THE DAWN OF

ERA

VERSION 4

TRANSFORMERS

ERA V4 introduces a new course structure which is **exceptional**, forward-looking and ambitious in a way that **no mainstream curriculum** is right now.

Real-World, Full-Scale LLM Training

- Training a **70B model end-to-end + instruction tuning** is unheard of in open courses - this alone will make your course legendary, especially with QAT and compute credits.

Practical CoreSet Focus

- You're not just learning about the right "datasets" - you're learning **CoreSet thinking**, which is at the bleeding edge of data efficiency.

Multi-GPU ImageNet Training

- Training from scratch on full ImageNet is *rare* even in advanced AI labs. This gives you real training and deployment experience.

Quantization Aware Training (QAT) as first-class citizen

- Covering **full QAT, not just LoRA/PEFT**, is a massive differentiator - real engineering, not shortcuts. You can now not only dream but also actually train a 100B+ parameter model!!

Balanced Inclusion of RL + VLMs + Embeddings

- We've captured **most of the modern modalities and methods**: vision, language, reward, embeddings - with deployment in mind.

We hope you'll enjoy learning in ERA V4 as much as we've loved creating it!

Here's the full course structure:

Session 1: Introduction to AI, Neural Networks and Development Tools

- What is AI? Evolution and real-world applications.
- Neural Network fundamentals: perceptrons, activations, weights, bias.
- Overview of course flow: how we go from scratch to training a 70B LLM.
- Setting up dev environment: Python, VS Code, CUDA drivers.
- Install PyTorch, WandB, Git, and use **Cursor** for coding acceleration.

Session 2: Python Essentials, Version Control, and Web Development Basics

- Python for ML: Essential Python syntax and data structures relevant to AI programming.
- Git/GitHub workflow: Basic commands, branching, merging, and collaboration workflows.
- Basic HTML/CSS/JS and Flask – serve static frontend.
- Launch a web UI to visualize model outputs early.

Session 3: PyTorch Fundamentals and AWS EC2 101

- Introduction to PyTorch and Tensors: Understanding tensors, tensor operations, and PyTorch basics.
- AutoGrad and Computational Graphs: Mechanism of automatic differentiation in PyTorch.
- Building Simple Neural Networks: Constructing basic neural networks using PyTorch.
- Implementing Training Loops: Writing loops for training and validating models.
- Spin up EC2 instance and connect via SSH.

Session 4: Building First Neural Network and Training on Cloud

- Build first MLP with PyTorch for MNIST.
- Visualize loss curves with WandB.
- Train on Colab and EC2.
- Save checkpoints and load model weights.
- Build a Flask API + frontend to display predictions.

Session 5: CNNs and Backpropagation

- Basics of CNNs: Understanding convolution operations, filters, feature maps, and receptive fields.
- Implementing CNNs in PyTorch: Building and training CNN models on image datasets.
- Backpropagation: The fundamentals of the backbone of training Neural Networks
- Architectural Basics: How do we structure a neural network together?
- Training CNNs: Techniques for effective training and avoiding overfitting.

Session 6: In-Depth Coding Practice – CNNs

- Hands-On Practice with CNNs: Extensive coding session focused on deepening understanding of CNN implementation.
- Advanced CNN Architectures: Exploring more complex CNN structures like VGG and Inception networks.
- Data Augmentation for CNNs: Applying data augmentation techniques to improve CNN performance.
- Model Evaluation and Debugging: Practical examples on how to evaluate CNNs' performance, debug issues, and fine-tune models.
- Use WeightWatcher to analyze and visualize weight distributions.

Session 7: Advanced CNN Architectures & Training

- Advanced Concepts: Image Normalization, Batch, Group & Layer Normalization, Regularization

- Regularizations: Batch Size, Early Stopping, DropOut, and L1/L2 Regularizations
- Advanced Convolutions: Pointwise, Atrous, Transpose, Pixel Shuffle, Depthwise, and Group Convolutions
- Data Augmentation: PMDAs, Elastic Distortion, CutOut, MixUp, RICAP, RMDAs and Strategy.
- Use CoreSets to reduce dataset without losing performance.
- Compare performance of full vs subset training.

Session 8: One Cycle Policy and CoreSet Training

- Larger than Life Receptive Fields: What happens when we go deeeeeeper!
- Advent of "many" receptive fields: Modern Neural Networks have "many receptive fields" 🤤
- ResNets: The "final" Convolution architecture
- Learning rate schedules, warmups, cosine decay.
- One Cycle Policy training for fast convergence.
- Use CoreSet sampling for image data – improve generalization.
- Live run: CIFAR-10 or TinyImageNet with OneCycle + CoreSets.

Session 9: Multi-GPU Training of ResNet from Scratch on Full ImageNet

- Set up DDP (Distributed Data Parallel) in PyTorch.
- Train ResNet-50 from scratch on full ImageNet.
- Use EC2 for multi-GPU training.
- Visualize training progress, speedup from parallelism.

Session 10: Introduction to Transformers and Emergent Abilities in LLMs

- Self-attention, multi-head attention, positional encodings.
- Implement transformer block from scratch.
- Vision Transformers (ViT) vs CNNs – pros and cons.
- Introduction to emergent abilities of large models (in-context learning, tool use).

Session 11: Embeddings, Tokenization, and CoreSets

- Intro to tokenization and BPE.
- Implement BPE tokenizer from scratch.
- CoreSets for text datasets – token diversity preservation.
- Embedding spaces: cosine similarity, t-SNE/UMAP visualizations.
- Prep text dataset for LLM training.

Session 12: Transformer Architectures, MHA and LLM Training

- Decoder-only architecture (GPT-style): MHA, FFN, layer norm.
- RoPE (rotary positional embedding) – why and how.
- Implement training loop with causal masking.
- Visualize attention weights and outputs with hooks.

Session 13: Optimization Techniques, RoPE, CoreSets & LLM Evaluations

- Mixed-precision training with FP16/BF16.
- Training stabilization: gradient clipping, loss scaling.
- CoreSets for long text corpora (LLM pretraining).
- LLM eval: perplexity, BLEU, TruthfulQA, MMLU.
- Use of small eval tasks to detect training divergence early.

Session 14: Full Quantization-Aware Training (not LoRA or PEFT)

- Deep dive into QAT, and not just LoRA/PEFT which is for fine-tuning a pre-trained model. We’ll learn how to pre-train a model in Quantized mode!
- QAT Implementation.
- Use WeightWatcher to monitor quantization impact.
- Set up pipeline with single A100.

Session 15: CLIP and Vision-Language Models (VLMs)

- CLIP architecture: dual encoder (ViT + Transformer).
- Implement zero-shot classification using CLIP.
- Train mini-CLIP on image-caption dataset.
- Apply contrastive loss: cosine similarity in latent space.
- Evaluate zero-shot generalization.

Session 16: Reinforcement Learning 101

- Agent, environment, state, action, reward.
- Q-learning: discrete action spaces.

- Train an agent from scratch.
- Visualize reward curves, convergence.
- Use replay buffers, epsilon-greedy policy.

Session 17: Continuous Action Spaces in RL

- Asynchronous Advantage Actor-Critic Algorithm
- Policy Optimization: T3D
- DDPG and PPO: algorithms for continuous actions.
- Train a simple driving agent from scratch.

Session 18: RLHF, GPRO and Instruction Fine-Tuning

- Reward Modeling: ranking vs scoring.
- Reinforcement Learning with Human Feedback pipeline.
- GPRO optimizer: generalized advantage optimization.
- Instruction tuning: SFT + reward modeling + PPO.
- LLM Alignment: prompt attacks, reward hacking, honest QA.

Session 19: Pretraining a 70B LLM End-to-End, followed by Instruction Tuning

- Full 70B pretraining:
    - Token budget planning, context length, gradient checkpointing.
    - Model parallelism strategy.
- Run sample batches, log loss/attention.
- Instruction-tune using cleaned dataset.
- vLLM inference deployment for optimized serving on A100.

Session 20: Capstone

- Team-based or solo full-stack AI project.
- Project must integrate:
    - Training a model (LLM)
    - Deployment (frontend/backend)
- Final demo + paper-style write-up + GitHub repo.

ERA V3 → ERA V4: What’s Changed and Why It Matters

ERA V4 is a major step forward from ERA V3 - not just in content, but in mindset. Where ERA V3 focused on building a strong foundation and helping students gain confidence in deploying AI models, **ERA V4 is designed for those ready to train, optimize, and understand AI systems at scale**.

It reflects how the field has evolved - and how our teaching needs to keep pace with what engineers actually need to know today.

1. From Small LLMs to Real Training at Scale

In V3, we introduced LLMs by showing how transformers work and walking through simple training loops with small datasets. That gave students a helpful mental model - but the real challenge of working with large models was abstracted away.

In V4, we don't just talk about large language models - we train one. Using **QAT**, students walk through the actual steps of **pretraining a 70B model**, including:

- token budgeting,
- checkpointing strategies,
- memory-efficient model parallelism, and
- post-training instruction tuning.

This shift - from understanding to doing - is one of the most meaningful changes in the course.

2. CoreSets, Not Just Data Cleaning

Data handling in V3 focused on cleaning, augmenting, and splitting - useful, but mostly standard. In V4, we introduce **CoreSet thinking**: how to represent large datasets compactly without sacrificing model performance.

Students apply CoreSets to both image and text datasets, learning how to:

- train on subsets that preserve learning quality,
- compare full-data vs CoreSet runs,
- think critically about data efficiency (not just compute efficiency).

It's a newer way to reason about scale, and it fits well with how real-world labs operate today.

3. ImageNet Training as a First-Class Assignment

In ERA V3, students trained CNNs on datasets like CIFAR-10, and learned about ImageNet mainly in theory. In V4, students **actually train ResNet-50 on the full ImageNet dataset using multi-GPU setups** (e.g. EC2, DDP). This includes:

- syncing across GPUs,
- debugging parallel training issues,
- benchmarking speedups.

This is something typically reserved for lab researchers, but it’s now integrated into the student journey.

4. Quantization as a Core Skill, Not an Add-On

ERA V3 introduced optimization through pruning, fine-tuning, and lightweight deployment, often using LoRA or PEFT.

ERA V4 teaches **full quantization-aware training** (QAT) as a primary technique. Not a shortcut - but a core capability.

Students implement QAT for large models and use tools like **WeightWatcher** to analyze weight behavior before and after quantization.

They gain a practical sense of what low-bit training really looks like - and where it breaks down.

5. vLLM for Inference - Built-In, Not Post-Course

While V3 covered inference in general terms (e.g. APIs, optimization tips), V4 includes **hands-on deployment using vLLM**, the fastest open-weight inference engine for LLMs today.

Students deploy large models in a way that mimics how modern systems (like OpenAI or Mistral) serve responses at scale - with kv-caching, streaming, and high-throughput workloads.

6. Alignment and RLHF - Introduced Early and Clearly

V3 explored fine-tuning and optimization in a more general sense. In V4, we go further and include **RLHF pipelines**, **GPRO**, and basic alignment strategies during instruction tuning.

We’re not trying to cover all of alignment theory here (that’s for EAG V4), but we do want students to see:

- how reward modeling works,
- what can go wrong (e.g. reward hacking, hallucinations),
- and how simple safeguards can be built into training loops.

7. Cleaner Focus - No Stretch Modules

In ERA V3, we had a broader footprint - agents, RAG, multimodal UX, MLOps, even edge deployment. These were valuable but diluted focus.

In V4, we’ve **pulled those topics into their own specialized tracks** (EAG V2), allowing this course to stay tightly focused on:

- neural networks,
- CNNs,
- transformers,
- RL, and
- LLMs at scale.

That makes the experience more cohesive - and more satisfying - for learners focused on core engineering.

ERA V3 gave students a strong start in AI. **ERA V4** gives them the skills to build, train, and deploy modern systems - including full-scale LLMs - with confidence and context.

It reflects how far the field has come, and what engineers now need to be able to do.

Not everything is harder - some things are just clearer now. And this course reflects that clarity.

Hope to see you in the class very soon!