ParhoAI

PARHOAI.COM

7 SPRINTS

Sprints

Short, focused courses for working engineers. Take the one you need, not the eight months around it.

7	9–15	3–5	1	52
SPRINTS	HOURS EACH	WEEKS EACH	PROJECT EACH	SESSIONS IN TOTAL

Seven sprints

Each sprint is a single subject taught to production standard, ending in one project that goes on your GitHub. Two evening sessions a week, ninety minutes each. **Together the seven are the three-month bootcamp** — take them in order for the full path, or take one and leave.

CODE	SPRINT	SESSIONS	HOURS	LENGTH	ASSUMES KNOWLEDGE OF
PML	Production ML	6	9 hrs	3 weeks	Programming · Machine learning · Tools
DLP	Deep Learning with PyTorch	8	12 hrs	4 weeks	Programming · Mathematics · Machine learning · Hardware
CV	Computer Vision	10	15 hrs	5 weeks	Programming · Skills · Machine learning · Hardware
NLP	NLP and Transformers	7	10.5 hrs	4 weeks	Programming · Skills · Machine learning · Hardware
RAG	RAG in Production	7	10.5 hrs	3 weeks	Programming · Machine learning · Accounts
AGT	AI Agents and Tool Use	6	9 hrs	3 weeks	Programming · Skills · Prior sprint · Accounts
OPS	MLOps and LLMOps	8	12 hrs	4 weeks	Programming · Assets · Accounts

WHAT DEPENDS ON WHAT


```
RAG → AGT          (no deep learning needed — start here if you build products)

OPS                (needs a trained model of your own, from anywhere)
```

RAG and AGT require no deep learning at all. Most engineers shipping AI features today need those two and nothing else — so they are deliberately placed off the main path.

SPRINT

PML

Production ML

The gap between a notebook model and one that ships.

6 SESSIONS

9 HOURS

3 WEEKS

WHO IT IS FOR

Engineers who can train a model but have never had one audited, deployed or challenged in review.

PREREQUISITES

Programming	Python at working level — functions, classes, virtual environments. pandas for loading and reshaping data.
Machine learning	You have trained and evaluated at least one supervised model. You know what a train/test split, overfitting and a confusion matrix are.
Tools	scikit-learn, Jupyter or an IDE. No GPU needed.
Not required	Deep learning. Mathematics beyond school algebra.

Not for people new to machine learning — take the diploma instead.

AFTER THIS SPRINT YOU CAN

- Choose a metric and a threshold from business cost, not from a tutorial
- Audit any pipeline for target leakage and prove the score is real
- Design validation that survives time, groups and distribution shift
- Explain a model to a stakeholder who will never open the notebook

01 **From notebook to production ML**

What breaks when a model leaves the notebook · assumptions that hold in training and fail in deployment · the model is 10% of the system

02 **Evaluation beyond accuracy**

Cost-sensitive thresholds · PR-AUC vs ROC-AUC under imbalance · probability calibration · decision curves · picking the metric from the business, not the paper

03 **Validation design and leakage audits**

Temporal, group and nested cross-validation · backtesting · auditing a pipeline for target leakage · why your 0.98 is a bug

04 **Boosting at a professional level**

XGBoost, LightGBM, CatBoost compared · Optuna · native categorical handling · early stopping · when to stop tuning

05 **Interpretability and trust**

SHAP · permutation importance · partial dependence · counterfactuals · explaining a model to someone who will not read the notebook

06 **Feature engineering + LAB** LAB

Target encoding without leakage · interactions · feature stores · leakage-safe pipelines · supervised build clinic

BUILD

A model that would survive review — leakage-audited, cost-thresholded, SHAP-explained, limitations stated.

SPRINT

DLP

Deep Learning with PyTorch

Train networks. Do not just call them.

8 SESSIONS

12 HOURS

4 WEEKS

WHO IT IS FOR

Engineers moving beyond scikit-learn who need to build and debug their own training loops.

PREREQUISITES

Programming	Python and NumPy. You think in arrays, not loops.
Mathematics	Matrices and dot products. Partial derivatives and the chain rule. You can read notation without panic.
Machine learning	The PML sprint, or equivalent — you have trained and evaluated a model before.
Hardware	GPU access. Colab free tier is enough to start.

No prior PyTorch assumed. Exposure to any framework helps.

AFTER THIS SPRINT YOU CAN

- Write, run and debug a training loop from scratch
- Diagnose a run that will not converge, and know why
- Choose losses, optimizers and schedules deliberately
- Run experiments with baselines, seeds and ablations, like a professional

01 **Networks and backpropagation**

Depth and representation · activations · initialization · vanishing and exploding gradients · when a network is the wrong choice

02 **PyTorch**

Tensors, devices, autograd · nn.Module · writing a training loop from scratch · checkpointing and resuming

03 **Losses, optimizers and schedules**

Cross-entropy, focal, custom losses · SGD, Adam, AdamW · learning-rate finding · warmup and cosine decay

04 **Regularization and normalization**

Dropout · weight decay · batch, layer and group norm · early stopping · what actually reduces overfitting in practice

05 **Data pipelines**

Dataset and DataLoader · transforms and augmentation · samplers for imbalance · workers, prefetching, and I/O bottlenecks

06 **Training at scale**

GPU memory accounting · batch size · mixed precision · gradient accumulation · multi-GPU in principle · Colab and cloud GPUs

07 **Debugging training runs**

Loss will not move · NaNs · silent label bugs · overfitting a single batch as a sanity check · gradient inspection

08 **Experiment discipline + LAB** LAB

Seeds and determinism · baselines before models · ablations · tracking runs · supervised build clinic

BUILD

A trained network with a full experiment log — curves, baseline, one ablation, reproducible seed.

SPRINT
CV

Computer Vision

Detection, tracking and video, on real cameras.

10 SESSIONS
15 HOURS
5 WEEKS

WHO IT IS FOR

Engineers building detection, tracking or video analytics on real footage and edge hardware.

PREREQUISITES

Programming	Python and PyTorch basics — tensors, nn.Module, autograd.
Skills	You can write, run and debug a training loop unaided.
Machine learning	The DLP sprint, or equivalent deep learning experience.
Hardware	GPU access. Plus a camera or your own video footage for the project.

Not an introduction to deep learning. Take DLP first if training loops are unfamiliar.

AFTER THIS SPRINT YOU CAN

- Collect, label and version your own image dataset
- Train a custom object detector and evaluate it honestly with mAP
- Handle live RTSP streams, dropped frames and latency budgets
- Deploy to edge hardware with ONNX or TensorRT and benchmark real FPS

01 **Images as tensors**

Pixels, channels, color spaces · resizing and normalization · OpenCV · classical CV where it still beats deep learning

02 **Convolutions and CNN architectures**

Kernels, stride, padding, receptive field · ResNet and skip connections · EfficientNet · Vision Transformers in brief

03 **Transfer learning**

Pretrained backbones · freezing and unfreezing · fine-tuning schedules · what to do when you have 200 images

04 **Data collection and labeling**

Sourcing and class balance · CVAT, Roboflow, Label Studio · label quality and inter-annotator agreement · dataset versioning

05 **Object detection**

Bounding boxes · IoU · non-max suppression · anchor vs anchor-free · mAP explained properly

06 **YOLO in practice**

Architecture overview · Ultralytics · inference · confidence and NMS thresholds · latency vs accuracy

07 **Training a custom detector**

Dataset formats · augmentation for detection · hyperparameters · evaluating on your own held-out data

08 **Segmentation, pose and keypoints**

Semantic vs instance segmentation · U-Net · Segment Anything · human pose · masks, IoU and Dice

09 **Video, tracking and live streams**

Frames vs streams · RTSP and ONVIF · buffering, dropped frames, latency budgets · ByteTrack · re-identification

10 **Edge deployment + LAB** LAB

ONNX · TensorRT · quantization and pruning · CPU and Jetson inference · FPS benchmarking · supervised build clinic

BUILD

A vision system on live video — custom detector, measured mAP and FPS, demo clip.

SPRINT

NLP

NLP and Transformers

Fine-tune models. Do not only prompt them.

7 SESSIONS
10.5 HOURS
4 WEEKS

WHO IT IS FOR

Engineers who need a model tuned to their own domain, language or task.

PREREQUISITES

Programming	Python and PyTorch basics.
Skills	Training loops. Fine-tuning as a concept, if not yet in practice.
Machine learning	The DLP sprint, or equivalent deep learning experience.
Hardware	GPU access. Colab Pro or better for the fine-tuning sessions.

Independent of the CV sprint. Tokenization is taught from scratch, not assumed.

AFTER THIS SPRINT YOU CAN

- Fine-tune a transformer for a specific task on your own data
- Use LoRA and QLoRA where full fine-tuning is unnecessary
- Build an evaluation set that survives contact with reality
- Handle multilingual and Urdu text without silent tokenization damage

01 Tokenization and embeddings

BPE and WordPiece · vocabularies · word2vec to sentence transformers · similarity · Unicode and Urdu pitfalls

02 Attention and the transformer

Queries, keys, values · multi-head attention · positional encoding · encoder vs decoder · BERT and GPT families

03 The Hugging Face ecosystem

Model hub · tokenizers · pipelines · datasets · reading a model card properly · licensing traps

04 Fine-tuning

Task heads · training arguments · freezing layers · catastrophic forgetting · overfitting on small datasets

05 LoRA and QLoRA

Parameter-efficient tuning · adapters · rank and alpha · merging weights · when full fine-tuning is unnecessary

06 Evaluating NLP systems

F1 and accuracy · BLEU and ROUGE · perplexity · human evaluation · designing a test set that survives contact

07 NLP build clinic LAB

Supervised build and evaluation of your own fine-tuned model

BUILD A fine-tuned language model with a baseline comparison, error analysis and model card.

SPRINT

RAG

RAG in Production

Retrieval that actually retrieves the right thing.

7 SESSIONS

10.5 HOURS

3 WEEKS

WHO IT IS FOR

Engineers building question-answering or knowledge systems over documents a business owns.

PREREQUISITES

Programming	Python. REST APIs and JSON. You have built or consumed an API.
Machine learning	Basic literacy only — what a model is, and what an embedding represents.
Accounts	An LLM API key, or a machine that can run a small open-weights model.
Not required	Deep learning. PyTorch. A GPU.

The most self-contained sprint we run. Start here if you build products rather than models.

AFTER THIS SPRINT YOU CAN

- Parse, chunk and index real documents including PDFs and tables
- Build hybrid retrieval with reranking, and return citations users can check
- Diagnose why a RAG system returns the wrong passage
- Measure retrieval quality separately from generation quality

01 **How LLMs work**

Pretraining and next-token prediction · instruction tuning · RLHF and DPO · what emerges, what does not, and what they will confidently invent

02 **Choosing a model**

Hosted vs open weights · size vs cost vs latency · licensing · reading benchmarks skeptically · when a small model is enough

03 **Embeddings and vector stores**

Embedding model choice · similarity metrics · pgvector, Qdrant, Chroma · index types · recall vs latency

04 **RAG I – ingestion**

Document parsing including PDFs and tables · chunking strategies · overlap · metadata design · incremental updates

05 **RAG II – retrieval**

Dense and hybrid retrieval · BM25 · reranking · query rewriting and expansion · returning citations users can check

06 **RAG III – failure modes**

Hallucination · retrieval misses · stale data · chunk boundary problems · measuring retrieval quality separately from generation

07

Evaluating LLM systems

Golden sets · LLM-as-judge and its limits · regression testing prompts · human review loops · what to log

BUILD A RAG system over a real document corpus, with citations and a measured retrieval score.

SPRINT

AGT

AI Agents and Tool Use

Systems that decide and act, with guardrails.

6 SESSIONS

9 HOURS

3 WEEKS

WHO IT IS FOR

Engineers putting tool-using LLM features into products people rely on.

PREREQUISITES

Programming	Python, including async. JSON schemas and typed validation.
Skills	API design and error handling. You have shipped something that calls another service.
Prior sprint	RAG recommended, not required.
Accounts	An LLM API key with a usable budget.

No deep learning, PyTorch or GPU required.

AFTER THIS SPRINT YOU CAN

- Get reliable structured output from a model that wants to write prose
- Design tool schemas and handle failure without the agent looping forever
- Choose between a prompt, a chain, an agent, and plain code
- Defend against prompt injection and keep a human in the loop where it matters

01 **Prompting as engineering**

System prompts · few-shot examples · chain of thought · decomposition · prompt versioning and diffing

02 **Structured output**

JSON mode · schemas · Pydantic validation · retries on malformed output · building type-safe LLM pipelines

03 **Tool and function calling**

Tool schemas · execution and error handling · sandboxing · deciding when a tool is genuinely needed

04 **Tool calling and agents**

Tool schemas · execution and error handling · sandboxing · ReAct · planning · loop control and stopping conditions

05 **Memory, multi-agent and orchestration**

Short vs long-term memory · conversation state · summarization · supervisor patterns · hand-offs · when it is plainly overkill

06 **Guardrails and safety + LAB** LAB

Prompt injection · jailbreaks · PII handling · content filtering · human in the loop · supervised build clinic

BUILD

An agentic application with tool calling, evaluation and a measured cost per query.

SPRINT
OPS

MLOps and LLMOps

Ship it, run it, and know the moment it breaks.

8 SESSIONS
12 HOURS
4 WEEKS

WHO IT IS FOR

Engineers responsible for models in production, and for the pager when they drift.

PREREQUISITES

Programming	Python, Git and the command line.
Assets	A trained model of your own to deploy. Bring one from any earlier sprint or from work.
Accounts	A cloud account — free tier is sufficient. GitHub for CI.
Not required	Prior Docker or Kubernetes. Docker is taught from first principles.

Not a DevOps introduction, but no DevOps background is assumed either.

AFTER THIS SPRINT YOU CAN

- Containerize and serve a model behind an API that survives load testing
- Build CI/CD that tests, builds and deploys without a human
- Detect data and concept drift before a user reports it
- Trace, version and evaluate LLM features where there is no fixed test set

01 Docker and packaging

Images and containers · Dockerfile · multi-stage builds · slim and GPU images · registries · reproducible environments

02 Serving models

FastAPI plus a model · batch vs real-time · ONNX Runtime · concurrency and workers · load testing before launch

03 CI/CD and cloud

GitHub Actions · test, build and deploy pipelines · environments and secrets · IAM and least privilege · controlling cost

04 Experiment tracking and registry

MLflow and Weights & Biases · runs, params, artifacts · model registry · staging and promotion · rolling back a bad model

05 Monitoring and drift

Metrics, logs and traces · data drift vs concept drift · alerting that fires · Prometheus and Grafana · shadow deployment

06 LLMOps

Why MLOps habits do not transfer · non-determinism · tracing with Langfuse or LangSmith · prompt version control · online evaluation

07 **Cost, caching and reliability**

Semantic caching · model routing · fallbacks and graceful degradation · rate limits and outages · cost per request as an SLO

08 **DEMO DAY**

Present the capstone · live demo · questions from a panel and invited guests

BUILD A deployed, monitored service with CI/CD, model registry and alerting that fires.