FIRST EDITION

MATHEMATICAL FOUNDATIONS OF NEURAL NETWORKS

From CNNs to PINNs, QINNs, and Deep Reinforcement Learning

RAPHAEL NORIEGA

Preface

For as long as I can remember, I have dreamed of writing a book where no step is skipped, where every concept finds its place in a logical chain, and where the reader is never left with the feeling that something essential has been hidden between the lines. Too often, in mathematics and in the sciences, explanations assume too much, or leave gaps that only those with prior training can fill. This book is my answer to that problem.

When I began teaching and researching neural networks, I was struck by how fragmented the explanations often were. Some texts went straight to the formulas, as if the reader should already know why those symbols mattered. Others remained on the surface, offering analogies without showing the rigorous backbone of mathematics that sustains the theory. The result is that many students, researchers, or professionals are left either overwhelmed by abstraction or undernourished by oversimplification. My conviction is that knowledge should not be fragmented: it should flow as a continuous path where each stone is visible and firmly placed for the next step.

That is the spirit of this book. Here, I aim to construct a narrative where the reader walks hand in hand with the mathematics, never jumping over a void of reasoning. Concepts are not presented as isolated facts, but as elements of a larger structure, built slowly with logic as mortar. The aim is not just to show *what* neural networks are, but to reveal *why* they are built the way they are, and how mathematics—linear algebra, calculus, probability, optimization—becomes the invisible skeleton that gives them life.

This is a book for a wide audience. It is written for the undergraduate student taking their first steps in artificial intelligence, for the graduate researcher who needs a rigorous reference, and for professionals in the industry who want to understand the deeper principles behind the tools they use. My hope is that this book will serve as both a guide and a reference: a text you can study systematically from beginning to end, but also one you can revisit to clarify details or to build new insights.

Writing this book has also been a personal journey. I wanted to produce something that stands apart from textbooks that either rush through derivations or leave entire arguments as an exercise for the reader. This will not be a book of shortcuts. It will demand effort. There will be formulas, derivations, and proofs. But I promise that every line is there for a reason. Nothing is left to chance, and nothing is presented without context. If you walk with me through these pages, you will not only learn what neural networks are and how they work—you will also discover the rhythm and poetry of mathematics itself, the logic that binds the abstract to the real.

Finally, this book is also a gesture of trust. Trust in the reader's intelligence, in their patience, and in their hunger for depth. I have chosen not to underestimate you, whoever you are, because I believe that clarity is not about removing complexity but about guiding through it.

May these pages serve as a bridge: between mathematics and intuition, between logic and imagination, and between the academic world and real-world applications.

Contents

Part I.

Mathematical Preliminaries

Introduction | 1.

1.1. Historical Context of AI and Neural Networks

The story of neural networks is inseparable from the broader history of artificial intelligence. In the mid-twentieth century, pioneers began to ask a radical question: could machines learn, reason, and perhaps even think?

Early models of computation and the brain appeared in the 1940s and 1950s. Warren McCulloch and Walter Pitts proposed a mathematical model of the neuron, reducing it to a binary threshold unit. Their work suggested that networks of such units could, in principle, compute any logical function. Around the same time, Alan Turing speculated about "learning machines," planting seeds that would later grow into the foundations of AI.

The perceptron era began in 1957, when Frank Rosenblatt introduced a trainable model capable of learning linear decision boundaries. It captured the imagination of both scientists and the public, sparking optimism that machines could soon replicate the brain's capacity for learning. Yet mathematics also revealed the perceptron's limits. In the late 1960s, Marvin Minsky and Seymour Papert proved that single-layer perceptrons could not solve even simple nonlinear problems such as XOR. This was a sobering reminder that without mathematical rigor, bold claims collapse under scrutiny.

These early attempts reveal an important lesson: science moves in cycles of enthusiasm and skepticism. Each generation rediscovers that true progress requires a marriage between creative vision and mathematical clarity.

1.2. AI Winters and the Deep Learning Revolution

The collapse of optimism after the perceptron marked the first "AI winter" of the 1970s. Funding dried up, and public interest waned. Limited computing power, scarce data, and inflated promises led many to dismiss neural networks as a dead end. A second AI winter followed in the late 1980s, as symbolic methods, once thought to be the future of AI, also struggled to deliver.

Yet beneath the surface, mathematics was preparing a renaissance. In the 1980s, the backpropagation algorithm was formalized and popularized, allowing multilayer perceptrons to model complex nonlinear functions. Still, adoption was slow, because hardware had not yet caught up with theory. Neural networks were powerful on paper, but impractical in real-world applications.

The deep learning explosion of the 2010s changed everything. With the rise of GPUs, massive datasets, and architectures such as convolutional and recurrent networks, machines suddenly outperformed classical methods in vision, language, and speech. Soon after, transformers redefined the field altogether, enabling large-scale models that blurred the line between statistics and creativity. At the core of these breakthroughs was not magic, but mathematics: linear algebra for representation, probability for modeling uncertainty, and optimization theory for training vast networks.

The history of neural networks is therefore not merely technical—it is also the history of human patience. What once looked like failure was in fact a pause, waiting for mathematics and technology to converge.

1.3. Why Mathematics Matters in Deep Learning

If philosophy gave us the first questions about intelligence, mathematics gave us the tools to answer them. Galileo once wrote that the universe "is written in the language of mathematics, and its characters are triangles, circles, and other geometrical figures." In the same spirit, deep learning is written in the language of vectors, matrices, and functions. Every model is a translation from the world's complexity into mathematical form, and every training process is an attempt to solve an equation that nature has posed.

Linear algebra is the grammar of representation, calculus is the machinery of change, probability is the measure of uncertainty, and optimization is the path to improvement. Without them, neural networks would be shapeless intuitions. With them, they become structured systems capable of learning patterns from the world.

Mathematics is therefore not a peripheral tool but the very skeleton of deep learning. It gives rigor to vision, coherence to creativity, and structure to intuition. Just as the Greeks once sought logic to discipline thought, we now rely on mathematics to discipline learning.

This book begins here—at the intersection of history, philosophy, and mathematics—because to understand neural networks is not merely to know how they work, but to see them as part of humanity's timeless attempt to comprehend intelligence itself.

Linear Algebra Essentials | 2.

Multivariable Calculus and Analysis

3.

4. Probability, Statistics, and Information Theory

4.1. Random variables and distributions

4.2. Expectation, variance, covariance

4.3. Gaussian and exponential families

4.4. Entropy, KL divergence, mutual information

Optimization Theory | 5.

6. Functional Analysis Foundations

6.1. Normed spaces, Banach and Hilbert spaces

6.2. Orthogonal polynomials (Hermite, Laguerre, Legendre)

6.3. Special functions (Gamma, Beta, Bessel)

6.4. Operator theory foundations for PINNs and QINNs

Part II.

Classical Neural Networks

7. The Perceptron and Linear Models

Feedforward Networks (MLPs) 8.

Convolutional Neural Networks (CNNs) 9.

10. Recurrent Neural Networks (RNNs)

11. Autoencoders and Representation Learning

12. Graph Neural Networks (GNNs)

Part III.

Neural Networks for Differential Equations

Mathematical Methods for Differential Equations 13.

13.1. Classification of ODEs and PDEs

13.2. Boundary and initial conditions

13.3. Separation of variables

13.4. Sturm–Liouville problems and orthogonal expansions

13.5. Fourier and Laplace transforms

13.6. Spectral methods (Chebyshev, Legendre)

13.7. Galerkin and Finite Element Methods (FEM)

13.8. Method of Frobenius and special functions

14. Physics-Informed Neural Networks (PINNs)

15. Inverse Physics-Informed Neural Networks (iPINNs)

16. Quantum Neural Networks (QINNs)

17. Neural Operators and DeepONets

17.1. Learning operators between function spaces

17.2. Comparison with PINNs, iPINNs, FEM

17.3. Applications in PDEs and scientific computing

Part IV.

Reinforcement Learning

Classical Reinforcement Learning 18.

Deep Reinforcement Learning 19.

Part V.

Modern Architectures

Generative Models 20.

21. Transformers and Attention Mechanisms

Part VI.

Advanced Topics

22. Optimization Beyond Gradient Descent

22.1. Variational inference

22.2. Expectation-Maximization (EM)

22.3. Federated optimization challenges

23. Mathematical Frontiers of Neural Networks

24. Meta-Learning and Transfer Learning

24.1. Few-shot learning

24.2. Pretraining and fine-tuning

24.3. Continual learning

25. Explainability and Interpretability

25.1. Saliency maps and Grad-CAM

25.2. SHAP and LIME

25.3. Interpretable PINNs and DRL policies

26. Ethical and Societal Aspects

Part VII.

Practical Implementation

Computational Frameworks 27.

27.1. PyTorch fundamentals

27.2. TensorFlow and Keras

27.3. JAX and differentiable programming

28. Efficient Training and Scaling

28.1. Hardware acceleration: GPUs, TPUs

28.2. Parallelization and distributed training

28.3. Memory-efficient backpropagation

29. Case Studies in Scientific Machine Learning

A. Mathematical Notation and Symbols

Linear Algebra Toolbox | B.

Probability Distributions C.

Special Functions (Gamma, Beta, Bessel, etc.) D.

E. Implementations in PyTorch and TensorFlow