

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

Contents	v
I. MATHEMATICAL PRELIMINARIES	1
1. Introduction	3
1.1. Historical Context of AI and Neural Networks	3
1.2. AI Winters and the Deep Learning Revolution	3
1.3. Why Mathematics Matters in Deep Learning	4
2. Linear Algebra Essentials	5
2.1. Vector spaces and inner products	5
2.2. Eigenvalues, eigenvectors, diagonalization	5
2.3. Singular value decomposition (SVD)	5
2.4. Tensor notation and operations	5
3. Multivariable Calculus and Analysis	7
3.1. Gradients, Jacobians, Hessians	7
3.2. Taylor expansions in multiple variables	7
3.3. Divergence, curl, and Laplacians	7
3.4. Variational principles	7
4. Probability, Statistics, and Information Theory	9
4.1. Random variables and distributions	9
4.2. Expectation, variance, covariance	9
4.3. Gaussian and exponential families	9
4.4. Entropy, KL divergence, mutual information	9
5. Optimization Theory	11
5.1. Convexity, duality, and Lagrangians	11
5.2. Gradient descent and its variants	11
5.3. Stochastic optimization and convergence	11
5.4. Newton and quasi-Newton methods	11
6. Functional Analysis Foundations	13
6.1. Normed spaces, Banach and Hilbert spaces	13
6.2. Orthogonal polynomials (Hermite, Laguerre, Legendre)	13
6.3. Special functions (Gamma, Beta, Bessel)	13
6.4. Operator theory foundations for PINNs and QINNs	13
II. CLASSICAL NEURAL NETWORKS	15
7. The Perceptron and Linear Models	17
7.1. McCulloch–Pitts neurons	17
7.2. Rosenblatt’s perceptron and linear separability	17
7.3. Logistic regression as probabilistic perceptron	17
8. Feedforward Networks (MLPs)	19
8.1. Activation functions (ReLU, sigmoid, tanh, GELU, softmax)	19
8.2. Universal Approximation Theorem	19
8.3. Forward and backward propagation	19
8.4. Vanishing and exploding gradients	19

9. Convolutional Neural Networks (CNNs)	21
9.1. Mathematical basis of convolution	21
9.2. Feature maps, receptive fields, pooling	21
9.3. Modern CNN architectures (AlexNet, VGG, ResNet)	21
9.4. Applications in vision, audio, and physics	21
10. Recurrent Neural Networks (RNNs)	23
10.1. Sequences as dynamical systems	23
10.2. Gradient vanishing and exploding	23
10.3. LSTMs and GRUs	23
10.4. Applications in NLP, speech, time-series	23
11. Autoencoders and Representation Learning	25
11.1. Linear autoencoders and PCA	25
11.2. Nonlinear autoencoders	25
11.3. Variational Autoencoders (VAEs)	25
11.4. Latent space geometry	25
12. Graph Neural Networks (GNNs)	27
12.1. Graph Laplacians and spectral methods	27
12.2. Message passing frameworks	27
12.3. Applications in chemistry, materials, biology	27
III. NEURAL NETWORKS FOR DIFFERENTIAL EQUATIONS	29
13. Mathematical Methods for Differential Equations	31
13.1. Classification of ODEs and PDEs	31
13.2. Boundary and initial conditions	31
13.3. Separation of variables	31
13.4. Sturm–Liouville problems and orthogonal expansions	31
13.5. Fourier and Laplace transforms	31
13.6. Spectral methods (Chebyshev, Legendre)	31
13.7. Galerkin and Finite Element Methods (FEM)	31
13.8. Method of Frobenius and special functions	31
14. Physics-Informed Neural Networks (PINNs)	33
14.1. Embedding PDEs into loss functions	33
14.2. Collocation and weak formulations	33
14.3. Elliptic, parabolic, and hyperbolic PDEs	33
14.4. Applications: fluids, electromagnetism, quantum mechanics	33
14.5. Extensions: XPINNs, VPINNs, Bayesian PINNs	33
14.6. Inverse problems (intro to iPINNs)	33
15. Inverse Physics-Informed Neural Networks (iPINNs)	35
15.1. Motivation: the role of inverse problems	35
15.2. Formulation with unknown coefficients	35
15.3. Loss functions for parameter estimation	35
15.4. Applications: heat, waves, Schrödinger, materials	35
15.5. Ill-posedness and regularization	35
15.6. Sensitivity to noise and data quality	35
16. Quantum Neural Networks (QINNs)	37
16.1. Hilbert spaces and Dirac notation	37
16.2. Quantum perceptron and gates as layers	37
16.3. Variational quantum circuits (VQE, QAOA)	37
16.4. Quantum Boltzmann Machines, QCNNs, Quantum Reservoirs	37
16.5. Parameter-shift rule for gradients	37

16.6. Challenges: barren plateaus, NISQ hardware	37
16.7. Applications in optimization, chemistry, cryptography	37
17. Neural Operators and DeepONets	39
17.1. Learning operators between function spaces	39
17.2. Comparison with PINNs, iPINNs, FEM	39
17.3. Applications in PDEs and scientific computing	39
IV. REINFORCEMENT LEARNING	41
18. Classical Reinforcement Learning	43
18.1. Agents, environments, states, actions, rewards	43
18.2. Markov Decision Processes (MDPs)	43
18.3. Value functions and Bellman equations	43
18.4. Tabular methods: SARSA, Q-learning	43
19. Deep Reinforcement Learning	45
19.1. Deep Q-Networks (DQN)	45
19.2. Policy gradient methods (REINFORCE, PPO)	45
19.3. Actor-Critic architectures (A2C, A3C)	45
19.4. Landmark systems: AlphaGo, AlphaZero, MuZero	45
V. MODERN ARCHITECTURES	47
20. Generative Models	49
20.1. GANs and minimax optimization	49
20.2. Wasserstein GANs, StyleGAN	49
20.3. Diffusion models and stochastic processes	49
20.4. Applications in synthesis and design	49
21. Transformers and Attention Mechanisms	51
21.1. Self-attention: queries, keys, values	51
21.2. Multi-head attention	51
21.3. Positional encodings	51
21.4. Transformer architectures: BERT, GPT, multimodal	51
21.5. Applications in PDEs and symbolic regression	51
VI. ADVANCED TOPICS	53
22. Optimization Beyond Gradient Descent	55
22.1. Variational inference	55
22.2. Expectation-Maximization (EM)	55
22.3. Federated optimization challenges	55
23. Mathematical Frontiers of Neural Networks	57
23.1. Neural Tangent Kernels (NTK)	57
23.2. Infinite-width limits and mean-field theory	57
23.3. Geometry of loss landscapes	57
23.4. Generalization bounds and capacity	57
24. Meta-Learning and Transfer Learning	59
24.1. Few-shot learning	59
24.2. Pretraining and fine-tuning	59
24.3. Continual learning	59

25. Explainability and Interpretability	61
25.1. Saliency maps and Grad-CAM	61
25.2. SHAP and LIME	61
25.3. Interpretable PINNs and DRL policies	61
26. Ethical and Societal Aspects	63
26.1. Bias and fairness in AI	63
26.2. Privacy and security	63
26.3. AI regulation and governance	63
VII. PRACTICAL IMPLEMENTATION	65
27. Computational Frameworks	67
27.1. PyTorch fundamentals	67
27.2. TensorFlow and Keras	67
27.3. JAX and differentiable programming	67
28. Efficient Training and Scaling	69
28.1. Hardware acceleration: GPUs, TPUs	69
28.2. Parallelization and distributed training	69
28.3. Memory-efficient backpropagation	69
29. Case Studies in Scientific Machine Learning	71
29.1. Navier–Stokes with PINNs	71
29.2. QINNs for quantum chemistry	71
29.3. DRL for robotics and control	71
29.4. CNNs/GNNs for materials science	71
A. Mathematical Notation and Symbols	73
B. Linear Algebra Toolbox	75
C. Probability Distributions	77
D. Special Functions (Gamma, Beta, Bessel, etc.)	79
E. Implementations in PyTorch and TensorFlow	81

Part I.

**MATHEMATICAL
PRELIMINARIES**

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.

2.1. Vector spaces and inner products

2.2. Eigenvalues, eigenvectors, diagonalization

2.3. Singular value decomposition (SVD)

2.4. Tensor notation and operations

Multivariable Calculus and Analysis

3.

- 3.1. Gradients, Jacobians, Hessians
- 3.2. Taylor expansions in multiple variables
- 3.3. Divergence, curl, and Laplacians
- 3.4. Variational principles

Probability, Statistics, and Information Theory

4.

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.

5.1. Convexity, duality, and Lagrangians

5.2. Gradient descent and its variants

5.3. Stochastic optimization and convergence

5.4. Newton and quasi-Newton methods

Functional Analysis Foundations

6.

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**

The Perceptron and Linear Models

7.

- 7.1. McCulloch–Pitts neurons
- 7.2. Rosenblatt's perceptron and linear separability
- 7.3. Logistic regression as probabilistic perceptron

Feedforward Networks (MLPs)

8.

- 8.1. Activation functions (ReLU, sigmoid, tanh, GELU, softmax)
- 8.2. Universal Approximation Theorem
- 8.3. Forward and backward propagation
- 8.4. Vanishing and exploding gradients

Convolutional Neural Networks (CNNs) | 9.

9.1. Mathematical basis of convolution

9.2. Feature maps, receptive fields, pooling

9.3. Modern CNN architectures (AlexNet, VGG, ResNet)

9.4. Applications in vision, audio, and physics

Recurrent Neural Networks (RNNs)

10.

10.1. Sequences as dynamical systems

10.2. Gradient vanishing and exploding

10.3. LSTMs and GRUs

10.4. Applications in NLP, speech, time-series

Autoencoders and Representation Learning

11.

11.1. Linear autoencoders and PCA

11.2. Nonlinear autoencoders

11.3. Variational Autoencoders (VAEs)

11.4. Latent space geometry

Graph Neural Networks (GNNs)

12.

12.1. Graph Laplacians and spectral methods

12.2. Message passing frameworks

12.3. Applications in chemistry, materials,
biology

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

- 14.1. Embedding PDEs into loss functions**
- 14.2. Collocation and weak formulations**
- 14.3. Elliptic, parabolic, and hyperbolic PDEs**
- 14.4. Applications: fluids, electromagnetism,
quantum mechanics**
- 14.5. Extensions: XPINNs, VPINNs, Bayesian
PINNs**
- 14.6. Inverse problems (intro to iPINNs)**

Inverse Physics-Informed Neural Networks (iPINNs)

15.

- 15.1. Motivation: the role of inverse problems**
- 15.2. Formulation with unknown coefficients**
- 15.3. Loss functions for parameter estimation**
- 15.4. Applications: heat, waves, Schrödinger,
materials**
- 15.5. Ill-posedness and regularization**
- 15.6. Sensitivity to noise and data quality**

- 16.1. Hilbert spaces and Dirac notation**
- 16.2. Quantum perceptron and gates as layers**
- 16.3. Variational quantum circuits (VQE, QAOA)**
- 16.4. Quantum Boltzmann Machines, QCNNs, Quantum Reservoirs**
- 16.5. Parameter-shift rule for gradients**
- 16.6. Challenges: barren plateaus, NISQ hardware**
- 16.7. Applications in optimization, chemistry, cryptography**

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.

18.1. Agents, environments, states, actions, rewards

18.2. Markov Decision Processes (MDPs)

18.3. Value functions and Bellman equations

18.4. Tabular methods: SARSA, Q-learning

19.1. Deep Q-Networks (DQN)

19.2. Policy gradient methods (REINFORCE, PPO)

19.3. Actor–Critic architectures (A2C, A3C)

19.4. Landmark systems: AlphaGo, AlphaZero, MuZero

Part V.

MODERN ARCHITECTURES

20.1. GANs and minimax optimization

20.2. Wasserstein GANs, StyleGAN

20.3. Diffusion models and stochastic processes

20.4. Applications in synthesis and design

Transformers and Attention Mechanisms

21.

21.1. Self-attention: queries, keys, values

21.2. Multi-head attention

21.3. Positional encodings

**21.4. Transformer architectures: BERT, GPT,
multimodal**

**21.5. Applications in PDEs and symbolic
regression**

Part VI.

ADVANCED TOPICS

Optimization Beyond Gradient Descent

22.

22.1. Variational inference

22.2. Expectation-Maximization (EM)

22.3. Federated optimization challenges

23.1. Neural Tangent Kernels (NTK)

**23.2. Infinite-width limits and mean-field
theory**

23.3. Geometry of loss landscapes

23.4. Generalization bounds and capacity

Meta-Learning and Transfer Learning

24.

24.1. Few-shot learning

24.2. Pretraining and fine-tuning

24.3. Continual learning

25.1. Saliency maps and Grad-CAM

25.2. SHAP and LIME

25.3. Interpretable PINNs and DRL policies

26.1. Bias and fairness in AI

26.2. Privacy and security

26.3. AI regulation and governance

Part VII.

PRACTICAL IMPLEMENTATION

27.1. PyTorch fundamentals

27.2. TensorFlow and Keras

27.3. JAX and differentiable programming

28.1. Hardware acceleration: GPUs, TPUs

28.2. Parallelization and distributed training

28.3. Memory-efficient backpropagation

29.1. Navier–Stokes with PINNs

29.2. QINNs for quantum chemistry

29.3. DRL for robotics and control

29.4. CNNs/GNNs for materials science

Mathematical Notation and Symbols

A.

Linear Algebra Toolbox

B.

Probability Distributions

C.

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

D.

Implementations in PyTorch and TensorFlow

E.

