Modern Materials Informatics: From Foundations to Generative AI

Textbook Proposal & Outline

February 12, 2026

Proposal Overview

Key Differentiation Points

- **Complete Beginner Support:** Starts with a review of high school mathematics, unlike existing textbooks that often assume university-level mathematical background.
- **Balanced Structure:** Places equal weight on Crystal Structure Analysis and Property Prediction, whereas existing resources tend to skew heavily towards property prediction.
- **Modern Tech Stack:** Designed from the ground up for collaboration with LLM/AI, incorporating Generative AI usage methods throughout the curriculum.
- **Real Project-Based:** Features multiple functional, integrated case studies rather than abstract or isolated examples.
- **Continuous Update Strategy:** Leverages digital publishing (e.g., Kindle) for easy revisions, enabling potential updates every 2-3 months.

Overall Structure

- **Total Chapters:** 11 Chapters
- **Estimated Length:** Equivalent to 400-500 pages

Contents

Part I

Foundations and Preparation (Theory 20% / Implementation 80%)

Chapter 1

From High School Math to Machine Learning (40 pages)

1.1 Essential Mathematics Overview

This section provides a bridge from high school mathematics to the foundational concepts required for machine learning and materials informatics. We purposefully avoid excessive rigor and instead focus on intuitive understanding and practical application.

Vector & Matrix Operations

We begin with the basics of linear algebra, focusing on vectors and matrices as data structures. The concept of dimensions and spaces is introduced visually. Key operations such as dot products, matrix multiplication, and transposition are explained not just mathematically, but also through their computational implementation using NumPy. This ensures that readers can immediately relate mathematical notation to code.

Calculus

Calculus is revisited with a singular goal: understanding optimization. We focus on derivatives and partial derivatives, explaining their physical meaning as "rates of change" or "slopes." This knowledge is directly applied to the concept of Gradient Descent, which is the engine behind training neural networks. We visualize how finding the minimum of a function relates to minimizing error in a model.

Probability Distributions

The section concludes with probability and statistics, essential for dealing with the inherent noise and uncertainty in material data. We cover the Normal (Gaussian) distribution as a fundamental tool and introduce the basics of Bayesian statistics to help readers understand how to update beliefs (models) based on new evidence (data).

1.2 Chemistry Basics Review

To ensure this book is self-contained, we review key concepts from chemistry and solid-state physics that are prerequisites for materials informatics.

Atoms, Molecules, and Crystal Structures

We start from the atomic scale, discussing bonding types (ionic, covalent, metallic) and how they influence material properties. The concept of the "Unit Cell" is introduced as the repeating building block of a crystal. We explain Lattice Constants $(a, b, c, \alpha, \beta, \gamma)$ and how they define the geometry of the crystal.

Miller Indices

We explain Miller Indices (hkl) as a standard notation for describing crystallographic planes and directions. Understanding this is crucial for interpreting X-ray diffraction patterns and modeling surfaces later in the book.

1.3 What is Materials Informatics?

Materials Informatics (MI) is the application of data science and machine learning techniques to the discovery and development of new materials. This field aims to accelerate the traditional trial-and-error process of experimentation.

History and Context

We trace the evolution of MI, highlighting the U.S. Materials Genome Initiative (MGI) as a catalyst for global research. We also discuss how the paradigm has shifted from "observation" to "rational design."

Industry Examples

We present real-world success stories, such as Toyota's "WAVEBASE," to demonstrate how data-driven approaches are being adopted in industry to solve complex problems like battery degradation and catalyst efficiency.

Practical Workflow

The chapter outlines the standard workflow of an MI project:

Data Collection → Descriptor Generation → Machine Learning → Bayesian Optimization → Validation Experi

This framework serves as a mental map for the rest of the book.

1.4 Learning Path in This Book

We provide a roadmap for the reader. The estimated completion time is 4-6 months for self-study. A dependency map illustrates which chapters are prerequisites for others, allowing readers to customize their learning path based on their interests (e.g., focusing on Deep Learning vs. Classical ML).

Implementation Focus: We emphasize that this book is "code-first." The first chapter concludes with hands-on exercises in NumPy and Pandas to warm up the reader's programming skills.

Chapter 2

Python Environment and Tools (30 pages)

2.1 Installation

We guide the reader through setting up a robust Python development environment. We recommend Python 3.9 or higher and cover installation on Windows (via Anaconda or WSL), macOS, and Linux. The importance of virtual environments (conda or venv) is emphasized to prevent dependency conflicts.

2.2 Essential Libraries

We introduce the "Standard Stack" of libraries used in Materials Informatics:

- **Data Processing:** NumPy for numerical computation and Pandas for tabular data manipulation.
- **Machine Learning:** scikit-learn for traditional algorithms and PyTorch for deep learning.
- **Materials Analysis:** pymatgen (Python Materials Genomics) for handling crystal structures and PySCF for quantum chemistry calculations.
- **Visualization:** Matplotlib and Seaborn for static plots, and Plotly for interactive 3D crystal visualizations.

2.3 Google Colab Introduction (Recommended)

For beginners or those without powerful local hardware, we introduce Google Colab. We explain how to leverage free GPU and TPU resources to accelerate training of deep learning models. We also cover how to mount Google Drive to save datasets and models persistently.

2.4 Jupyter Notebook Basics

Since Jupyter Notebooks are the de facto standard for data science experimentation, we provide a crash course. Topics include cell types (Code vs. Markdown), keyboard shortcuts, and magic

commands (e.g., `%timeit`).

2.5 Code Management

We introduce Version Control as a critical skill for reproducibility.

GitHub Basics

We cover the basics of Git and GitHub: creating a repository, cloning, staging changes, committing, and pushing code. We encourage readers to maintain their own portfolio of MI projects on GitHub as they progress through the book.

Implementation: The chapter ends with a "Hello World" style setup script that verifies all libraries are correctly installed and imports a crystal structure from the Materials Project database to test connectivity.

Part II

Crystal Structure Analysis & Image Processing (Theory 40% / Implementation 60%)

Chapter 3

Crystal Structure Representation and pymatgen Basics (45 pages)

3.1 Crystal Structure File Formats

Data formats are the language of Materials Informatics. We examine the structure of the essential file types:

- **CIF (Crystallographic Information File):** The industry standard for sharing crystal structures. We parse its key blocks (cell parameters, symmetry, atomic positions).
- **POSCAR:** The input format for VASP, the most widely used DFT code. Understanding this is crucial for first-principles datasets.
- **XYZ & JSON:** Simple formats often used for molecules or machine learning datasets.

3.2 pymatgen Basics

We introduce `pymatgen` (Python Materials Genomics) as the core library for structure manipulation.

Core Objects

We define the `Structure`, `Lattice`, and `Site` objects. Readers learn how to instantiate a crystal structure from scratch code and how to calculate basic properties like density and volume.

Structure Manipulation

We demonstrate how to generate supercells (expanding the lattice) and creating slab models (surfaces) from bulk crystals, which are essential for surface science and catalysis studies.

3.3 Data Acquisition from Materials Project

The Materials Project (MP) is the largest open database of calculated material properties.

MP API (MPRester)

We walk through obtaining an API key and using the `MPRester` class to programmatically query the database.

Querying Strategy

Readers learn how to filter the 150,000+ entries by criteria such as elements (e.g., "Li-Fe-O"), stability (energy above hull), and band gap. We discuss best practices for handling large datasets and API rate limits.

3.4 Local Environment Analysis

Understanding the local neighborhood of an atom is more predictive than global structure.

coordination Number & Polyhedra

We define coordination numbers and use algorithms to automatically detect them.

CrystalNN

We introduce the `CrystalNN` class in pymatgen, a robust algorithm for finding nearest neighbors and characterizing local geometries (e.g., distinguishing between tetrahedral and octahedral sites).

3.5 Dimensionality Assessment (0D/1D/2D/3D)

Not all materials are 3D bulk solids. We classify materials into 0D (molecules), 1D (chains), 2D (layers like graphene), and 3D (bulk). We implement functions using pymatgen's dimensionality analysis tools to screen for 2D materials, which are hot candidates for electronics.

Implementation: The chapter culminates in a project where readers download 100 structures from MP, filter them by dimensionality, and visualize their unit cells in 3D using Plotly.

Chapter 4

X-ray Diffraction and Electron Microscopy Image Analysis (50 pages)

4.1 Principles of X-ray Diffraction and Experiments

X-ray Diffraction (XRD) is the primary method for identifying crystal structures. We explain Bragg's Law ($n\lambda = 2d\sin\theta$) and how constructive interference creates diffraction patterns. We touch upon the concepts of Scattering Factors and Structure Factors to understand peak intensities.

4.2 Calculation of Diffraction Patterns

We bridge theory and code by calculating theoretical XRD patterns from known crystal structures.

Simulating XRD with pymatgen

Using the `XRDCalculator`, we generate patterns for ideal crystals. We study how specific structural changes (e.g., lattice expansion) shift the peak positions (Bragg angles).

4.3 Image Processing Basics

Before analyzing complex electron microscopy data, we cover fundamental image processing techniques using OpenCV.

- **Preprocessing:** Noise removal using Gaussian and Median filters.
- **Normalization:** Rescaling pixel intensities (Min-Max, Z-score) for neural network inputs.
- **Feature Extraction:** Simple edge detection to identify grain boundaries.

4.4 Traditional Methods: Rietveld Analysis

We provide a high-level overview of Rietveld refinement, the standard method for fitting a theoretical profile to experimental data to refine structure parameters. While we don't implement a full Rietveld code, understanding its "profile matching" logic is crucial for understanding the AI approaches that seek to replace or augment it.

4.5 Analysis Assistance with LLM

We demonstrate how Generative AI can assist in the analysis workflow.

Parameter Extraction

We build a script using an LLM to automatically parse scientific papers and extract experimental XRD parameters (e.g., wavelength, scan rate).

Code Generation

We show effective prompts to generate Python scripts for specific diffraction calculations, speeding up the implementation process.

Chapter 5

Crystal Structure Estimation via Deep Learning (55 pages)

5.1 Foundations of CNN (Convolutional Neural Networks)

We introduce CNNs, the powerhouse of computer vision, tailored for materials scientists.

Architecture

We explain the role of Convolutional Layers (feature detection), Pooling Layers (spatial down-sampling), and Activation Functions (ReLU for non-linearity).

Optimizing Learning

We explain Backpropagation and the danger of Overfitting, introducing techniques like Dropout and Data Augmentation as countermeasures.

5.2 Crystal Orientation Estimation from Electron Diffraction

We focus on a concrete problem: determining the orientation of a crystal from its electron backscatter diffraction (EBSD) or transmission electron microscopy (TEM) image.

The Task

The goal is to map a 2D diffraction image to a 3D Euler angle or rotation matrix. We cite recent successes achieving $R^2 \approx 0.95$.

Dataset Construction

Since experimental labeling is expensive, we use a "Synth-to-Real" approach, training on simulated diffraction patterns generated in Chapter 4.

5.3 Implementation with PyTorch

We build a complete Deep Learning pipeline.

- **Model Definition:** Subclassing `torch.nn.Module` to define a simple CNN.
- **Training Loop:** Writing the standard PyTorch boilerplate: `forward` pass, `loss` calculation, `backward` pass, and `optimizer.step()`.

5.4 Transfer Learning and Fine-tuning

Training from scratch requires massive data. We demonstrate Transfer Learning: taking a model pre-trained on ImageNet (like ResNet18) and adapting it to diffraction images. We show how this technique yields high accuracy even with a small dataset (e.g., $<$500 images).

5.5 Model Evaluation and Uncertainty Quantification

We don't just inspect the loss curve. We verify performance on a held-out test set using metrics like RMSE for regression tasks. We also discuss how to estimate the reliability of the model's prediction, a step often overlooked but critical in experimental science.

Part III

Property Prediction & Machine Learning (Theory 50% / Implementation 50%)

Chapter 6

Descriptor Design and Feature Extraction (50 pages)

6.1 Hierarchy of Descriptors

"Garbage in, garbage out" is the rule of Machine Learning. To feed materials into an algorithm, we must convert them into vectors of numbers called descriptors or features. We categorize them into three levels:

1. **Low-level:** Compositional features (e.g., average electronegativity) that ignore geometry.
2. **Mid-level:** Structural features (e.g., density, packing fraction).
3. **High-level:** Encoded identifiers of the local atomic environment or learned representations from deep learning.

6.2 Composition-based Descriptors

For many tasks, the chemical formula alone carries 80% of the signal. We learn to construct vectors based on periodic table properties.

Feature Generation

We calculate statistics (mean, max, min, variance) for properties like Atomic Radius, Electronegativity, and Valence Engineers.

One-hot Encoding

We explain how to handle categorical data (like "Element Name") using One-hot encoding, though we discuss its limitations in high-dimensional spaces.

6.3 Structure-based Descriptors

When structure is known (e.g., from XRD), we can use it to improve predictions significantly.

- **Density:** A simple yet powerful predictor for properties like Band Gap and Melting Point.
- **Radial Distribution Function (RDF):** A statistical description of interatomic distances.

6.4 Local Environment Descriptors (Conceptual Introduction)

We introduce advanced descriptors that capture the nuanced geometry around an atom. We discuss the conceptual basis of SOAP (Smooth Overlap of Atomic Positions), which allows comparing atomic environments invariant to rotation and translation.

6.5 Automated Feature Extraction with XenonPy

Instead of hand-crafting features, we use libraries like XenonPy that contain pre-built libraries of thousands of compositional descriptors.

Transfer Learning in Features: We discuss how descriptors learned on one dataset can be transferred to another.

6.6 Feature Selection and Dimensionality Reduction

Too many features lead to the "Curse of Dimensionality."

Selection Strategies

We implement correlation analysis to remove redundant features.

PCA (Principal Component Analysis)

We implement PCA to project high-dimensional material data into 2D plots, allowing us to visually cluster materials (e.g., separating metals from insulators).

Chapter 7

Basics of Machine Learning Model Construction (60 pages)

7.1 Supervised Learning Workflow

We define the rigorous workflow required for publication-quality MI research.

Data Splitting

We emphasize the 60/20/20 split (Train/Validation/Test) and why "testing on training data" is the cardinal sin of ML.

7.2 Linear Regression and Regularization

We start simple with Linear Regression.

Regularization (L1/L2)

We explain how Ridge (L2) and Lasso (L1) regression prevent overfitting and, in the case of Lasso, automatically select relevant physical features.

7.3 Non-linear Models

Most material properties are non-linear.

- **Decision Trees:** Easy to interpret but prone to overfitting.
- **Random Forest:** The "Switzerland" of MI algorithms—robust, requires little tuning, and often yields excellent results.
- **Gradient Boosting:** XGBoost and LightGBM for squeezing out the last % of performance in competitions.

7.4 Support Vector Machines (SVM)

We cover SVMs for tasks with smaller datasets, explaining the Kernel Trick as a way to find linear boundaries in high-dimensional feature spaces.

7.5 Model Evaluation Metrics

How do we know a model is "good"?

- **Regression:** RMSE (Root Mean Squared Error) vs MAE (Mean Absolute Error). We explain when to use which.
- **Classification:** Accuracy is often misleading in imbalanced materials data (imbalanced classes). We introduce Precision, Recall, and F1-score.

7.6 Hyperparameter Optimization

We move beyond manual tuning. We implement Grid Search and Random Search to automatically find the best parameters (e.g., number of trees in a Random Forest). We use Learning Curves to diagnose Bias (underfitting) vs Variance (overfitting).

Chapter 8

Property Prediction with Deep Learning (60 pages)

8.1 Basics of Neural Networks

We transition from classical ML to Deep Learning.

The Neuron

We build up from the single perceptron to the Multi-Layer Perceptron (MLP).

Activation Functions

We visualize what ReLU, Tanh, and Sigmoid do to the signal, and why ReLU is the default for modern hidden layers.

8.2 Implementation with PyTorch

We provide a template for a generic Property Predictor in PyTorch.

- **Architecture:** Stacking `nn.Linear` tailored to the input descriptor size.
- **The Loop:** A reusable training loop function that handles batching and logging.

8.3 Property Prediction Tasks

We define three core challenges for the reader to solve:

1. **Band Gap:** A regression task critical for solar/electronics. Target Mean Absolute Error (MAE) < 0.5 eV.
2. **Density:** A "sanity check" task where physics should make prediction easy.
3. **Melting Point:** A complex property that challenges simple models.

8.4 Regularization Techniques and Early Stopping

Deep networks easily memorize small datasets. We introduce:

- **Dropout:** Randomly killing neurons to force redundancy.
- **Early Stopping:** Monitoring the validation loss and stopping training before the model begins to overfit.

8.5 Transfer Learning and Pre-trained Models

We explore the strategy of "Pre-training." We explain how a model can learn the "rules of chemistry" from a massive dataset (like OQMD or Materials Project) and then be fine-tuned on a small experimental dataset (e.g., 100 lab values).

8.6 Validation Loop with Experiments

We close with the philosophy of "The Loop." A model is useless if it lives only in a notebook. We describe the cycle of Prediction $\rightarrow$ Experimental Verification $\rightarrow$ Feedback, which is the heart of the Materials Informatics revolution.

Part IV

Applications, Optimization, and Real Projects (Theory 40% / Implementation 60%)

Chapter 9

Uncertainty Quantification and Bayesian Optimization (50 pages)

9.1 Uncertainty in Machine Learning Models

Standard ML models are often "confidently wrong." In materials discovery, knowing *what you don't know* is as important as the prediction itself.

1. **Aleatoric Uncertainty:** Noise inherent in the data (e.g., experimental error).
2. **Epistemic Uncertainty:** Uncertainty due to lack of data (model ignorance).

We implement Ensemble Methods (training 5+ models with different seeds) to estimate variance as a proxy for uncertainty.

9.2 Gaussian Process Regression (GP)

The gold standard for uncertainty. We visualize how GPs fit a function not as a single line, but as a "cloud" of possibilities.

Kernel Functions

We explain how the choice of kernel (RBF, Matérn) dictates the "smoothness" and "similarity" assumptions of the chemical space.

9.3 Basics of Bayesian Optimization

Experimental materials science is expensive. Bayesian Optimization (BO) solves the problem: "Where should I sample next to find the best material in the fewest steps?"

The Surrogate Model

Using a GP to approximate the expensive objective function (e.g., DFT calculation or Synthesis).

Acquisition Functions

We define the strategies for selecting the next experiment:

- **Exploitation:** Going where the model predicts high performance (High Mean).
- **Exploration:** Going where the model is uncertain (High Variance).
- **Expected Improvement (EI):** The mathematical balance of the two.

9.4 Application to Efficient Experimental Planning

We simulate a "Virtual Lab." The reader starts with a dataset of 10 points and uses BO to find the material with the highest Melting Point in a dataset of 1000, typically finding the optimum in <20 iterations compared to >100 for random search.

9.5 Automated Parameter Proposal by LLM

We introduce a hybrid approach:

1. Use an LLM to scan the literature for reasonable bounds of synthesis parameters (e.g., "Sintering temperature for excess Li").
2. Feed these bounds as priors into the Bayesian Optimization loop.

Chapter 10

Collaborative Style with Generative AI (LLM) (40 pages)

10.1 Materials Informatics in the AI Era

The release of ChatGPT changed the coding landscape. We discuss the shift from "writing code" to "reviewing code" and "architecting systems."

10.2 Practical Prompt Engineering

We provide a library of prompts specifically for Materials Science:

- "Act as a senior DFT expert and explain the convergence failure in this VASP OUTCAR."
- "Convert this IUPAC name to a SMILES string and check for validity."

10.3 Automation of Literature and Database Search

Semantic Scholar API

We build a tool that takes a keyword (e.g., "Solid State Battery"), searches the latest 100 papers using the API, and uses GPT-4 to generate a tabular summary of Cathode Materials mentioned.

10.4 Generation of Material Design Proposals

Can AI invent materials?

Generative Chemistry

We experiment with asking LLMs to modify existing crystal structures (e.g., "Substitute half the Co with high-valence doping elements") and validate the chemical feasibility (Charge neutrality, Tolerance factor) using simple Python scripts.

10.5 Limitations and Reliability of LLM

We strictly warn against "Hallucinations." An LLM might invent a reference or a physical constant. We teach the pattern of "LLM Propose, Physics Verify," where the LLM is the creative engine and a physics simulator (or calculator) is the ground truth.

10.6 Ethics, IP, and Security

We cover the basics of data privacy when using cloud-based LLMs in an R&D context, advising on when to use local open-source models (like Llama 3) versus public APIs.

Chapter 11

Integrated Project Case Studies (70 pages)

11.1 Case Study 1: Band Gap Prediction for Li-ion Battery Cathode Materials

A complete end-to-end project.

- **Goal:** Find high-voltage cathodes with appropriate electronic conductivity (correlated with Band Gap).
- **Data:** We pull 1,000 transition metal oxides from Materials Project.
- **Feature Engineering:** We create a hybrid vector of "Magpie" (elemental) and "Voronoi" (structural) features.
- **Model:** We compare Random Forest (interpretable) vs Neural Networks.
- **Outcome:** We identify 5 candidate materials that are not in the training set but predicted to have optimal properties.

11.2 Case Study 2: Stability Evaluation of Perovskite Solar Cells

A more advanced, multi-modal project.

- **Input:** Simulated Electron Diffraction images.
- **Task:** Classification (Is it the stable alpha-phase or the unstable delta-phase?).
- **Modeling:** We fine-tune a ResNet-18 vision model.
- **Integration:** We combine the visual classification with thermodynamic stability data to create a "Stability Score" map.

11.3 Collaboration Flow with Experiments

We diagram the "Human-in-the-Loop" workflow.

1. Computationalist trains model on existing database.

2. Model proposes 5 experiments.
3. Experimentalist synthesizes materials.
4. Results (Success/Failure) are fed back to retrain the model.

11.4 Future Learning Areas

We conclude by pointing the reader to advanced topics:

- **First-Principles Calculation (DFT):** The source of ground truth data.
- **Graph Neural Networks (GNN):** The state-of-the-art for crystal property prediction (e.g., CGCNN, M3GNet).
- **Generative Models:** Diffusion models for inverse design.

Appendix