent detection analysis, more directions are required to capture the same proportion of spectral energy in the adversarially trained model's final layer, while the opposite holds in the earlier layers. The total energy values add an independent observation, they are consistently lower for the adversarially trained model across all layers, regardless of depth, indicating an overall reduction in representation magnitude that holds even where the spectral shape itself does not show a consistent pattern.

To identify which part of the singular spectrum is most informative for adversarial detection, the next step analyses the mean normalized energy of clean and adversarial samples along the singular directions. This analysis is shown in Figure 17. The directions are ordered by decreasing singular value. Thus, the left side of each plot corresponds to the dominant large-singular-value directions, whereas the right side corresponds to the lower singular-value tail of the spectrum.



(a) WRN28-10 standard



(b) WRN70-16 standard

Figure 17: Singular values and mean normalized energy along singular directions for clean and adversarial samples in selected layers of the standard WRN28-10 and WRN70-16 models.

The results show that the leading singular directions are not always the most discriminative. In several selected layers, particularly in intermediate and final layers, adversarial samples deviate more strongly from clean samples toward the tail of the spectrum. This means that tail directions, although associated with lower singular values, can contain useful information for distinguishing clean representations from adversarial representations.

This behavior appears in both WRN28-10 and WRN70-16. The exact shape of the energy curves varies across layers and attacks, but the general pattern is consistent: adversarial perturbations are often more visible in lower-energy directions than in dominant singular directions. This observation motivates the use of tail energy as the main representation-based statistic in the following analysis.

The next step tests whether detection performance is simply explained by the amount of spectral mass contained in a selected subspace. For each layer, groups of SVD components are randomly selected, and the energy of each sample is computed after restricting the representation to those selected directions. Given a set of selected component indices S, the corresponding subspace energy is defined as

$$E_S^{(l)}(x) = \sum_{i \in S} \alpha_{l,i}(x)^2,$$

where $\alpha_l(x)$ denotes the projected coefficients of the layer-l. For each selected subspace, the AUROC is then computed by using this energy score to separate clean and adversarial samples. AUROC refers to the Area Under the Receiver Operating Characteristic curve. In this context, it measures how well the energy score ranks clean and adversarial samples separately, with values close to chance indicating weak separation and higher values indicating stronger separation.

In addition, each selected subspace is associated with the sum of its corresponding singular values,

$$\Sigma_S^{(l)} = \sum_{i \in S} \sigma_{l,i}.$$

This quantity measures the amount of spectral mass contained in the selected directions. Plotting AUROC against this value allows us to test whether better detection is mainly caused by selecting directions with larger singular values.

Figure 18 shows that detection performance is not determined only by the amount of spectral mass in the selected directions. Random subspaces with larger singular-value sums do not necessarily produce better AUROC. In several intermediate layers, the tail subspace provides stronger separation than both the dominant head directions and most randomly selected subspaces, especially for APGD-CE and APGD-T.

The final fully connected layer is less informative in this analysis because it is already a compressed class-decision representation. As a result, random SVD subspaces in this layer provide less meaningful variation than those in intermediate layers.

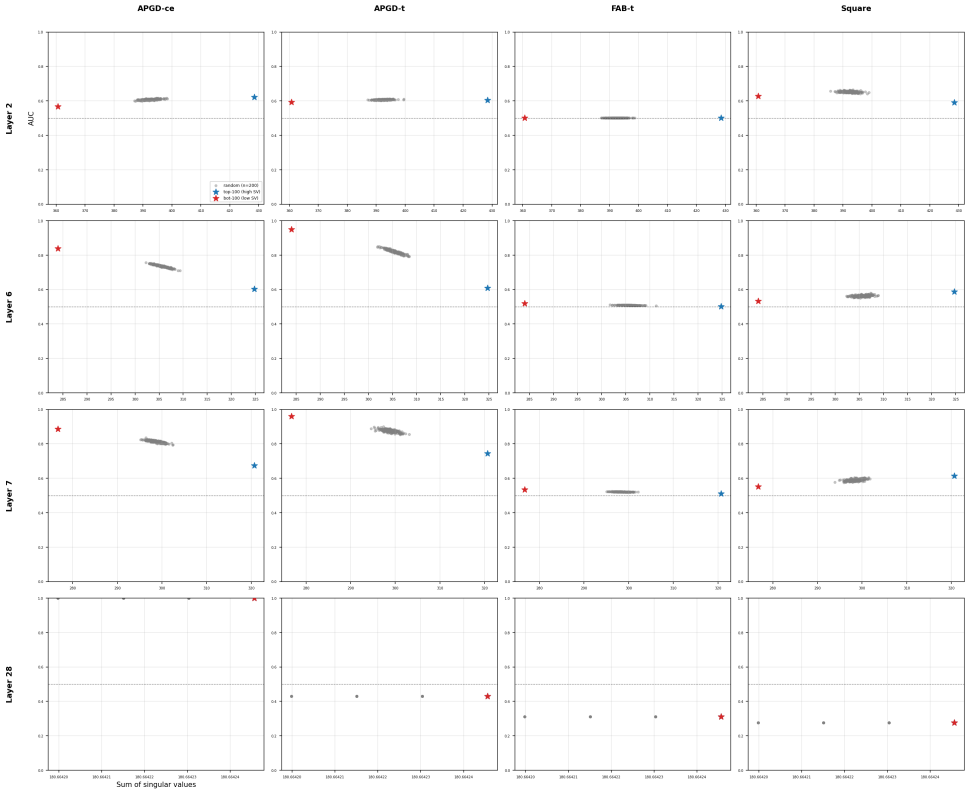
These findings support the earlier observation from the energy-profile analysis. Although useful adversarial information can sometimes appear in intermediate parts of the spectrum, the most consistent separation for the attacks of interest is obtained from lower singular-value tail directions. The tail subspace therefore provides a more stable and interpretable detector feature than either randomly selected SVD components or leading large-singular-value directions.

Corevector Tail Energy

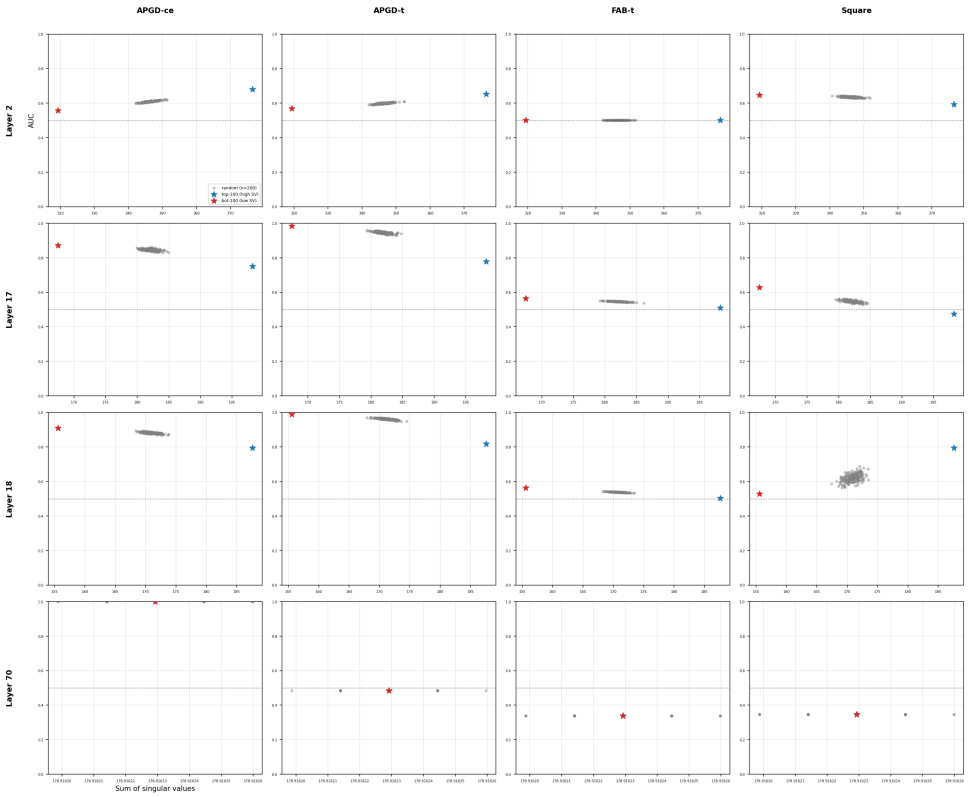
After analysing the singular-value profiles and the distribution of clean and adversarial energy along the singular directions, the analysis focuses on the tail of the SVD basis. The previous results showed that adversarial samples often differ more clearly from clean samples in lower-energy directions than in dominant singular directions. For this reason, the full representation is not used directly, and the analysis does not rely only on the leading components. Instead, tail energy is used as the main corevector-based statistic.

In this chapter, a corevector refers to the layer-wise representation vector extracted for a given sample and used in the corevector analysis. For each layer, the SVD components are ordered according to their singular values. The first components correspond to dominant large-singular-value directions. The last components correspond to lower singular-value directions, which form the tail of the spectrum. These tail directions should not be interpreted as irrelevant. Rather, they capture components that are less dominant in the SVD projection basis and may therefore reveal atypical changes introduced by adversarial perturbations.

For each layer, the tail subspace is defined as the subspace spanned by the last 100 retained SVD components. The flattened activation of each sample is projected onto the SVD basis to obtain its corevector. The energy contained in the tail components of this corevector is then computed.



(a) WRN28-10 standard



(b) WRN70-16 standard

Figure 18: Random SVD subspace probes for the standard WRN28-10 and WRN70-16 models. Each point corresponds to a randomly selected group of 100 SVD directions.

Let $a_l(x)$ be the flattened activation of sample x at layer l,

$$a_l(x) = \mathrm{vec}\left(h^{(l)}(x)\right),$$

and let $V_l = [v_1, \ldots, v_d]$ be the SVD basis of that layer. The corevector is defined as

$$\alpha_l(x) = V_l^\top a_l(x).$$

The tail-energy score is then defined as

$$E_{\text{tail}}^{(l)}(x) = \sum_{i=d-99}^{d} \alpha_{l,i}(x)^2.$$

This score measures how much of a sample representation lies in the lower singular-value part of the retained SVD-projected corevector. The underlying motivation is that adversarial perturbations may introduce internal changes that are more visible in lower singular-value directions than in the dominant projected directions. When this occurs, adversarial samples are expected to produce higher energy in the tail subspace than clean samples. Tail energy can therefore serve as a representation-level signal for adversarial detection.

Figure 19 and Figure 20 show the tail-energy distributions for the standard WRN28-10 and WRN70-16 models, respectively. For each architecture, the four selected layers introduced earlier are reported: layers 2, 6, 7, and 28 for WRN28-10, and layers 2, 17, 18, and 70 for WRN70-16. The columns correspond to the four attacks considered in this work: APGD-CE, APGD-T, FAB-T, and Square.

The separation between clean and adversarial samples is strongly layer-dependent. In the earliest layers, the clean and adversarial tail-energy distributions often overlap. This indicates that early representations do not provide a stable separation signal. In contrast, several intermediate layers show a clear shift of successful adversarial samples toward higher tail-energy values, especially for APGD-CE and APGD-T. This suggests that these attacks introduce internal changes that are more visible in the lower singular-value directions of the representation.

The same general trend is observed for both WRN28-10 and WRN70-16. However, the strength of the separation varies across attacks. APGD-CE and APGD-T usually produce the clearest shift, whereas FAB-T and Square show more overlap in several layers. The intermediate layers consistently provide more useful tail-energy separation than the earliest layers. The final layer can also show strong separation for specific attacks, but its behavior is less consistent across all attacks.

The quantitative AUC values reported in Table 2 confirm the visual trends observed in Figure 19 and Figure 20. For both architectures, the earliest layer provides weak separation, with AUC values close to chance for most attacks. In contrast, the intermediate layers produce much stronger separation for APGD-CE and APGD-T. For WRN28-10, layers 6 and 7 reach AUC values of 83.8% and 96.1%, respectively. For WRN70-16, layers 17 and 18 reach AUC values of 87.1% and 99.0%, respectively. These results confirm that the most useful tail-energy signal is concentrated in the intermediate layers.

The results are less consistent for FAB-T and Square, where the AUC values remain closer to chance in most layers. The final layer also behaves differently across attacks. It perfectly

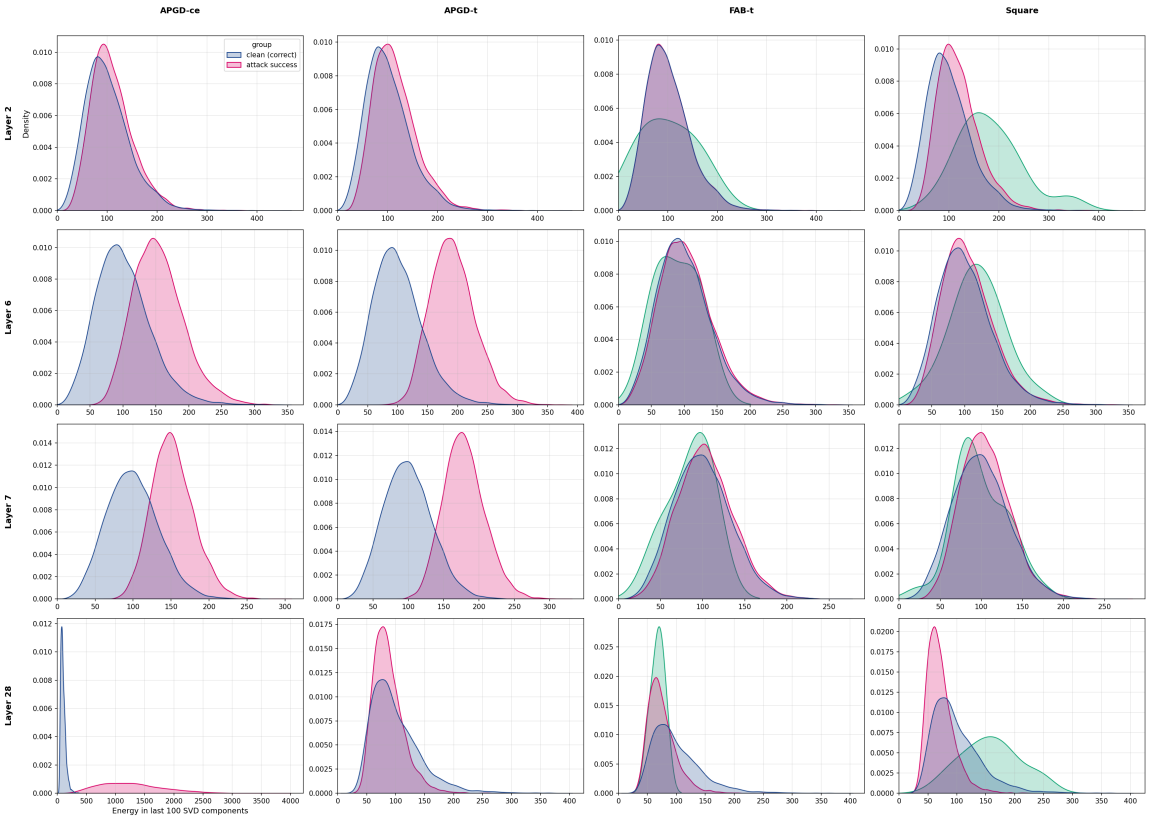


Figure 19: Tail-energy distributions for the standard WRN28-10 model.

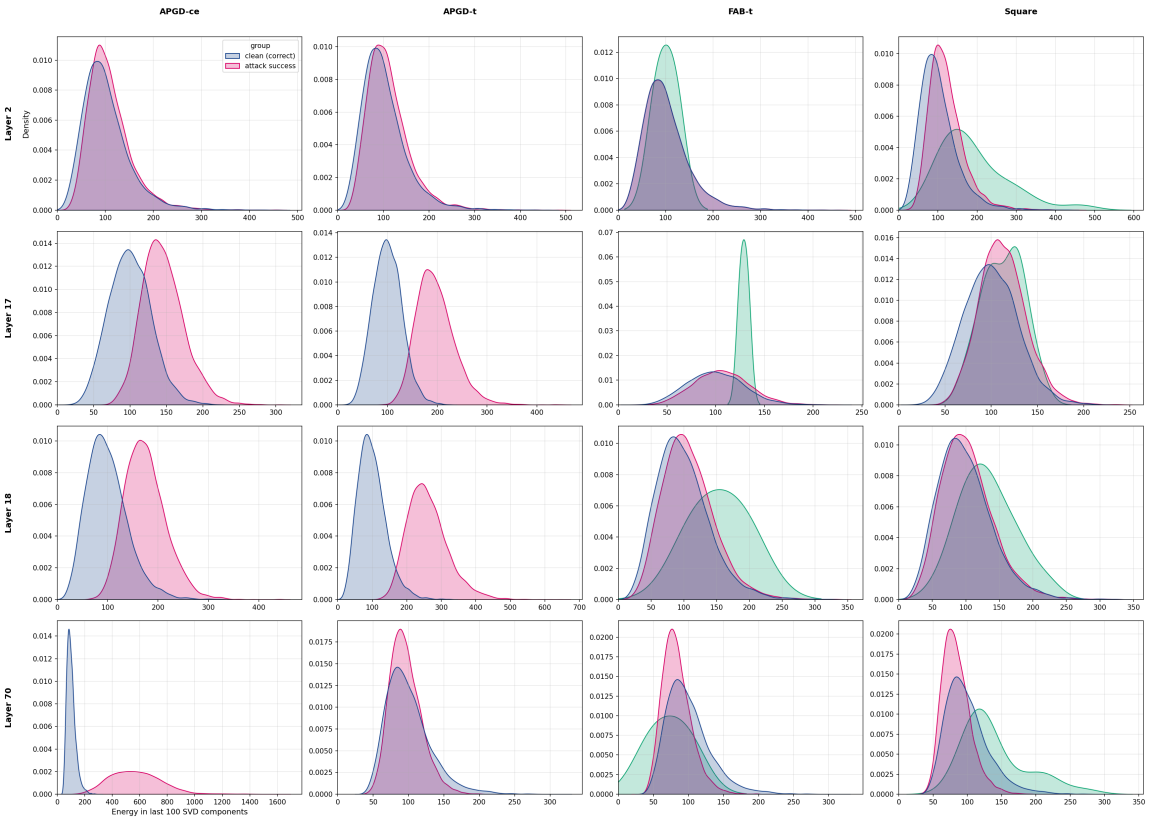


Figure 20: Tail-energy distributions for the standard WRN70-16 model.

Table 2: Tail-energy detection AUC values for selected layers of the standard WRN28-10 and WRN70-16 models. Values are reported in percent.

Model	Layer	APGD-CE	APGD-T	FAB-T	Square
WRN28-10	Layer 2	56.6	59.2	50.1	62.8
WRN28-10	Layer 6	83.8	94.9	52.0	53.3
WRN28-10	Layer 7	88.7	96.1	53.5	55.3
WRN28-10	Layer 28	100.0	43.0	31.1	27.7
WRN70-16	Layer 2	55.7	56.8	50.0	64.6
WRN70-16	Layer 17	87.1	98.4	56.3	62.9
WRN70-16	Layer 18	91.1	99.0	56.4	52.8
WRN70-16	Layer 70	100.0	48.4	34.0	34.6

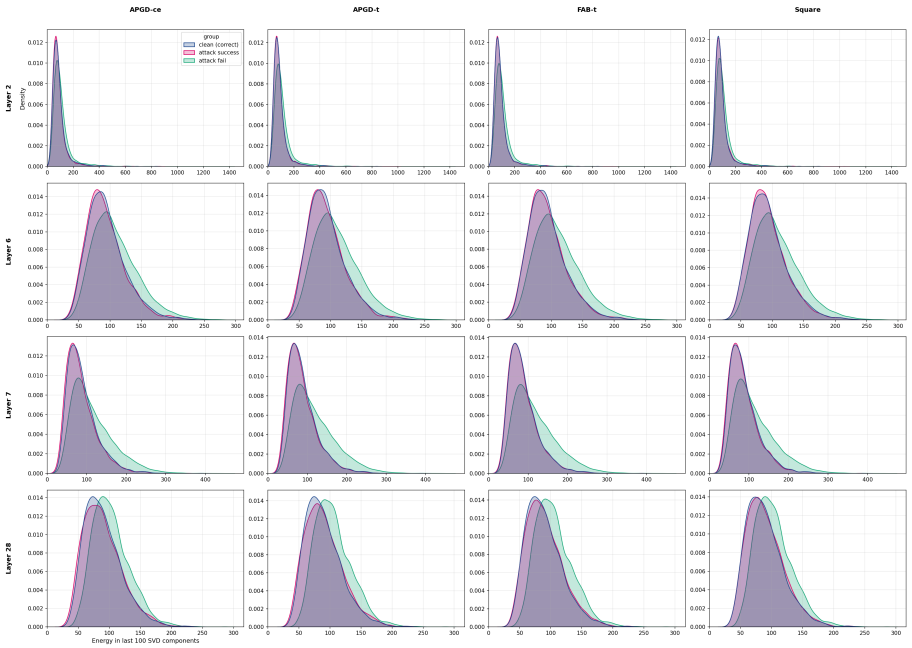
separates APGD-CE, but performs poorly for APGD-T, FAB-T, and Square. This indicates that tail energy is informative, but that its effectiveness depends strongly on both the selected layer and the attack type.

These findings also clarify the scope of the detector developed in the following chapters. Since the tail-energy signal is consistently strong for APGD-CE and APGD-T in the intermediate layers, the final detector is focused on these two attacks. In contrast, FAB-T and Square show weaker and less consistent separation. Therefore, a single tail-energy detector that reliably covers all attacks considered here would not be well supported by the present results. For this reason, FAB-T and Square are retained as diagnostic evaluations, while the final detector is developed for the attacks where the representation-level signal is sufficiently stable.

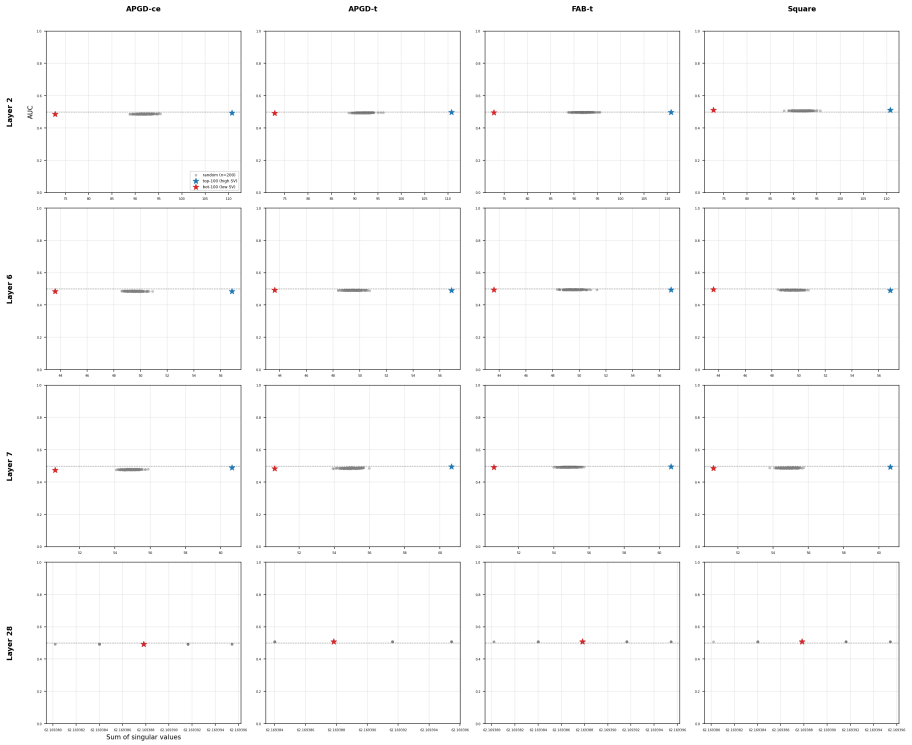
The same analysis was also performed on the adversarially trained WRN28-10 model, as shown in Figure 21. Unlike the standard models, the robust model does not show the same clear and consistent tail-energy separation between clean and adversarial samples. The tail-energy distributions overlap strongly across the analyzed layers and attacks, and the random SVD subspace analysis remains close to chance-level separation. Similarly, the energy profiles along the singular directions do not reveal a stable tail-region deviation comparable to the one observed in the standard models. This suggests that adversarial training changes the internal representation geometry in a way that reduces the usefulness of the proposed tail-energy statistic. Therefore, the detector developed in the following chapters is focused on the standard models, where the representation-level signal is empirically more stable.

Conclusion

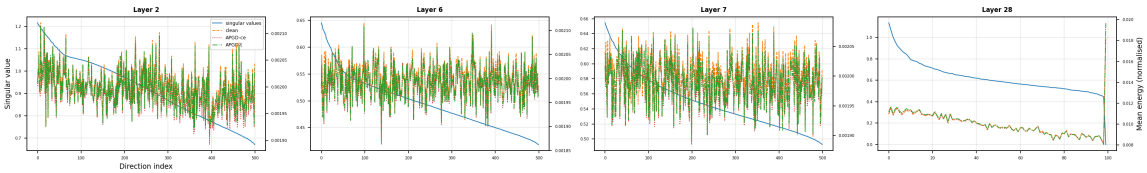
This chapter has shown that adversarial detection can benefit from examining internal model representations rather than relying only on output-level confidence. The SVD analysis demonstrated that the spectral structure of layer representations changes across depth and that adversarially trained models can distribute spectral energy more broadly across singular directions. More importantly, the energy-profile analysis showed that adversarial samples are not always most visible in the dominant large-singular-value directions. In several cases, the most informative differences appear in the lower singular-value tail of the spectrum.



(a) Tail-energy distributions



(b) Random SVD subspace analysis



(c) Singular values and mean energy along singular directions

Figure 21: Corevector and SVD-based analysis for the adversarially trained WRN28-10 model.

The random subspace analysis further showed that detection performance is not simply determined by the amount of spectral mass contained in a selected group of SVD components. Subspaces with larger singular-value sums do not necessarily produce stronger separation. Instead, the lower singular-value tail directions provide a more stable and interpretable basis for detection, particularly in intermediate layers.

The tail-energy results confirm this conclusion. Intermediate layers provide the strongest and most consistent separation for APGD-CE and APGD-T because they contain richer feature representations than the earliest layers while still preserving internal geometric changes introduced by adversarial perturbations. The earliest layers are generally weak because they mainly capture low-level visual patterns, such as edges and local textures, which are often similar for clean and adversarial samples. The final layer is less consistent across attacks because it is already compressed toward the class-decision space, so much of the intermediate representation structure has been reduced. FAB-T and Square remain more difficult to detect using this single statistic, suggesting that their perturbations do not consistently produce the same tail-energy shift observed for APGD-CE and APGD-T. Overall, the chapter establishes corevector tail energy as a meaningful representation-level signal and motivates its use in the final detector for the attacks where the signal is empirically stable.

Chapter 6: Tail-Energy Detector

Introduction

The previous chapter showed that adversarial samples can produce distinctive internal representation patterns, especially in the lower singular-value directions of intermediate layers. This observation motivates the construction of an explicit adversarial detector based on tail energy. Instead of modifying the classifier itself, the detector uses a representation-level statistic to decide whether an input should be accepted or rejected.

The aim of this chapter is to evaluate whether tail energy can serve as a practical detection score. The analysis focuses on the standard WRN28-10 and WRN70-16 models because the earlier corevector evaluation showed that these models contain clear tail-energy separation for some attacks. The main question is whether this separation can be converted into an operational detector that reduces attack success while preserving clean accuracy.

This chapter is organized in two stages. First, a single-layer detector is defined using the tail energy of a selected intermediate layer. Second, the detector is evaluated at different clean false-positive-rate operating points in order to study the trade-off between clean-sample rejection and adversarial-sample rejection. An additional accuracy-matched comparison is then presented to compare tail-energy detection with adversarial training under the same clean-accuracy budget.

Single-Layer Tail-Energy Detector

Based on the previous energy-based corevector analysis, we now use the tail-energy score to construct an adversarial detector. The earlier results showed that, in the standard models, adversarial samples often produce larger energy in the lower singular-value SVD directions than clean samples, especially in the intermediate layers. Therefore, tail energy is used as the detector score.

For each input sample, the flattened activation at a selected layer is projected onto the SVD basis to obtain the corevector. The detector score is then computed as the energy contained in the last 100 SVD components:

$$E_{\text{tail}}^{(l)}(x) = \sum_{i=d-99}^{d} \alpha_{l,i}(x)^2.$$

Here, $E_{\text{tail}}^{(l)}(x)$ denotes the tail-energy score of sample x at layer l. The coefficients $\alpha_{l,i}(x)$ are the components of the corevector in the SVD basis. The final 100 components correspond to the tail of the spectrum, meaning the lower singular-value directions of the SVD-projected representation. A larger score indicates that more of the sample representation lies in these lower singular-value directions.

The detector applies a fixed threshold τ to this score. A sample is rejected when its tail energy exceeds the threshold:

$$D(x) = \begin{cases} 1, & E_{\text{tail}}^{(l)}(x) > \tau, \\ 0, & E_{\text{tail}}^{(l)}(x) \leq \tau, \end{cases}$$

where $D(x) = 1$ denotes a rejected sample and $D(x) = 0$ denotes an accepted sample.

This decision rule is simple and interpretable. If the tail-energy score is below or equal to the threshold, the sample is treated as clean or acceptable. If the score is above the threshold, the sample is treated as suspicious and is rejected. The threshold therefore determines the operating point of the detector. The detector is instantiated separately for the two standard architectures. For WRN28-10, Layer 7 is used. For WRN70-16, Layer 18 is used. These layers are selected because they showed the strongest and most consistent tail-energy separation for APGD-CE and APGD-T among the analysed intermediate layers.

To evaluate the effect of the operating point, two thresholds are tested. These thresholds correspond to target clean false-positive rates of 2% and 1%. The false-positive rate (FPR) measures the fraction of clean samples that are incorrectly rejected by the detector. A lower target FPR makes the detector more conservative. In this case, the threshold moves to the right, fewer clean samples are rejected, and clean accuracy is better preserved. However, fewer adversarial samples are rejected as well. The operating point therefore controls the trade-off between preserving clean accuracy and detecting adversarial inputs.

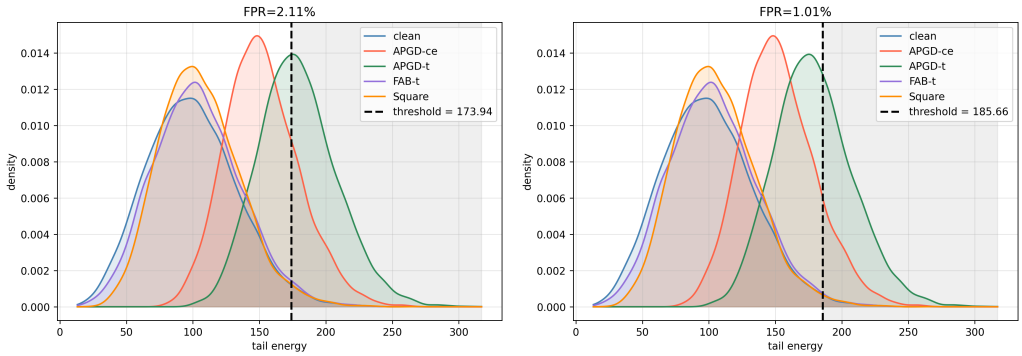
Figure 22 shows this threshold trade-off for the selected detector layers: Layer 7 for WRN28-10 and Layer 18 for WRN70-16.

At the higher-FPR operating point, the threshold is lower. This allows the detector to reject a larger proportion of adversarial samples, but it also rejects more clean samples. When the target clean FPR is reduced to approximately 1%, the threshold moves to the right. This makes the detector more selective with respect to clean samples, but it also reduces the fraction of adversarial samples that fall into the rejected region.

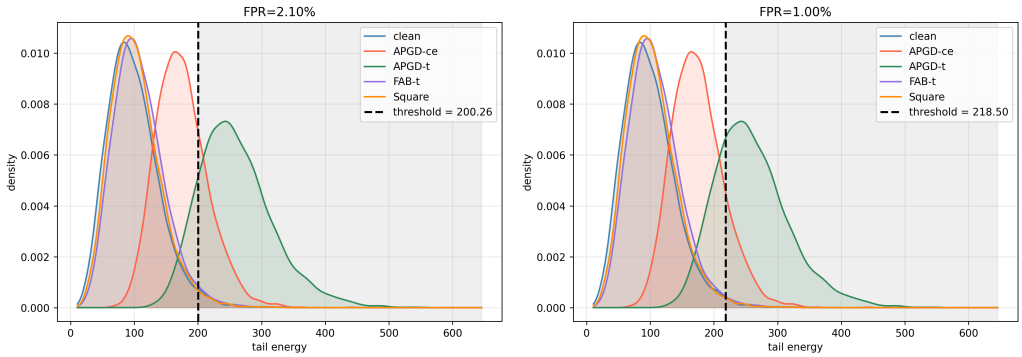
For both WRN28-10 and WRN70-16, APGD-T is more clearly shifted toward high tail-energy values than APGD-CE. As a result, APGD-T is rejected more effectively by the detector. APGD-CE also shifts toward higher tail energy, but it overlaps more with the clean distribution. Its detection is therefore more sensitive to the threshold choice. FAB-T and Square remain close to the clean distribution, which is consistent with the previous AUC analysis. This explains why these two attacks are not the main target of the single-layer detector.

Table 3: Comparison between the standard model, adversarially trained model, and tail-energy detector operating points. Values reported are in percentages

Model	Mode	Standard Acc.	APGD-CE ASR	APGD-T ASR	FAB-T ASR	Square ASR
WRN28-10	Standard training	76.20	100.00	100.00	99.96	99.73
WRN28-10	Adversarial training	72.57	39.21	46.56	60.81	55.57
WRN28-10	FPR = 1% detector	75.44	88.32	61.50	98.87	98.60
WRN28-10	FPR = 2% detector	74.60	79.23	45.53	97.51	97.56
WRN70-16	Standard training	76.62	100.00	100.00	99.98	99.73
WRN70-16	Adversarial training	75.22	51.85	57.30	56.92	51.07
WRN70-16	FPR = 1% detector	75.85	86.06	24.81	98.76	98.64
WRN70-16	FPR = 2% detector	75.01	75.39	13.99	97.68	97.78



(a) WRN28-10



(b) WRN70-16

Figure 22: Tail-energy detector operating points for WRN28-10 and WRN70-16

Table 3 reports the clean accuracy and attack success rate obtained by the standard models, the adversarially trained models, and the tail-energy detector. The standard models achieve the highest clean accuracy, but they are fully vulnerable to APGD-CE and APGD-T, both of which reach an attack success rate of 100%.

The adversarially trained models substantially reduce the attack success rate, but this robustness comes with a clean-accuracy cost. By comparison, the tail-energy detector preserves clean accuracy close to that of the standard model and comparable to, or higher than, that of the adversarially trained model. For WRN28-10, both detector operating points maintain higher clean accuracy than the adversarially trained model. For WRN70-16, the detector gives essentially comparable clean accuracy: the 1% FPR setting is slightly higher than the adversarially trained model, while the 2% FPR setting is only slightly lower.

The main benefit of the detector appears for APGD-T. For WRN28-10, the 2% FPR detector reduces APGD-T ASR to 45.53%, which is slightly below the adversarially trained model value of 46.56%. For WRN70-16, the improvement is stronger: APGD-T ASR is reduced to 24.81% at 1% FPR and 13.99% at 2% FPR, compared with 57.30% for the adversarially trained model.

APGD-CE remains more difficult for the single-layer detector. Although the detector reduces APGD-CE ASR relative to the undefended standard models, it does not match the adversarially trained models for this attack. FAB-T and Square are also not effectively handled by this detector, as expected from the earlier tail-energy AUC results. Overall, the single-layer detector is most effective for APGD-T, while preserving clean accuracy close to the standard model and

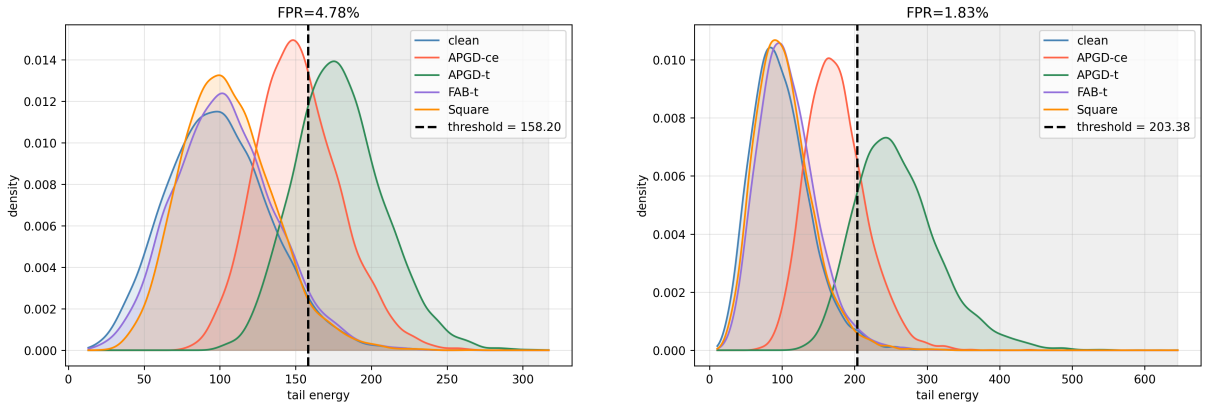
comparable to the adversarially trained model.

Accuracy-Matched Evaluation

As an additional comparison, the detector is evaluated under a different constraint. Instead of fixing the clean false-positive rate directly, the detector threshold is selected so that the final clean accuracy matches the clean accuracy of the corresponding adversarially trained model. This gives a more direct comparison between tail-energy detection and adversarial training under the same clean-accuracy budget.

For WRN28-10, the adversarially trained model achieves a clean accuracy of 72.57%. Therefore, the tail-energy threshold at Layer 7 is lowered until the standard model with rejection reaches the same clean accuracy. This corresponds to a clean FPR of 4.78%. For WRN70-16, the adversarially trained model achieves a clean accuracy of 75.22%. The corresponding accuracy-matched detector at Layer 18 gives a clean FPR of 1.83%.

Figure 23 shows the resulting accuracy-matched detector thresholds for both architectures. Compared with the stricter 1% and 2% FPR operating points, the threshold is moved further to the left, especially for WRN28-10. This makes the detector more aggressive: more clean samples are rejected, but a larger fraction of adversarial samples also falls inside the rejected region.



(a) WRN28-10 (b) WRN70-16

Figure 23: Tail-energy detector under the accuracy-matched setting for WRN28-10 and WRN70-16

Table 4 compares the accuracy-matched detector with the adversarially trained models. Since the clean accuracy is matched within each architecture, the comparison focuses directly on the remaining attack success rates.

For APGD-T, the detector performs substantially better than adversarial training. In WRN28-10, the APGD-T ASR is reduced to 24.56%, compared with 46.56% for the adversarially trained model. In WRN70-16, the improvement is even stronger, with APGD-T ASR reduced to 15.69%, compared with 57.30% for adversarial training. This confirms that the intermediate-layer tail-energy detector is particularly effective against APGD-T when both methods are compared at the same clean accuracy.

Table 4: Accuracy-matched comparison between adversarial training and the tail-energy detector. Values are in percentages.

Model	Mode	Standard Acc.	APGD-CE ASR	APGD-T ASR	FAB-T ASR	Square ASR
WRN28-10	Standard training	76.20	100.00	100.00	99.96	99.73
WRN28-10	Adversarial training	72.57	39.21	46.56	60.81	55.57
WRN28-10	Accuracy-matched detector, FPR = 4.78%	72.57	61.59	24.56	94.52	94.82
WRN70-16	Standard training	76.62	100.00	100.00	99.98	99.73
WRN70-16	Adversarial training	75.22	51.85	57.30	56.92	51.07
WRN70-16	Accuracy-matched detector, FPR = 1.83%	75.22	77.80	15.69	97.87	97.92

However, APGD-CE remains more difficult for the single-layer detector. Although the detector reduces APGD-CE ASR compared with the undefended standard model, it does not match the adversarially trained model. APGD-CE ASR remains 61.59% for WRN28-10 and 77.80% for WRN70-16. FAB-T and Square are also only weakly affected, which is consistent with the previous observation that these attacks do not produce a stable tail-energy separation in the selected intermediate layers.

Overall, the accuracy-matched setting highlights the main trade-off of the single-layer detector. By allowing a larger clean rejection rate, the detector becomes much stronger against APGD-T while preserving the same clean accuracy as adversarial training. Nevertheless, APGD-CE, FAB-T, and Square remain limiting cases. This shows that one intermediate tail-energy score is not sufficient to capture all adversarial objectives equally well.

Conclusion

This chapter developed and evaluated a tail-energy detector based on the representation-level findings from the previous corevector analysis. The detector uses the energy contained in the final 100 SVD components of a selected intermediate layer as an adversarial score. Samples with tail energy above a fixed threshold are rejected, while samples below the threshold are accepted. This creates a simple and interpretable detection rule based on the hypothesis that adversarial perturbations can produce abnormal energy in lower singular-value projected directions.

The single-layer detector preserves clean accuracy close to that of the standard model while reducing attack success for APGD-T. This result is strongest in WRN70-16, where the APGD-T ASR is reduced substantially at both fixed-FPR and accuracy-matched operating points. The detector is less effective for APGD-CE, FAB-T, and Square. This limitation is consistent with the previous chapter, where these attacks did not show the same stable intermediate-layer tail-energy separation as APGD-T.

Overall, the chapter shows the main trade-off of the single-layer tail-energy detector. By adjusting the threshold, the detector can preserve clean accuracy close to the standard model while reducing the success rate of selected attacks, especially APGD-T. However, APGD-CE, FAB-T, and Square remain limiting cases. This shows that one intermediate tail-energy score is not sufficient to capture all adversarial objectives equally well, and motivates the ensemble-based strategy developed in the following chapter.

Chapter 7: Ensemble Tail-Energy Detector

Introduction

The previous chapters developed the argument for a representation-based adversarial defense step by step. Chapter 2 established the importance of evaluating adversarial robustness under clearly defined attacks and reliable experimental conditions. Chapter 3 then introduced the mathematical framework used to describe internal representations through activations, SVD-projected corevectors, and tail energy. Chapter 4 showed that output-level indicators, such as softmax confidence, calibration, and prediction margins, are not sufficient for stable adversarial detection.

Chapter 5 therefore introduced corevector evaluation, where the internal activations of the network were analyzed layer by layer. The main finding was that adversarial samples can differ from clean samples in the lower singular-value directions of the SVD representation, especially in intermediate layers. This motivated the tail-energy statistic, which measures how much of a sample representation lies in the tail of the SVD basis. Chapter 6 then converted this statistic into a detector. The detector rejected samples whose tail-energy score exceeded a fixed threshold.

However, rejection alone creates an important limitation. When a sample is rejected, the system avoids making a potentially unsafe prediction, but it also fails to classify the input. This is useful for detection, but it is incomplete as a classification strategy. The present chapter addresses this limitation by extending the detector into an ensemble routing mechanism. Instead of rejecting suspicious samples, the system sends them to an adversarially trained model. In this way, the standard model is used when the input appears reliable, while the robust model is used only when the intermediate representation appears suspicious.

The aim of this chapter is therefore to present the final solution of this work: an ensemble detector that combines the clean accuracy advantage of the standard model with the robustness advantage of the adversarially trained model. The chapter first explains the problem of using layers in deep networks for detection. It then reviews the original detector and its limitations. Finally, it presents the ensemble detector and evaluates its performance under several operating points.

From Layer Analysis to the Final Detector

Deep neural networks do not represent inputs in the same way at every layer. Early layers usually capture simple visual features, such as edges, colors, and local texture patterns. These features are often similar for clean and adversarial samples because the adversarial perturbation is designed to remain visually small. For this reason, early layers tend to provide weak adversarial separation. Intermediate layers are more informative because they contain richer feature representations. At this stage, the model has moved beyond simple local patterns and begins to encode more

abstract class-related information. The previous corevector analysis showed that adversarial samples often become more visible in these intermediate representations, especially in the lower singular-value tail directions of the SVD basis. This explains why Layers 7 and 18 were selected for WRN28-10 and WRN70-16, respectively.

The final layer behaves differently. It is closer to the class-decision space and is therefore more compressed than the intermediate representation. This compression can make the final layer useful for some attacks, especially APGD-CE, but less reliable for others. As a result, no single layer provides equally strong separation for all attacks. This is one of the central problems of multi-layer deep learning analysis: different layers contain different types of information, and different attacks leave their strongest signatures at different depths.

The original detector in Chapter 6 used this layer-dependent behavior in a minimal way. It selected one strong intermediate layer and used tail energy as a rejection score. This approach was simple and interpretable. It showed that intermediate lower singular-value directions contain useful adversarial information, especially for APGD-T. Nevertheless, the detector had two limitations. First, rejected samples were not classified. Second, the detector was less effective for APGD-CE, FAB-T, and Square, because these attacks did not always produce the same intermediate tail-energy shift.

The final solution proposed in this chapter keeps the useful part of the original detector but changes its role. The detector no longer acts only as a rejection rule. Instead, it becomes a routing rule. If the input appears normal, the prediction of the standard model is used. If the input appears suspicious, the sample is routed to the adversarially trained model. This allows the system to preserve the clean performance of the standard model while using the robust model only when needed.

The Ensemble Detector

The previous detector rejected samples whose tail-energy score exceeded a fixed threshold. While this reduced the attack success rate, rejected samples were not classified. The ensemble detector extends this idea by using the detector as a routing mechanism rather than as a pure rejection rule.

The proposed ensemble first evaluates each input using the standard model and computes the tail-energy score at the selected intermediate layer. For WRN28-10, this corresponds to Layer 7. For WRN70-16, this corresponds to Layer 18. If the tail-energy score is below the threshold, the sample is considered reliable and the prediction of the standard model is kept. If the score is above the threshold, the sample is considered suspicious and is passed to the adversarially trained model, which acts as a robust fallback classifier.

Figure 24 illustrates the proposed ensemble routing mechanism. The tail-energy detector acts as a gate that keeps reliable samples with the standard model and sends suspicious samples to the robust model.

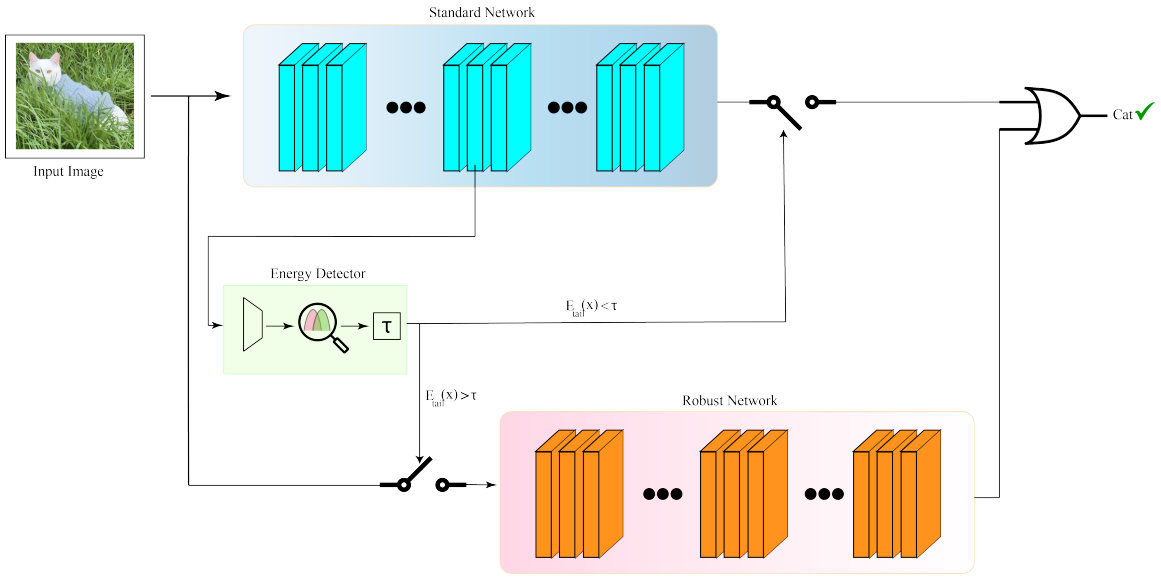


Figure 24: Overview of the ensemble tail-energy routing mechanism

Therefore, the final classifier is defined as

$$f_{\text{ens}}(x) = \begin{cases} f_{\text{std}}(x), & E_{\text{tail}}(x) \leq \tau, \\ f_{\text{rob}}(x), & E_{\text{tail}}(x) > \tau. \end{cases}$$

In this equation, $f_{\text{ens}}(x)$ denotes the final ensemble prediction, $f_{\text{std}}(x)$ denotes the prediction of the standard model, and $f_{\text{rob}}(x)$ denotes the prediction of the adversarially trained model. The term $E_{\text{tail}}(x)$ is the tail-energy score of the input, and τ is the threshold that determines whether the input should remain with the standard model or be routed to the robust model.

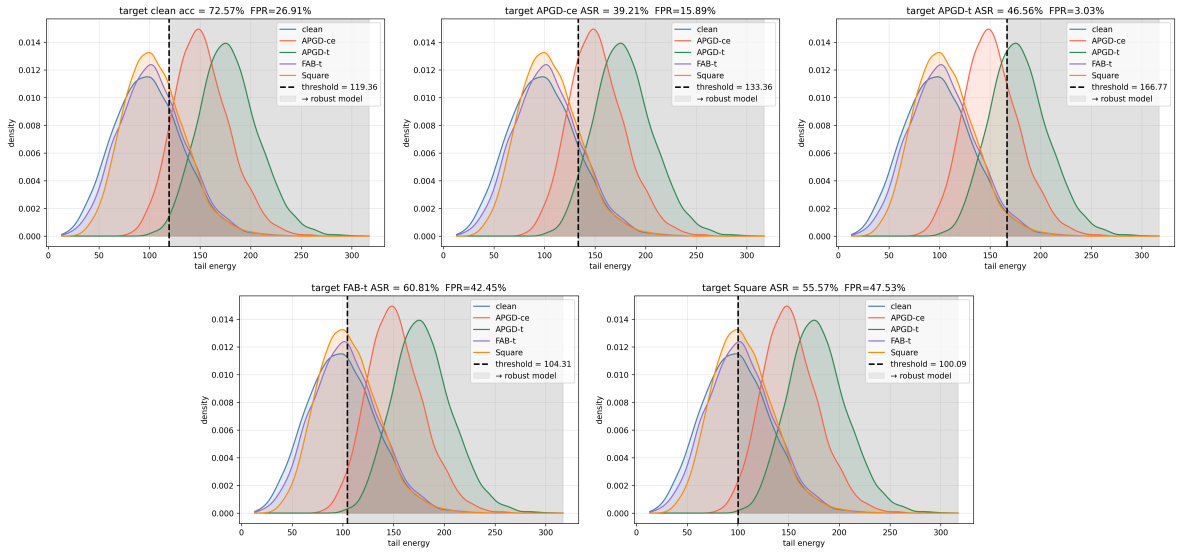
This construction preserves the advantage of the standard model on clean samples while using the robust model only for samples that appear atypical in the intermediate representation. In this sense, the tail-energy detector does not directly decide the class label. Instead, it decides which classifier should be trusted.

Operating Point Selection

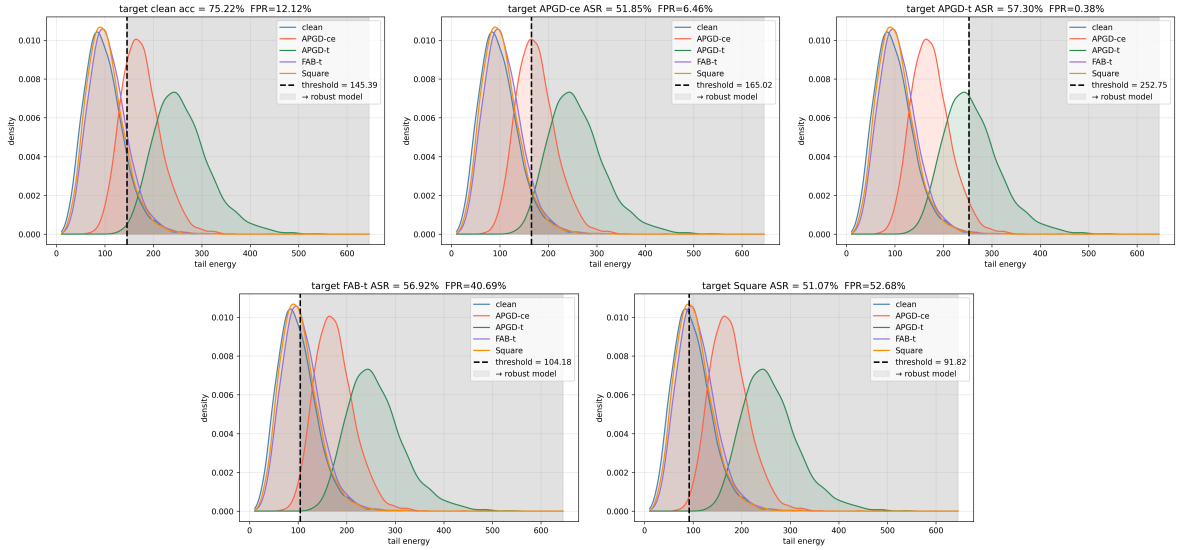
The threshold τ determines how often the robust model is used. A low threshold makes the ensemble more cautious because more samples are routed to the robust model. This can improve robustness, but it may reduce clean accuracy and increase dependence on the robust fallback. A high threshold makes the ensemble more similar to the standard model because fewer samples are routed to the robust model. This preserves clean accuracy, but it provides less protection against adversarial inputs.

To compare the ensemble with adversarial training, different operating points are selected by matching one target metric at a time, as shown in Figure 25.

Five operating points are considered. In the first case, the threshold is selected so that the ensemble reaches the same clean accuracy as the adversarially trained model. This setting asks



(a) WRN28-10



(b) WRN70-16

Figure 25: Operating points of the ensemble network for WRN28-10 and WRN70-16. For each model, the detector threshold is selected to match the clean accuracy or the robustness under a specific attack.

whether the ensemble can provide stronger robustness under the same clean-accuracy budget. In the second, third, fourth, fifth cases, the threshold is selected so that the ensemble matches the attack success rate of the adversarially trained model under APGD-CE, APGD-T, FAB-T, and Square Attack, respectively. These settings ask whether the ensemble can preserve higher clean accuracy while achieving the same attack-specific robustness as adversarial training. Considering the attacks separately also makes it possible to analyze how the trade-off between clean accuracy, adversarial robustness, and dependence on the robust fallback model changes across different attack objectives.

Table 5: Performance of the ensemble with tail-energy routing. Values are in percentages

Model	Operating point	Standard Acc.	APGD-CE ASR	APGD-T ASR	FAB-T ASR	Square ASR	Robust Contribution
WRN28-10	Standard model	76.20	100.00	100.00	99.96	99.73	0.00
WRN28-10	Robust model	72.57	39.21	46.56	60.81	55.57	100.00
WRN28-10	Ensemble, accuracy matched, FPR = 26.91%	72.57	26.86	17.86	74.83	75.28	45.40
WRN28-10	Ensemble, APGD-CE matched, FPR = 15.89%	74.12	39.21	21.10	84.77	85.62	36.42
WRN28-10	Ensemble, APGD-T matched, FPR = 3.03%	75.92	77.02	46.56	96.82	96.97	16.98
WRN28-10	Ensemble, FAB-T matched, FPR = 42.45%	70.59	20.29	17.15	60.81	60.41	55.23
WRN28-10	Ensemble, Square matched, FPR = 47.53%	69.94	19.50	17.08	56.10	55.57	58.16
WRN70-16	Standard model	76.62	100.00	100.00	99.98	99.73	0.00
WRN70-16	Robust model	75.22	51.85	57.30	56.92	51.07	100.00
WRN70-16	Ensemble, accuracy matched, FPR = 12.12%	75.22	35.68	15.18	86.13	88.41	36.93
WRN70-16	Ensemble, APGD-CE matched, FPR = 6.46%	75.85	51.85	16.54	92.94	94.01	29.97
WRN70-16	Ensemble, APGD-T matched, FPR = 0.38%	76.57	96.52	57.30	99.65	99.37	9.42
WRN70-16	Ensemble, FAB-T matched, FPR = 40.69%	72.23	16.97	14.79	56.92	62.15	56.81
WRN70-16	Ensemble, Square matched, FPR = 52.68%	70.92	15.82	14.79	45.67	51.07	63.57

Results and Discussion

Table 5 reports the resulting performance. The table compares the standard model, the adversarially trained model, and the ensemble under the five operating points described above. The final column, robust contribution, reports the percentage of samples routed to the adversarially trained model. This value measures how strongly the ensemble depends on the robust fallback classifier.

The results should be interpreted as a trade-off between clean accuracy, adversarial robustness, and dependence on the robust fallback model. Lowering the threshold makes the ensemble more defensive because more samples are routed to the adversarially trained model. This generally reduces attack success rates, but it also reduces clean accuracy and increases reliance on the robust model. This reliance is not always desirable. As shown in the previous output-level analysis, the adversarially trained models improve robustness, but they also have lower clean accuracy, weaker calibration, and less reliable confidence behavior under clean in-distribution conditions. Therefore, the goal of the ensemble is not to use the robust model as much as possible. Instead, the goal is to use it selectively, only when the intermediate representation suggests that the input is suspicious.

In the accuracy-matched setting, the ensemble reaches the same clean accuracy as the adversarially trained model, but achieves lower APGD-CE and APGD-T ASR. For WRN28-10, APGD-CE ASR decreases from 39.21% to 26.86%, and APGD-T ASR decreases from 46.56% to 17.86%. For WRN70-16, APGD-CE ASR decreases from 51.85% to 35.68%, and APGD-T ASR decreases from 57.30% to 15.18%. Thus, under the same clean-accuracy budget, the ensemble provides stronger robustness against the APGD-based attacks. However, FAB-T and Square remain less controlled in this operating point, as expected from the weaker tail-energy separation observed for these attacks.

When the ensemble is matched to the APGD-CE ASR of the adversarially trained model, it preserves higher clean accuracy than the robust model. This is visible for both architectures. For WRN28-10, clean accuracy improves from 72.57% to 74.12%. For WRN70-16, it improves from 75.22% to 75.85%. At the same time, APGD-T ASR is substantially reduced compared with the adversarially trained model. This means that the ensemble can match the robust model on APGD-CE while improving both clean accuracy and APGD-T robustness. Nevertheless, FAB-T and Square ASR remain high in this setting, which again confirms that matching APGD-CE does not automatically provide robustness against all attack mechanisms.

When the ensemble is matched to the APGD-T ASR of the adversarially trained model, the clean accuracy becomes very close to that of the standard model, but APGD-CE, FAB-T, and Square remain high. This operating point uses the robust model only rarely, especially for WRN70-16, where dependence on the robust model is only 9.42%. This shows the expected trade-off. A weaker detector threshold preserves clean accuracy and reduces reliance on the robust fallback, but it provides less protection against attacks whose samples are not strongly separated by the intermediate tail-energy score.

The FAB-T-matched and Square-matched operating points show a different behavior. In these cases, the ensemble can match the corresponding attack success rate of the adversarially trained model, but only by using a much lower threshold. As a result, a larger fraction of samples is routed to the robust model, and clean accuracy drops below that of the adversarially trained model. For WRN28-10, matching FAB-T and Square requires robust contributions of 55.23% and 58.16%, respectively. For WRN70-16, the corresponding robust contributions increase to 56.81% and 63.57%. This indicates that FAB-T and Square can be controlled by the ensemble, but only at the cost of greater dependence on the robust fallback classifier and a larger clean-accuracy loss.

These results show that the ensemble provides a more flexible trade-off than using the adversarially trained model alone. The standard model handles most clean samples when the threshold is moderate, while the robust model is used selectively on suspicious inputs. The strongest advantage appears for APGD-CE and APGD-T, where the ensemble can improve robustness or preserve higher clean accuracy with only partial use of the robust model. FAB-T and Square are more difficult: matching their robust-model ASR is possible, but it requires more aggressive routing and therefore weakens the clean-accuracy advantage of the ensemble.

Interpretation of the Final Solution

The ensemble detector addresses the main problems identified in the previous chapters. First, it avoids relying only on output-level confidence, which was shown to be unstable across attacks and model types. Second, it uses internal layer information, where adversarial perturbations are more visible than in the final softmax output. Third, it avoids the main weakness of the original rejection detector by assigning suspicious samples to a robust classifier instead of leaving them unclassified.

The method also clarifies the practical role of the intermediate layer. The intermediate tail-energy score is not treated as a complete classifier. It is used as a reliability signal. When the score is low, the standard model is trusted because the sample appears consistent with normal internal representations. When the score is high, the robust model is trusted because the sample appears

atypical and may be adversarial.

This design is especially useful because standard and adversarially trained models have different strengths. The standard model generally provides stronger clean accuracy, while the adversarially trained model provides stronger robustness. The ensemble combines these strengths through conditional routing. Rather than choosing one model for all inputs, it selects the model according to the representation behavior of each sample.

At the same time, the ensemble does not solve every limitation. FAB-T and Square remain difficult because they do not consistently produce the same tail-energy shift as APGD-CE and APGD-T. This means that the final detector is strongest for the attacks that create detectable intermediate representation changes. Therefore, the ensemble should be interpreted as a targeted improvement over the original detector, not as a universal defense against all adversarial attacks.

Conclusion

This chapter presented the final detector proposed in this work: an ensemble tail-energy detector that uses intermediate representation energy as a routing mechanism. The detector first computes the tail-energy score of the input at a selected intermediate layer. If the score is below the threshold, the standard model is used. If the score is above the threshold, the adversarially trained model is used as a robust fallback classifier.

This approach draws a clear line through the previous chapters. The earlier output-level analysis showed that softmax confidence alone is not reliable enough for adversarial detection. The corevector analysis then showed that intermediate lower singular-value directions contain stronger adversarial signals. The original tail-energy detector converted this signal into a rejection rule, but rejected samples remained unclassified. The ensemble detector solves this final problem by replacing rejection with routing.

The results show that the ensemble provides a more flexible balance between clean accuracy and robustness than adversarial training alone. Under the accuracy-matched setting, it achieves the same clean accuracy as the robust model while reducing APGD-CE and APGD-T attack success rates. Under the APGD-CE-matched setting, it preserves higher clean accuracy while also improving APGD-T robustness. Under the APGD-T-matched setting, it recovers clean accuracy close to the standard model, although APGD-CE, FAB-T, and Square remain less controlled. When the ensemble is instead matched to FAB-T or Square, it can reach the robust model's attack success rate for those attacks, but only by routing many more samples to the robust model and sacrificing clean accuracy. This confirms that FAB-T and Square are harder cases for the proposed tail-energy routing mechanism.

Overall, the ensemble tail-energy detector is the most complete form of the proposed method. It keeps the interpretability of the intermediate tail-energy signal, avoids the classification gap created by rejection, and combines the complementary strengths of standard and adversarially trained models. Its main limitation is that it remains attack-dependent, with weaker performance on FAB-T and Square. Nevertheless, it provides a clear and practical final solution for using internal representation analysis to improve adversarial robustness.

Conclusion

Summary

Deep neural networks have achieved remarkable success in modern artificial intelligence, especially in tasks such as image classification and pattern recognition. Their strength comes from their ability to transform raw input data into increasingly abstract internal representations. However, this thesis has shown that the same representational power also creates a significant challenge. A model may classify clean images accurately while remaining vulnerable to small adversarial perturbations that are almost invisible to human observers. This contrast between external similarity and internal instability is one of the central problems in adversarial robustness.

The main aim of this thesis was to investigate whether the intermediate activations of deep neural networks can be statistically characterized and used for adversarial attack detection. Instead of analyzing only the final prediction or confidence score of a classifier, the thesis focused on what happens inside the model before the final decision is produced. This internal view is important because adversarial examples may not always be clearly distinguishable from clean samples at the input level or at the output level. Their abnormal behavior may instead appear in the hidden representation spaces formed by the network across its layers.

The thesis first reviewed the historical development of neural networks, from early logical models and biological inspiration to modern deep learning architectures. This background showed that neural networks have always been shaped by the attempt to model intelligence as a process of representation, learning, and adaptation. The discussion then moved to the fragility of deep neural networks and the discovery of adversarial examples. These examples demonstrated that high accuracy on clean data does not necessarily imply robustness, reliability, or genuine stability under small input changes.

Main Contribution

The main contribution of this thesis is the development and evaluation of an adversarial detection method based on the statistical characterization of intermediate activations. Layer activations were represented as feature maps and then flattened into activation vectors. For each selected layer, a layer-wise SVD projection basis was obtained from the corresponding layer transformation. For convolutional layers, this transformation was represented through an equivalent operator form, while linear layers were handled through their weight transformation. The resulting SVD bases were used to project sample activations into corevectors, which were then analyzed through their energy in the lower singular-value tail directions.

The central statistic introduced in this work was tail energy. Tail energy measures how much of a sample representation lies in the lower singular-value directions of the SVD-projected corevector. The motivation behind this statistic is that clean samples define the reference behavior of the tail-energy score, while adversarial samples may move through less typical internal directions.

In this sense, tail energy provides a representation-level measure of abnormality. It does not depend directly on the final predicted class or on the model's confidence score. Instead, it measures whether the internal representation of a sample behaves unusually with respect to the clean reference distribution.

The proposed detector used this information to classify samples as accepted or suspicious. In its simplest form, the detector computes the tail-energy score of a sample at a selected layer and compares it with a fixed threshold. If the score exceeds the threshold, the sample is rejected as suspicious. This rule is simple, interpretable, and directly connected to the geometry of the internal representation space. The detector was evaluated in single-layer, accuracy-matched, and ensemble settings. These evaluations showed that combining information from selected layers can improve the detection process and provide a more stable representation-level signal.

Interpretation of the Results

The experimental analysis supported the value of studying adversarial behavior through internal representations. The output-level analysis showed that confidence, calibration, and risk-coverage behavior provide useful information, but they do not fully explain adversarial behavior. The corevector evaluation then showed that selected intermediate layers contain stronger signals for separating clean and adversarial samples. This finding supports the main argument of the thesis: the hidden layers of a neural network should not be treated as an inaccessible black box, because they contain structured information that can help reveal adversarial behavior.

The results show that the proposed detector is particularly effective against APGD-CE and APGD-T. These attacks are gradient-based components of AutoAttack and generate adversarial examples by iteratively optimizing a loss objective. For these attacks, adversarial samples tend to produce changes that are visible in the selected tail-energy representation. This suggests that loss-optimization-based attacks can produce detectable changes in the lower singular-value components of intermediate activations in a way that can be detected by the proposed method.

Limitations

At the same time, the thesis also identified important limitations. When the detector was evaluated against the remaining AutoAttack components, FAB-T and Square Attack, a different behavior emerged. As illustrated in Figure 19 and Figure 20, the tail-energy distributions of adversarial samples generated by FAB-T and Square Attack largely overlap with those of clean samples. This overlap makes detection significantly more difficult. The limitation is important because it shows that the proposed detector does not respond equally to all adversarial attack mechanisms.

This difference can be explained by the different strategies used by these attacks. APGD-CE and APGD-T rely on gradient-based loss optimization. In contrast, FAB-T searches for perturbations that move the input across the decision boundary while keeping the perturbation magnitude small, rather than directly maximizing a classification loss in the same way. Square Attack differs even further because it is a black-box attack that does not rely on gradient information. Instead, it repeatedly perturbs randomly selected image regions and observes the model response in order to find successful adversarial perturbations. These different mechanisms can produce adversarial examples that do not create the same tail-energy signature as APGD-based attacks.

This result leads to a broader conclusion. A detector may be effective against one family of attacks, but this does not guarantee that it will be equally effective against attacks based on different optimization principles. Designing a single adversarial detector that is robust against all possible attack methodologies remains highly challenging. The diversity of adversarial attacks means that detection methods must be evaluated not only by their average performance, but also by their behavior against different attack mechanisms.

Another limitation concerns the layer-wise behavior of the proposed energy metric. The experimental results showed that certain intermediate layers provide stronger separation between clean and adversarial samples, while earlier and later layers are less informative. The reason for this phenomenon was not investigated in depth within the scope of this thesis. A possible explanation is related to the hierarchical structure of deep neural networks. Early layers generally capture low-level features such as edges, textures, colors, and simple local patterns, which may be only weakly affected by adversarial perturbations. Final layers, by contrast, have already compressed the representation toward the classification decision, which may reduce the usefulness of the tail-energy metric. Intermediate layers may represent a more informative stage, where adversarial perturbations have already influenced the learned representation but the representation has not yet fully collapsed into the final class prediction.

However, this interpretation remains speculative. The present thesis provides empirical evidence that intermediate layers can be useful for detection, but it does not provide a complete theoretical explanation of why some layers are more informative than others.

Future Work

Future work should investigate the layer-wise behavior of adversarial perturbations more systematically. This could include analyzing representation geometry across a larger set of architectures, datasets, threat models, and attack families. Such work may help explain why certain layers produce stronger detection signals and whether this behavior is consistent across different neural network designs.

Future work should also extend the proposed detector beyond the attacks studied here. Since FAB-T and Square Attack produce tail-energy distributions that overlap strongly with clean samples, additional representation-level statistics may be needed. Tail energy captures one type of abnormal movement in the lower singular-value SVD-projected directions, but adversarial examples generated by different mechanisms may affect other parts of the representation space.

A further direction is the development of adaptive evaluation against the detector itself. If an attacker knows that the detector is based on tail energy, the attack could be modified to reduce the tail-energy score while still causing misclassification. Testing the detector under such adaptive conditions would provide a stronger estimate of its robustness. This is especially important because adversarial defense research has repeatedly shown that methods can appear effective under non-adaptive evaluation but fail when the attacker directly accounts for the defense.

Final Remarks

Despite its limitations, this thesis contributes to the study of adversarial robustness by showing that intermediate activations contain useful statistical information for detection. The proposed

method is not a complete solution to adversarial robustness, but it demonstrates that the internal geometry of neural representations can provide meaningful signals beyond final output confidence. This is valuable because it shifts the analysis from asking only what the model predicts to asking how the model internally represents the input before making that prediction.

Overall, this thesis supports the view that adversarial robustness should be studied inside the network, not only at its output. Deep neural networks are not only decision functions; they are layered representation systems. Their intermediate activations contain traces of how an input is transformed, compressed, and prepared for classification. By statistically characterizing these internal representations, this work provides a step toward more interpretable and reliable adversarial detection. The broader goal remains open: to build deep learning systems that are accurate under normal conditions, stable under perturbation, transparent in their internal behavior, and trustworthy in real-world environments.

Bibliography

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[2] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York, NY: John Wiley and Sons, 1949.

[3] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[4] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[5] W. W. Bledsoe and I. Browning, "Pattern recognition and reading by machine," in *Proceedings of the Eastern Joint Computer Conference*, pp. 225–232, 1959.

[6] O. G. Selfridge, "Pandemonium: A paradigm for learning," in *Proceedings of the Symposium on Mechanization of Thought Processes*, vol. 2, (London, UK), pp. 513–526, Her Majesty's Stationery Office, 1959.

[7] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[8] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[10] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

[13] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2014.

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, arXiv preprint arXiv:1412.6572, 2015.

[17] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Mądry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[18] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[19] F. Tramèr, "Detecting adversarial examples is (nearly) as hard as classifying them," in *Proceedings of the 39th International Conference on Machine Learning (ICML)*, PMLR, 2022.

[20] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," in *International Conference on Learning Representations (ICLR) Workshop*, 2015. arXiv:1412.5068.

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 10, pp. 1977–1984, 2017. Originally published as arXiv:1511.04599.

[22] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 582–597, IEEE, 2016.

[24] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 39–57, IEEE, 2017.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2018.

[26] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 274–283, PMLR, 2018.

[27] F. Tramèr, N. Carlini, W. Brendel, and A. Mądry, "On adaptive attacks to adversarial example defenses," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[28] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 2206–2216, PMLR, 2020.

[29] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: A standardized adversarial robustness benchmark," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[31] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, 2016.

[32] G. Cazenavette, C. Murdock, and S. Lucey, "Architectural adversarial robustness: The case for deep pursuit," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7150–7158, 2021.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, 2014.

[35] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 4 ed., 2013.

[36] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, p. 20150202, 2016.

[37] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan, "Better diffusion models further improve adversarial training," in *Proceedings of the 40th International Conference on Machine Learning*, vol. 202 of *Proceedings of Machine Learning Research*, pp. 36246–36263, PMLR, 2023.

[38] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.