Absolutely! Here's a comprehensive roadmap for mastering Machine Learning (ML), Deep Learning (DL), and foundational Computer Vision (CV) concepts, tailored for someone who wants to deeply understand the math and implementation behind the scenes.

1. Mathematical Foundations

- Linear Algebra: Vectors, matrices, matrix multiplication, determinants, eigenvalues/eigenvectors
- Calculus: Derivatives, gradients, partial derivatives, chain rule (for backpropagation)
- Probability & Statistics: Distributions, mean/variance, Bayes' theorem, conditional probability
- Optimization: Gradient descent, stochastic gradient descent, loss functions

2. Python & Scientific Libraries

- Core Python: Control flow, functions, OOP basics
- NumPy & SciPy: Array operations, broadcasting, basic stats, linear algebra, manual model implementation
- Matplotlib/Seaborn: Data visualization

3. Machine Learning Fundamentals

- Data Preprocessing: Normalization, encoding, train/test split, feature selection
- Supervised Learning:
    - Linear Regression
    - Logistic Regression
    - K-Nearest Neighbors (KNN)
    - Decision Trees, Random Forests
    - Support Vector Machines (SVM)
- Unsupervised Learning:
    - K-Means Clustering
    - PCA (Principal Component Analysis)
- Model Evaluation: Cross-validation, confusion matrix, ROC/AUC, precision/recall, F1-score

Should you use scikit-learn?

YES!

- scikit-learn is fantastic for learning and experimenting with classic ML algorithms.
- It abstracts away some implementation, but you can always go back to NumPy for a “from-scratch” version if you want to see the math.
- Use both: Implement a simple model from scratch, then use scikit-learn to experiment and compare results.

4. Deep Learning

- Foundations: Perceptron, activation functions (sigmoid, ReLU, tanh), loss functions
- Neural Networks: Forward pass, backpropagation, gradient descent, weight initialization
- Frameworks:
    - PyTorch: For fine-grained control and understanding (recommended for learning)
    - Keras (TensorFlow backend): For quick prototyping and higher-level abstraction
- Practical Models:
    - Feedforward Neural Networks (MLP)
    - Convolutional Neural Networks (CNN)
    - Recurrent Neural Networks (RNN, LSTM, GRU) – for sequence data
- Training Tricks: Batch normalization, dropout, learning rate scheduling, regularization

5. Computer Vision (with OpenCV)

- Image Basics: Pixels, channels, color spaces (RGB, BGR, Grayscale)
- Image Operations: Resizing, cropping, rotation, flipping, filtering (blur, edge detection)
- Drawing & Annotation: Drawing shapes, adding text, bounding boxes
- Feature Detection: Contours, corners, histograms, thresholding
- Object Detection (basic): Haar cascades, template matching
- Integration with ML/DL: Preparing image data for models, augmentations, using OpenCV with PyTorch/Keras

6. Projects & Practice

- Mini-projects: Titanic prediction, MNIST digit recognition, basic image classification, face detection
- Kaggle Competitions: Participate to apply your skills

- OpenCV Projects: Image filters, real-time webcam manipulations, object detection, face recognition

7. Advanced Topics (Optional, for later)

- Transfer learning, fine-tuning
- GANs (Generative Adversarial Networks)
- Object detection (YOLO, SSD)
- Semantic segmentation (U-Net, Mask R-CNN)
- NLP (transformers, embeddings)

Recommended Learning Path

1. Math + Python/NumPy: Understand the theory, implement tiny models from scratch.
2. scikit-learn: Apply classic ML algorithms, focus on data handling and evaluation.
3. PyTorch/Keras: Re-implement models, build small neural networks, experiment with layers and custom loss functions.
4. OpenCV: Start with image manipulation, move to feature detection, and integrate with ML/DL models.
5. Projects: Try to combine everything by building full pipelines (e.g., image classification with OpenCV preprocessing and PyTorch model).

Resources

- Math: Khan Academy, 3Blue1Brown
- Python/NumPy: NumPy Documentation, W3Schools Python
- scikit-learn: Official Tutorials
- PyTorch: PyTorch Tutorials
- Keras: Keras Documentation
- OpenCV: OpenCV-Python Tutorials, PyImageSearch
- Projects: Kaggle, GitHub trending ML repos

Summary Table

Step	Tool/Library	Concepts/Topics	Resources
1	Python/NumPy	Math, manual model implementation	Khan Academy, 3Blue1Brown
2	scikit-learn	Classic ML, evaluation, pipelines	scikit-learn tutorials
3	PyTorch/Keras	DL, custom models, training tricks	PyTorch/Keras guides
4	OpenCV	Image ops, drawing, feature detection	OpenCV docs, PyImageSearch
5	Projects	Real datasets, end-to-end pipelines	Kaggle, GitHub

---

TL;DR

- Use raw Python/NumPy/SciPy for math and basic implementations.
- Use scikit-learn for classic ML and practical experimentation.
- Use PyTorch/Keras for neural networks and deep learning.
- Use OpenCV for image processing and computer vision.
- Alternate between “from scratch” and using frameworks for maximum understanding.