Goal

Build **MicroNet++**: A minimalist deep learning framework in C++ that supports:

- Custom tensor class or Eigen-based backend
- Manual autograd (reverse-mode)
- Linear layers, activations
- Loss functions, SGD/Adam
- Training loop with XOR + MNIST demos
- Clean CMake-based build system
- Documented GitHub repo (resume-ready)

Suggested Frameworks & Tools

Task	Suggested Tool/Library
Math/Tensors	Eigen or xtensor
Build system	CMake
Datasets (MNIST)	Use CSVs or `libtorch::data` or OpenCV
IDE	VSCode / CLion
Testing	Catch2 or GoogleTest
Optional GPU	CUDA (later)

Week-by-Week Plan

Week 1: Tensor + Build Setup

Goal: Set up project and tensor structure.

Tasks:

- Set up CMake project
- Write or integrate a `Tensor` class:
  - Use Eigen for matrix ops
  - Add shape, broadcast support (basic)
- Implement operators: `+`, `-`, `*`, `@`, etc.

Resources:

- Eigen Tutorial: https://eigen.tuxfamily.org/dox/GettingStarted.html

- CMake Guide: https://cliutils.gitlab.io/modern-cmake/

Week 2: Autograd System

Goal: Reverse-mode automatic differentiation engine

Tasks:

- Create `Variable` class to track `.grad`, `.backward()`
- Build computation graph with nodes
- Implement backward pass for basic ops

Resources:

- Micrograd (Karpathy): C++ port inspiration
- Blog: Implementing Backprop in C++

Week 3: Layers, Activations, Loss, Optimizers

Goal: Build reusable modules and training essentials.

Tasks:

- `Module` base class
- `Linear`, `ReLU`, `Tanh`
- `MSELoss`, `CrossEntropy`
- Optimizers: `SGD`, `Adam`

Resources:

- PyTorch source (C++): https://pytorch.org/cppdocs/
- Optional: Use Pimpl idiom for clean class separation

Week 4: Training Loop + Demo Models

Goal: Train models on XOR and MNIST

Tasks:

- Write forward/backward training loop
- XOR demo: basic proof-of-concept
- MNIST loader (CSV or OpenCV)
- CLI arguments (epochs, LR, etc.)

Resources:

- MNIST CSV: CSV Format Dataset
- OpenCV C++ for image loading (if preferred)

Week 5: Polish + Presentation

Goal: Make it recruiter-friendly and showcase-ready

Tasks:

- Write clean README (architecture, how to build, run)
- Add unit tests (Catch2 / GTest)
- Host code on GitHub
- Prepare GIFs/screenshots + Colab-style notebook (optional: use Jupyter C++ kernel)

Summary of Resources

Type	Name / Link
Linear Algebra	Eigen Docs
Deep Learning	Micrograd (logic)
Autograd C++	Tinygrad C++ Fork
Build Tools	Modern CMake
Data	MNIST CSV
Tensor Libraries	xtensor
