

Production LLM & Agentic AI Engineering

Master Program: Full Syllabus

A hands-on 7-8 month specialization led by Mian Ahmad Basit. You will fine-tune DeepSeek-R1 and LLaMA 3.1 with QLoRA and DPO, build multi-agent systems with MCP and A2A, ship ColPali vision RAG, add NeMo security guardrails and run it all on Kubernetes.

20

Phases

37

Modules

936

Topics & subtopics

Lead instructor: **Mian Ahmad Basit**. Live weekend sessions, 8-11 PM. 2 years of LMS access.

Contents

- Phase 0** Programming & AI Foundations (Weeks 1 - 2)
- ML Phase** Machine Learning Foundations (Weeks 3 - 4)
- DL Phase** Deep Learning Foundations (Weeks 5 - 6)
- NLP Phase** NLP Foundations (Week 7)
- Phase 1** AI Foundations & Transformers 101 (Weeks 8 - 9)
- Phase 2** Prompts to Programs & Programmatic LLMs (Weeks 10 - 11)
- Phase 3** KV Cache & Advanced Attentions (Weeks 12 - 13)
- Phase 4** Finetuning Language Models & PEFT (Weeks 14 - 16)
- Phase 5** Mixture of Experts & Reasoning Models (Week 17)
- Phase 6** Knowledge Distillation & Small Language Models (Week 18)
- Phase 7** Multimodal Vision Systems & Image Understanding (Week 19)
- Phase 8** Speech AI & Conversational Audio Systems (Week 20)
- Phase 9** Representation Learning & Embedding Systems (Week 21)
- Phase 10** Production RAG Systems (Weeks 22 - 24)
- Phase 11** Agentic Systems: Core Architectures & Foundations (Weeks 25 - 26)
- Phase 12** Agents: Orchestration, Protocols & Production (Weeks 27 - 28)
- Phase 13** Context Engineering & Harness Engineering (Weeks 29 - 30)
- Phase 14** LLMOps & Observability (Week 31)
- Phase 15** Production Infrastructure & Cloud Deployment (Weeks 32 - 33)
- Phase 16** AI Security, Guardrails & RBAC (Week 34)

Programming & AI Foundations

Python fundamentals, object-oriented design, async concurrency and the project engineering prerequisites required for AI systems.

Checkpoint: Python Engineering Checkpoint | **Milestone:** Master Production-Grade Python

Skills: Python, OOP, Async, Git | **Tools:** Python, Git, venv, Poetry

P0-01: Python Syntax & Core Fundamentals

Deep dive into Python language constructs, data structures and memory mechanics.

Build production-grade Python foundations covering syntax, data structures, functions, exceptions, file handling, environments, packages and dependency management.

Core Syntax & Control Flow

- Python syntax and fundamentals
- Variables and data types
- Arithmetic, logical and bitwise operators
- Conditional statements: if, elif, else
- for and while loops
- break and continue

Functions & Functional Paradigms

- Functions and function arguments
- *args and **kwargs
- Lambda functions and anonymous execution
- Lists, tuples, sets and dictionaries
- List, dictionary and set comprehensions
- String manipulation
- Regular expression parsing

Files, Exceptions & Environments

- File handling
- Context management
- try, except and finally
- Modules and packages
- Relative imports
- venv, Conda and Poetry
- pip package management
- Dependency locking

Key outcome: Build clean Python projects using production-grade syntax, data structures, packaging and dependency management.

P0-02: Object-Oriented & Advanced Python Paradigms

Advanced OOP principles, metaprogramming, type hints and asynchronous Python.

Master advanced Python mechanics including object-oriented programming, iterators, generators, decorators, typing, dataclasses, asynchronous execution, REST APIs and project engineering.

Object-Oriented Programming

- Classes and objects mechanics
- Inheritance
- Multiple inheritance
- Method Resolution Order (MRO)
- Encapsulation and private attributes

- Polymorphism
- Duck typing
- Abstraction
- Abstract Base Classes

Advanced Python Mechanics

- Iterator protocol
- Custom iterable objects
- Generators and yield
- Memory-efficient iteration
- Decorators and wrapper functions
- Stateful decorators
- Context managers
- `__enter__` and `__exit__`
- Type hints
- Static typing verification
- Dataclasses
- Immutable data structures

Async Programming & APIs

- asyncio event loop
- async and await
- Concurrent tasks
- `asyncio.gather`
- Semaphores
- HTTP and REST fundamentals
- Async HTTP requests
- JSON serialization and deserialization
- `.env` environment management
- Git and GitHub workflows
- Python project structure
- Package setup

Key outcome: Engineer asynchronous, typed and modular Python services ready for AI workloads.

Machine Learning Foundations

Complete mathematical foundations (linear algebra, calculus, statistics) and classical supervised and unsupervised ML algorithms.

Checkpoint: ML Foundations Checkpoint | **Milestone:** Train, Evaluate & Serialize Classical ML Models

Skills: Linear Algebra, Statistics, Classical ML | **Tools:** NumPy, Scikit-Learn

ML-01: Mathematics for Machine Learning

The exact mathematical concepts that power vector spaces, optimization loss landscapes and probabilistic models.

Build the mathematical intuition required to understand optimization, neural networks and modern AI systems.

Linear Algebra

- Scalars, vectors, matrices and tensors
- Matrix addition and transposition
- Dot product
- L1 and L2 vector norms
- Matrix multiplication
- Eigenvalues
- Eigenvectors

Probability & Statistics

- Probability fundamentals and axioms
- Conditional probability
- Joint probability
- Bayes theorem
- Bayesian inference
- Discrete and continuous random variables
- Gaussian, Bernoulli and Poisson distributions
- Mean, variance and standard deviation
- Covariance and correlation
- Descriptive vs inferential statistics
- Sampling techniques
- Central Limit Theorem
- Hypothesis testing
- p-values
- Confidence intervals

Calculus & Optimization

- Derivatives and rate of change
- Partial derivatives
- Gradient vectors
- Loss surface geometry
- Chain rule
- Optimization fundamentals
- Convexity

Key outcome: Build the mathematical intuition behind optimization and probabilistic modeling used across AI systems.

ML-02: Classical Machine Learning Algorithms & Engineering

Supervised and unsupervised learning, ensemble methods, feature engineering and model evaluation.

Master classical machine learning algorithms alongside the feature engineering and evaluation practices required to ship reliable models.

Classical Machine Learning

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement Learning overview
- Linear and multiple regression
- Polynomial regression
- Logistic regression
- KNN
- Naive Bayes
- Decision Trees
- Random Forest
- GBM
- XGBoost
- LightGBM
- CatBoost
- SVM and SVR
- K-Means
- Hierarchical clustering
- DBSCAN
- PCA

ML Engineering & Evaluation

- Feature selection
- One-Hot Encoding
- Target Encoding
- Ordinal Encoding
- MinMax normalization
- Standardization
- Missing-value imputation
- Outlier detection
- Data leakage prevention
- Train/validation/test splitting
- K-Fold cross-validation
- Stratified cross-validation
- Grid Search
- Random Search
- Confusion Matrix
- Accuracy, Precision, Recall and F1
- ROC-AUC and PR-AUC
- MAE, MSE, RMSE and R^2
- Bias-variance tradeoff
- Underfitting and overfitting
- L1 and L2 regularization
- Scikit-Learn Pipelines
- joblib and pickle serialization
- ML deployment fundamentals

Key outcome: Train, evaluate and serialize classical ML models with production-grade pipelines.

Deep Learning Foundations

Neural network architectures, forward and backpropagation, loss landscapes, PyTorch mechanics, CNNs and sequence models.

Checkpoint: Deep Learning Checkpoint | **Milestone:** Build Neural Networks in PyTorch

Skills: Neural Networks, PyTorch, CNNs, Sequence Models | **Tools:** PyTorch

DL-01: Neural Networks & Optimization Mechanics

Perceptrons, multi-layer networks, backpropagation calculus and optimization algorithms.

Understand neural network mechanics from perceptrons through backpropagation, optimizers and regularization.

Neural Network Mechanics

- Artificial neurons
- Perceptrons
- Linearity limitations
- Forward propagation
- Backpropagation
- Computation graphs
- MSE loss
- Cross-Entropy
- Binary Cross-Entropy
- Gradient Descent
- SGD
- Mini-batch Gradient Descent
- Momentum
- Nesterov acceleration
- Adam
- AdamW
- Cosine Annealing
- Learning-rate warmup
- Xavier initialization
- He initialization

Normalization & Regularization

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- GELU
- Softmax
- BatchNorm
- LayerNorm
- Dropout
- DropPath
- L1/L2 neural regularization
- Vanishing gradients
- Exploding gradients

Key outcome: Trace a neural network from forward propagation through backpropagation and modern optimization.

DL-02: PyTorch, CNNs & Sequence Models

Hands-on PyTorch engineering, convolutional networks for vision and recurrent architectures for sequences.

Engineer models in PyTorch and understand convolutional and recurrent architectures for vision and sequence data.

PyTorch Engineering

- PyTorch tensors
- Tensor memory layout
- Autograd
- Custom nn.Module models
- Dataset and DataLoader
- Training loops
- Validation loops
- CUDA and MPS acceleration
- Mixed precision training
- Checkpointing
- State Dicts
- Transfer learning

CNNs & Sequence Models

- Convolution operations
- Kernels and filters
- Feature maps
- Stride and padding
- Max and average pooling
- ResNet
- EfficientNet
- Image classification
- RNNs
- LSTMs
- GRUs
- Bidirectional RNNs
- Seq2Seq
- Encoder-decoder architectures
- Teacher forcing
- Introduction to attention

Key outcome: Build and train CNNs and sequence models in PyTorch, ready for Transformer architectures.

NLP Foundations

Classical text preprocessing, vector embeddings (Word2Vec, GloVe) and sequence modeling for NLP downstream tasks.

Checkpoint: NLP Foundations Checkpoint | **Milestone:** Represent Text as Dense Vector Spaces

Skills: Tokenization, Word2Vec, NER, Sentiment Analysis | **Tools:** NLTK, spaCy

NLP-01: NLP Core Concepts & Vector Spaces

Text representation techniques from lexical features to dense continuous vector spaces.

Master classical NLP preprocessing, representation and downstream tasks that lead directly into Transformers.

NLP Foundations

- Text normalization
- Unicode handling
- Tokenization
- Stopword removal
- Stemming
- Lemmatization
- N-grams
- Bag of Words
- TF-IDF
- Word2Vec CBOW
- Word2Vec Skip-Gram
- Negative sampling
- GloVe
- Contextual embeddings
- Text classification
- Topic modeling
- NER
- Sentiment analysis
- Intent recognition
- Language modeling
- Translation
- Summarization
- Attention in NLP

Key outcome: Build classical NLP pipelines and understand the representations that lead directly into Transformers.

AI Foundations & Transformers 101

Deep dive into Transformers, self-attention and full tokenization strategies (BPE, WordPiece, SentencePiece).

Checkpoint: Transformer Engineering Checkpoint | **Milestone:** Implement a Transformer Attention Block

Skills: Self-Attention, Tokenization, BPE | **Tools:** PyTorch, Transformers, tiktoken

01-A: Transformers 101 & Attention Mechanism

Embeddings, self-attention and Transformer architectures.

Understand the mathematics and implementation of modern Transformer blocks from embeddings through attention and positional encoding.

Embeddings

- High-dimensional semantic spaces
- Embedding lookup tables
- Projection matrices
- Cosine similarity
- Dot-product similarity

Attention Mechanics

- Scaled Dot-Product Attention
- Query, Key and Value mathematics
- Self-attention step-by-step
- Multi-Head Attention
- Parallel attention subspaces
- Masked causal attention
- Cross-attention

Transformer Architectures

- Sinusoidal positional encoding
- Learned positional encoding
- Original Encoder-Decoder Transformer
- BERT
- RoBERTa
- GPT architecture
- LLaMA architecture
- Encoder-only models
- Decoder-only models

Key outcome: Implement and mathematically trace a Transformer attention block from embeddings through attention outputs.

01-B: Tokenization Deep Dive

BPE, WordPiece, SentencePiece and byte-level tokenization.

Understand how modern LLM tokenizers convert raw text into token IDs and handle vocabulary construction, subwords and edge cases.

Tokenization Taxonomy

- Word-level tokenization
- Subword tokenization
- Character-level tokenization
- Byte-level tokenization

Subword Algorithms

- Byte Pair Encoding
- BPE algorithm step-by-step
- WordPiece
- Unigram tokenization
- SentencePiece
- Byte-level BPE
- tiktoken
- LLaMA tokenizer

Vocabulary Engineering

- Vocabulary construction
- Special tokens
- Unknown tokens
- Padding tokens
- Beginning/end-of-sequence tokens
- Tokenization edge cases
- Token counting
- Context-window budgeting

Key outcome: Build and analyze tokenizers while understanding exactly how text becomes model input tokens.

Prompts to Programs & Programmatic LLMs

Advanced prompt engineering, Pydantic structured schemas, Outlines FSA grammars and DSPy programmatic pipeline optimization.

Checkpoint: Programmatic LLM Checkpoint | **Milestone:** Compile an Optimized DSPy Program

Skills: Prompt Engineering, Structured Generation, DSPy | **Tools:** Pydantic, Instructor, Outlines, DSPy

02-A: Advanced Prompt Engineering Strategies

Systematic prompt design, chain-of-thought, meta-prompting and output enforcement.

Design robust production prompts using systematic prompting strategies, decomposition, structured outputs and iterative refinement.

Prompt Architecture

- Anatomy of a production prompt
- Zero-shot prompting
- One-shot prompting
- Few-shot prompting
- System prompt design
- Role assignment
- Output formatting

Reasoning Strategies

- Chain-of-Thought prompting
- Step-Back prompting
- Prompt decomposition
- Prompt chaining
- Meta-prompting
- Self-refinement loops

Prompt Reliability

- Prompt sensitivity
- Robustness testing
- Structured generation
- Prompt versioning
- Prompt evaluation

Key outcome: Design reliable production prompts instead of relying on ad-hoc prompt experimentation.

02-B: Instructor, Pydantic & Outlines for Typed Outputs

Guarantee 100% schema-compliant responses using typed validation contracts and token masking.

Build schema-controlled LLM applications using Pydantic, Instructor and Outlines with validation, retries and constrained decoding.

Instructor & Pydantic

- Why typed outputs matter
- Instructor client patching
- Pydantic execution contracts
- Automatic validation retries
- Validation-error feedback
- Nested schemas
- Streaming validation
- Async structured outputs
- TOOLS mode

- JSON_SCHEMA mode
- MD_JSON mode

Outlines Guided Generation

- Finite State Automata generation
- Logit-level token masking
- Pydantic schema enforcement
- JSON Schema generation
- Regex-constrained generation
- Context-free grammars
- SQL generation
- Code generation
- Custom DSL generation

Key outcome: Build LLM applications whose outputs conform to machine-verifiable schemas instead of unreliable free-form text.

02-C: DSPy: Programmatic LLM Pipeline Optimization

Replacing manual prompt engineering with compiled, self-optimizing DSPy programs.

Replace manual prompt engineering with declarative DSPy programs that automatically optimize instructions, demonstrations and pipeline behavior.

DSPy Core

- Prompt engineering vs programmatic LLMs
- DSPy signatures
- Declarative input/output definitions
- DSPy modules
- Predict
- ChainOfThought
- ReAct
- ProgramOfThought
- LM configuration
- Telemetry

DSPy Optimizers

- BootstrapFewShot
- Automatic few-shot selection
- BootstrapFewShotWithRandomSearch
- MIPROv2
- SIMBA
- GEPA
- BetterTogether
- BootstrapFinetune
- Model distillation

Key outcome: Compile an optimized DSPy program instead of manually tuning prompts.

KV Cache & Advanced Attentions

Transformer fine-tuning, FlashAttention, Multi-Query Attention (MQA), Grouped-Query Attention (GQA), MLA, PagedAttention and scaling laws.

Checkpoint: Attention Optimization Checkpoint | **Milestone:** Optimize LLM Inference Memory & Throughput

Skills: FlashAttention, GQA, MLA, PagedAttention | **Tools:** PyTorch, vLLM

03-A: Finetuning Transformer Architectures

Hands-on training of DistilBERT, DistilGPT and T5 on custom task datasets.

Implement Transformer attention components and fine-tune encoder, decoder and encoder-decoder architectures on custom datasets.

Transformer Fine-Tuning

- Coding attention mechanisms in PyTorch
- DistilBERT classification fine-tuning
- Custom classification datasets
- DistilGPT generation fine-tuning
- Custom generation datasets
- T5 Seq2Seq fine-tuning
- Custom sequence-to-sequence datasets

Key outcome: Fine-tune encoder, decoder and encoder-decoder Transformer architectures on custom tasks.

03-B: KV Cache & Modern Attention Variants

Optimizing inference speed and memory throughput using FlashAttention, GQA, MLA and vLLM PagedAttention.

Understand the memory and compute bottlenecks of autoregressive decoding and the modern attention mechanisms used to solve them.

KV Cache & Efficient Attention

- Naive decoding and $O(N^2)$ redundant computation
- KV Cache mechanics
- KV Cache memory calculations
- FlashAttention 1
- FlashAttention 2
- FlashAttention 3
- IO-aware GPU tiling
- PyTorch SDPA

Attention Variants

- MHA memory bottleneck
- Multi-Query Attention
- Grouped-Query Attention
- GQA in LLaMA-3
- Multi-Head Latent Attention
- MLA in DeepSeek models
- PagedAttention
- vLLM virtual memory management
- RoPE

Scaling Laws

- Kaplan scaling laws
- Chinchilla compute-optimal scaling
- Tokens-to-parameters ratio
- Dataset scaling mathematics
- Compute budget allocation

Key outcome: Analyze and optimize LLM attention memory, throughput and compute requirements for production inference.

Finetuning Language Models & PEFT

Pre-training lifecycle, SFT, LoRA, QLoRA, DoRA, preference alignment (DPO, GRPO, ORPO), quantization and synthetic datasets.

Checkpoint: Model Training & Alignment Checkpoint | **Milestone:** Align, Quantize & Ship an Open-Weight LLM

Skills: SFT, LoRA, QLoRA, DPO, GRPO | **Tools:** Hugging Face, PEFT, TRL, Unsloth, Axolotl

04-A: Pre-training vs Post-training Lifecycle & Data Prep

Understanding pre-training objectives, continued pre-training (CPT) and instruction dataset engineering.

Understand the complete model lifecycle from pre-training and continued pre-training through instruction tuning and synthetic dataset engineering.

Training Lifecycle

- Pre-training vs post-training
- Base model capabilities
- Base model limitations
- Causal Language Modeling
- Masked Language Modeling
- Prefix Language Modeling
- Continued Pre-Training
- Domain adaptation
- Compute budget tradeoffs
- When CPT is appropriate
- Multi-Token Prediction

Data Preparation

- CommonCrawl
- RefinedWeb
- Instruction-pair datasets
- Chat-format datasets
- ChatML templates
- LLaMA-3 chat format
- Mistral formatting
- Loss masking
- Dataset deduplication
- Filtering pipelines

Synthetic Data

- Self-Instruct
- Alpaca-style datasets
- Synthetic instruction generation
- Data quality filtering

Key outcome: Engineer high-quality datasets and understand exactly where CPT, SFT and synthetic data fit into the model lifecycle.

04-B: PEFT Techniques: LoRA, QLoRA, DoRA & AdaLoRA

Parameter-efficient fine-tuning math, 4-bit NF4 quantization and adapter architectures.

Master adapter-based fine-tuning techniques and the mathematics behind low-rank adaptation and quantized training.

PEFT Mathematics

- Intrinsic dimensionality
- LoRA mathematics
- $W = W_0 + BA$
- Rank selection
- Alpha scaling
- QLoRA
- NF4 quantization
- Double quantization
- Paged optimizers
- DoRA
- Magnitude and direction decomposition
- AdaLoRA
- Adaptive rank allocation
- LoRA+
- Separate A/B learning rates

Supervised Fine-Tuning

- SFT pipeline design
- Instruction tuning
- FLAN datasets
- Alpaca datasets
- OpenHermes datasets
- Chat fine-tuning
- Conversational fine-tuning
- Reasoning fine-tuning
- Domain-specific fine-tuning

Key outcome: Fine-tune an open-weight LLM efficiently using QLoRA and production-grade SFT datasets.

04-C: Preference Alignment & Advanced Post-Training

Aligning LLMs with human preferences using DPO, GRPO, ORPO and RLHF.

Master modern post-training methods for aligning language models with human and programmatic preferences while preparing models for efficient deployment.

Preference Alignment

- Why SFT alone is insufficient
- Distribution shift
- Verbosity bias
- RLHF
- PPO
- Reward models
- Critic networks
- KL penalties
- DPO
- GRPO
- ORPO

Quantization & Inference Preparation

- GPTQ
- AWQ
- BNB NF4
- FP8
- LoRA adapter merging
- Multi-LoRA serving
- vLLM

- SGLang
- GGUF
- GGUML
- llama.cpp
- Speculative decoding

Training Tooling

- Hugging Face TRL
- SFTTrainer
- DPOTrainer
- GRPOTrainer
- ORPOTrainer
- Unsloth
- Axolotl
- LLaMA-Factory
- Together AI
- AWS SageMaker fine-tuning

Synthetic Preference Data

- Instruction synthetic data
- Preference synthetic data
- Code synthetic data
- LLM-as-Judge scoring
- distilabel
- DataDreamer
- Argilla
- Model collapse risks
- Data poisoning prevention
- End-to-end SFT dataset pipeline

Key outcome: Align, quantize and prepare an open-weight LLM for efficient production inference.

Mixture of Experts & Reasoning Models

Mixture of Experts (MoE) routing mechanisms, DeepSeek-R1-Zero reasoning models and chain-of-thought reinforcement learning.

Checkpoint: MoE & Reasoning Checkpoint | **Milestone:** Explain Sparse Routing & RL-Driven Reasoning
Skills: MoE, GRPO, Reasoning Models | **Tools:** Mixtral, DeepSeek-R1

05-A: Mixture of Experts

Sparse expert activation, gating networks and load-balancing at scale.

Understand sparse expert architectures used by modern production models.

Mixture of Experts

- Dense-model scaling limitations
- Sparse expert activation
- Gating networks
- Top-K routing
- Router mathematics
- Load-balancing loss
- Auxiliary losses
- Expert capacity
- MoE training at scale
- MoE inference
- Memory vs compute tradeoffs
- Mixtral 8x7B
- DeepSeek-V3 256 experts
- Production MoE models
- MoE vs dense architectures

Key outcome: Explain how sparse expert routing enables larger models without proportional compute cost.

05-B: Reasoning Models

DeepSeek-R1-Zero, rule-based rewards and GRPO-driven chain-of-thought reasoning.

Understand reinforcement-learning approaches used by modern reasoning models.

Reasoning Models

- Reasoning model concept
- System 1 vs System 2
- Chain-of-Thought
- Pure RL reasoning
- SFT + RL reasoning
- DeepSeek-R1-Zero
- Rule-based rewards
- GRPO reasoning
- Reasoning trace distillation

Key outcome: Explain how reinforcement learning produces modern reasoning-capable models.

Knowledge Distillation & Small Language Models

SLM design philosophy, Liquid Foundation Models, soft labels, KL divergence and student-teacher pipeline construction.

Checkpoint: Distillation Checkpoint | **Milestone:** Build a Compact Specialized Model

Skills: SLMs, Distillation, KL Divergence | **Tools:** Phi-3, Gemma-2B

06-A: Small Language Models (SLMs) & Edge AI

Building ultra-compact 1B-3B models with performance rivaling large LLMs.

Explore SLM design philosophy, edge deployment and training/fine-tuning of small models.

Small Language Models

- SLM design philosophy
- Phi-3
- Gemma-2B
- LLaMA-3.2-1B
- Latency optimization
- Privacy advantages
- Edge deployment
- Data density
- Liquid Foundation Models
- Dynamical systems
- Training SLMs from scratch
- SLM fine-tuning
- Reasoning trace distillation
- SLM vs LLM tradeoffs

Key outcome: Design and deploy compact, edge-ready small language models.

06-B: Knowledge Distillation Pipelines

Compressing teacher model knowledge into student networks using temperature scaling and KL loss.

Compress foundation-model capabilities into smaller student models.

Knowledge Distillation

- Teacher-student paradigm
- Hard labels
- Soft labels
- Temperature scaling
- KL divergence
- Attention transfer
- Hidden-state matching
- Teacher logit generation
- Student architecture design
- Student training pipeline

Key outcome: Build a teacher-student distillation pipeline that compresses a large model into a compact one.

Multimodal Vision Systems & Image Understanding

Vision Transformers (ViT), CLIP, SigLIP, DINOv2 and Visual Language Models (VLMs) like LLaVA and Qwen-2-VL.

Checkpoint: Multimodal Vision Checkpoint | **Milestone:** Connect Vision Encoders to LLM Backbones

Skills: ViT, CLIP, VLMs | **Tools:** LLaVA, Qwen-2-VL

07-A: Vision Foundations & Vision Transformers (ViT)

Patch embeddings, 2D positional encodings, self-attention on visual tokens and pre-trained encoders.

Bridge from CNNs to Vision Transformers and modern visual encoders.

Vision Transformers

- CNN-to-ViT conceptual bridge
- ViT architecture
- 16x16 image patches
- Patch embeddings
- CLS token
- 2D positional encoding
- Visual self-attention
- Attention maps
- CNN vs ViT inductive bias
- CLIP
- SigLIP
- DINO
- DINOv2

Key outcome: Implement and reason about Vision Transformer patch embeddings and self-attention.

07-B: Visual Language Models (VLMs)

Connecting visual encoders with LLM backbones via linear projectors and cross-attention aligners.

Understand how visual encoders connect to LLM backbones to produce multimodal reasoning.

Visual Language Models

- VLM architecture
- LLaVA
- Qwen-2-VL
- Gemini Vision
- Visual encoder
- ViT/SigLIP encoders
- MLP projectors
- Cross-attention aligners
- Perceiver alignment
- Vision-language feature spaces
- Visual grounding

Key outcome: Explain and build the encoder-projector-LLM pipeline behind modern VLMs.

Speech AI & Conversational Audio Systems

Speech-to-Text (STT) foundations, Whisper architecture, acoustic dataset preparation and custom fine-tuning.

Checkpoint: Speech AI Checkpoint | **Milestone:** Fine-Tune Whisper on Domain Audio

Skills: Whisper, STT, Audio Processing | **Tools:** Whisper API

08-A: Speech AI & Fine-Tuning Whisper

Acoustic feature extraction, log-mel spectrograms, Whisper architecture and fine-tuning on domain audio.

Understand speech signal processing and build custom speech-to-text pipelines with Whisper.

Speech AI

- Speech signal processing
- Log-Mel spectrograms
- Speech-to-Text
- Whisper API
- Whisper architecture
- Audio Encoder-Decoder Transformer
- Streaming chunking
- Audio dataset preparation
- Whisper fine-tuning
- Domain accents
- Domain-specific audio

Key outcome: Fine-tune Whisper on domain-specific audio and build streaming speech-to-text systems.

Representation Learning & Embedding Systems

Dense, sparse, quantized, binary and multi-vector (ColBERT) embeddings, Matryoshka Representation Learning (MRL) and embedding fine-tuning.

Checkpoint: Embedding Systems Checkpoint | **Milestone:** Select, Fine-Tune & Benchmark Embeddings

Skills: Embeddings, ColBERT, MRL | **Tools:** BGE, E5

09-A: Advanced Embedding Taxonomy & MRL

Dense vs sparse signals, float32 to int8/binary quantization, ColBERT multi-vector and MRL dynamic dimensioning.

Understand modern embedding architectures from dense vectors through sparse lexical signals, quantized representations, ColBERT multi-vectors and Matryoshka embeddings.

Embedding Taxonomy

- Dense embeddings
- BGE
- E5
- OpenAI text-embedding models
- Sparse embeddings
- SPLADE
- BM25
- Quantized embeddings
- Float32 to Int8
- Uint8 compression
- Binary embeddings
- Hamming distance
- Multi-vector embeddings
- ColBERT
- ColPali

Matryoshka Representation Learning

- MRL concept
- Flexible embedding dimensions
- Nested dimension loss
- 1536 to 256 dimension retrieval
- Dynamic query-time dimensions

Embedding Fine-Tuning & Evaluation

- Multiple Negatives Ranking Loss
- Embedding fine-tuning
- Domain adaptation
- MTEB benchmark
- Embedding leaderboard analysis
- State-of-the-art embedding evaluation

Key outcome: Select, fine-tune, compress and benchmark embeddings for enterprise retrieval workloads.

Production RAG Systems

LangChain 2026, LlamaIndex, Hybrid RAG, Self-RAG, Corrective RAG, Adaptive RAG, ColPali multimodal RAG, GraphRAG and RAG security.

Checkpoint: Production RAG Architecture Checkpoint | **Milestone:** Ship a Secure Enterprise RAG System

Skills: Hybrid Search, GraphRAG, Multimodal RAG, RAG Security | **Tools:** LangChain, LlamaIndex, Pinecone, Qdrant, Neo4j, Redis

10-A: LangChain & LlamaIndex Framework Ecosystems

Production RAG frameworks and data connectors.

Build production retrieval pipelines using LangChain and LlamaIndex with document loaders, vector stores, metadata filtering and query engines.

LangChain Ecosystem

- LangChain architecture
- LLM integrations
- Model abstraction
- LCEL
- Structured outputs
- Output parsers
- Tool calling
- Function calling
- Conversation memory
- Document loaders
- Text splitters
- Embeddings
- PGVector
- Pinecone
- Qdrant
- Retrieval pipelines
- Text-to-SQL
- Callbacks
- LangSmith telemetry
- Multimodal integrations
- Deep Agents

LlamaIndex Ecosystem

- Documents
- Nodes
- Chunking strategies
- PDF/API/web/filesystem loaders
- Embedding configuration
- Chroma
- FAISS
- Pinecone
- Qdrant
- Vector Index
- Summary Index
- Keyword Index
- Retrievers
- Query engines
- Metadata filtering
- Hybrid search

- Reranking
- Response synthesis

Key outcome: Build production RAG applications using both major Python retrieval ecosystems.

10-B: Advanced RAG Architectures & Vector Quantization

Hybrid, agentic, Self-RAG, CRAG and adaptive retrieval.

Move beyond vanilla RAG into sophisticated retrieval architectures with query transformation, reranking, reflection and vector compression.

RAG Foundations

- Vanilla RAG pipeline
- RAG limitations
- Embedding model selection
- Fixed chunking
- Sentence chunking
- Recursive chunking
- Semantic chunking

Advanced Retrieval

- BM25
- SPLADE
- ColBERT
- Hybrid dense + sparse retrieval
- Reciprocal Rank Fusion
- Meta Hybrid RAG
- Query rewriting
- Sub-query decomposition
- HyDE
- Contextual retrieval
- Anthropic contextual chunking
- Agentic RAG

Self-Reflective RAG

- RAG evaluation
- Context precision
- Context recall
- Faithfulness
- Answer relevance
- Cohere Rerank
- BGE-Reranker-v2
- Cross-encoder reranking
- Self-RAG
- Corrective RAG
- Fallback web search
- Adaptive RAG
- Complexity-based routing

Vector Quantization

- Scalar Quantization
- SQ8
- Binary Quantization
- BQ
- Product Quantization
- PQ

Key outcome: Design retrieval architectures that dynamically select, transform, rerank and validate information.

10-C: Multimodal RAG, Graph RAG, Caching & RAG Security

Visual retrieval, knowledge graphs and secure retrieval.

Build enterprise RAG systems capable of understanding complex documents, knowledge graphs, multimodal content and security boundaries.

Multimodal Document RAG

- ColPali visual page retrieval
- Single-stage parsing
- Dual-stage parsing
- Layout detection
- OCR
- Structured layout chunking
- Vision-language embeddings
- Vision-language rerankers
- Multimodal LLM document understanding
- AWS Textract
- Azure AI Document Intelligence
- Google Cloud Document AI
- LlamaParse
- Docling
- LiteParse

Graph RAG & Vectorless RAG

- Neo4j GraphRAG
- Knowledge graph construction
- Entity extraction
- Relation extraction
- Graph traversal
- Vectorless RAG
- PageIndex

Caching & Security

- Redis caching
- GPTCache
- Semantic caching
- PII masking
- Anonymization pipelines
- Input guardrails
- Output guardrails
- Prompt injection defense
- Retrieval-interface security

Key outcome: Ship a secure multimodal and graph-enhanced enterprise RAG system with caching and access controls.

Agentic Systems: Core Architectures & Foundations

Pydantic validation, LangGraph stateful graphs, PydanticAI type-safe agents, Pydantic Evals, MCP and A2A protocols.

Checkpoint: Agentic Systems Architecture Checkpoint | **Milestone:** Build a Stateful, Checkpointed Agent

Skills: LangGraph, PydanticAI, Evals | **Tools:** LangGraph, PydanticAI, Logfire

11-A: Pydantic Validation & LangGraph Stateful Agents

State management, dynamic routing, ReAct loops, human-in-the-loop checkpointing and LangGraph Platform.

Build reliable stateful agents using Pydantic contracts and LangGraph graph-based orchestration.

Pydantic Validation

- Pydantic installation
- Models
- Fields
- Validators
- JSON parsing
- Type coercion
- Dataclasses
- Nested models
- Validation decorators
- Custom validators
- Settings management

LangGraph Architecture

- LangGraph foundations
- StateGraph
- START and END
- State management
- Reducers
- Nodes
- Edges
- Conditional routing
- Tool calling
- ReAct loops

Agent State & Memory

- Human-in-the-loop interrupts
- Approval workflows
- Edit State
- Checkpointers
- Postgres persistence
- MemorySaver
- Subgraphs
- Supervisor architectures
- Multi-agent swarms
- Streaming
- Observability
- LangGraph Platform

Key outcome: Build a stateful, checkpointed LangGraph agent with tool execution and human approval workflows.

11-B: PydanticAI, Evals & Logfire Observability

Type-safe agent contracts, dependency injection, ModelRetry, Pydantic Evals and Logfire tracing.

Engineer type-safe agents using PydanticAI with structured outputs, dependency injection, retries, evaluations and tracing.

PydanticAI Architecture

- PydanticAI architecture
- Agents
- Models
- Runnable contracts
- Provider configuration
- Fallback models
- Result types
- Pydantic structured results
- Union result types
- Discriminated unions

Agent Tools & Validation

- Field validators
- Model validators
- Static system prompts
- Dynamic system prompts
- Instruction patterns
- Dependency injection
- `deps_type`
- `RunContext`
- Type-safe context
- `@agent.tool`
- `RunContext` tools
- `@agent.tool_plain`
- `ModelRetry`
- Validation error handling

Streaming & Multi-Agent Systems

- `stream_text`
- `stream`
- `astream_events`
- Multi-turn conversations
- Message history
- Context window management
- Agent pipelines
- Delegation
- Handoffs
- `TestModel`
- `FunctionModel`
- Agent unit testing

Evals & Observability

- Pydantic Evals
- Datasets
- Evaluation cases
- Experiment model
- Built-in evaluators
- Custom evaluators
- LLM-as-a-Judge
- Online evaluation

- Logfire integration
- Tracing
- Metrics
- Cost monitoring
- FastAPI deployment
- Sessions
- Containerization

Key outcome: Build type-safe, testable agents with automated evaluation and production-grade observability.

Agents: Orchestration, Protocols & Production

Model Context Protocol (MCP), Agent-to-Agent Protocol (A2A), function calling mechanics and agent observability tools.

Checkpoint: Agent Protocols Checkpoint | **Milestone:** Connect Agents & Tools Through MCP and A2A

Skills: MCP, A2A, Function Calling | **Tools:** FastMCP, LangSmith, Langfuse

12-A: MCP (Model Context Protocol) Deep Dive

Architecture, host-client-server primitives, FastMCP, stdio and SSE transport layers, security boundaries and multi-server orchestration.

Master the standardized protocol for connecting agents to tools and resources across distributed systems.

Model Context Protocol

- Why MCP exists
- Tool fragmentation problem
- MCP Hosts
- MCP Clients
- MCP Servers
- MCP Tools
- MCP Resources
- MCP Prompts
- stdio transport
- SSE transport
- FastMCP server development
- MCP client development
- Remote MCP servers
- MCP authentication
- MCP authorization
- Trust boundaries
- Multi-server orchestration
- MCP with LangGraph
- MCP with Claude

Key outcome: Build and connect MCP servers and clients across a multi-tool agent system.

12-B: A2A (Agent-to-Agent Protocol) & Function Calling

Inter-agent communication, Agent Cards discovery, task lifecycle states, parallel tool calls and custom tool execution loops.

Master agent interoperability and robust function execution alongside end-to-end observability of agent workflows.

Agent-to-Agent Protocol

- Why A2A exists
- Agent interoperability
- Agent Cards
- Agent discovery
- `/.well-known/agent.json`
- A2A task lifecycle
- Submitted state
- Working state
- Completed state
- Failed state
- Streaming states

- HTTP transport
- SSE transport
- JSON-RPC messages
- A2A server
- A2A client
- Remote agent delegation
- A2A vs MCP

Function Calling

- LLM-to-tool communication
- JSON Schema tool definitions
- Parameter typing
- Tool descriptions
- Function calling lifecycle
- Parallel tool calls
- Forced tool selection
- Required tool choice
- No-tool choice
- Specific tool selection
- Structured output vs function calling
- Tool executor loops

Agent Observability

- LangSmith tracing
- LangSmith span telemetry
- Dataset annotation
- Logfire tracing
- Span-level telemetry
- Langfuse monitoring
- Token monitoring
- Cost monitoring
- Latency monitoring
- Multi-agent tracing
- Tool execution traces
- Agent handoff tracing
- Failure analysis
- Cost attribution
- Latency analysis
- Protocol-level observability

Key outcome: Connect multiple agents through A2A, execute robust function calls and trace multi-agent workflows end to end.

Context Engineering & Harness Engineering

Anatomy of context windows, context compression (LLMLingua, RECOMP), evaluation harnesses (lm-eval, Inspect AI) and E2B sandboxes.

Checkpoint: Context & Harness Engineering Checkpoint | **Milestone:** Evaluate Agents & Execute Code in Secure Sandboxes

Skills: Context Engineering, Agent Evals, Sandboxing | **Tools:** Inspect AI, E2B, Docker

13-A: Context Engineering & Compression

Maximizing context window efficiency, memory architectures (episodic, semantic, procedural) and context pruning.

Engineer the context supplied to LLMs and agents using retrieval, memory, structured tool results and compression.

Context Construction

- Anatomy of the context window
- System instructions
- Conversation history
- Retrieved context
- Tool results
- Structured data
- Output constraints
- RAG as context construction
- Chunking impact on context quality

Memory Architectures

- Short-term memory
- Episodic memory
- Semantic memory
- Procedural memory
- Conversation state
- Context accumulation

Context Compression

- Context pruning
- Context compression
- LLMLingua
- RECOMP
- Long-context optimization
- Tool-result compression
- Retrieval context selection

Key outcome: Construct compact, relevant and high-signal context for long-running AI agents.

13-B: Harness Engineering & Sandbox Execution

Building evaluation harnesses, Inspect AI task scorers, E2B/Docker code execution sandboxes and ReAct loop safeguards.

Build automated evaluation harnesses and secure execution environments for tool-using and coding agents.

Evaluation Harnesses

- Harness engineering
- Why ad-hoc testing fails
- Evaluation harness fundamentals
- lm-evaluation-harness

- MMLU
- GSM8K
- TruthfulQA
- Inspect AI
- Task abstraction
- Solver abstraction
- Scorer abstraction
- LLM-as-a-Judge
- Judge calibration
- Evaluation bias mitigation

Secure Execution

- E2B sandboxes
- Modal sandboxes
- Docker code execution
- Secure tool execution
- Agent checkpointing
- LangGraph persistence
- Interrupts

ReAct Safety Harnesses

- Iteration guards
- Fallback handlers
- Token budgets
- Execution time limits
- Tool failure handling
- Supervisor-worker contracts
- Multi-agent execution harnesses

Key outcome: Evaluate agents automatically and execute untrusted agent-generated code inside controlled sandboxes.

LLMOps & Observability

Span-level tracing, LangSmith, Langfuse, Logfire, RAGAS evals, prompt caching, semantic Redis caching and token budgeting.

Checkpoint: LLMOps Checkpoint | **Milestone:** Instrument Production AI With End-to-End Telemetry

Skills: LLMOps, Tracing, Cost Control | **Tools:** LangSmith, Langfuse, Logfire, RAGAS

14-A: Tracing, Evals, Cost & Caching

Production observability, cost optimization, token budget enforcement and semantic caching.

Operate production AI systems using tracing, evaluation, token accounting, caching, rate limiting and financial controls.

Tracing & Evaluation

- LangSmith tracing
- Span telemetry
- Langfuse token monitoring
- Langfuse latency monitoring
- Logfire span-level traces
- RAGAS evaluation
- Inspect AI evaluation
- RAG validation
- Agent validation

Caching & Cost Control

- Prompt caching
- Anthropic prompt prefix caching
- OpenAI prompt caching
- Semantic caching
- Redis caching
- GPTCache
- Token accounting
- Rate limiting
- Token budgets
- Financial budget enforcement

Key outcome: Instrument production AI workloads with end-to-end telemetry, evaluation and automated cost controls.

Production Infrastructure & Cloud Deployment

Inference serving (vLLM, SGLang, llama.cpp), speculative decoding, Kubernetes (EKS), Docker, Helm and GitHub Actions CI/CD.

Checkpoint: Infrastructure Checkpoint | **Milestone:** Deploy a Scalable LLM Inference Platform

Skills: Kubernetes, vLLM, CI/CD | **Tools:** vLLM, SGLang, Docker, Kubernetes, Helm, AWS EKS

15-A: Inference Serving & Cloud Orchestration

Deploying high-throughput model endpoints on AWS EKS, GCP Cloud Run and Kubernetes clusters.

Deploy high-throughput model-serving infrastructure using modern inference engines, Kubernetes and cloud-native CI/CD.

Inference Serving

- vLLM
- SGLang
- llama.cpp
- High-throughput inference
- Dynamic LoRA loading
- Multi-adapter serving
- Speculative decoding
- Quantized inference
- GGUF
- AWQ
- GPTQ
- Continuous batching

Cloud Infrastructure

- AWS EKS
- AWS ECR
- AWS Batch
- ECS Fargate
- Kubernetes clusters
- Helm charts
- Cloud-native AI architecture

Containerization & CI/CD

- Docker
- Multi-stage Docker builds
- Lightweight AI containers
- GitHub Actions
- AI application CI/CD
- Automated deployments
- Production configuration management

Key outcome: Deploy a scalable LLM inference platform on Kubernetes with automated container builds and CI/CD.

AI Security, Guardrails & RBAC

Prompt injection defense, NVIDIA NeMo Guardrails, LlamaFirewall, Presidio PII, LiteLLM gateways and retrieval/prompt RBAC.

Checkpoint: Senior AI Engineer & Architect Checkpoint | **Milestone:** Deploy Secure Enterprise AI Infrastructure

Skills: AI Security, Guardrails, RBAC, Multi-Tenancy | **Tools:** NeMo Guardrails, LiteLLM, Presidio

16-A: Guardrails, LLM Gateways & RBAC Security

Securing AI applications against prompt injection, data poisoning, PII leaks and unauthorized document access.

Secure enterprise AI applications against prompt injection, jailbreaks, data poisoning, PII exposure and unauthorized information access.

Guardrails & Defense

- Direct prompt injection
- Indirect prompt injection
- Jailbreak prevention
- PII filtering
- Toxicity filtering
- Output validation
- NVIDIA NeMo Guardrails
- Colang
- LlamaFirewall
- LLM Guard
- AWS Bedrock Guardrails
- Microsoft Presidio
- PII detection
- PII redaction
- Data poisoning defense

LLM Gateways & Routing

- LLM gateway architecture
- Load balancing
- Automatic retries
- Gateway metadata
- Gateway observability
- Gateway caching
- LiteLLM Proxy
- AWS Bedrock routing
- Provider routing
- Model fallback strategies

Identity, RBAC & Enterprise Security

- Identity fundamentals
- Authentication
- Authorization
- SAML assertions
- JWT
- RBAC
- Retrieval-layer RBAC
- Document-level tenant filtering
- Prompt-layer RBAC
- Dynamic system prompts by role

- Multi-tenancy
- Tenant isolation
- Data isolation guarantees

Security Evaluation

- RAG security evaluation
- Agent security evaluation
- Prompt injection stress testing
- Unauthorized retrieval testing
- PII leakage testing
- Tool authorization testing
- Multi-tenant isolation testing

Key outcome: Deploy enterprise AI behind guardrails, secure gateways, PII protection and tenant-aware RBAC controls.