A Collaborative Initiative by **Open Minds Technologies** & **Discensys**

Generative AI Course Syllabus

Comprehensive · Practical · Industry-Ready

45	40	9	5	Mon-Fri
Total Sessions	Teaching Sessions	Weeks	Sessions / Week	Schedule

Course Overview

Program Structure

This course runs across 9 weeks (45 sessions, Mon-Fri). Weeks 1-8 are 40 dedicated teaching sessions covering 13 modules -- from generative model foundations through VAEs, GANs, Diffusion Models, Transformers, LLMs, Prompt Engineering, Multimodal AI, Fine-Tuning, RAG, and Production Deployment. Week 9 (sessions 41-45) is the Capstone & Career Week -- project development, a Resume & Career Readiness Workshop, final presentations, and course wrap-up.

Prerequisite: *Machine Learning fundamentals are required -- Python, NumPy, pandas, sklearn, and core ML concepts (supervised learning, cross-validation, model evaluation). Completion of the Open Minds Technologies ML course is strongly recommended.*

Assignment Submission -- Centralised GitHub Organisation Repo

All assignments -- daily, weekly, bi-weekly, and the capstone -- are submitted as pull requests to the centralised course organisation repository on GitHub. Each learner works on their own named branch, pushes their work, and opens a PR for instructor review. This mirrors exactly how professional AI engineering teams collaborate on shared codebases.

Daily assignments are pushed to the learner's branch and a PR is opened by end of each session. Weekly assignments are submitted as a PR at the end of each week, reviewed by the instructor before the next week begins. Bi-weekly team assignments are raised from a shared team branch. The capstone project lives in its own dedicated repo under the organisation. This single centralised workflow gives instructors full visibility of every learner's progress in one place -- and gives learners a real-world PR review experience from day one.

Daily Assignments (Individual)

Every session has a daily assignment -- short, targeted coding tasks that reinforce the exact concept taught that day. Pushed to the learner's branch on the org repo and a PR opened by end of day. These are the primary vehicle for developing hands-on proficiency.

Weekly Assignments (Individual)

At the end of each teaching week, learners raise a PR on the org repo covering all topics from that week. The instructor reviews, leaves inline comments, and either requests changes or merges -- giving learners structured, code-level feedback on their understanding.

Bi-Weekly Assignments (Team)

Every two weeks, teams push their collaborative solution to a shared team branch on the org repo and raise a PR. The team manages the branch together -- dividing work, reviewing each other's commits, and resolving conflicts before the PR is submitted. Mirrors real team delivery.

Capstone Project (Team) -- Week 9 (Sessions 41-45)

Teams build an end-to-end Generative AI application -- from model selection and fine-tuning through to a deployed, production-ready API. The project lives in a dedicated repo under the organisation. Mentor reviews happen via PR comments. Certificate of completion issued upon successful delivery and final presentation.

Assessment Type	Frequency	Submission Method	Mode
Daily Assignment	Every session	Push to learner branch + PR on org repo	Individual

Weekly Assignment	End of each teaching week	PR on org repo -- instructor reviewed	Individual
Bi-Weekly Assignment	Every 2 weeks	Team branch PR on org repo	Team
Capstone Project	Week 9 (Sessions 41-45)	Dedicated org repo + Final Presentation	Team

Weekly Module Breakdown

Week 1 | Sessions 1-5

Foundations of Generative AI & Neural Network Basics

Mon	Tue	Wed	Thu	Fri
S1	**S2**	**S3**	**S4**	**S5**
Gen AI introduction -- generative vs discriminative, applications & landscape	Probability distributions, latent variables & model evaluation metrics	Neural networks review -- feedforward nets, activations, backprop & PyTorch	Autoencoders -- encoder-decoder architecture, reconstruction loss & hands-on	Latent space representation -- interpolation, visualisation & limitations

Module 1: Foundations of Generative AI

Sessions 1-2

- **What is Generative AI:**
  - Learning the data distribution P(X) to generate new, realistic samples
  - Generative vs Discriminative -- P(X) vs P(Y|X); fundamentally different learning objectives
  - Key applications -- image synthesis, text generation, audio creation, drug discovery, code generation
  - The generative AI landscape -- VAEs, GANs, Diffusion Models, Transformers, LLMs; when to use each
- **Probability foundations:**
  - Probability distributions -- Gaussian, Bernoulli, Categorical; how data distributions are modelled
  - Sampling -- ancestral sampling, rejection sampling; drawing new examples from a learned distribution
  - KL divergence -- measuring distance between two distributions; central to VAE training and RLHF
  - Monte Carlo estimation -- approximating intractable integrals via random sampling
- **Latent variable models:**
  - Observed variables X vs latent (hidden) variables Z -- the generative story
  - Generative process: sample Z from prior P(Z), then generate X from P(X|Z)
  - Marginal likelihood P(X) requires integrating over all Z -- why exact inference is intractable
  - Posterior inference P(Z|X) -- the core learning challenge; approximation methods are necessary
- **Evaluation of generative models:**
  - Likelihood -- valid when tractable; cannot be computed for GANs and Diffusion Models
  - Frechet Inception Distance (FID) -- joint quality and diversity measure for image generation
  - Inception Score (IS) -- sharpness + variety; ignores the real data distribution
  - BLEU, ROUGE, BERTScore -- text evaluation; each captures different quality dimensions
  - Human evaluation -- gold standard for creative and open-ended generation tasks
  - No single metric is sufficient -- always use multiple complementary measures

Module 2: Neural Network Basics for Generation

Sessions 3-5

Prerequisite: ML fundamentals assumed -- Python, NumPy, pandas, sklearn

- **Feedforward networks review:**
  - Layers, weights, biases -- the computation graph; parameters learned via gradient descent
  - Activation functions -- ReLU, GELU, Sigmoid, Tanh; introducing non-linearity into the network
  - Forward pass -- input to hidden layers to output; matrix multiplications + activations
  - Backpropagation -- chain rule; computing gradients with respect to every parameter layer-by-layer

Vanishing and exploding gradients -- batch normalisation, skip connections, gradient clipping as remedies

- **PyTorch for deep learning:**
  Tensors -- creation, shapes, dtype, device management (CPU/GPU); GPU acceleration with .cuda()
  nn.Module -- defining models; forward() method; composing layers with nn.Sequential
  Optimizers -- Adam, AdamW; learning rate scheduling (cosine decay, warmup steps)
  Training loop -- zero_grad, forward pass, loss computation, backward, optimizer step
  DataLoader -- batching, shuffling, num_workers; custom Dataset class for any data format
- **Autoencoders:**
  Encoder -- compresses input X to a bottleneck latent vector Z via learned mappings
  Decoder -- reconstructs X-hat from Z; learning the inverse Z to X mapping
  Reconstruction loss -- MSE for continuous data; BCE for binary or pixel data
  Bottleneck dimension -- controls compression vs information retention tradeoff
  Denoising autoencoders -- adds noise to input; forces robust, generalizable feature learning
  Limitations -- irregular latent space; cannot reliably sample novel data from Z
- **Latent space representation:**
  What the latent space captures -- semantic structure; similar inputs cluster nearby in Z
  Interpolation -- linear path between two Z points produces smooth semantic transitions
  t-SNE and UMAP -- 2D visualisation of high-dimensional latent spaces
  Disentanglement -- individual Z dimensions correspond to interpretable data factors
  Why plain autoencoders fail for generation -- no prior on Z; gaps in latent space produce garbage
- **Git & GitHub -- centralised org repo workflow from day one:**
  git clone -- clone the course organisation repo to your local machine
  git checkout -b firstname-lastname/daily/session-01 -- your personal branch naming convention
  git add, git commit, git push -- the daily assignment submission cycle
  Pull Requests -- open a PR from your branch to main after every assignment; instructor reviews inline
  .gitignore -- excluding venvs, __pycache__, .ipynb_checkpoints, model weights, .env files

Week 2 | Sessions 6-10

Variational Autoencoders & GANs -- Part 1

Mon	Tue	Wed	Thu	Fri
S6	**S7**	**S8**	**S9**	**S10**
VAE I -- probabilistic interpretation & reparameterization trick	VAE II -- KL divergence & ELBO objective	VAE III -- PyTorch implementation, variants & limitations	GAN I -- generator & discriminator, minimax game formulation	GAN II -- training instability, mode collapse & DCGAN architecture

Module 3: Variational Autoencoders (VAE)

Sessions 6-8

- **Probabilistic interpretation:**
  Encoder as approximate posterior q(Z|X; phi) -- outputs mean (mu) and std (sigma) of Z
  Decoder as likelihood P(X|Z; theta) -- generates reconstruction from a sampled Z
  Goal -- learn parameters theta and phi to maximise data log-likelihood log P(X)
  Intractability -- marginal P(X) = integral of P(X|Z)P(Z) dZ requires approximation
- **Reparameterization trick:**
  Direct sampling Z ~ N(mu, sigma^2) is non-differentiable -- gradients cannot flow back through it
  Rewrite as Z = mu + sigma * eps, eps ~ N(0,1) -- stochasticity shifted to eps; mu and sigma stay differentiable
  Enables end-to-end backpropagation through the sampling operation -- critical for training VAEs
- **KL Divergence:**
  Measures divergence between approximate posterior q(Z|X) and standard prior P(Z) = N(0,1)
  KL term regularises latent space -- pulls the posterior toward a compact, smooth Gaussian prior
  Closed-form for Gaussians: KL = -0.5 * sum(1 + log sigma^2 - mu^2 - sigma^2)
  KL annealing -- gradually increase KL weight during training; prevents posterior collapse early on
  Posterior collapse -- decoder ignores Z entirely; occurs when KL weight overpowers reconstruction
- **ELBO Objective:**

ELBO = E[log P(X|Z)] - KL(q(Z|X) || P(Z))
Reconstruction term -- how faithfully X is reconstructed from a sampled Z
Regularisation term -- how close the learned posterior is to the prior
Maximising ELBO = maximising a lower bound on log P(X); tightens as the posterior improves
Beta-VAE -- ELBO with beta > 1 on KL; stronger disentanglement at cost of reconstruction quality

- **VAE variants & limitations:**
  Conditional VAE (CVAE) -- condition encoder & decoder on class label; enables controlled generation
  VQ-VAE -- discrete latent codes via vector quantisation; used in DALL-E 1 and AudioLM
  Blurry outputs -- MSE reconstruction loss averages over modes; misses sharp high-frequency detail
  VAEs vs GANs vs Diffusion -- VAEs: stable training, interpretable Z, blurry outputs; Diffusion: highest quality

Module 4: Generative Adversarial Networks (GANs)

Sessions 9-11

- **Generator and Discriminator:**
  Two-network adversarial game -- G and D trained simultaneously against each other
  G: noise Z maps to synthetic sample X_fake; aims to fool D into classifying fakes as real
  D: binary classifier -- real (1) vs generated (0); trained to tell them apart
  Alternating training -- D updated k steps per G update; maintaining balance is key
- **Minimax Game:**
  Objective: min_G max_D E[log D(X_real)] + E[log(1 - D(G(Z)))]
  D maximises -- correctly classifying real as real and fake as fake
  G minimises -- making D classify G(Z) as real; non-saturating loss used in practice
  Nash equilibrium -- optimal G produces a distribution indistinguishable from real data
  Why difficult -- saddle point optimisation; no convergence guarantee in practice
- **Training Instability & DCGAN:**
  Mode collapse -- G produces limited variety; ignores large parts of the real data distribution
  Vanishing gradients for G -- when D is too strong; G receives near-zero gradient signal
  Practical fixes -- label smoothing (0.9 not 1.0), instance noise, spectral normalisation
  DCGAN -- convolutional G (transposed convolutions) and D (strided convolutions); no FC hidden layers
  DCGAN rules -- batch normalisation in both G and D; LeakyReLU in D; ReLU in G; Tanh output

Week 3 | Sessions 11-15

GANs -- Completion & Diffusion Models

Mon	Tue	Wed	Thu	Fri
S11	**S12**	**S13**	**S14**	**S15**
GAN III -- WGAN, StyleGAN & conditional GANs	Diffusion I -- forward process, noise scheduling & Markov chain	Diffusion II -- DDPM, score-based models & reverse process	Diffusion III -- Stable Diffusion, CLIP conditioning & CFG	Diffusion IV -- DDIM sampling, implementation walkthrough & hands-on

Module 4: GANs -- continued

Session 11

- **WGAN (Wasserstein GAN):**
  Wasserstein-1 distance -- more meaningful metric when real and fake supports do not overlap
  Critic -- unbounded real-valued score replaces the sigmoid discriminator
  Lipschitz constraint -- weight clipping (original) or gradient penalty (WGAN-GP); stabilises training
  More stable training -- meaningful gradient for G even when distributions are far apart
- **StyleGAN:**
  Mapping network -- 8-layer MLP maps noise Z to intermediate latent W; disentangles style factors
  AdaIN (Adaptive Instance Normalisation) -- injects W at each conv layer to control style at that scale
  Stochastic variation -- per-pixel Gaussian noise at each layer for fine texture details
  Style controls scale -- coarse layers control shape and pose; fine layers control texture and colour
- **Conditional GAN (cGAN):**
  Condition G and D on class label, attribute embedding, or reference image

Applications -- class-conditional image generation, image-to-image translation (Pix2Pix, CycleGAN)
Why GANs matter historically -- understanding adversarial training underpins modern alignment concepts
GANs in 2025+ -- largely superseded by Diffusion for images; still used in video, audio, domain adaptation

Module 5: Diffusion Models

Sessions 12-15

- **Forward Diffusion Process:**
  - Gradually adds Gaussian noise over T timesteps -- a fixed Markov chain on data X
  - q(X_t | X_{t-1}) = N(X_t; sqrt(1-beta_t) * X_{t-1}, beta_t * I) -- each step adds small noise
  - Noise schedule -- linear (original DDPM) or cosine; controls how fast the signal is destroyed
  - Closed-form: X_t = sqrt(alpha_bar_t) * X_0 + sqrt(1-alpha_bar_t) * eps -- jump to any timestep
  - By step T = 1000, X_T approximates N(0,I) -- X_0 is fully destroyed into pure Gaussian noise
- **Reverse Diffusion Process:**
  - Learn to denoise -- neural network predicts X_{t-1} given noisy X_t and timestep t
  - Parameterize as noise prediction eps_theta(X_t, t) -- more stable than predicting X_0 directly
  - At inference: start from pure Gaussian noise X_T; iteratively apply learned denoising T times
  - U-Net backbone -- skip connections preserve spatial detail; timestep embedding injected at each block
- **DDPM (Denoising Diffusion Probabilistic Models):**
  - Training -- uniformly sample t; add noise to X_0; train network to predict the added noise eps
  - Loss -- MSE between predicted noise and actual noise; simple and very stable to optimise
  - Sampling -- 1000 denoising steps; high quality but slow at inference time
  - DDIM (Denoising Diffusion Implicit Models) -- deterministic sampling; 50-100 steps without quality loss
- **Score-based Models:**
  - Score function grad_x log P(x) -- gradient of log-density; points toward higher-probability regions
  - Score matching -- train a neural net to estimate the score function directly from data
  - Langevin dynamics -- iterative refinement using estimated score + injected noise
  - SDE perspective -- forward and reverse diffusion as continuous stochastic differential equations
  - Equivalence -- score-based models and DDPM are two views of the same underlying framework
- **Stable Diffusion:**
  - Latent Diffusion Model (LDM) -- diffusion in VAE latent space, not pixel space; ~8x faster
  - VAE encoder/decoder -- compress 512x512 image to 64x64x4 latent; decompress after denoising
  - U-Net denoiser -- cross-attention layers incorporate text conditioning at every resolution level
  - CLIP text encoder -- embeds prompt text; fed into U-Net cross-attention to guide denoising
  - Classifier-Free Guidance (CFG) -- blend conditional and unconditional score; controls quality vs diversity
  - CFG scale -- low (1-3): loose and diverse; high (7-15): sharp and tightly prompt-adherent

Week 4 | Sessions 16-20

Transformers for Generation & LLMs -- First Contact

Mon	Tue	Wed	Thu	Fri
S16	**S17**	**S18**	**S19**	**S20**
Transformers I -- self-attention & multi-head attention mechanism	Transformers II -- decoder-only architecture, positional encoding & KV cache	Transformers III -- autoregressive generation & sampling strategies	Transformers IV -- scaling laws, architecture variants & emergent abilities	LLMs I -- first LLM API call hands-on + pretraining objectives & web-scale data

Module 6: Transformers for Generation

Sessions 16-19

- **Self-Attention Mechanism:**
  - Attention as soft retrieval -- Query asks, Key matches, Value is the returned content
  - Scaled dot-product: Attention(Q,K,V) = softmax(QK^T / sqrt(d_k)) * V
  - Captures long-range dependencies -- any two positions interact directly; no recurrence bottleneck
  - Quadratic complexity O(n^2) in sequence length -- efficient variants (FlashAttention) for long contexts
  - Attention patterns -- different heads learn syntactic, semantic, and positional relations
- **Multi-Head Attention:**

H parallel attention heads -- each projects Q, K, V to a lower-dimensional subspace
Each head attends to different aspects -- syntax, coreference, positional structure simultaneously
Concatenate head outputs then linear projection back to model dimension d_model
Richer representation than single-head attention of the same total compute budget

- **Positional Encoding:**
  Attention is permutation-invariant -- must explicitly inject position information into tokens
  Sinusoidal encoding -- fixed; deterministic; generalises to sequences longer than seen in training
  Learned positional embeddings -- trainable; used in GPT-2; limited length generalisation
  RoPE (Rotary Position Embedding) -- encodes relative position via rotation; used in LLaMA and Mistral
  ALiBi -- linear attention bias by distance; strong extrapolation to longer sequences
- **Decoder-only Architecture:**
  Causal (masked) self-attention -- each token attends only to past tokens; triangular mask enforced
  Enables autoregressive generation -- predict one token at a time, strictly left to right
  Used in GPT-2/3/4, LLaMA, Mistral -- dominant paradigm for open-ended language generation
  Layer structure -- pre-norm: RMSNorm, attention, residual; RMSNorm, FFN, residual
  KV cache -- store past key/value tensors during inference; avoids recomputation; critical for speed
- **Autoregressive Generation & Sampling:**
  Generate one token at a time -- P(x_t | x_1...x_{t-1}); product gives full sequence probability
  Greedy decoding -- always pick argmax; fast but repetitive and sub-optimal
  Temperature -- divide logits by T before softmax; T > 1 creative, T < 1 focused and deterministic
  Top-k sampling -- sample only from the top k most probable tokens; removes long-tail noise
  Top-p (nucleus) sampling -- dynamic top set covering cumulative probability mass p; more adaptive
  Repetition penalty -- downweight previously generated tokens; reduces looping and repetition
- **Scaling Laws & Architecture Variants:**
  Chinchilla scaling laws -- tokens = 20x parameters for compute-optimal training efficiency
  Performance scales predictably with compute C, data D, and model parameters N
  Emergent abilities -- multi-step reasoning and arithmetic appear suddenly at sufficient scale
  Mixture of Experts (MoE) -- sparse activation; Mixtral 8x7B; efficient parameter scaling
  Encoder-decoder (T5, BART) -- better for seq2seq tasks; decoder-only dominates open generation

Module 7: Large Language Models (LLMs)

Sessions 20-24

- **First LLM API Call -- hands-on before theory:**
  Call GPT-4o-mini or Claude API in Python -- see the model respond in under 5 lines of code
  Set up API keys securely -- .env file + python-dotenv; never hardcode credentials
  Inspect the response object -- role, content, finish_reason, usage tokens, model ID
  Try different prompts -- observe how prompt wording changes output drastically
  This session anchors the entire LLMs module -- practical first, theory second
- **Pretraining Objectives:**
  Causal Language Modelling (CLM) -- predict next token; used by GPT family; simple and scalable
  Masked Language Modelling (MLM) -- predict masked tokens; bidirectional context; BERT-style
  Span corruption -- mask and predict contiguous spans; used in T5 and UL2
  Web-scale data -- CommonCrawl, Books3, Wikipedia, GitHub, arXiv; aggressive quality filtering critical
  Data mixture -- domain balance shapes downstream capabilities; code data improves reasoning broadly

Week 5 | Sessions 21-25

Large Language Models -- Deep Dive & Prompt Engineering

Mon	Tue	Wed	Thu	Fri
S21	**S22**	**S23**	**S24**	**S25**
LLMs II -- tokenization (BPE, SentencePiece, tiktoken)	LLMs III -- supervised fine-tuning (SFT) & dataset preparation	LLMs IV -- instruction tuning & RLHF alignment	LLMs V -- open vs closed models, HuggingFace & Ollama	Prompt Eng. I -- design principles, zero-shot & few-shot learning

Module 7: LLMs -- continued

Sessions 21-24

- **Tokenization:**
  - Byte-Pair Encoding (BPE) -- merge frequent byte pairs iteratively; tiktoken used by GPT-4
  - SentencePiece -- language-agnostic; handles CJK, Arabic, Hindi without pre-tokenisation step
  - Vocabulary size -- 32K-128K tokens; larger vocab: shorter sequences but more embedding parameters
  - Tokenisation artefacts -- split words, leading spaces, case sensitivity; affects prompt design
  - Special tokens -- end-of-text, user, assistant; critical for chat fine-tuning format
  - Counting tokens before API calls -- tiktoken; estimate cost; prevent unexpected truncation
- **Supervised Fine-Tuning (SFT):**
  - Train on (prompt, response) pairs -- teach the model what high-quality responses look like
  - Learning rate -- much smaller than pretraining (1e-5 to 5e-5); warmup + cosine decay
  - Catastrophic forgetting -- full SFT may overwrite general capabilities; PEFT methods mitigate this
  - Dataset quality -- 100 high-quality examples often outperform 10,000 noisy ones
  - When to fine-tune vs prompt -- fine-tune for consistent format and style; prompt for flexibility
- **Instruction Tuning:**
  - Training on diverse instruction-following examples -- generalises to unseen task formats
  - FLAN -- template-based reformulation of 62 NLP datasets across many task types
  - Chat templates -- system/user/assistant turn structure; model must see this format during SFT
  - Multi-turn dialogue -- maintaining conversation context across turns; role tags are critical
- **RLHF & Alignment:**
  - Reward Model (RM) -- trained on human preference pairs; learns to score responses as humans would
  - Human preference data -- pairwise comparisons (preferred vs rejected); quality matters more than quantity
  - PPO (Proximal Policy Optimisation) -- fine-tune LLM to maximise RM score; clipping prevents large updates
  - KL penalty -- keeps fine-tuned model close to the reference policy; prevents reward hacking and collapse
  - Alignment goal -- Helpful, Harmless, Honest (HHH); reducing harmful and sycophantic outputs
  - RLHF limitations -- reward hacking; human labeller disagreement; expensive preference collection
  - DPO (Direct Preference Optimisation) -- no separate RM needed; trains directly on preference pairs; read the paper for the derivation
- **Open-source vs Closed Models:**
  - Open-source -- LLaMA 3, Mistral, Phi-3, Gemma, Qwen; fine-tuneable; local and private deployment
  - Closed -- GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro; API-only; highest capability tier currently
  - HuggingFace Hub -- model cards, weights, tokeniser configs; AutoModelForCausalLM pattern
  - Ollama -- run open models locally; simple REST API; great for development and privacy-sensitive data
  - Selection criteria -- task complexity, latency, cost, data privacy, customisation requirements

Week 6 | Sessions 26-30

Prompt Engineering -- Completion & Multimodal Generative AI

Mon	Tue	Wed	Thu	Fri
S26	**S27**	**S28**	**S29**	**S30**
Prompt Eng. II -- chain-of-thought & advanced prompting techniques	Prompt Eng. III -- system prompts, injection attacks & prompt optimisation	Multimodal I -- text-to-image models (DALL-E, Stable Diffusion, CLIP)	Multimodal II -- image-to-text, vision-language models & VQA	Multimodal III -- audio, video generation & cross-modal learning

Module 8: Prompt Engineering

Sessions 25-27

- **Prompt Design Principles:**
  - Clarity -- unambiguous task description; specify output format, length, and tone explicitly
  - Context -- provide relevant background; include examples of expected input and output pairs
  - Constraints -- what to avoid, required output schema, language and style specifications
  - Role assignment -- "You are a senior ML engineer..." -- shapes model behaviour consistently
  - Iterative refinement -- test on diverse examples; analyse failure modes; adjust systematically
- **Zero-shot & Few-shot Learning:**
  - Zero-shot -- model performs task from description alone; relies on instruction-following capability

Few-shot -- 1-8 input-output examples in the prompt; demonstrates format and expected style
Example selection -- diverse, representative, correctly labelled; avoid misleading patterns
Format consistency -- examples must exactly match expected output format; model learns by mimicry
In-context learning -- no weight updates; knowledge retrieved purely from the context window

Module 8: Prompt Engineering -- continued

Sessions 26-27

- **Chain-of-Thought (CoT) Prompting:**
  - "Let's think step by step" -- triggers explicit intermediate reasoning before the final answer
  - Zero-shot CoT -- single trigger phrase; effective on arithmetic, logic, and multi-step reasoning
  - Few-shot CoT -- provide full worked examples with complete reasoning chains in the prompt
  - Self-consistency -- sample multiple reasoning paths; majority-vote the final answer for reliability
  - Least-to-most prompting -- decompose complex problem into ordered sub-problems; solve sequentially
- **Advanced Prompting Techniques:**
  - Tree of Thought (ToT) -- explore multiple reasoning branches in parallel; evaluate and select best
  - ReAct -- interleave reasoning and tool use (web search, calculator, code execution) in one prompt
  - Retrieval-Augmented Prompting -- prepend retrieved context chunks; factual grounding for answers
  - DSPy -- declarative prompt programming; compiles and optimises prompts automatically
  - Soft prompts and prefix tuning -- trainable continuous embeddings prepended to input sequence
- **System Prompts & Safety:**
  - System prompt -- sets model persona, behaviour and constraints; persists across all conversation turns
  - Prompt injection attacks -- adversarial user inputs that override system instructions; defence patterns
  - Jailbreaking -- attempts to bypass safety guidelines; why alignment robustness remains hard
  - Output validation -- always validate LLM output format and content before downstream system use
  - Prompt optimisation -- APE (Automatic Prompt Engineer); iterative evaluation-based prompt search

Module 9: Multimodal Generative AI

Sessions 28-30

- **Text-to-Image Models:**
  - DALL-E 3 -- GPT-4 recaptioning for dense prompt adherence; diffusion backbone for generation
  - Stable Diffusion -- open-source latent diffusion; massive community fine-tune and ControlNet ecosystem
  - CLIP (Contrastive Language-Image Pretraining) -- aligns text and image in shared embedding space
  - Contrastive training -- 400M image-text pairs; maximise similarity of matched pairs; cosine objective
  - Prompt engineering for images -- art style, lighting, camera angle, composition, negative prompts
  - ControlNet -- conditioning on depth map, edge map, or pose; precise spatial control over output
- **Image-to-Text Models:**
  - Image captioning -- describe image content in natural language; encoder-decoder architecture
  - Visual Question Answering (VQA) -- answer free-form questions about image content
  - Vision-Language Models -- LLaVA, GPT-4o Vision, Gemini; image tokens projected into LLM space
  - BLIP-2 -- Q-Former bridges frozen vision encoder and frozen LLM; efficient cross-modal alignment
  - OCR capabilities -- reading text within images; document understanding and structured extraction
- **Audio, Video Generation & Cross-modal Learning:**
  - Text-to-speech (TTS) -- Tacotron 2, VITS; mel-spectrogram + neural vocoder pipeline
  - MusicGen -- autoregressive on EnCodec audio tokens; conditioned on text and melody
  - Voice cloning -- few seconds of reference audio maps to a personalised voice embedding for TTS
  - Video generation -- Sora (diffusion transformer, space-time patch tokens); Runway Gen-2 for practical use
  - Cross-modal contrastive learning -- align embeddings from different modalities via matched pairs
  - Unified tokenisation -- treat images (VQ-VAE tokens), audio (codec tokens), text as one sequence

Week 7 | Sessions 31-35

Fine-Tuning, Adaptation & Evaluation

Mon	Tue	Wed	Thu	Fri
S31	S32	S33	S34	S35

Fine-Tuning I -- full fine-tuning & dataset preparation	Fine-Tuning II -- LoRA, QLoRA & PEFT methods	Fine-Tuning III -- domain adaptation & instruction fine-tuning	Evaluation I -- metrics, LLM-as-judge & hallucination detection	Evaluation II -- bias, fairness, toxicity & alignment problems

Module 10: Fine-Tuning & Adaptation

Sessions 31-33

- **Full Fine-Tuning:**
  - Update all parameters -- most flexible; requires proportional compute to original pretraining
  - Memory requirements -- model weights + gradients + optimiser states; approximately 4x model size in FP32
  - Dataset preparation -- instruction format, quality filtering, deduplication, train/val split
  - Evaluation -- held-out task metrics; check for capability regression on general benchmarks
  - When appropriate -- small models, abundant labelled data, domain very different from pretraining
- **LoRA (Low-Rank Adaptation):**
  - Inject trainable rank-r matrices into frozen attention weight matrices of the base model
  - W_new = W + BA; B has shape (d, r), A has shape (r, k); only A and B are trained; base W frozen
  - Parameter reduction -- r=8 gives approximately 99% fewer trainable parameters vs full fine-tuning
  - Rank selection -- r=4 to 16 typical; higher r gives more capacity; diminishing returns beyond 64
  - Target modules -- q_proj, v_proj in attention layers; sometimes also k_proj, o_proj, FFN layers
  - Merging at inference -- W_new = W + BA; zero latency overhead compared to the base model
- **QLoRA:**
  - 4-bit quantisation of base model (NF4 format) -- dramatically reduces memory footprint
  - LoRA adapters computed in BFloat16 -- high-precision gradients despite quantised base weights
  - Double quantisation -- quantise the quantisation constants for further memory savings
  - Paged optimisers -- CPU offload for optimiser states; enables 70B models on single consumer GPU
  - Practical impact -- fine-tune 7B to 70B models on a single GPU with 24GB VRAM
- **Other PEFT Methods:**
  - Prefix Tuning -- prepend trainable soft token embeddings to K and V; no architecture change
  - Prompt Tuning -- trainable task prefix embedding only; lightest-weight PEFT approach
  - Adapter layers -- small bottleneck MLP inserted between transformer sub-layers; only adapters trained
  - IA3 -- scales attention keys, values and FFN activations; very few parameters, strong performance
- **Domain Adaptation & Instruction Fine-Tuning:**
  - Continued pretraining -- further train on unlabelled domain text; adapts vocabulary distribution
  - Domain data sources -- medical (PubMed), legal (court opinions), finance, code (GitHub)
  - Instruction datasets -- Alpaca, Dolly, OpenAssistant, ShareGPT; quality over quantity always
  - Prompt templates -- consistent format for all training examples; must match inference format exactly
  - Evaluation -- MT-Bench (multi-turn); MMLU (knowledge breadth); HumanEval (code generation)

Module 11: Evaluation & Safety

Sessions 34-35

- **Evaluation Metrics:**
  - BLEU -- n-gram precision; designed for translation; poor proxy for open-ended generation quality
  - ROUGE -- recall-oriented; used for summarisation; ROUGE-L captures longest common subsequence
  - Perplexity -- how well model predicts held-out text; lower is better; not always task-aligned
  - BERTScore -- semantic similarity via contextual embeddings; better than surface n-gram overlap
  - FID and IS -- Frechet Inception Distance and Inception Score; standard for image generation quality
  - LLM-as-judge -- GPT-4 or Claude evaluates outputs on defined criteria; scalable evaluation proxy
- **Hallucinations:**
  - Factual hallucination -- confident generation of false or unverifiable information
  - Faithfulness hallucination -- response contradicts the provided source context
  - Causes -- overconfident next-token prediction; knowledge gaps; reward hacking in RLHF
  - Detection -- NLI-based consistency checking; retrieval verification; TruthfulQA benchmark
  - Mitigation -- RAG for factual queries; citation requirements; uncertainty quantification
- **Bias, Fairness & Safety:**
  - Social biases -- gender, racial, religious stereotypes encoded from web-scale training data
  - Measurement benchmarks -- WinoBias, StereoSet, BBQ (Bias Benchmark for QA)
  - Toxic content categories -- hate speech, violence, self-harm; safety classifiers (Llama Guard)
  - Constitutional AI -- model uses written principles to self-critique and revise its own outputs

Red-teaming -- structured adversarial testing to find safety failures before deployment

- **Alignment Problems:**
  Goodhart's Law -- when a measure becomes a target it ceases to be a good measure
  Reward hacking -- model exploits the reward function without satisfying the true objective
  Sycophancy -- model agrees with user even when wrong; artefact of human-preference training signal
  Specification gaming -- maximising proxy objective while violating true human intent
  Ongoing research -- interpretability, scalable oversight, debate, iterated amplification

Week 8 | Sessions 36-40

Deployment, RAG in Depth & Capstone Kickoff

Mon	Tue	Wed	Thu	Fri
S36	**S37**	**S38**	**S39**	**S40**
Deployment I -- API-based deployment & FastAPI for LLM serving	Deployment II -- quantisation, KV cache, vLLM & latency optimisation	Deployment III -- real-world applications, monitoring & cost optimisation	RAG in Depth -- architecture, vector DBs, chunking, reranking & evaluation	Agents, Memory in LLMs & Capstone Kickoff -- project brief & team formation

Module 12: Deployment & Applications

Sessions 36-38

- **API-based Deployment:**
  OpenAI API -- GPT-4o, structured outputs, function calling, vision, assistants API
  Anthropic API -- Claude; tool use, extended thinking, vision, 200K token context window
  HuggingFace Inference API -- open models without self-hosting; serverless and dedicated endpoints
  Rate limits and cost management -- tokens/min limits; estimate cost before calling; set usage alerts
  API key security -- .env file + python-dotenv; never hardcode in source; rotate keys regularly
  Structured outputs -- JSON mode, response_format schema; reliable parsing for downstream systems
- **FastAPI for LLM Serving:**
  Loading open-source models -- AutoModelForCausalLM on startup; singleton pattern to avoid reload
  Streaming responses -- Server-Sent Events (SSE); token-by-token output for low perceived latency
  Request schema -- Pydantic models; input validation; structured error responses (422, 500)
  Authentication -- API key middleware; bearer token header; per-key rate limiting
  Async endpoints -- non-blocking inference; handle multiple concurrent generation requests
  Swagger UI at /docs -- auto-generated interactive API documentation; share with team instantly
- **Latency Optimisation:**
  Quantisation -- INT8 (bitsandbytes), GPTQ, AWQ; 2-4x smaller model with faster inference
  KV cache -- store past attention key/value tensors; avoids O(n^2) recomputation per token
  Speculative decoding -- small draft model proposes tokens; large model verifies batch in parallel
  vLLM -- PagedAttention manages KV cache memory like OS paging; high-throughput production serving
  Flash Attention -- IO-aware attention; avoids materialising full O(n^2) matrix in GPU memory
  Continuous batching -- dynamically batch requests of different lengths; maximise GPU utilisation
- **Real-world Applications & Cost Optimisation:**
  Chatbots and assistants -- multi-turn conversation; tool use; conversation memory management
  Code generation -- GitHub Copilot pattern; context window management; edit distance metrics
  Document processing -- extraction, summarisation, classification; structured output pipelines
  Monitoring and drift -- log requests/responses; track latency, error rate, and output quality
  Token counting -- tiktoken; estimate cost before calling; cache identical prompt responses
  Model selection -- GPT-4o-mini or Claude Haiku for simple tasks; frontier models for complex reasoning
  Prompt compression -- LLMLingua; remove redundant tokens while preserving semantic meaning

Module 13: RAG, Agents & Future Directions

Sessions 39-40

- **Retrieval-Augmented Generation (RAG) -- core pipeline:**
  Motivation -- LLM knowledge frozen at training cutoff; hallucinations on current and factual queries
  Architecture -- retriever finds relevant documents; LLM generates a grounded response from them

Dense retrieval -- DPR, Contriever; encode query and docs to vectors; cosine similarity search
Sparse retrieval -- BM25; TF-IDF based; efficient; strong baseline especially for keyword queries
Vector databases -- FAISS (local/free), Chroma (local dev), Pinecone/Qdrant (managed cloud); ANN search
Chunking strategy -- fixed size, sentence-level, semantic; chunk overlap maintains context continuity
Reranking -- cross-encoder reranks top-k retrieved chunks for precision before passing to LLM
Hybrid retrieval -- combine dense + sparse scores; gets benefits of both; standard in production RAG
RAGAS evaluation -- faithfulness, answer relevancy, context precision, context recall
Advanced RAG (self-study) -- HyDE, multi-hop RAG, corrective RAG; reference links in course repo

- **Agents and Tool Use:**
  - LLM as reasoning engine -- decides which action to take given a goal and available tools
  - Tool calling -- structured function calls: web search, calculator, code interpreter, external APIs
  - ReAct pattern -- Reasoning + Acting; interleaved thought and action traces in the context window
  - LangChain and LlamaIndex -- orchestration frameworks; chains, agents, memory components
  - Multi-agent systems -- planner agent + specialised executor agents; roles and handoffs
- **Memory in LLMs:**
  - In-context memory -- everything in the context window; bounded by token limit (4K to 1M)
  - External memory -- vector store; retrieve relevant past interactions via similarity search
  - Episodic memory -- selective storage of important context; recency and relevance scoring
  - Long-context models -- Gemini 1.5 (1M tokens), Claude (200K); tradeoffs in cost and latency
- **Future Directions:**
  - Multimodal foundation models -- any-to-any generation; embodied and action AI models
  - Efficient architectures -- Mamba (state space models); MoE (Mixture of Experts) at scale
  - AI agents at scale -- autonomous multi-step task completion; long-horizon planning
  - Interpretability -- understanding what LLMs know and how; mechanistic interpretability research
  - Regulation and governance -- EU AI Act; model cards; responsible disclosure; output watermarking

Week 9 | Sessions 41-45

Capstone & Career Week

Mon	Tue	Wed	Thu	Fri
S41	S42	S43	S44	S45
Capstone Work -- mentored project development & doubt-solving (session 1)	Capstone Work -- mentored project development & doubt-solving (session 2)	Resume, Portfolio & Career Readiness Workshop	Final Presentations -- team demos, Q&A & peer review	Course Wrap-up -- feedback, certificates & next steps

Capstone Project (Sessions 41-45)

Sessions 41-45

- **End-to-end Generative AI application delivery:**
  - Teams build a production-grade Gen AI application -- chatbot, RAG system, fine-tuned model API, or multimodal app
  - All work on the organisation GitHub repo -- PRs used for mentor review and integration throughout
  - Mentor-led doubt-solving sessions (S41 and S42) -- feedback delivered via PR comments
  - Mid-capstone checkpoint -- progress review; mentor merges or requests changes
- **Final Presentation (Session 44):**
  - 10-minute team demo + 5-minute Q&A
  - Present: problem statement, model choice and architecture, results, limitations, deployment
  - Peer review -- structured feedback form for cross-team assessment
  - Certificate of Completion issued upon successful delivery

Resume, Portfolio & Career Readiness Workshop (Session 43)

Session 43

- **Gen AI Resume Structure:**
  - One-page format for 0-3 years experience; two-page maximum for senior roles

Skills section -- Python, PyTorch, HuggingFace, LangChain, FastAPI, Docker, cloud platforms
Projects section -- problem, model and approach, result, impact (quantified where possible)
ATS optimisation -- keyword alignment with Gen AI job descriptions (LLM, RAG, fine-tuning, prompt engineering)

- **GitHub Portfolio:**
  By this point: 9 weeks of daily PRs on the org repo -- a live, reviewable learning track record
  Capstone repo under the org -- deployed Gen AI application with a live demo link
  Personal profile README -- pinned repos, tech stack summary, deployed project links
  Contribution graph -- consistent activity signals reliability and commitment to recruiters
  README structure for each project -- problem, model used, approach, results, how to run
- **LinkedIn Optimisation:**
  Headline -- "Generative AI Engineer | Python | HuggingFace | LangChain | FastAPI" style
  About section -- 3-5 sentences: background, what you build, key tools and what you are seeking
  Featured section -- link to deployed app, org GitHub repo, or a demo video
  Skills and endorsements -- request endorsements from cohort peers for Gen AI, LLMs, Python
- **Interview Guidance:**
  Technical rounds -- explain model architectures (Transformer, VAE, GAN, Diffusion), tradeoffs and training dynamics
  LLM case studies -- when to use RAG vs fine-tuning; how to handle hallucinations in production
  Coding rounds -- Python, PyTorch, HuggingFace; implement attention mechanism; write a training loop
  Behavioural -- STAR method; align stories to Gen AI project experience from the capstone
  Common questions: attention mechanism, RLHF, LoRA vs full fine-tuning, latency optimisation strategies