

# Course Summary & Future Trends

Week 15 · CS 203: Software Tools and Techniques for AI

Prof. Nipun Batra

*IIT Gandhinagar*

# The Journey We've Taken

## Part 1: The Data Foundation (Weeks 1-5)

- Week 1: Data Collection (Web Scraping, HTTP APIs)
- Week 2: Data Validation (Pydantic, Schema Enforcement)
- Week 3: Data Labeling (Label Studio, Annotation Workflows)
- Week 4: Active Learning (Smart Data Selection)
- Week 5: Data Augmentation (Synthetic Data Generation)

## Part 2: Model Engineering (Weeks 6-8)

- Week 6: LLM APIs (OpenAI, Anthropic, Prompt Engineering)
- Week 7: Model Development (Training, Fine-tuning, PEFT/LoRA)
- Week 8: Reproducibility (Docker, MLflow, DVC)

# The Journey (Continued)

## Part 3: Deployment & MLOps (Weeks 9-14)

- Week 9: Interactive Demos (Streamlit, Gradio)
- Week 10: HTTP APIs (FastAPI, REST Principles)
- Week 11: Git & CI/CD (GitHub Actions, Automated Testing)
- Week 12: Edge Deployment (Quantization, Pruning, ONNX)
- Week 13: Profiling & Optimization (Performance Tuning)
- Week 14: Model Monitoring (Drift Detection, Observability)

# The Full Stack AI Engineer

You now have the toolkit to build end-to-end systems:

1. **Scrape** data from the web (requests, BeautifulSoup)
2. **Validate** it rigorously (Pydantic schemas)
3. **Label** efficiently (Label Studio, Active Learning)
4. **Train** a model (PyTorch, fine-tune LLMs with LoRA)
5. **Package** it reproducibly (Docker, requirements.txt)
6. **Serve** it via APIs (FastAPI, async endpoints)
7. **Deploy** with CI/CD (GitHub Actions, automated tests)
8. **Optimize** for production (quantization, profiling)
9. **Monitor** for drift (Evidently AI, Prometheus)

# Week-by-Week Key Lessons

## Week 1 (Data Collection)

- **Lesson:** Good data beats fancy algorithms
- **Tool:** `requests`, BeautifulSoup, Selenium
- **Pitfall:** Violating robots.txt, missing rate limiting

## Week 2 (Data Validation)

- **Lesson:** Validate early, fail fast
- **Tool:** Pydantic for schema enforcement
- **Pitfall:** Trusting external data without validation

## Week 3 (Data Labeling)

- **Lesson:** Quality > Quantity for labels
- **Tool:** Label Studio for annotation

# Week-by-Week Key Lessons (2)

## Week 4 (Active Learning)

- **Lesson:** Let the model tell you what to label next
- **Tool:** Uncertainty sampling, query strategies
- **Pitfall:** Using random sampling when data is expensive

## Week 5 (Data Augmentation)

- **Lesson:** Synthetic data can fill gaps in real data
- **Tool:** albumentations, TextAttack, nlpaug
- **Pitfall:** Unrealistic augmentations that hurt generalization

## Week 6 (LLM APIs)

- **Lesson:** Start with APIs before self-hosting LLMs
- **Tool:** OpenAI API, Anthropic Claude API

# Week-by-Week Key Lessons (3)

## Week 7 (Model Development)

- **Lesson:** Start with baselines, iterate systematically
- **Tool:** PyTorch, Optuna for hyperparameter tuning
- **Pitfall:** Jumping to complex models without baselines

## Week 8 (Reproducibility)

- **Lesson:** "Works on my machine" is not acceptable
- **Tool:** Docker, MLflow, DVC for versioning
- **Pitfall:** Not pinning dependency versions

## Week 9 (Interactive Demos)

- **Lesson:** Demos accelerate feedback loops
- **Tool:** Streamlit for dashboards, Gradio for quick UIs

# Week-by-Week Key Lessons (4)

## Week 10 (HTTP APIs)

- **Lesson:** REST APIs are the universal interface
- **Tool:** FastAPI with automatic OpenAPI docs
- **Pitfall:** Missing input validation and error handling

## Week 11 (Git & CI/CD)

- **Lesson:** Automate everything that can be automated
- **Tool:** GitHub Actions for CI/CD pipelines
- **Pitfall:** Not testing before deploying

## Week 12 (Edge Deployment)

- **Lesson:** Optimization is required for resource-constrained devices
- **Tool:** ONNX Runtime, quantization, pruning

# Week-by-Week Key Lessons (5)

## Week 13 (Profiling & Optimization)

- **Lesson:** Measure first, optimize second
- **Tool:** PyTorch Profiler, cProfile, line\_profiler
- **Pitfall:** Premature optimization

## Week 14 (Model Monitoring)

- **Lesson:** Models degrade over time in production
- **Tool:** Evidently AI, Prometheus, Grafana
- **Pitfall:** No alerts for drift or performance degradation

# Common Pitfalls Across the Course

## Data Issues:

- Not validating data schemas early
- Collecting biased or unrepresentative data
- Ignoring class imbalance

## Model Issues:

- No baseline comparison
- Overfitting on small datasets
- Not using cross-validation

## Engineering Issues:

- Hardcoding configurations
- Not using version control for data

# Integration: How Weeks Connect

## Data Pipeline (Weeks 1-5):

Raw Web Data → Validation → Labeling → Active Learning → Augmentation → Clean Dataset

## Model Pipeline (Weeks 6-8):

Clean Dataset → LLM API / Training → Experiment Tracking → Reproducible Model

## Deployment Pipeline (Weeks 9-14):

Model → Demo (Streamlit) → API (FastAPI) → CI/CD → Optimization → Monitoring

**Key Insight:** Each week builds on previous weeks. You need ALL pieces for production ML.

# Real-World Case Study: Image Classification Service

**Scenario:** Build a production image classifier for plant disease detection.

**Week 1-2:** Scrape plant images, validate image formats/sizes

**Week 3-4:** Label diseases, use active learning for rare classes

**Week 5:** Augment with rotations, crops (realistic transformations)

**Week 7:** Fine-tune ResNet-50 with transfer learning

**Week 8:** Package in Docker, track experiments with MLflow

**Week 9:** Build Streamlit demo for farmers

**Week 10:** Create FastAPI endpoint for mobile app

**Week 11:** Set up CI/CD to auto-test on new data

**Week 12:** Quantize model to INT8 for mobile deployment

**Week 13:** Profile and optimize inference latency

**Week 14:** Monitor for drift as seasons change

# Real-World Case Study: Text Classification API

**Scenario:** Build sentiment analysis API for customer reviews.

**Week 1-2:** Scrape reviews, validate JSON schemas

**Week 3-4:** Label sentiment, use uncertainty sampling

**Week 5:** Augment with back-translation, synonym replacement

**Week 6:** Start with OpenAI API for prototyping

**Week 7:** Fine-tune DistilBERT with LoRA for cost savings

**Week 8:** Containerize with Docker, track with W&B

**Week 9:** Build Gradio demo for stakeholders

**Week 10:** Deploy FastAPI with rate limiting

**Week 11:** Automate testing and deployment

**Week 12:** Convert to ONNX for faster inference

**Week 13:** Profile and enable batch processing

**Week 14:** Monitor for concept drift (changing language patterns)

# Best Practices: The Golden Rules

## Data:

1. Always validate inputs (Pydantic)
2. Version your datasets (DVC)
3. Measure label quality (inter-annotator agreement)

## Models:

4. Start simple, baseline first
5. Track all experiments (MLflow/W&B)
6. Use cross-validation, not single train/test split

## Code:

7. Pin all dependency versions
8. Use type hints and docstrings
9. Write tests before deploying

## Production:

10. Monitor everything (metrics, logs, drift)
11. Automate testing and deployment (CI/CD)

# Career Paths in AI/ML

## ML Engineer:

- Focus: Training and deploying models
- Skills: PyTorch, TensorFlow, MLOps tools
- This course: Weeks 6-8, 12-14

## MLOps Engineer:

- Focus: Infrastructure and automation
- Skills: Docker, Kubernetes, CI/CD
- This course: Weeks 8, 11, 14

## Data Engineer:

- Focus: Data pipelines and infrastructure
- Skills: Data validation, ETL, databases

# Future Trends in AI Engineering

## 1. LLM Ops (LLOps)

- Managing prompts as code (version control for prompts)
- Eval-driven development (RAGAS, TruLens for LLM evaluation)
- Vector Database management (Chroma, Pinecone, Weaviate)
- Prompt caching and optimization
- **Emerging tools:** LangChain, LlamalIndex, DSPy

## 2. AI Agents

- Systems that take action, not just chat
- Tool use and function calling (ReAct pattern)
- Planning and reasoning (Chain-of-Thought, Tree-of-Thoughts)
- Multi-agent systems and collaboration
- **Emerging tools:** LangGraph, AutoGPT, CrewAI

# Future Trends (Continued)

## 3. Edge AI & Small Language Models

- Running SLMs on phones/laptops (Phi-3, Gemma, Llama 3.2)
- ExecuTorch (PyTorch for mobile/edge)
- MLX (Apple Silicon optimization)
- WebGPU for browser-based inference
- **Use cases:** Offline translation, on-device assistants

## 4. Multimodal AI

- Vision + Language models (GPT-4V, Claude 3, Gemini)
- Speech + Vision + Text integration
- Video understanding and generation
- **Tools:** CLIP, Whisper, Stable Diffusion

## 5. AI Safety & Alignment

# Project Ideas: Beginner Level

## 1. Personal Document Q&A System

- Week 1-2: Upload and parse PDFs
- Week 6: Use LLM API for RAG (Retrieval Augmented Generation)
- Week 9: Build Streamlit interface
- **Complexity:** Low | **Impact:** High for personal productivity

## 2. Image Classification Web App

- Week 5: Augment limited dataset
- Week 7: Fine-tune pre-trained model
- Week 9-10: Streamlit demo + FastAPI backend
- **Complexity:** Medium | **Impact:** Portfolio piece

## 3. Sentiment Analysis API

# Project Ideas: Intermediate Level

## 4. Smart Web Scraper with Active Learning

- Week 1: Scrape e-commerce sites
- Week 4: Use active learning for price extraction
- Week 11: Automate with CI/CD to run daily
- **Complexity:** Medium | **Impact:** Practical automation

## 5. Plant Disease Detector (Mobile App)

- Week 3-5: Label and augment plant images
- Week 7: Train CNN with transfer learning
- Week 12: Quantize for mobile deployment
- Week 13: Optimize inference speed
- **Complexity:** High | **Impact:** Agriculture tech

## 6. Code Review Assistant

# Project Ideas: Advanced Level

## 7. Real-time Anomaly Detection System

- Week 1-2: Collect and validate streaming data
- Week 7: Train autoencoder for anomaly detection
- Week 13: Optimize for real-time processing
- Week 14: Monitor for drift with Evidently AI
- **Complexity:** Very High | **Impact:** Production ML system

## 8. Multi-Model Ensemble API

- Week 7: Train multiple models (CNN, Transformer, Gradient Boosting)
- Week 8: Package all models in Docker
- Week 10: FastAPI with model selection endpoint
- Week 14: A/B test models in production
- **Complexity:** Very High | **Impact:** Advanced MLOps

# Tools & Technologies Summary

## Data Tools:

- Collection: `requests`, `BeautifulSoup`, `Selenium`
- Validation: `Pydantic`, `pandera`
- Labeling: Label Studio, Prodigy
- Augmentation: `albumentations`, `nlpaug`, `TextAttack`

## Model Tools:

- Training: PyTorch, TensorFlow, Hugging Face Transformers
- Optimization: Optuna, Ray Tune
- Tracking: MLflow, Weights & Biases

## Deployment Tools:

- Containerization: Docker, Docker Compose

# Tools & Technologies Summary (2)

## Production Tools:

- Optimization: ONNX Runtime, TensorRT, OpenVINO
- Profiling: PyTorch Profiler, `cProfile`, `line_profiler`
- Monitoring: Evidently AI, Prometheus, Grafana, Sentry
- Versioning: Git, DVC

## Emerging Tools to Watch:

- LangChain/Llamaindex (LLM orchestration)
- Weights & Biases (experiment tracking evolution)
- Modal, Replicate (serverless ML deployment)
- Hugging Face Inference Endpoints
- Vertex AI, SageMaker (managed ML platforms)

# Learning Resources

## Courses & Books:

- "Designing Data-Intensive Applications" by Martin Kleppmann
- "Designing Machine Learning Systems" by Chip Huyen
- "Machine Learning Engineering" by Andriy Burkov
- fast.ai (Practical Deep Learning)
- Full Stack Deep Learning (FSDL)

## Newsletters:

- The Batch (DeepLearning.AI)
- Import AI (Jack Clark)
- TLDR AI
- Ahead of AI

## Podcasts:

# Learning Resources (2)

## Conferences:

- **MLOps focus:** MLOps World, apply()
- **Research:** NeurIPS, ICML, ICLR (Datasets & Benchmarks track)
- **Systems:** MLSys, SysML
- **Industry:** AI Summit, Gartner AI Summit

## Communities:

- r/MachineLearning, r/MLOps (Reddit)
- Hugging Face Discord
- MLOps Community Slack
- Papers with Code

## Practice Platforms:

# What We Didn't Cover (But You Should Learn)

## Infrastructure:

- Kubernetes for orchestration
- Terraform for infrastructure as code
- Airflow for workflow orchestration

## Advanced ML:

- Reinforcement Learning
- Federated Learning
- Self-supervised Learning

## Specialized Topics:

- Time series forecasting (ARIMA, Prophet, Temporal Fusion Transformers)
- Recommendation systems (collaborative filtering, matrix factorization)

# Key Takeaways

## 1. Data is King

- 80% of ML work is data collection, cleaning, and validation
- Good data > fancy algorithms
- Active learning and augmentation multiply your data value

## 2. Reproducibility is Non-Negotiable

- Pin versions, use Docker, track experiments
- Future you (and your team) will thank you

## 3. Start Simple, Iterate

- Baseline → Simple model → Complex model
- Profile before optimizing
- Monitor before scaling

# The Uncomfortable Truth

**Most ML projects fail not because of bad models, but because of bad engineering.** The model is often 5% of the code. The other 95% is data pipelines, validation, APIs, deployment, monitoring, and error handling. The course focused on this 95% because that's where projects succeed or fail. Models are easy. Systems are hard.

What people think ML is:

Model  
Training  
  
★ MAGIC ★

What ML actually is:

Data Collection (Week 1)  
Data Validation (Week 2)  
Data Labeling (Week 3)  
Active Learning (Week 4)  
Data Augmentation (Week 5)  
Model (Week 7) ← only here!  
Reproducibility (Week 8)  
Deployment (Weeks 9-11)  
Optimization (Week 12-13)  
Monitoring (Week 14)

# The ML Development Lifecycle

## Stage 1: Exploration

- Understand the problem
- Collect and explore data
- Build baselines

## Stage 2: Development

- Train and validate models
- Track experiments
- Optimize hyperparameters

## Stage 3: Deployment

- Package model (Docker, ONNX)
- Build API (FastAPI)

# Final Thoughts

## You've learned to:

- Build end-to-end ML systems from scratch
- Use industry-standard tools and frameworks
- Deploy models to production
- Monitor and maintain ML systems

## What's next?

- Build a portfolio project using these skills
- Contribute to open-source ML tools
- Join the MLOps community
- Keep learning - AI moves fast!

## Remember:

# Course Statistics

## What we covered:

- 14 weeks of content
- 15+ tools and frameworks
- 3 phases: Data, Models, Production
- 100+ code examples
- Dozens of best practices

## What you built:

- Web scrapers
- Data validation pipelines
- ML models (classical + deep learning + LLMs)
- REST APIs
- Deployment pipelines

# Staying Updated

## Daily:

- Follow key researchers on Twitter/X
- Browse Hugging Face daily papers
- Check r/MachineLearning

## Weekly:

- Read newsletters (The Batch, Import AI)
- Try new tools released on GitHub
- Participate in community discussions

## Monthly:

- Read a paper from ArXiv
- Build a small project with new tech

# Parting Wisdom

## From experienced ML engineers:

"Shipping a model to production teaches you more than any course." - Random ML Engineer

"Always have a baseline. You'd be surprised how often it wins." - Another ML Engineer

"If you can't explain it to a stakeholder, you don't understand it well enough." - Yet Another ML Engineer

"Monitor everything. The model you don't monitor is the one that breaks." - Wise MLOps Engineer

"The best model is the one that's in production." - Pragmatic ML Engineer

# Thank You!

"The best way to predict the future is to invent it." - Alan Kay

Keep building. Keep learning. Keep shipping.

**Questions?**

# Additional Resources

## Course materials:

- All lecture slides on GitHub
- Lab notebooks and solutions
- Example projects and code

## Recommended next steps:

1. Build a portfolio project
2. Contribute to open-source ML projects
3. Write blog posts about what you learned
4. Join MLOps communities
5. Keep experimenting with new tools

## Stay in touch: