Aalto-yliopisto
Aalto-universitetet
Aalto University

Bachelor's Programme in Electrical Engineering

Guarantees in deep neural networks predictions: probabilistic bounds and robustness methods

Jaakko Niemi

Bachelor's Thesis
2025

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Jaakko Niemi

Title of thesis Guarantees in deep neural networks predictions: probabilistic bounds and robustness methods

Programme Bachelor's Programme in Electrical Engineering

Major Automation and Robotics

Thesis supervisor Doctor of Science (Technology) Pekka Forsman

Thesis advisor(s) Doctoral Researcher Abdullah Tokmak

Date 26 August 2025 **Number of pages** 33 **Language** English

Abstract

Deep neural networks (DNNs) are increasingly deployed in several applications and often achieve impressive performance. However, their behavior can become unpredictable when deployed in real-world, safety-critical environments. Real-world reliability requirements highlight the need for methods aiming to certify DNNs' outputs.

This thesis is a literature review. The thesis introduces classical, sampling-based probabilistic methods. These approaches offer NN model-agnostic techniques to quantify the probability of prediction failure. While broadly applicable, their utility is often limited by the trade-off between the tightness of the guarantee and the required sample size. Next, the thesis reviews guarantee robustness certification methods, which provide more deterministic prediction guarantees. The thesis also investigates both soft- and hard-constrained architectures, followed by a high-level comparative overview.

The thesis concludes no single method is solely adequate to fulfil prediction guarantee robustness requirements for safety-critical applications. In addition, the selection of an appropriate methods requires a careful balance of computational cost, architectural assumptions, and the desired level of guarantee tightness.

Keywords machine learning, deep learning, neural network, uncertainty, probabilistic guarantees, deterministic guarantees, guarantee robustness

Aalto-yliopisto
Aalto-universitetet
Aalto University

Tekijä Jaakko Niemi

Työn nimi Syvien neuroverkkojen ennusteiden luotettavuus: todennäköisyyspohjaiset keinot ja varmennusmenetelmät

Koulutusohjelma Sähkötekniikan kandidaattiohjelma

Pääaine Automaatio ja robotiikka

Vastuuopettaja/valvoja Tekniikan tohtori Pekka Forsman

Työn ohjaaja(t) Tohtorikoulutettava Abdullah Tokmak

Päivämäärä 26.8.2025 **Sivumäärä** 33 **Kieli** englanti

Tiivistelmä

Syväoppiminen (engl. *deep learning*) on koneoppimisen osa-alue, jossa hyödynnetään useista kerroksista koostuvia neuroverkkoja hierarkkiseen oppimiseen ja ennusteiden tekemiseen. Neuroverkkoja, joissa on useampi kerros, kutsutaan syviksi neuroverkoiksi. Syviä neuroverkkoja (engl. *deep neural networks*, DNNs) hyödynnetään nykyään laaja-alaisesti eri alojen tehtävissä ja sovelluksissa. Syvät neuroverkot ovat erittäin suorituskykyisiä. Ne hoitavatkin muun muassa kuvan, tekstin ja äänen tunnistamistehtävät tehokkaasti. Syvien neuroverkkojen ennusteiden luotettavuus voi kuitenkin kärsiä reaalimaailman käytännön tehtävissä ja sovelluksissa. Turvallisuuskriittisissä sovelluksissa ennusteiden luotettavuudella on suuri merkitys. Reaalimaailman käytännön sovelluksissa on tärkeää pystyä varmentamaan ennusteiden luotettavuus ja näin ollen antamaan takeet syvien neuroverkkojen tuloksista.

Tämä kandidaatintyö on kirjallisuustutkimus menetelmistä, jotka tarjoavat joko takeet neuroverkkojen ennusteiden luotettavuudesta tai parantavat niiden kestävyyttä häiriöisessä ympäristössä. Työssä tarkastellaan klassisia todennäköisyyspohjaisia keinoja luotettavuuden tarkasteluun. Näitä keinoja ovat Hoeffdingin, Chebyshevin ja Chernoffin epäyhtälöihin perustuvat rajat (engl. *bounds*), joilla kvantifioidaan ennustevirheen todennäköisyyttä. Tällaiset keinot ovat neuroverkkoagnostisia eli eivät ole riippuvaisia neuroverkon mallista tai syvyydestä. Huolimatta siitä, että todennäköisyyspohjaiset keinot ovat laajasti sovellettavissa, niiden hyödyllisyyttä rajoittaa usein takeiden tarkkuus ja vaadittu suuri otoskoko. Työssä käsitellään myös ennustetakeiden kestävyyden varmentamismenetelmiä häiriöisessä ympäristössä. Tällaiset menetelmät tarjoavat deterministisiä ennustetakeita. Työssä esitellään CROWN-menetelmä, jolla on kyky tuottaa ennusteiden luotettavuustakeet tiettyjä aktivaatiofunktioita hyödyntäville neuroverkoille. PROVEN on puolestaan varmennusmenetelmä, joka osoittaa, kuinka deterministiset ennusteiden luotettavuusrajat voidaan muuntaa tiukemmiksi todennäköisyystakeiksi häiriöisessä ympäristössä. Työssä tarkastellaan myös Lipschitzin ehtoon perustuvia rajoja, joilla hallitaan neuroverkon ennusteiden herkkyyttä syötteen häiriöille. Lisäksi työssä tutkitaan menetelmiä, joilla pyritään rajoittamaan joko neuroverkkojen syötettä (engl. *input*), tuotosta (engl. *output*) tai molempia. Tällaiset menetelmät yleisesti jaetaan pehmeisiin ja koviin rajoitteisiin. Pehmeään rajoitteeseen perustuvaa menetelmää tässä työssä edustaa PINN-neuroverkot (engl. *Physics-Informed Neural Networks*, PINNs), jotka pakottavat osittaisdifferentiaaliyhtälöitä oppimisprosessin tappiofunktioon (engl. *loss function*). Kovaa rajoitetta hyödyntävää arkkitehtuuria edustaa HardNet, joka pakottaa tiukat ehdot neuroverkon syötteeseen. HardNet kuitenkin säilyttää neuroverkon toimintatehon.

Kandidaatintyössä esitellään ylätason vertaileva analyysi menetelmien oletuksista, soveltuvista neuroverkkoarkkitehtuureista, skaalautuvuudesta ja takeiden tarkkuudesta. Vertailevan analyysin lopputuloksena todetaan, ettei yksikään menetelmä pelkästään riitä täyttämään turvallisuuskriittisten käyttösovellusten ennustetakeiden kestävyyttä. Sopivien menetelmien valinta vaatii huolellista tasapainottelua laskennallisen tehon, neuroverkko-oletusten ja halutun takeen tarkkuuden välillä. Jatkotutkimussuunnaksi kandidaatintyö ehdottaa eri menetelmien yhdistämistä, esimerkiksi liittämällä deterministiset takeet todennäköisyyspohjaisiin keinoihin.

Avainsanat koneoppiminen, syväoppiminen, neuroverkko, epävarmuus, todennäköisyyspohjaiset varmennuskeinot, deterministiset varmennuskeinot, varmennusmenetelmät

Table of Contents

Operators and abbreviations

Operators

$\frac{\partial}{\partial t}$	Partial derivative operator with respect to the variable t
$\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n})$	The gradient of a scalar function f of several with vector of all its higher-order partial derivatives
$[A]^T$	The transpose operation of matrix A

Abbreviations

AI	Artificial intelligence
ANN	Artificial neural network
AOL	Almost orthogonal Lipschitz
BNN	Bayesian neural network
CD	Confidence dimension
CNN	Convolutional neural network
CPU	Central processing unit
DNN	Deep neural network
LLP	Lipschitz locomotion policy
ML	Machine learning
MLP	Multilayer perceptron
MNIST	Modified National Institute of Standards and Technology
MPC	Model predictive control
MSE	Mean squared error
NN	Neural network
PDE	Partial differential equation
PINN	Physics-informed neural network
PPO	Proximal policy optimization
ReLU	Rectified linear unit
RKHS	Reproducing kernel Hilbert space
SGD	Stochastic gradient descent
SDP	Semidefinite programming
SN	Spectral normalization
TTA	Test-time data augmentation
VC	Vapnik-Chervonenkis

1 Introduction

Over the past decade, deep learning has transformed several fields, including image and speech recognition, as well as reinforcement learning. Deep learning, in the context of this thesis, is considered a subfield of machine learning (ML) that utilizes artificial neural networks (ANNs) with multiple layers, i.e., deep neural networks (DNNs) to learn hierarchical representations of data automatically.

A neural network (NN) consists of interconnected layers of artificial neurons and, in general, is a model for processing information that imitates the human brain (Bishop & Bishop 2023, 16 – 17). Like a human brain, an NN consists of neurons that are connected. In its essence, a neuron combines the signals it receives from other neurons into a single scalar value using an activation function and then transmits the scalar to the neurons in the next layer. When there are enough neurons in the NN, the NN can learn to recognize, for example, images, objects, or speech. A traditional neural network produces a single model tailored to the data. In such a model, a common problem is that the model often makes predictions with excessive confidence. As NNs have become widely utilized in science and various industries and found their way into high-assurance domains, such as autonomous vehicles, medical devices, and aerospace systems, robustness verification methods are increasingly essential (Taylor 2006, 5).

Although DNNs often achieve impressive performance in controlled simulations, their behavior can become unpredictable when deployed in real-world, safety-critical environments. The gap between simulated success and real-world reliability highlights the need for methods that aim to certify DNNs' outputs. DNNs are black-box models without extensive guarantees and robustness (Pauli 2025, 11). An example of a model lacking robustness and safety guarantees is generative artificial intelligence (AI) models such as ChatGPT, which tend to hallucinate and produce incorrect results.

This thesis first introduces methods that either provide guarantees on predictions or enhance their robustness, followed by a high-level comparison of guarantee and robustness methods. Additionally, the thesis includes a review of existing conference proceedings and research publications. It begins with an introduction to deep learning, neural networks, and the guarantees and robustness of predictions. Then, the thesis examines sampling-based approaches that apply classical probabilistic bounds to certify network guarantees, including Hoeffding, Chebyshev, and Chernoff bounds. The fourth chapter introduces strategies for improving robustness, such as Lipschitz bounds, constrained architectures, a framework for certifying the robustness of neural networks (CROWN), and a novel framework called Probabilistically Verify Neural Networks (PROVEN). The fifth chapter offers a high-level comparative overview of guarantee and robustness methods. The final chapter concludes the thesis and summarizes its main findings.

2 Background

This chapter begins by tracing the origins of DNNs and the definition of how DNNs compute an output layer and result from an input layer. By outlining key concepts such as feedforward NN, backpropagation, and probabilistic bounds and predictions robustness, this chapter establishes the principles that inform the thesis's subsequent exploration of methods for DNNs guarantees.

2.1 Deep neural networks

Dreams of intelligent and thinking machines have existed long before the first computer was invented. Today's artificial intelligence (AI) has made that dream a reality with automation of routine tasks, interpreting speech and images, aiding medical diagnoses, and advancing scientific discovery. Early AI succeeded in solving problems that were expressible through formal rules. However, it struggled with tasks human beings perform intuitively, such as recognizing faces or understanding natural language, because these processes are hard to codify. Deep learning addresses these challenges by enabling machines to learn from experience (Goodfellow et al. 2016, 1).

Defining deep neural networks starts with understanding the relationship between artificial intelligence, machine learning, and deep learning. Deep learning utilizes neural network computation models and is a subset of machine learning that has become an essential field of technology and is growing at a fast pace (Bishop & Bishop 2023, 1). Deep neural networks are often denoted as deep learning (Mermillod 2020, 80).

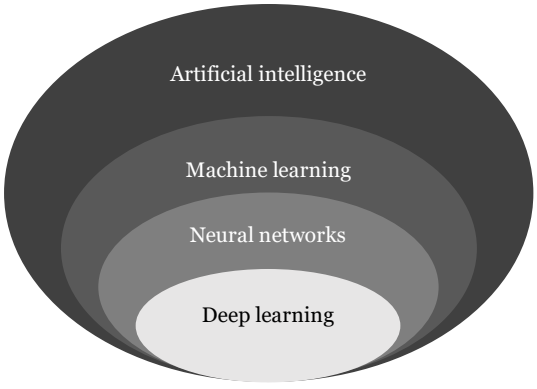

Figure 1. Simplified and illustrative Venn diagram of deep learning in relation to artificial intelligence, machine learning and neural networks (Mermillod 2020, 80).

Deep learning paradigms are distinguished by the role of labels in learning: supervised learning applies previous experience to compute expertise and label the test data under predefine labels, whereas unsupervised learning does not distinguish training and test data instead aiming to summarize or cluster a compressed version or a subset of the test data (Shalev-Shwartz & Ben-David 2014, 22-23). Reinforcement learning scouts its surroundings to define the target label and output with the guidance of provided feedback on output's success or failure (Du & Swamy 2014, 547). Reinforcement learning has similarities to supervised learning. Besides these core paradigms, other learning methods exist, however, they are less utilized.

DNNs typically follow feedforward architecture, where the data flows in one direction and does not loop (Shalev-Shwartz & Ben-David 2014, 268). Some sources denote feedforward deep neural networks as multilayer perceptrons (MLP). Deep neural networks are a representation-learning method that is composed of multiple layers of non-linear transformations between the input and output layer, which layer by layer extract the raw data input into a more abstract level (LeCun et al. 2015, 436). More closely, raw signals at the input layer are fed into hidden units where each neuron computes a weighted sum of its inputs and applies a non-linear function computed with the bias terms to produce an output to be passed forward (LeCun et al. 2015, 437-438).

Bishop & Bishop (2023, 235) define the first step in forward propagation of a feedforward neural network as computing a weighted sum of the input a_j to neuron j:

$$a_j = \sum_i w_{ji} z_i + w_{0j} \tag{2.1}$$

where z_i is the output of the neuron i, and w_{ji} is the weight in that connection. A bias function w_{0j} can be opted for. Next, the input a_j is transformed by the non-linear function:

$$z_j = h(a_j). \tag{2.2}$$

Common non-linear transformation functions in DNNs by LeCun at al. (2015, 437-437) are:

Rectified linear unit (ReLU): $h(z) = \max(0, z)$ (2.3)

Hyperbolic tangent: $h(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ (2.4)

Sigmoid function: $h(z) = \frac{1}{1+e^{-z}}$. (2.5)

Figure 2 shows an example of a feedforward NN with a bias function and a Sigmoid function to illustrate the operations of an NN.

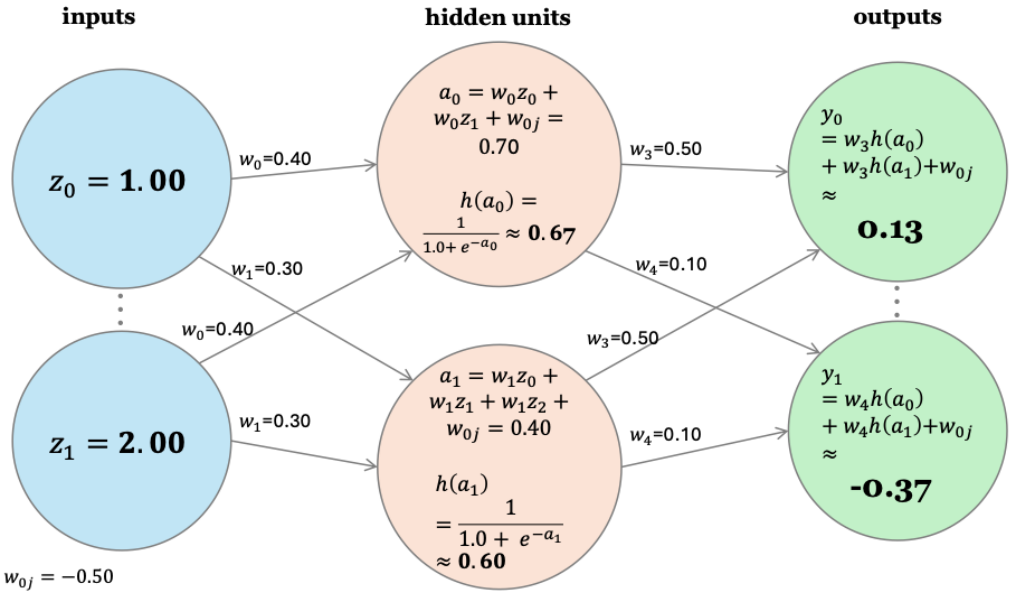

Figure 2. Example of feedforward NN with bias function and Sigmoid function as transformation function.

Following the transformation, the backpropagation process provides the learning of a neural network (Bishop & Bishop 2023, 235 – 236). Backpropagation computes the gradients, i.e., how each weight in a neural network affects the error by evaluating the partial derivatives of the error E_n of a certain input n concerning a weight w_{ji} as

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} \times \frac{\partial a_j}{\partial w_{ji}}. \tag{2.6}$$

Based on earlier, $\delta_k = y_k - t_k$ can be computed for the output layer. Bishop & Bishop (2023, 236) then introduce a notation that becomes useful when moving forward in the backpropagation process over k units, where δ denotes as loss function or error:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \times \frac{\partial a_k}{\partial a_j}. \tag{2.7}$$

Utilizing equations 2.7, 2.1, and 2.2, the backpropagation formula can be obtained:

$$\delta_j = h'(a_j) \sum_i w_{kj} \delta_k, \tag{2.8}$$

which denotes that the value of a certain error δ can be found by sending the δ's backwards from the units higher in the network. The error is typically defined utilizing mean squared error (MSE) by the following formula (Du & Swamy 2014, 16).

Feedforward DNN learns by adjusting the weights of the neurons, aiming to minimize the error between the predicted output and the actual output, which is typically performed using gradient descent. Stochastic gradient descent (SGD) is the most prevalent model in DDNs due to its sequential nature, yet other models exist (Bishop & Bishop 2023, 214 – 215). In SGD, each data point's weight vector is updated one by one at a time based on the following formula through the whole training data:

$$w^{\tau} = w^{\tau-1} - \eta \nabla E_n(w^{\tau-1}) \tag{2.9}$$

where η is the learning rate parameter, w^{τ} the final weight vector, $w^{\tau-1}$ and the previous weight vector. E_n is the error.

The training of feedforward DNNs involves calculating a loss function or error by optimizing the weights with gradient descent methods such as stochastic gradient descent (SGD) and then testing the accuracy with the test data. However, training of DNNs provides a sense of the quality of NN prediction, yet it does not provide guarantees (Wang et al. 2022, 1). In safety-critical domains, predictions made by DNNs must be highly reliable. The next subsection of the chapter presents theory around the uncertainty of deep neural network predictions, and after that, the thesis presents theory around guarantees and robustness.

2.2 Uncertainty of predictions

Uncertainty of predictions is present in nearly all machine learning applications, including applications with infinitely large data sets (Bishop & Bishop 2023, 23). Gawlikowski et al. (2023, 1516 – 1517) identify several potential causes of uncertainty. Such causes are not limited only to the real-world situation variability, but also errors inherited in the measurement systems or in the DNN architecture, or errors DNN training process, or by any unknown data. In addition, DNNs are exposed to adversarial examples whose input has been classified correctly, yet misclassified due to a perturbation (Kwiatkowska 2019, 1). Acknowledging the noise and adversarial perturbation in DNN prediction guarantees matters especially in safety-critical applications. Even an obscure input tweak can change a network's prediction. For example, changing only a few pixels in a vision model misreads a traffic light from red to green. Such a slip could lead an autonomous vehicle into a crash with a crossing vehicle.

Gawlikowski et al. (2023, 1524 – 1527) define four categories for quantification methods to estimate the uncertainty based on the number, single or multiple, or the nature, stochastic or deterministic, of the utilized DNNs. Single deterministic networks operate with a single forward pass and either embed uncertainty inside the network or attach it afterwards and possess low computational effort and memory consumption during training. Bayesian

methods and Bayesian Neural Networks (BNNs) explicitly treat the weights as random variables for a single forward pass, and therefore, the model is stochastic. According to the study, BNNs perform with high computational effort and low memory consumption. Ensemble methods gain uncertainty quantification by averaging the predictions of several trained deterministic networks, and the study shows them to possess high computation performance and high memory consumption. The last method the study presents is test-time data augmentation (TTA). TTA mimics an ensemble by perturbing each input and feeding the variants through one fixed network. TTA's variance across the augmented predictions quantifies the uncertainty, and TTAs possess low computation performance combined with low memory consumption. The next subsection of the chapter presents guarantees of predictions and examines the theory of prediction guarantee techniques.

2.3 Probabilistic bounds and robustness of prediction guarantees

In the context of this thesis, DNN prediction guarantees are mathematically proved statements that the outputs will behave within pre-specified bounds. Moreover, this thesis reviews both probabilistic bounds and prediction robustness methods.

Probability theory offers a way to represent and work with uncertainty, making it an essential machine learning methodology (Bishop & Bishop 2023, 24 – 26). Furthermore, probabilities follow two rules: the sum rule and the product rule. When these rules are combined with decision theory, they enable optimal predictions from whatever information is available. The sum rule tells how to obtain the probability of one variable X by summing the uncertainty of another variable Y that appears in the joint distribution:

$$p(X) = \sum_Y p(X, Y) \tag{2.10}$$

where $p(X, Y)$ is a joint probability of X and Y. The product rule shows how joint probabilities factor into a conditional distribution $p(Y|X)$ combined with the quantity of $p(X)$:

$$p(X, Y) = p(Y|X)p(X). \tag{2.11}$$

Another essential theory in machine learning is Bayes' theorem that follows from the product rule (Bishop & Bishop 2023, 28):

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \tag{2.12}$$

Furthermore, using the equation (2.10), Bayes' theorem presents a normalization format:

$$p(X) = \sum_Y p(X|Y)p(Y). \quad (2.13)$$

Taylor (2006, 9 – 10) presents verification and validation (V&V) methods providing guarantees of DNN predictions. The paper proposes to begin with verification, asking the question whether the DNN has been deployed correctly, with a focus on the results, whereas validation examines whether the DNN meets the real-world need. Furthermore, more mathematical methods to provide guarantees of DNN predictions are needed.

Probabilistic DNN prediction verification is concerned with comparing the output distribution of DNN under a probability of the distribution of inputs (Boetius et al. 2024, 1). In practice, a guarantee for a DNN is a proof that the chance of a failure is insignificant and can be written down as a number, such as failure probability is less than 5 % with 95 % confidence level. These numbers are called probabilistic bounds, and they are the bridge between laboratory accuracy and real-world safety (Taylor 2016, 137 – 138). Furthermore, probabilistic bounds provide limits on the loss function or error that the DNN can predict (Boetius et al. 2024, 2).

In terms of generalization error bounds, the simplest bound starts with the generalization gap, that is, the difference between the error on the test data and the unknown true error (Goodfellow et al. 2016, 107). In practice, the gap is examined to identify underfitting that denotes whether the model can obtain a low error value on the training set, whereas overfitting denotes the gap between the training error and test error being too large (Goodfellow et al. 2016, 104).

Hoeffding's inequality is one way to provide quantification guarantees, obtaining the empirical averages of the input data and how the predictions deviate from the expected outcome (Shalev-Shwartz & Ben-David 2014, 56). Moreover, Hoeffding's bound provides a probabilistic guarantee for the deviation of the empirical mean from the expected outcome of random variables. Specifically, Hoeffding's bound states the confidence level on the true error of the NN on any random data.

Lemma 2.1 (Hoeffding's inequality). *Let $\bar{X}$ be a mean of a sequence $X_i, \dots, X_m$ of random variables bounded by a and b. Assume that for all $i, E(\bar{X}) = \mu$ and $P(a \leq X_i \leq b) = 1$. Then for any desired deviation $\epsilon > 0$,*

$$P(|\bar{X} - \mu| > \epsilon) \leq 2e^{-\frac{2m\epsilon^2}{(b-a)^2}}.$$

Chebyshev's inequality, instead, provides probability guarantees based on variance (Shalev-Shwartz & Ben-David 2014, 423). Furthermore, it places an

upper bound on the probability that an experimental value of a random variable with finite mean and variance will differ from the mean by more than a fixed positive number.

Lemma 2.2 (Chebyshev's inequality). *Let X be a random variable bounded by a and assume that $E(X) = \mu$ and $Var(X) = \sigma^2$. Then for bound $a > 0$,*

$$P[(X - \mu)^2 \geq a^2] \leq \frac{\sigma^2}{a^2}.$$

Chernoff's inequality is utilized to provide exponentially deteriorating probabilities on deviations from the expected outcome of a sum of random variables (Shalev-Shwartz & Ben-David 2014, 423 – 424).

Lemma 2.3 (Chernoff's inequality). *Let $X_i, \dots, X_m$ be a sequence of random variables. Assume that $E(X) = \mu$. Let $X = \sum_{i=1}^{m} X_i$. Then for any deviation $\delta > 0$,*

$$P[X > (1 + \delta)\mu] \leq e^{-\frac{\delta^2 \mu}{2+\delta}}.$$

The ability to preserve high accuracy of DNN predictions when being exposed to adversarial perturbation or noise is called the robustness of deep neural networks (Pauli 2025, 44). Such adversarial perturbation can be injected either into the input data set or into the weights (Goodfellow et al. 2016, 216). Theory denotes the latter technique as a stochastic implementation of a Bayesian inference and is utilized especially in recurrent neural networks. Such robustness is delivered with Lipschitz bounds (Du & Swamy 2014, 789):

Lemma 2.4 (Lipschitz bounds). *Let vector $f: \mathbb{R} \to \mathbb{R}$ and random variables $x_1, x_2 \in \mathbb{R}$. Then for any constant $\mathcal{L} > 0$,*

$$|f(x_1) - f(x_2)| \leq \mathcal{L}|x_1 - x_2|,$$

where $f(x)$ is the Lipschitz continuous. Specifically, $f: \mathbb{R} \to \mathbb{R}$ is called Lipschitz continuous if there exists a positive $\mathcal{L}$-Lipschitz constant such that, for variables x_1 and x_2 Lemma 2.4 is true. Intuitively, $\mathcal{L}$-Lipschitz constant is a measure of how fast the function can change, hence it is both a sensitivity and robustness measure (Pauli 2025, 17).

Probabilistic bounds translate DNN predictions into quantitative guarantees. Generalization error bounds control the frequency of predictions, while the Lipschitz condition identifies the robustness of prediction guarantees. In the next chapter, the thesis examines sampling-based approaches providing prediction guarantees, including classical probabilistic methods.

3 Sampling-based approaches providing prediction guarantees

Chapter 2 defines the fundamentals of DNN prediction guarantees. This chapter begins by reviewing the literature of Hoeffding's, Chebyshev's, and Chernoff's probabilistic inequalities. The chapter ends with remarks on heuristic methods in NNs.

3.1 Probabilistic bounds

Sampling-based approaches aim to provide tools for quantifying statistical guarantees in DNN predictions. In the context of this thesis, such approaches are Hoeffding's, Chebyshev's, and Chernoff's bounds.

3.1.1 Hoeffding's bound

Hoeffding's bound, derived from Hoeffding's inequality, aims to provide probabilistic guarantees. Furthermore, Hoeffding's bound provides a method for quantifying the deviation of empirical averages from the true expectation, specifically applied in machine learning to evaluate the difference between empirical risk and expected risk (Shalev-Shwartz & Ben-David 2014, 56). Mathematically, Hoeffding's inequality is expressed as Lemma 2.1. Hoeffding's bound is commonly used in the literature of NNs to provide statistical guarantees.

Wang et al. (2022, 3 – 4) utilize Hoeffding's bound and Vapnik-Chervonenkis Dimension (VC-dimension) in the framework for evaluating the relative generalization ability of DNN. Moreover, the approach measures generalization error by computing the difference between training and test data via the cross-tasks of image classification and object detection. In practice, the Confidence Dimension (CD) approach starts with training DNNs using data that is labelled both randomly and accurately to determine the VC-dimension. Next, the approach utilizes Hoeffding's inequality and incorporates the specifics of the optimization process to compute the corrected generalization term to obtain the CD. The paper proposes CD concept as a probabilistic upper bound on the generalization ability with a probability of

$$P \geq 1 - \frac{1}{(2+E)^4}, \tag{3.1}$$

where E denotes as training error. While Wang et al. (2022, 6) conclude that CD enables consistent and reliable results of DNN in image classification and object detection, the approach has not been validated with other

applications. Moreover, the approach requires a large number of experiments to deliver stable performance.

Than & Phan (2025, 2) introduce a generalization bound on the test error of large-scale DNNs using a new concentration inequality like Hoeffding's inequality. The paper presents an approach that addresses conditionally independent variables, whereas Hoeffding's inequality is for the sum of independent variables. Unlike other approaches that require model modifications such as compression or quantization, the paper presents a technique that is applied directly to pretrained models and establishes an upper bound on the generalization error using training data without strict assumptions required. The paper proposes a probabilistic upper bound on the generalization error with a probability of

$$P \geq 1 - \gamma^{-\alpha} - \delta, \tag{3.2}$$

for any constants $\gamma \geq 1, \delta > 0$ and $\alpha \in [\frac{\gamma n(K+\gamma n)}{K(4n-3)}]$, where n denotes the sample size and K is the number of partition regions. While Than & Phan (2025, 11) conclude the bound provides non-vacuous theoretical guarantees for a trained NN model using only the training data set without external support, the study has limited scope in terms of taking other NN model properties into consideration and lacks proof on the practical implementation of different applications.

Furthermore, Hoeffding's inequality is utilized in optimization engineering. Hertneck et al. (2018, 546) apply Hoeffding's inequality to validate the stability of an NN-based learned approximate Model Predictive Controller (MPC) and to deliver guarantees. Furthermore, the framework utilizes Hoeffding's inequality to obtain a lower bound for the probability of framework stability and constraints from to empirical risk. Similarly, Nubert et al. (2020, 3050) utilize Hoeffding's inequality in MPC stability and constraint. Furthermore, the study employs Hoeffding's inequality for statistically ensuring that the deviation between the NN approximation and the robust MPC control remains within acceptable limits and guarantees safety in real-time robotic tracking control. While Hoeffding's inequality is proposed to provide guarantees in safety-critical applications such as robot tracking control, more general verification and validation procedures are needed (Herbert et al. 2018, 548 & Nubert et al. 2020, 3057).

Tokmak et al. (2024, 12) demonstrate the application of Hoeffding's inequality in Bayesian optimization for robotic control and utilize Hoeffding's inequality to create probabilistic guarantees for the estimation of an upper bound on the reproducing kernel Hilbert space (RKHS) norm. Moreover, the paper presents Hoeffding's inequality to quantify the maximum deviation between the empirical mean RKHS norm computed from randomly generated RKHS functions and its true expected value. The paper provides an

approximately correct bound to ensure that the estimated RKHS norm is greater than or equal to the true unknown RKHS norm with high probability and guarantees safety in Bayesian optimization. The approach allows the algorithm to estimate RKHS norms directly from data and reduces reliance on overly conservative a priori bounds.

Hoeffding's bound provides loose probabilistic guarantees. Therefore, it should not be applied as a sole prediction guarantee method for safety-critical real-world applications. While the bound has increased computation time and needs bounded random variables with defined bound, it is widely utilized in deep learning. However, the prediction guarantee tightness and computation time trade-off should be carefully evaluated.

3.1.2 Chebyshev's bound

Chebyshev's inequality bounds the probability of observing deviations and states that the probability bound is directly proportional to the variance and inversely proportional to the square of the chosen fixed positive number (Shalev-Shwartz & Ben-David 2014, 423). Mathematically, Chebyshev's inequality is expressed as Lemma 2.2. Despite the property in the variance-only method, Chebyshev's bound is typically less frequently applied in DNN literature compared to Hoeffding's and Chernoff's bounds due to the generally weaker guarantees provided for large deviations.

In practical applications, Ali & Angelov (2018, 3190) utilize Chebyshev's inequality within an empirical data analytics framework for anomalous behavior detection that the paper proposes to be utilized in image detection. Furthermore, the paper introduces a metric called standardized eccentricity that is derived from Chebyshev's inequality and detects anomalous data points without prior assumptions. The paper suggests the approach allows robust detection of anomalies based solely on the data's variance structure and therefore simplifies analysis of complex and heterogeneous datasets.

Jasour et al. (2022, 273) apply Chebyshev's inequality in risk assessment scenarios for autonomous vehicles together with the sum of squares method. The paper proposes employing a nonlinear variant of Chebyshev's inequality to compute upper bounds on the probability of safety constraint violations. Specifically, the paper defines a probability bound for the safety constraint $g(x_t)$ as

$$P(g(x_t) \leq 0) \leq \frac{E[g(x_t)^2]-E[g(x_t)]^2}{E[g(x_t)^2]}. \tag{3.3}$$

Chebyshev's bound provides loose probabilistic guarantees. Loose bounds typically are too conservative for applications requiring high-performance, however are utilized in deep learning.

3.1.3 Chernoff's bound

Chernoff's inequality provides bounds on the probability that the sum of independent random variables deviates from its expected value (Shalev-Shwartz & Ben-David 2014, 424). Furthermore, Chernoff's inequality provides a bound on the probability that a sum of independent random variables takes values between 0 and 1 has a value far away from its expected value (Dillencourt et al. 2025, 1).

Cardelli et al. (2019, 4) utilize parameterized Chernoff's bound to analyze the robustness of Bayesian Neural Networks (BNNs). The paper presents a framework that evaluates the probability of the existence of adversarial perturbations within bounded input regions, thus quantifying statistical uncertainty and robustness. Furthermore, Chernoff's bound is applied to quantify the likelihood that sampled deterministic NNs deviate substantially in predictions, thus aiming to provide statistical guarantees for robustness in adversarial scenarios. However, the amount of samples for Chernoff's bound probabilities between 0.23 and 0.79 requires a larger sample size than Massart's bounds, which are also presented in the paper (Cardelli et al. 2019, 8).

Dillencourt et al. (2025, 2) introduce a parameterized Chernoff's inequality for simplified algorithm analysis and demonstrate practical application of the inequality in analyzing randomized algorithms and data structures. The paper presents simplified parameterized Chernoff's inequality to deliver probabilities expressed as powers of two, enhancing the interpretability and practical utility of these bounds in algorithmic contexts. The paper highlights the practical efficiency and clarity gained through the bound and overcomes the limitations inherent in traditional Chernoff's bound. Such limitations are powers of Euler's number and the δ^2 term in the (Dillencourt et al. 2025, 1).

Chernoff's bound offers moderately tight probabilistic guarantees. However, the bound can become computationally intensive, as can be concluded from Lemma 2.3. Therefore, it is not optimal for large DNNs.

3.2 Remarks on heuristic methods in NNs

In the context of this thesis, heuristic methods in NNs refer to empirical methods aiming to identify predictions within acceptable computational performance. However, heuristic methods do not guarantee optimal predictions nor deliver guarantees. Typically, heuristic methods guide the design and training processes to improve performance and prediction capacity. Various heuristic prediction verification methods have been utilized throughout the history of ML.

Walczak & Cerpa (1999, 107) conduct a heuristic method in ANNs to optimize the performance. The method selects input variables, determines the

number of hidden layers, nodes per layer, and chooses suitable learning methods. Moreover, the approach simplifies complex decisions and reduces the time needed compared to formal methods. By effectively reducing unnecessary input variables, the paper defines heuristic methods to enhance generalization performance and to reduce the chances of under- or overfitting (Walczak & Cerpa 1999, 111), ultimately leading to improved predictive accuracy of an ANN.

Ozsoydan & Gölcük (2022, 1) present a hyper-heuristic framework to train feedforward NNs. Furthermore, the method emphasizes the combination of various metaheuristic algorithms such as Particle Swarm Optimization, Differential Evolution, and the Flower Pollination Algorithm. The method addresses issues of local minima and slow convergence common in traditional training methods and offers improvements in both function approximation and classification tasks (Ozsoydan & Gölcük 2022, 9).

Chen & Wei (2011, 1768) iteratively train NNs to approximate a heuristic function that reduces search times and computational complexity. The paper proposes a method that updates the heuristic values during the search process based on previous performance.

Another way to quantify uncertainty in a heuristic manner is the bootstrapping method, which is based on statistical inference, such as confidence intervals and standard error evaluations (Shin et al. 2021, 1). The challenge in the bootstrapping method is the computational cost that limits the practical utilization of bootstrapping, especially in DNNs.

4 Prediction guarantees robustness approaches

The robustness guarantee indicates how much the perturbed input to an NN can change without significantly altering its predictions, which is critical for deploying NNs in safety-critical systems (Barbara et al. 2025, 1). The third chapter presented literature on classical probabilistic bounds. This chapter focuses on robustness methods providing DNN prediction guarantees. The chapter starts with a framework that certifies the robustness of NNs (CROWN) and presents Probabilistically Verify Neural Network (PROVEN). In addition, the chapter presents Lipschitz continuity, constrained architectures, and physics-informed neural networks (PINN).

4.1 Certify robustness of neural networks with general activation functions (CROWN)

Zhang et al. (2018, 1) introduce a general framework designed to provide robustness guarantees for NNs with general activation functions (CROWN). Furthermore, CROWN bounds the outputs of NN with linear and quadratic approximation methods. Unlike other verification approaches that only support a limited number of activation functions, CROWN deals with various activation functions such as ReLU, tanh, sigmoid, and arctan (Zhang et al. 2018, 2).

Zhang et al. (2018, 4) highlight that CROWN provides certified lower bounds on adversarial perturbations. These bounds guarantee the NN's predictions to remain consistent within a defined range of input perturbations, quantifying the minimal distortion needed for adversarial perturbations to mislead the NN. Moreover, the paper presents that for a given input x_0 and adversarial perturbations within a ℓ_p-norm region, the robustness bounds deterministically certify that no adversarial perturbation within the ℓ_p-norm region can change the model's output for any distance norm $p \geq 1$ (Zhang et al. 2018, 2).

The performance-computing time trade-off in CROWN is notable. CROWN reduces approximation errors compared to earlier methods by employing adaptive linear upper and lower bounds and provides tighter and more reliable robustness certificates. However, CROWN balances computational efficiency with the tightness of its robustness bounds (Zhang et al. 2018, 3). More closely, the paper proposes up to 28 % improvements in lower bound certificates compared to older methods Fast-Lin and Fast-Lip (Zheng et al. 2018, 16). However, the average computation time increases to 81 %. Specifically, CROWN computes certified robustness bounds for large networks, as demonstrated in experiments with NNs over 10 000 neurons, where robustness guarantees were computed. The computing time was approximately one minute on a single central processing unit (CPU) core. The

paper defines CROWN computational efficiency as improved compared to Reluplex, which provides exact robustness certificates, however is computationally infeasible for large networks (Zheng et al. 2018, 2).

CROWN is, in fact, a special case of a convolutional neural network (CNN) robustness certifying framework called CNN-Cert (Boopathy et al. 2019, 3244). Moreover, CNN-Cert improves CNN computational speed by exploiting CNN structures. The paper proposes CNN-Cert to produce the same robustness certificates as CROWN when identical linear bounds are used for ReLU activations. However, for general activation functions, CNN-Cert applies more precise linear bounds, resulting in up to 260 % improvement in certificate quality compared to CROWN. In addition, CNN-Cert outperforms CROWN in computational speed and achieves 2.5 to 11.4 times computational speed using convolutional bounds.

CROWN seems to offer a balanced approach for computational efficiency and the quality of robustness guarantees for DNNs. Hence, CROWN would be suitable for real-world applications requiring fast and reliable NN prediction verification. Its adaptability to various NNs and activation functions is further notable. However, more research on the practical applicability of CROWN and CNN-Cert is advisable.

4.2 Probabilistically Verify Neural Networks (PROVEN)

Probabilistically Verify Neural Networks (PROVEN) introduces a statistical approach to verify the robustness of NNs under adversarial perturbations (Weng et al. 2019, 6727). Furthermore, PROVEN integrates probabilistic uncertainty to yield tighter robustness guarantees instead of only providing deterministic worst-case bounds. Such deterministic worst-case methods like CROWN and CNN-Cert might provide a hard guarantee. However, they often possess too loose bounds, which states they might unnecessarily limit the deployment of DNNs. Moreover, Weng et al. (2019, 6728) highlight that they fail to perform in events in which perturbations follow a Gaussian and sub-Gaussian distribution. PROVEN addresses this limitation by integrating probabilistic certificates and hence provides tighter robustness guarantees that are statistically notable.

Weng et al. (2019, 6729) introduce in their paper first the worst-case scenario, in which input examples can be perturbed by any noise contained within a ℓ_p ball. Next, the paper extends the worst-case analysis to a probabilistic scenario and assumes input perturbations follow a Sub-Gaussian and Gaussian distribution characterization. Then, probabilistic bounds with robustness guarantees are defined as follows (Weng et al. 2019, 6732).

Lemma 4.1 (PROVEN: Gaussian adversarial perturbation). *Let X follow a multivariate Gaussian distribution with mean x_0 and covariance Σ. Define*

$$\mu_L = A_t^L x_0 + d^L, \sigma_L^2 = A_t^L \sum (A_t^L)^T,$$

$$\mu_U = A_t^U x_0 + d^U, \sigma_U^2 = A_t^U \sum (A_t^{LU})^T,$$

where T denotes the transpose operator. Then for any $a \in \mathbb{R}$*, the certified lower bound* γ_L *and upper bound* γ_U *are:*

$$\gamma_L \approx \frac{1}{2} - \frac{1}{2}\delta\left(\frac{a - \mu_L}{\sigma_L\sqrt{2}}\right),$$

$$\gamma_U \approx \frac{1}{2} - \frac{1}{2}\delta\left(\frac{a - \mu_U}{\sigma_U\sqrt{2}}\right),$$

where δ *denotes the error function.*

Intuitively, Lemma 4.1 denotes that if input perturbations follow a Gaussian distribution, common to real-world applications, the robustness guarantee can be expressed clearly in terms of familiar statistical quantities. In practice, the quantification tells how likely the model is to maintain its original prediction given. While Lemma 4.1 addresses scenarios where adversarial perturbations follow Gaussian distribution with bounded support, Lemma 4.2 extends to Sub-Gaussian perturbations.

Lemma 4.2 (PROVEN: Sub-Gaussian adversarial perturbation). *Let* $f : \mathbb{R}^{n_0} \to \mathbb{R}^K$ *be a K-class NN classifier with K number of output classes, and let* $x_0 \in \mathbb{R}^{n_0}$ *be an input with a predicted class* $c = argmin_i f_i(x_0)$*. Let margin function* $g_t(x) \coloneqq f_c(x) - f_t(x)$*, and* $t \neq c$*. Assume the margin function's linear lower bound function is given by* $g_t^L(x) = A_t^L x + d^L$ *and its linear upper bound function* $g_t^U(x) = A_t^U x + d^U$*.*

Let X_i *be independent random variables. Assume a sub-Gaussian distribution with bounded support* $X_i \in [x_{0_i} - \epsilon, x_{0_i} + \epsilon]$*,* $\forall i \in [n_0]$ *and symmetric around the mean* x_{0_i}*. Then for any* $a \in \mathbb{R}$ *, the certified lower bound* γ_L *and upper bound* γ_U *are:*

$$\mathrm{P}[g_t(x) > a] \geq \gamma_L \coloneqq \begin{cases} 1 - e^{\frac{(\mu_L - a)^2}{2\epsilon^2 \|A_t^L\|_2^2}}, if\ \mu_L - a \geq 0 \\ 0, otherwise; \end{cases}$$

$$\mathrm{P}[g_t(x) > a] \geq \gamma_U \coloneqq \begin{cases} e^{-\frac{(\mu_L - a)^2}{2\epsilon^2 \|A_t^U\|_2^2}}, if\ \mu_U + a \leq 0 \\ 1, otherwise. \end{cases}$$

Lemma 4.2 provides probabilistic robustness guarantees when the inputs are perturbed by independent, bounded random variables. Specifically, when each input is independently perturbed within an interval around its mean, the probability that the NN maintains its correct prediction can be quantified. In practice, quantification is the degree of certainty that the NN's prediction will remain correct.

PROVEN guarantees that with 99.99 % confidence, no adversarial perturbation within a certain radius will change DNN predictions. The key result behind PROVEN is the translation of linear bounds obtained from deterministic robustness verification techniques into probabilistic bounds (Weng et al. 2019, 6731), and PROVEN operates without a larger computation overhead (Weng et al. 2019, 6728).

As stated by Weng et al. (2019, 6727), PROVEN can offer robustness certificates that are tighter than those achieved through worst-case analysis alone, which might make it viable in applications demanding high safety and reliability standards.

4.3 Lipschitz bounds

Lipschitz bounds are well-known in the DNN literature and are an important method to guarantee the robustness of DNNs. Next, three use cases of Lipschitz bounds are presented.

Barbara et al. (2025, 1) examine whether Lipschitz-bounded policy networks can enhance the robustness of reinforcement learning against adversarial perturbations. In addition, the paper examines how the choice of Lipschitz architecture influences the performance–robustness trade-off. Specifically, the study investigates the benefits of Lipschitz bounds using Proximal Policy Optimization (PPO) in trained pendulum swing-up and Atari Pong tasks. The pendulum swing-up is a classic continuous control problem, while Atari Pong is a proxy for vision-based autonomous decision-making tasks.

Barbara et al. (2025, 2 – 4) define the Lipschitz constant $Lip(f)$ as the smallest Lipschitz bound $\mathcal{L}$ that satisfies Lemma 2.4 and show that the smaller $Lip(f)$ it is, the more robust the NN is against adversarial perturbations. The paper demonstrates that Lipschitz-bounded NNs show lower sensitivity to input perturbations since a small change at the input can propagate only as a proportionally insignificant change at the output. This is not the case for unconstrained NNs.

Barbara et al. (2025, 13) highlight that the Lipschitz-bounded policy architecture matters when gaining robustness guarantees. The spectral normalization (SN) and almost orthogonal Lipschitz (AOL) layer architectures in the Atari Pong experiment were conservative, destructive, and for small $Lip(f) \leq 20$ even failed to learn for small datasets (Barbara et al. 2025, 13).

In contrast, Cayley and the Sandwich layer architecture deliver robustness without sacrificing performance.

Pauli (2025, 11) develops a system-theoretic method for analyzing, synthesizing, and certifying robustness of NNs with Lipschitz bounds. The study's main contributions are a semidefinite programming (SDP) based Lipschitz constant estimation tool, training of NNs with user-specified $\mathcal{L}$-Lipschitz constant through semidefinite constraints, and stability guarantees for closed-loop systems that contain neural components. At the end of the study, Pauli (2025, 155) argues that the proposed system-theoretic framework provides rigorous guarantees for NN robustness and stability guarantees for closed-loop systems.

For the Lipschitz constant estimation tool, Pauli (2025, 49) presents an SDP-based analysis that enables tighter and scalable Lipschitz bounds for feedforward NNs. The SDP-based analysis over-approximates slope-restricted activation functions and captures characteristic properties of gradient norm preserving activation functions GroupSort and Householder, by deriving quadratic constraints. Next, Pauli (2025, 87) turns the SDP-based analysis into design procedures. Firstly, an optimization problem is induced with semidefinite constraints and was solved by alternating direction method of multipliers, and a custom interior point method to deliver decreased computation time. Lastly, Pauli (2025, 111) introduces a direct parameterization of Lipschitz-bounded NNs called LipKernel that enforces linear matrix inequalities constraints by construction, unconstrained variables to the parameters of the NN.

Zhang et al. (2025, 273) propose a Lipschitz Locomotion Policy (LLP) for quadruped robots with adaptive Lipschitz constraint. The study states LLP reduces the legged robot's erratic actions and improves robustness to observation noise and external disturbances. The LLP is parameterized with multi-dimensional gradient normalization that guarantees a bounded Lipschitz constant. Unlike previous studies in the field, LLP is directly integrated into the robot's locomotion training framework and thus does not require modifications to RL algorithms.

More specifically, Zhang et al. (2025, 279) confirm in their Unitree Go1 robot experiment that across flat ground, grass, a 32° slope, and 16 cm stairs, LLP lowers erratic actions, improves velocity tracking, and increases energy efficiency. In addition, under a 1.5 kg lateral impact, LLP shows reduced joint-trajectory oscillation and better recovery from potential external disturbances.

While Lipschitz bounds are notably visible in NN robustness, especially in perturbed environment literature, scalability to real-world applications remains an important question. Although maintaining tight Lipschitz constraints involves computational trade-offs, recent advances have improved these methods' effectiveness, which might reinforce their practical value in real-world scenarios.

4.4 Constrained architectures

In this chapter, the thesis distinguishes soft and hard constrained architectures and introduces a representative method for both. For soft constraint architecture, Physics-Informed Neural Networks (PINNs) are presented. Furthermore, HardNet is presented for a hard constraint architecture. Both representative methods are visible in the constrained architecture literature. Constrained architectures aim to improve NN's generalization abilities, reduce under- and overfitting, and enhance robustness and reliability.

Constrained architectures are NN methods that enforce structural or mathematical constraints (Min & Azizan 2024, 1; Cuomo et al. 2022, 1). Furthermore, constrained architectures typically follow either encoding prior knowledge, satisfying safety conditions, or respecting the known relationship between NN input and output. These constraints can be equality or inequality conditions and may depend on either the output alone or both the input and output.

Constrained architectures are typically categorized into soft and hard constraints (Min & Azizan 2024, 2). Soft constraints guide the learning process by penalizing deviations through loss functions, however, do not strictly enforce these constraints. Contrastingly, hard constraints enforce requirements to guarantee that inputs and outputs strictly meet the predetermined requirements. Overall, the difference between a soft and hard constrained architecture is the level of restriction.

4.4.1 Physics-Informed Neural Networks (PINNs)

Physics-Informed Neural Networks (PINNs) represent a specialized type of DNN that integrates physical laws into its training process (Cuomo et al. 2022, 2). As PINNs are trained using physical equations and boundary conditions for forward problems, they are unsupervised learning methods (Cuomo et al. 2022, 8). However, when a physical property of a DNN is perturbed, PINN can be considered as a supervised learning method. Due to the physical equations and representation, training a PINN comes with complexity (de Oliveira et al. 2023, 8)

PINNs operate by embedding partial differential equations (PDEs) directly into NN to minimize the loss function. Specifically, the residual term is embedded in the NN's loss function. PINNs are particularly effective in solving PDEs and related scientific problems (Cuomo et al. 2022, 8; de Oliveira et al. 2023, 17). Mathematically, a general PDE defined on a domain $\Omega \in \mathbb{R}^d$ with boundary $\partial\Omega$ can be represented as follows:

$$F(u(z);\ \gamma) = f(z) \qquad z \in \Omega, \tag{4.1}$$

$$B\big(u(z)\big) = g(z) \qquad z \in \partial\Omega \qquad (4.2)$$

where $z := [x_1, \dots, x_{d-1}; t]$ and denotes the space-time coordinate vector, F the PDE operator, u the unknown solution, γ physics-related parameters, B the operator defining initial or boundary conditions, and g the boundary function. To achieve the result, PINNs approximate the PDE by finding θ as follows:

$$\theta^* = argmin(\omega_F L_F(\theta) + \omega_B L_B(\theta) + \omega_d L_{data}(\theta)), \qquad (4.3)$$

where L_F denotes differential equation, L_B the boundary conditions, and L_{data} data, and where $\omega_F, \omega_B, \omega_d$ are associated weights (Cuomo et al. 2022, 8). Thus, θ^* can be considered as the setup of the weights, boundary conditions, and unknown physical constants inside the PINN after the NN training has been conducted.

While PINNs do not provide strict guarantees of constraint satisfaction and can potentially lead to learning inefficiency (Cuomo et al. 2022, 23), imposing hard constraints on NNs may affect the NN performance (Min & Azizan 2024, 1).

4.4.2 HardNet

HardNet framework is a representative model of hard-constrained architectures. The framework incorporates NN input-dependent constraints into NN model and ensures outputs strictly satisfy predefined conditions (Min & Azizan 2025, 1). Furthermore, HardNet provides constraint satisfaction guarantees, unlike soft constrained architectures. More specifically, HardNet appends a differentiable projection of affine constraints directly into NN outputs via differentiable enforcement layers. Such input-dependent affine constraints take a form (Min & Azizan 2025, 6):

$$b_l(x) \leq A(x)f(x) \leq b_u(x), \qquad (4.4)$$

where $A(x) \in \mathbb{R}^{n_c \times n_{out}}$ is a matrix representing linear relationships between input and output, and $b_l(x)$ and $b_u(x)$ represent lower and upper bounds respectively. The projection operation in HardNet has a closed-form solution:

$$P(f_\theta)(x) = f_\theta(x) + A(x)^+[ReLU(b_l(x) - A(x)f_\theta(x) - ReLU\big(A(x)f_\theta(x)\big) - b_u(x))], \quad (4.5)$$

where $A(x)^+$ denotes the inverse matrix of A. As a conclusion, this projection guarantees that the output satisfies all specified affine constraints for every input (Min & Azizan 2025, 5).

While HardNet enforces strict conditions, it does not compromise NN's performance. Min & Azizan (2025, 6–7) demonstrated through universal approximation theorems that NNs with HardNet layers maintain full expressive capabilities. Specifically, the study showed that if traditional unconstrained NNs universally approximate a given function, HardNet retains this capability within feasible constraint sets.

However, imposing hard constraints on NN has some limitations and challenges. Márquez-Neila et al. (2017, 1) highlight that although theoretically advantageous, hard constraints are typically computationally intensive and challenging to enforce within DNNs. In addition, the study showed that soft constraints can sometimes yield better performance due to computational simplicity and avoidance of issues like dependency or incompatibility among constraints.

Min & Azizan (2025, 5 – 7) provide proof that HardNet preserves the universal approximation ability while maintaining expressive power despite the strict enforcement of constraints. In addition, the study shows that NNs with HardNet layers can be applied to any NN, and they approximate any continuous function within the feasible set defined by affine constraints, which are needed if the network has sufficient width. In addition, HardNet-Cvx provides a differentiable convex optimization solver for more general convex constraints.

5 High-level comparative overview

The fourth chapter discussed methods for NN prediction guarantee robustness and constrained architectures. This chapter provides a high-level comparative overview of prediction guarantee and robustness methods. This overview is shown in Table 1 and compares guarantee type, assumptions, NN architectures, scalability, guarantee type, and qualitative tightness, which refers to the accuracy of the methods presented in this thesis.

Classical probabilistic bounds offer a probabilistic description of uncertainty in their predictions and are generally architecture-agnostic methods. However, they do not provide certified guarantee robustness like deterministic methods aim to provide.

Table 1. High-level comparative overview of prediction guarantee and robustness methods.

Method	Guarantee type	Assumptions	Architectures	Scalability	Tightness
Hoeffding's bound	Probabilistic framework (Hertneck et al. 2018, 544).	Sequence of independent bounded random variables (Hertneck et al. 2018, 546).	Model-agnostic	Yes	Loose. Tightness between Chernoff and Chebyshev.
Chebyshev's bound	Probabilistic framework (Jasour et al. 2022, 270).	Any random variable with a finite variance (Jasour et al. 2022, 273).	Model-agnostic	Yes	The loosest of the three probabilistic guarantee methods.
Chernoff's bound	Probabilistic framework (Dillencourt et al. 2025, 1).	Sequence of independent random variables with {0,1} values (Dillencourt et al. 2025, 1).	Model-agnostic	Yes, however bound can be computationally intensive.	The tightest of the three probabilistic guarantee methods (Mitzenmacher & Upfal 2017, 87).
CROWN	Deterministic framework (Zhang et al. 2018, 1).	ReLU and general activation functions sigmoid, tanh and arctan (Zhang et al. 2018, 2; Boopathy et al. 2019, 3241).	Feed-forward NNs, efficient for CNNs (Boopathy et al. 2019, 3241).	Yes, scalable for NNs over 10 000 neurons (Zhang et al. 2018, 2).	Tight
PROVEN	Probabilistic framework (Weng et al. 2019, 1).	Adversarial perturbed input with Gaussian and sub-Gaussian distribution (Weng et al. 2019, 2).	Agnostic, large NNs (Weng et al. 2019, 2).	Limited computational overhead when incorporated in CROWN and CNN-Cert (Weng et al. 2019, 2).	Tighter than worst-case certificates such as CROWN and CNN-Cert (Weng et al. 2019, 2).
Lipschitz bounds	Deterministic framework (Pauli 2025, 44).	NN is Lipschitz continuous (Pauli et al. 2022, 122).	Feed-forward NNs, CNNs (Pauli 2025, 39 – 40).	Computing the exact constant is NP-hard, thus other techniques are utilized to compute upped bounds (Pauli 2025, 21).	Bounds are typically loose (Barbara et al. 2025, 1).
PINNs	Soft-constrained (Raissi et al. 2019, 688).	A known physical model described by PDEs is required (Raissi et al. 2019, 687)	Feed-forward DNNs (Raissi et al. 2019, 687).	PINNs are computationally expensive (de Oliveira et al. 2023, 16).	No hard guarantee of exact satisfaction, only penalty (Raissi et al. 2019, 690).
HardNet	Hard-constrained (Min & Azizan 2024, 1).	Input-dependent affine/convex constraints (Min & Azizan 2024, 3).	Agnostic (Min & Azizan 2024, 1).	Yes, efficient forward pass and closed-form layer (Min & Azizan 2024, 1).	Constraints are always satisfied exactly (Min & Azizan 2024, 3).

6 Conclusion

Deep learning is a powerful technology that is broadly applicable and demonstrates the capability to solve complex tasks across several fields. The ability of DNNs to learn patterns directly from large datasets has expanded the boundaries of what AI can achieve. Despite these advantages, DNNs are black-box models and lack profound robustness and stability guarantees, which makes them prone to adversarial perturbations. This limits the use of DNNs in safety-critical applications, such as autonomous driving or surgical robots. They often fail to provide measures of uncertainty and therefore lead to situations where their predictions cannot always be trusted.

To address reliability issues, methods that can provide prediction guarantees and ensure robustness in perturbed environments are crucial. This thesis reviewed sampling-based prediction guarantee methods as Hoeffding's, Chebyshev's, and Chernoff's bounds, and robustness methods, CROWN, PROVEN, and Lipschitz bounds. In addition, both soft-constrained architecture PINNs and hard-constrained architecture HardNet were reviewed.

As a conclusion of the literature review, no single method is solely adequate to provide a full guarantee robustness certification. In practice, combining methods is advisable in safety-critical applications such as medical diagnosis and autonomous driving. This motivates combining approaches that, e.g., layer probabilistic guarantees on top of deterministic certificates.

Another conclusion is that not all guarantee methods apply to fast-reacting real-world problems due to their computational performance. In addition, the required tightness, computational cost and architectural assumptions should be considered when evaluating the suitable methods for a particular application.

Finally, a more systematic study across DNNs is needed to understand how architectural choices differentiate guarantee robustness, certificate tightness, and computational cost. In addition, research on method combinations such as probabilistic with deterministic should take place.

References

Ali, A. M. & Angelov, P. (2018). Anomalous Behaviour Detection Based on Heterogeneous Data and Data Fusion. Soft Computing : A Fusion of Foundations, Methodologies and Applications. Vol. 22:10. Pp. 3187 – 3201. Available from: DOI: 10.1007/s00500-017-2989-5

Barbara, N. H., Wang, R. & Manchester, I. R. (2025). On robust reinforcement learning with Lipschitz-bounded policy networks. 17 p. Available from: DOI: 10.48550/arXiv.2405.11432

Bishop, C. M. & Bishop, H. (2023). Deep learning : foundations and concepts. Cham: Springer. 649 p. ISBN : 978-3-031-45467-7.

Boetius, D., Leue, S. & Sutter, T. (2024). Probabilistic verification of neural networks using branch and bound. Available from: DOI: 10.48550/arXiv.2405.17556

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S. & Daniel, L. (2019). CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks. AAAI Conference on Artificial Intelligence. Menlo Park, California, United States of America : AAAI Press. P. 3240–3247. Available from: DOI: 10.1609/aaai.v33i01.33013240

Cardelli, L., Kwiatkowska, M., Laurenti, L., Paoletti, N., Patane, A. & Wicker, M. (2019). Statistical guarantees for the robustness of Bayesian neural networks. 9 p. Available from: DOI: 10.48550/arXiv.1903.01980

Chen, H.-C. & Wei, J.-D. (2011). Using neural networks for evaluation in heuristic search algorithm. In Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 25:1. Pp. 1768 – 1769.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M. & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. Journal of Scientific Computing. Vol. 92:3. 88 p. Available from: DOI: 10.48550/arXiv.2201.05624

Dillencourt, M., Goodrich, M. T. & Mitzenmacher, M. (2025). Leveraging parameterized Chernoff bounds for simplified algorithm analyses. Information Processing Letters. Vol. 187. 7 p. Available from: DOI: 10.1016/j.ipl.2024.106516

Du, K.-L. & Swamy, M. N. S. (2014). Neural Networks and Statistical Learning. London, United Kingdom: Springer. 824 p. ISBN : 978-1-4471-5570-6.

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R. & Zhu, X. X. (2023). A survey of uncertainty in deep neural networks.

Artificial Intelligence Review. Vol. 56. P. 1513 – 1589. Available from: DOI: 10.1007/s10462-023-10562-9

Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep learning. Cambridge, Massachusetts, United States of America: The MIT Press. 775 p. ISBN : 978-0-262-03561-3.

Hertneck, M., Köhler, J., Trimpe, S. & Allgöwer, F. (2018) Learning an Approximate Model Predictive Controller with Guarantees. IEEE Control Systems Letters. Vol. 2. P. 543 – 548. Available from: DOI: 10.1109/LCSYS.2018.2843682

Jasour, A., Huang, X., Wang, A. & Williams, B. C. (2022). Fast nonlinear risk assessment for autonomous vehicles using learned conditional probabilistic models of agent futures. Autonomous Robots. Vol. 46. Pp. 269 – 282. Available from: DOI: https: 10.1007/s10514-021-10000-1

Kwiatkowska, M. (2019). Safety and robustness for deep learning with provable guarantees. ESEC/FSE '19 : proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering : August 26-30, 2019, Tallinn, Estonia. New York, NY, United States of America : Association for Computing Machinery, Inc.,. 3 p. Available from: DOI: 10.1145/3338906.3342812

LeCun Y., Bengio Y. & Hinton G. (2015). Deep learning. Nature. Vol. 521:7553. P. 436-444. Available from: DOI: 10.1038/nature14539

Márquez-Neila, P., Salzmann, M. & Fua, P. (2017). Imposing hard constraints on deep networks: Promises and limitations. 9 p. Available from: DOI: 10.48550/arXiv.1706.02025

Mermillod, M. (2020). The Hold-Up of the Century: Neural Networks are Coming from Cognitive Science and not Machine Learning. Perspectives to Avoid a New Dark Age of Artificial Intelligence. Trends in Artificial Intelligence. Vol. 4:1. P. 80-86. Available from: DOI: 10.36959/643/306

Min, Y. & Azizan, N. (2024). HardNet: Hard-Constrained Neural Networks with Universal Approximation Guarantees. arXiv preprint arXiv:2410.10807. 27 p. Available from: DOI: 10.48550/arXiv.2410.10807

Mitzenmacher, M. & Upfal, E. (2017). Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis. Cambridge university press. 401 p.

Nubert, J., Köhler, J., Berenz, V., Allgöwer, F. & Trimpe, S. (2020). Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. IEEE Robotics and Automation Letters. Vol 5:2. P. 3050 – 3057. Available from: DOI: 10.1109/LRA.2020.2975727

de Oliveira, L. S., Kunstmann, L., Pina, D., de Oliveira, D. & Mattoso, M. (2023). PINNProv: Provenance for Physics-Informed Neural Networks. 2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW). Porto Alegre, Brazil : Proceedings / IEEE. P. 16 – 23. Available from: DOI: 10.1109/SBAC-PADW60351.2023.00013

Ozsoydan, F. B. & Gölcük, İ. (2022). A hyper-heuristic based reinforcement-learning algorithm to train feedforward neural networks. Engineering Science and Technology, an International Journal. Vol. 35. 10 p.

Pauli, P., Koch, A., Berberich, J., Kohler, P. & Allgöwer, F. (2022). Training Robust Neural Networks Using Lipschitz Bounds. IEEE Control Systems Letters. Vol. 6. Pp. 121 – 126. Available from: DOI: 10.1109/LCSYS.2021.3050444

Pauli, P. (2025). Robust neural networks: Analysis, synthesis, and control. Doctoral thesis. Universität Stuttgart, Institut für Systemtheorie und Regelungstechnik. Stuttgart. 211 p.

Raissi, M., Perdikaris, P. & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics. Vol. 378. Pp. 686 – 707.

Shalev-Shwartz, S. & Ben-David, S. (2014). Understanding machine learning : from theory to algorithms. New York, NY: Cambridge University Press. 449 p. ISBN : 978-1-107-05713-5.

Shin, M., Cho, H., Min, H.-S. & Lim, S. (2021). Neural bootstrapper. Advances in Neural Information Processing Systems. Vol. 34. 20 p. Available from: DOI: 10.48550/arXiv.2010.01051

Taylor, B. J. (editor) (2006). Methods and Procedures for the Verification and Validation of Artificial Neural Networks. New York (NY), United States of America: Springer Science & Business Media. 277 p. Available from: DOI: 10.1007/0-387-29485-6

Than, K. & Phan, D. (2025). Non-vacuous Generalization Bounds for Deep Neural Networks without any modification to the trained models. 21 p. Available from: DOI: 10.48550/arXiv.2503.07325

Tokmak, A., Schön, T. B. & Baumann, D. (2024). PACSBO: Probably approximately correct safe Bayesian optimization. 15 p. Available from: DOI: 10.48550/arXiv.2409.01163

Walczak, S. & Cerpa, N. (1999). Heuristic principles for the design of artificial neural networks. Information and software technology. Vol. 41:2. Pp. 107-117.

Wang, R., Yang, L., Zhang, B., Zhu, W., Doermann, D. & Guo, G. (2022). Confidence dimension for deep learning based on Hoeffding inequality and relative evaluation. 9 p. Available from: DOI: 10.48550/arXiv.2203.09082

Weng, T.-W., Chen, P.-Y., Nguyen, L. M., Squillante, M. S., Boopathy, A., Oseledets, I. & Daniel, L. (2019). PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach. Proceedings of Machine Learning Research. Vol. 97. P. 6727-6736. San Diego, California, United States of America: Journal Machine Learning Research.

Zhang, H., Weng, T.-W., Chen, P. Y., Hsieh, C.-J. & Daniel, L. (2018). Efficient neural network robustness certification with general activation functions. Neural information processing systems. Vol. 31. 17 p. Available from: DOI: 10.48550/arXiv.1811.00866

Zhang, Y., Nie, B. & Gao, Y. (2025). Robust Locomotion Policy With Adaptive Lipschitz Constraint for Legged Robots. IEEE Robotics & Automation Letters.Vol. 10:1. Pp. 272 – 279. Available from: DOI: 10.1109/LRA.2024.3506278