EXAM & INTERVIEW GUIDE

Deep Learning Cheatsheet Ebook

Neural Networks, CNNs, NLP, Transformers, LLMs, MLOps, Model Evaluation, AI Safety, and Production Prep

From fundamentals to production - master the deep learning topics that matter for exams and interviews

Lamhot Siagian

AI Engineering Insider • Free resources: aiengineeringinsider.substack.com/subscribe

How to use this ebook

Use this guide as a fast revision companion. Read the definition column first, then practice explaining the interview tip out loud in simple language. For exams, focus on formulas, definitions, and differences. For interviews, focus on tradeoffs, failure modes, and when to use each method.

The 5-minute deep learning mental model

Deep learning systems learn useful representations from data. A model receives inputs, transforms them through layers, produces predictions, compares predictions with targets using a loss function, computes gradients through backpropagation, and updates weights with an optimizer. Production deep learning also requires data pipelines, evaluation, deployment, monitoring, security, and continuous improvement.

Free Resources + Premium Guides

Subscribe for free resources: aiengineeringinsider.substack.com/subscribe
Explore premium and individual guides: beacons.ai/aiengineeringinsider

Contents

Chapter 1

Best Learning Order

Chapter Introduction

This chapter gives the recommended order for learning deep learning without feeling overwhelmed. Use it as your study checklist.

Step	Topic	What to Master
1	**Python**	Become comfortable with scripts, notebooks, packages, functions, and classes.
2	**Math for ML**	Focus on linear algebra, derivatives, gradients, probability, and optimization.
3	**Machine learning basics**	Learn supervised/unsupervised learning, splits, metrics, overfitting, and baselines.
4	**Neural networks**	Understand neurons, layers, activation, loss, forward pass, and backpropagation.
5	**PyTorch or TensorFlow**	Pick one primary framework and learn its training workflow deeply.
6	**CNNs**	Learn computer vision fundamentals and image model architectures.
7	**RNNs and LSTMs**	Understand sequence modeling before moving fully into Transformers.
8	**Transformers**	Master attention, encoder/decoder structures, and positional encoding.
9	**NLP and computer vision**	Apply models to real tasks with datasets and evaluation.
10	**Generative AI**	Study autoencoders, GANs, diffusion models, and generation tradeoffs.
11	**LLMs and RAG**	Learn embeddings, vector search, retrieval, prompts, agents, and evaluation.
12	**MLOps and deployment**	Package, serve, monitor, test, and version models.
13	**AI safety and evaluation**	Measure quality, safety, bias, robustness, and hallucinations.
14	**Advanced specialization**	Choose LLMs, CV, multimodal, RL, graph ML, audio, or edge AI.

Chapter 2

Foundations Before Deep Learning

Chapter Introduction

Build the math, programming, and ML base needed before training neural networks.

Topic	Simple Definition	Exam / Interview Tip
Linear algebra	Math of vectors, matrices, spaces, and transformations used to represent data and neural network parameters.	Expect matrix shapes, dot products, and why tensors matter.
Vectors	Ordered lists of numbers that represent features, embeddings, gradients, or model parameters.	Know vector dimension and similarity.
Matrices	Two-dimensional arrays used for batches, weights, transformations, and model computations.	Be ready to explain matrix multiplication in a layer.
Matrix multiplication	Operation that combines inputs and weights to produce linear transformations inside neural layers.	Core formula: output = input times weights plus bias.
Eigenvalues and eigenvectors	Values and directions that describe how a matrix stretches space; useful in PCA and stability analysis.	Useful for dimensionality reduction concepts.
Calculus	Math of change used to compute gradients and update neural network weights.	Backpropagation depends on calculus.
Derivatives	Measure how a function changes when one input changes.	Used to minimize loss.
Partial derivatives	Derivatives with respect to one variable while holding others constant.	Used for each weight during training.
Chain rule	Rule for differentiating composed functions; the backbone of backpropagation.	Must explain how errors flow backward.
Gradients	Vectors of partial derivatives showing the direction of steepest increase of loss.	Training moves opposite the gradient.

Topic	Simple Definition	Exam / Interview Tip
Probability	Framework for uncertainty, distributions, likelihoods, predictions, and model confidence.	Important for classification and generative models.
Random variables	Variables whose values are outcomes of uncertain processes.	Common in probabilistic modeling.
Distributions	Functions describing possible values and their probabilities.	Know normal, Bernoulli, categorical.
Bayes theorem	A rule for updating belief using evidence.	Useful for probabilistic reasoning and classifiers.
Statistics	Methods for summarizing data, estimating patterns, and evaluating uncertainty.	Used in experiments and model evaluation.
Mean	Average value of a dataset.	Often used in normalization and metrics.
Variance	Measure of how spread out values are from the mean.	High variance can indicate instability.
Standard deviation	Square root of variance, expressed in the same units as data.	Useful for standardization.
Correlation	Measure of relationship between two variables.	Correlation does not prove causation.
Hypothesis testing	Statistical process for deciding whether observed differences are likely meaningful.	Useful in A/B testing and experiments.
Optimization basics	Methods for finding parameters that minimize or maximize an objective.	Deep learning training is optimization.
Python	Primary language used for deep learning experimentation and production workflows.	Know functions, classes, environments, and packages.
NumPy	Python library for fast numerical arrays and matrix operations.	Good for understanding tensors.
Pandas	Data analysis library for tables, cleaning, joins, and preprocessing.	Used before model training.

Topic	Simple Definition	Exam / Interview Tip
Matplotlib	Visualization library for plotting losses, metrics, distributions, and predictions.	Useful for debugging models.
Scikit-learn	Machine learning library for preprocessing, metrics, baseline models, and pipelines.	Use it for baseline comparison.
Jupyter Notebook	Interactive environment for experiments, visualization, and exploratory analysis.	Great for prototypes, not always production.
Git and GitHub	Version control tools for tracking code, experiments, and collaboration.	Important for reproducible ML projects.
Linux and terminal basics	Command-line skills for running scripts, managing files, installing packages, and using servers.	Essential for GPU/cloud workflows.
Supervised learning	Learning from labeled examples to predict outputs from inputs.	Classification and regression are common types.
Unsupervised learning	Learning patterns from unlabeled data.	Includes clustering, dimensionality reduction, representation learning.
Classification	Predicting a discrete class label.	Use accuracy, precision, recall, F1, ROC-AUC.
Regression	Predicting a continuous numeric value.	Use MAE, MSE, RMSE, R-squared.
Clustering	Grouping similar examples without labels.	Common algorithms: k-means, DBSCAN.
Feature engineering	Creating useful input variables from raw data.	Deep learning reduces but does not eliminate this need.
Model training	Process of fitting model parameters to data by minimizing loss.	Know epochs, batches, loss, optimizer.
Model validation	Evaluating model performance on data not used for weight updates.	Prevents overfitting decisions.
Overfitting	When a model memorizes training data and performs poorly on unseen data.	Use regularization, more data, early stopping.

Topic	Simple Definition	Exam / Interview Tip
Underfitting	When a model is too simple or poorly trained to capture patterns.	Use better features, larger model, longer training.
Bias-variance tradeoff	Balance between overly simple models and overly sensitive models.	Explain generalization using this.
Train/validation/test split	Data partitioning strategy for training, tuning, and final evaluation.	Never tune on the test set.
Cross-validation	Repeated train/evaluation splits to estimate model reliability.	Less common for huge DL but useful for small data.
Evaluation metrics	Quantitative measures that show whether a model meets the task goal.	Choose metrics based on business and error cost.

Chapter 3

Neural Network Basics

Chapter Introduction

Understand how neural networks compute predictions and learn from errors.

Topic	Simple Definition	Exam / Interview Tip
Artificial neuron	A computational unit that applies weights, bias, and an activation to inputs.	Basic building block of neural networks.
Perceptron	Simple linear classifier that maps inputs to a binary output.	Historical foundation of neural networks.
Multilayer perceptron	Feedforward neural network with one or more hidden layers.	Good baseline for tabular and simple tasks.
Input layer	Layer that receives raw features or embeddings.	Its size matches input feature dimension.
Hidden layers	Intermediate layers that learn internal representations.	Depth increases representational power.
Output layer	Final layer that produces prediction values, logits, probabilities, or embeddings.	Activation depends on task type.
Weights	Learnable parameters that scale input signals.	Most of learning updates weights.
Biases	Learnable offsets added to weighted sums.	Help shift activation boundaries.
Forward propagation	Passing input through the network to compute predictions and loss.	Used in both training and inference.
Backpropagation	Algorithm that computes gradients of loss with respect to parameters.	Core training mechanism.
Loss function	Objective that measures prediction error and guides training.	Lower loss usually means better fit, but verify metrics.
Gradient descent	Optimization method that updates parameters opposite the gradient direction.	Foundation for most optimizers.

Topic	Simple Definition	Exam / Interview Tip
Sigmoid	Activation that maps values to 0 through 1.	Often used for binary probability outputs.
Tanh	Activation that maps values to -1 through 1.	Zero-centered but can saturate.
ReLU	Activation that outputs max(0, x).	Fast, common, but can create dead neurons.
Leaky ReLU	ReLU variant that allows a small negative slope.	Reduces dead ReLU problem.
ELU	Activation with exponential behavior for negative values.	Can improve learning stability.
GELU	Smooth activation commonly used in Transformer models.	Seen in BERT/GPT-style networks.
Softmax	Converts logits into a probability distribution across classes.	Used for multi-class classification.
Swish	Smooth activation defined roughly as x times sigmoid(x).	Sometimes improves deep networks.
Mean squared error	Average squared difference between predictions and targets.	Common for regression.
Mean absolute error	Average absolute difference between predictions and targets.	More robust to outliers than MSE.
Binary cross-entropy	Loss for binary classification probabilities.	Common with sigmoid output.
Categorical cross-entropy	Loss for multi-class classification with one-hot labels.	Common with softmax output.
Sparse categorical cross-entropy	Multi-class cross-entropy using integer class labels.	Avoids one-hot encoding.
Hinge loss	Margin-based loss used in SVM-style classifiers.	Useful for max-margin thinking.
Contrastive loss	Loss that pulls similar pairs together and pushes dissimilar pairs apart.	Used in representation learning.
Triplet loss	Loss that compares anchor, positive, and negative examples.	Common in face recognition and embeddings.
Focal loss	Loss that focuses training on hard or misclassified examples.	Useful for imbalanced detection tasks.

Chapter 4

Optimization in Deep Learning

Chapter Introduction

Learn how models are trained efficiently and stably.

Topic	Simple Definition	Exam / Interview Tip
Gradient descent	Uses the full dataset to compute gradients and update parameters.	Accurate but expensive for large data.
Stochastic gradient descent	Updates parameters using one sample or small random samples.	Noisy but scalable.
Mini-batch gradient descent	Updates parameters using small batches of examples.	Standard training approach.
Momentum	Optimizer technique that accumulates past gradient direction.	Speeds training and reduces zigzagging.
RMSProp	Adaptive optimizer that normalizes updates by recent gradient magnitudes.	Good for non-stationary objectives.
Adam	Adaptive optimizer combining momentum and RMSProp ideas.	Default starting point for many models.
AdamW	Adam variant with decoupled weight decay.	Common for Transformers.
Adagrad	Adaptive optimizer that gives larger updates to rare features.	Can decay learning rate too much.
Adadelta	Adaptive method designed to reduce manual learning-rate tuning.	Less common today.
LAMB	Layer-wise adaptive optimizer for large-batch training.	Used for large-scale training.
Lion optimizer	Memory-efficient optimizer using sign-based updates.	Modern alternative in some experiments.
Learning rate scheduling	Changing learning rate during training.	Often improves convergence.
Warmup	Starting with a small learning rate and gradually increasing it.	Important for large Transformers.

Topic	Simple Definition	Exam / Interview Tip
Step decay	Reducing learning rate at fixed milestones.	Simple and effective schedule.
Exponential decay	Continuously reducing learning rate by a factor.	Smooth schedule for long training.
Cosine annealing	Learning rate follows a cosine curve from high to low.	Popular in vision and LLM training.
ReduceLROnPlateau	Reduces learning rate when validation metric stops improving.	Good practical callback.
Cyclical learning rates	Learning rate cycles between bounds.	Can escape shallow minima.
Vanishing gradients	Gradients become too small to train early layers effectively.	Use ReLU, normalization, residual connections.
Exploding gradients	Gradients become too large and destabilize training.	Use gradient clipping and careful initialization.
Gradient clipping	Limits gradient magnitude before optimizer updates.	Important for RNNs and large models.
Weight initialization	Choosing starting parameter values before training.	Bad initialization can stop learning.
Xavier initialization	Initialization designed for tanh/sigmoid-like activations.	Balances signal variance across layers.
He initialization	Initialization designed for ReLU-family activations.	Common in CNNs and MLPs.
Batch normalization	Normalizes activations across a mini-batch.	Speeds training, common in CNNs.
Layer normalization	Normalizes features within each example.	Common in Transformers.
Weight normalization	Reparameterizes weights to improve optimization.	Less common but useful conceptually.

Chapter 5

Regularization Techniques

Chapter Introduction

Reduce overfitting and improve generalization.

Topic	Simple Definition	Exam / Interview Tip
L1 regularization	Penalty on absolute weight values that can encourage sparsity.	Useful when feature selection matters.
L2 regularization	Penalty on squared weight values that discourages large weights.	Often implemented as weight decay.
Dropout	Randomly disables neurons during training to reduce co-adaptation.	Common in MLPs and older architectures.
Early stopping	Stops training when validation performance stops improving.	Simple and powerful anti-overfitting method.
Data augmentation	Creates modified training examples to improve robustness.	Essential in vision, audio, and text.
Label smoothing	Softens hard labels to reduce overconfidence.	Useful for classification and calibration.
Weight decay	Regularization that shrinks parameters during optimization.	AdamW handles this cleanly.
Noise injection	Adds noise to inputs, weights, or activations during training.	Can improve robustness.
Mixup	Combines examples and labels using interpolation.	Improves smooth decision boundaries.
CutMix	Replaces image regions and mixes labels.	Strong vision augmentation.
Stochastic depth	Randomly skips layers during training.	Regularizes very deep networks.
Model pruning	Removes unnecessary weights or units from a model.	Used for compression and inference speed.

Chapter 6

Deep Learning Frameworks

Chapter Introduction

Use practical libraries to build, train, deploy, and debug models.

Topic	Simple Definition	Exam / Interview Tip
TensorFlow	End-to-end deep learning framework with training, serving, and deployment tools.	Strong production ecosystem.
Keras	High-level API for building neural networks quickly.	Beginner-friendly and readable.
PyTorch	Dynamic deep learning framework popular for research and production.	Know tensors, autograd, modules, DataLoader.
JAX	High-performance numerical computing library with automatic differentiation and compilation.	Popular for research and large-scale training.
MXNet	Deep learning framework historically used in production and research.	Less common now but still appears in legacy systems.
Tensors	Multi-dimensional arrays used as the main data structure in deep learning.	Know shape, dtype, device.
Autograd	Automatic differentiation engine that computes gradients.	PyTorch training relies on it.
Dataset and DataLoader	PyTorch abstractions for data access and batching.	Important for scalable training loops.
Training loop	Repeated process of forward pass, loss calculation, backpropagation, and optimizer step.	Be able to write one from scratch.
Model class	Code structure that defines layers and forward logic.	In PyTorch, usually extends nn.Module.
GPU training	Using GPUs to accelerate tensor operations and model training.	Know device placement.
Saving and loading models	Persisting model weights or full models for reuse and deployment.	Save weights plus config when possible.

Topic	Simple Definition	Exam / Interview Tip
TorchScript	PyTorch model serialization and optimization path for deployment.	Useful for production portability.
PyTorch Lightning	Training framework that organizes PyTorch code and reduces boilerplate.	Helpful for reproducible experiments.
Sequential API	Keras API for stacking layers linearly.	Best for simple models.
Functional API	Keras API for flexible graphs with multiple inputs/outputs.	Use for complex architectures.
Custom models	User-defined model classes for custom training behavior.	Needed for advanced research patterns.
Custom layers	Reusable layer definitions with custom computation.	Useful for novel architectures.
TensorBoard	Visualization tool for training metrics, graphs, images, and embeddings.	Good for monitoring experiments.
tf.data pipeline	TensorFlow input pipeline API for efficient loading and preprocessing.	Important for performance.
SavedModel	TensorFlow standard format for exporting trained models.	Used with serving and deployment.
TensorFlow Lite	Framework for running models on mobile and edge devices.	Know quantization and constraints.
TensorFlow Serving	Production serving system for TensorFlow models.	Handles scalable model APIs.

Chapter 7

Computer Vision Deep Learning

Chapter Introduction

Apply neural networks to images and visual understanding tasks.

Topic	Simple Definition	Exam / Interview Tip
Convolution	Operation that slides filters over an image to detect local patterns.	Core of CNNs.
Filters/kernels	Small learnable matrices that detect features such as edges, textures, or shapes.	Number of filters controls feature maps.
Feature maps	Outputs produced by applying filters to input images or activations.	Represent detected patterns.
Padding	Adding borders around inputs to control output size.	Same padding preserves dimensions.
Stride	Step size used when sliding a filter.	Higher stride reduces spatial size.
Pooling	Downsampling operation that reduces spatial resolution.	Improves efficiency and invariance.
Max pooling	Pooling that keeps the maximum value in each region.	Captures strongest activation.
Average pooling	Pooling that averages values in each region.	Often used near classification heads.
Fully connected layers	Dense layers that combine learned features for final predictions.	Often appear after CNN feature extraction.
CNN architecture	Neural architecture designed around convolutional layers for vision tasks.	Explain local connectivity and parameter sharing.
LeNet	Early CNN architecture for digit recognition.	Classic historical model.
AlexNet	CNN that popularized deep learning for ImageNet-scale vision.	Known for ReLU and GPU training.

Topic	Simple Definition	Exam / Interview Tip
VGG	CNN family using many small 3x3 convolutions.	Simple but parameter-heavy.
GoogLeNet/Inception	Architecture using parallel convolution paths at different scales.	Introduced efficient multi-scale features.
ResNet	Architecture using residual connections to train very deep networks.	Key concept: skip connections.
DenseNet	Architecture connecting each layer to many later layers.	Encourages feature reuse.
MobileNet	Efficient CNN family for mobile and edge devices.	Uses depthwise separable convolutions.
EfficientNet	CNN family that scales depth, width, and resolution systematically.	Strong accuracy-efficiency tradeoff.
ConvNeXt	Modern CNN architecture influenced by Transformer design choices.	Shows CNNs remain competitive.
Image classification	Assigning one or more labels to an image.	Start with CNN/ViT baselines.
Object detection	Finding and classifying objects with bounding boxes.	Metrics include mAP and IoU.
Image segmentation	Assigning labels to pixels or regions.	Used in medical imaging and autonomous systems.
Instance segmentation	Detecting separate object instances and masks.	Mask R-CNN is classic.
Semantic segmentation	Classifying every pixel by category without separating object instances.	U-Net and DeepLab are common.
Face recognition	Identifying or verifying people using learned facial embeddings.	Often uses metric learning.
Image captioning	Generating text descriptions from images.	Combines vision and language models.
Image generation	Creating images from noise, text, or conditions.	GANs and diffusion dominate.
Super-resolution	Increasing image resolution and detail using neural networks.	Common in restoration.
Style transfer	Applying visual style from one image to another.	Uses content and style representations.
R-CNN	Object detection method that classifies region proposals.	Accurate but slow historically.
Fast R-CNN	Improved R-CNN that shares convolutional computation.	Faster training and inference.

Topic	Simple Definition	Exam / Interview Tip
Faster R-CNN	Detection model with a region proposal network.	Strong two-stage detector.
SSD	Single-shot detector that predicts boxes and classes in one pass.	Fast real-time detection.
YOLO	One-stage detector optimized for speed and practical deployment.	Popular real-time detector family.
RetinaNet	One-stage detector using focal loss to handle class imbalance.	Good accuracy-speed tradeoff.
DETR	Transformer-based object detector using set prediction.	Reduces hand-designed detection components.
FCN	Fully convolutional network for dense prediction tasks.	Foundation for segmentation.
U-Net	Encoder-decoder segmentation architecture with skip connections.	Very popular in medical imaging.
Mask R-CNN	Extension of Faster R-CNN that also predicts segmentation masks.	Instance segmentation classic.
DeepLab	Segmentation family using atrous convolutions and multi-scale context.	Strong semantic segmentation baseline.
Segment Anything Model	Promptable foundation model for image segmentation.	Useful for interactive segmentation workflows.

Chapter 8

Natural Language Processing Deep Learning

Chapter Introduction

Model language, meaning, and sequence data using neural networks.

Topic	Simple Definition	Exam / Interview Tip
Tokenization	Splitting text into units such as words, subwords, or characters.	Tokenization affects context length and cost.
Stemming	Reducing words to crude roots by chopping suffixes.	Less precise than lemmatization.
Lemmatization	Reducing words to dictionary forms using linguistic rules.	Better for classical NLP pipelines.
Stop words	Common words often removed in traditional NLP.	Do not blindly remove them for Transformers.
Vocabulary	Set of tokens a model recognizes.	Out-of-vocabulary handling matters.
Bag of words	Text representation based on token counts without order.	Simple baseline.
TF-IDF	Weighting method that emphasizes terms important to a document but rare overall.	Strong search/classification baseline.
Word embeddings	Dense vectors representing word meaning and relationships.	Foundation for neural NLP.
Word2Vec	Neural embedding method that learns word vectors from context.	Know skip-gram and CBOW.
GloVe	Embedding method based on global word co-occurrence statistics.	Classic pre-Transformer embedding.
FastText	Embedding method using subword information.	Handles rare and misspelled words better.

Topic	Simple Definition	Exam / Interview Tip
Sentence embeddings	Dense vectors representing full sentence meaning.	Used for semantic search and retrieval.
Contextual embeddings	Embeddings that change based on surrounding words.	BERT-style representations.
Positional embeddings	Vectors that encode token positions in a sequence.	Needed because attention has no natural order.
RNN	Neural network that processes sequences step by step with hidden state.	Classic sequence model.
LSTM	RNN variant with gates to preserve long-term information.	Addresses vanishing gradients.
GRU	Simpler gated RNN variant.	Often faster than LSTM.
Bidirectional RNN	RNN that reads sequence forward and backward.	Good when full context is available.
Encoder-decoder models	Architecture that maps input sequence to output sequence.	Used for translation and summarization.
Sequence-to-sequence learning	Learning to transform one sequence into another.	Foundation for translation and chat models.
Attention mechanism	Method that lets models focus on relevant parts of input.	Explain query, key, value.
Self-attention	Attention where tokens attend to other tokens in the same sequence.	Core of Transformers.
Multi-head attention	Runs multiple attention heads to capture different relationships.	Improves representation capacity.
Positional encoding	Injects order information into Transformer inputs.	Can be sinusoidal or learned.
Encoder	Transformer block stack that builds contextual representations of input.	BERT is encoder-only.
Decoder	Transformer block stack that generates tokens autoregressively.	GPT is decoder-only.
Transformer architecture	Attention-based neural architecture replacing recurrence for sequence modeling.	Core architecture of modern LLMs.
BERT	Encoder-only Transformer trained for language understanding tasks.	Good for classification and NER.
GPT	Decoder-only Transformer trained for autoregressive generation.	Good for text generation and chat.

Topic	Simple Definition	Exam / Interview Tip
T5	Text-to-text Transformer framing many NLP tasks as generation.	Flexible task format.
RoBERTa	Optimized BERT variant trained with stronger data and training choices.	Often stronger than BERT baseline.
DistilBERT	Compressed BERT model for faster inference.	Example of distillation.
XLNet	Permutation-based language model designed to improve bidirectional context learning.	Important historical model.
Text classification	Assigning labels to text documents or messages.	Use BERT or embeddings plus classifier.
Sentiment analysis	Classifying opinion polarity or emotion in text.	Common interview project.
Named entity recognition	Identifying entities such as people, locations, organizations, and dates.	Token-level classification task.
Text summarization	Producing shorter text that preserves key information.	Can be extractive or abstractive.
Machine translation	Translating text from one language to another.	Sequence-to-sequence task.
Question answering	Answering questions from context, knowledge, or retrieval.	RAG often improves factuality.
Chatbots	Conversational systems that respond to user messages.	Need memory, safety, evaluation.
Text generation	Producing coherent text from a prompt or context.	Decoder LLMs excel here.
Semantic search	Retrieval based on meaning using embeddings.	Better than keyword search for paraphrases.
Information retrieval	Finding relevant documents from a collection.	Hybrid search combines lexical and semantic methods.

Chapter 9

Large Language Models

Chapter Introduction

Understand modern generative language systems and production LLM applications.

Topic	Simple Definition	Exam / Interview Tip
Transformer decoder	Autoregressive Transformer stack that predicts the next token.	Foundation of GPT-style LLMs.
Pretraining	Training on massive general data to learn broad language patterns.	Expensive phase before adaptation.
Fine-tuning	Additional training on task-specific or domain-specific data.	Improves specialized behavior.
Instruction tuning	Fine-tuning on instruction-response examples.	Makes models follow user requests better.
RLHF	Reinforcement learning from human feedback to align model outputs with preferences.	Important alignment concept.
Direct preference optimization	Preference optimization method that can avoid explicit RL training.	Common alternative to RLHF.
Context window	Maximum amount of tokens a model can consider at once.	Limits long-document reasoning.
Prompt engineering	Designing prompts to guide model behavior.	Useful but not a substitute for evaluation.
In-context learning	Model learns task behavior from examples inside the prompt.	No weight update required.
Chain-of-thought prompting	Prompting style that encourages step-by-step reasoning.	Use carefully; final answers matter most.
Tool calling	Allowing a model to call external functions, APIs, or tools.	Needed for real-time actions.
Function calling	Structured tool calling where model outputs function name and arguments.	Improves reliability and integration.

Topic	Simple Definition	Exam / Interview Tip
RAG	Retrieval-augmented generation that adds relevant external context before generation.	Reduces hallucination when implemented well.
Embeddings	Dense vectors representing semantic meaning of text, images, or items.	Used for search, clustering, memory.
Vector databases	Databases optimized for storing and searching embeddings.	Examples: Pinecone, Qdrant, Weaviate, pgvector.
Chunking	Splitting documents into retrievable pieces.	Poor chunking breaks RAG quality.
Hybrid search	Combining keyword retrieval and vector search.	Improves recall and precision.
Reranking	Second-stage scoring of retrieved results for relevance.	Often improves RAG answers.
Prompt templates	Reusable prompt structures with variables.	Helps consistency and testing.
Guardrails	Rules, filters, validators, or policies that constrain model behavior.	Important for safety and compliance.
Hallucination reduction	Techniques to reduce unsupported or fabricated answers.	Use grounding, citations, evals, refusal rules.
LLM evaluation	Measuring quality, safety, grounding, latency, and cost of LLM systems.	Use task-specific test sets.
Agentic workflows	LLM systems that plan, use tools, remember state, and complete multi-step tasks.	Evaluate with end-to-end traces.
Multi-agent systems	Multiple specialized agents collaborating or debating.	Use only when complexity justifies it.
LLM observability	Tracing prompts, tool calls, outputs, latency, cost, and errors.	Essential for production debugging.
Cost optimization	Reducing inference and infrastructure expense.	Use caching, routing, smaller models.
Latency optimization	Reducing response time and improving user experience.	Use streaming, parallel retrieval, smaller models.
Full fine-tuning	Updating all model parameters on new data.	Powerful but expensive and risky.
Parameter-efficient fine-tuning	Updating a small subset of parameters or adapters.	Cheaper than full fine-tuning.

Topic	Simple Definition	Exam / Interview Tip
LoRA	Low-rank adaptation method for efficient fine-tuning.	Popular PEFT technique.
QLoRA	LoRA fine-tuning with quantized base models.	Reduces GPU memory needs.
Prefix tuning	Learning prefix vectors prepended to model activations.	Parameter-efficient approach.
Prompt tuning	Learning soft prompt embeddings while freezing model weights.	Useful for controlled adaptation.
Adapter tuning	Adding small trainable modules inside a frozen model.	Good for multi-task adaptation.
Dataset preparation	Cleaning, formatting, labeling, and splitting data for training or tuning.	Garbage data creates garbage models.
Evaluation datasets	Curated examples used to measure model performance.	Must reflect real user tasks.

Chapter 10

Generative Deep Learning

Chapter Introduction

Models that create new data such as images, text, audio, or structured samples.

Topic	Simple Definition	Exam / Interview Tip
Autoencoders	Networks that compress input into latent representations and reconstruct it.	Good for compression and anomaly detection.
Variational autoencoders	Probabilistic autoencoders that learn smooth latent spaces.	Generate samples from latent distributions.
Generative adversarial networks	Two-network setup where generator and discriminator compete.	Can produce sharp images but training is unstable.
Diffusion models	Generative models that learn to remove noise step by step.	Dominant for image generation.
Flow-based models	Generative models with invertible transformations and exact likelihoods.	Useful for density estimation.
Energy-based models	Models that assign low energy to likely data configurations.	Flexible but training can be hard.
Generator	GAN component that creates fake samples.	Tries to fool discriminator.
Discriminator	GAN component that distinguishes real from fake samples.	Provides learning signal to generator.
Minimax training	GAN objective where generator and discriminator optimize opposing goals.	Can suffer from instability.
DCGAN	GAN architecture using convolutional networks for images.	Classic GAN baseline.
Conditional GAN	GAN that generates outputs conditioned on labels or inputs.	Adds controllability.
CycleGAN	GAN for image-to-image translation without paired examples.	Uses cycle consistency.

Topic	Simple Definition	Exam / Interview Tip
StyleGAN	GAN family known for high-quality controllable image generation.	Important generative vision model.
Wasserstein GAN	GAN variant using Wasserstein distance for more stable training.	Often uses gradient penalty.
Forward diffusion	Process of gradually adding noise to data.	Training learns reverse process.
Reverse diffusion	Denoising process that turns noise into data.	Generation happens here.
Noise scheduler	Controls how noise is added or removed across diffusion steps.	Affects quality and speed.
Denoising U-Net	U-Net model trained to predict noise or denoised samples.	Common diffusion backbone.
Stable Diffusion	Latent diffusion model for text-to-image generation.	Runs diffusion in compressed latent space.
Latent diffusion	Diffusion performed in latent space instead of pixel space.	More efficient image generation.
Text-to-image generation	Creating images from natural language prompts.	Uses text encoder plus generative model.
Image-to-image generation	Transforming an input image guided by prompt or condition.	Used for editing and stylization.
ControlNet	Conditioning method that controls diffusion with edges, poses, depth, or maps.	Improves spatial control.

Chapter 11

Representation Learning

Chapter Introduction

Learn useful embeddings and internal features from data.

Topic	Simple Definition	Exam / Interview Tip
Feature learning	Automatically learning useful representations instead of hand-coding features.	Key advantage of deep learning.
Embedding spaces	Vector spaces where similar items are close together.	Used in search, recommendation, clustering.
Metric learning	Training embeddings so distances reflect semantic similarity.	Used for verification and retrieval.
Contrastive learning	Learning by comparing positive and negative pairs.	Important self-supervised method.
Self-supervised learning	Training using labels derived from the data itself.	Reduces need for manual labels.
SimCLR	Contrastive visual representation learning method using augmentations.	Classic SSL method.
MoCo	Contrastive method using a momentum encoder and queue of negatives.	Improves negative sampling.
BYOL	Self-supervised method that learns without explicit negative pairs.	Uses online and target networks.
CLIP	Vision-language model aligning images and text in a shared embedding space.	Enables zero-shot image classification.
Triplet networks	Networks trained with anchor, positive, and negative examples.	Used for face/product similarity.
Siamese networks	Two networks sharing weights to compare two inputs.	Used for verification and similarity.

Chapter 12

Reinforcement Learning with Deep Learning

Chapter Introduction

Train agents to make decisions through rewards and interaction.

Topic	Simple Definition	Exam / Interview Tip
Agent	Decision-making system that chooses actions.	Learns from environment feedback.
Environment	World or simulator where the agent acts.	Returns observations and rewards.
State	Information representing the current situation.	Can be fully or partially observable.
Action	Choice made by the agent.	Can be discrete or continuous.
Reward	Signal indicating how good an action or outcome was.	Shapes agent behavior.
Policy	Strategy mapping states to actions.	Can be deterministic or stochastic.
Value function	Estimate of expected future reward from a state or action.	Guides decisions.
Q-learning	Learns action values for state-action pairs.	Off-policy RL method.
Exploration vs exploitation	Tradeoff between trying new actions and using known good actions.	Core RL challenge.
Deep Q-Network	Uses neural networks to approximate Q-values.	Classic deep RL breakthrough.
Policy gradient	Directly optimizes the policy using gradient estimates.	Good for continuous or stochastic actions.
Actor-critic methods	Combine policy learning actor with value learning critic.	Balances stability and flexibility.

Topic	Simple Definition	Exam / Interview Tip
Advantage actor-critic	Actor-critic method using advantage estimates.	Reduces variance in policy updates.
A3C	Asynchronous advantage actor-critic using parallel workers.	Important historical algorithm.
PPO	Policy optimization method with clipped updates for stability.	Popular practical RL algorithm.
DDPG	Actor-critic method for continuous control.	Off-policy deterministic policy gradient.
SAC	Soft Actor-Critic maximizes reward and entropy.	Strong for continuous control.
Model-based RL	Learns or uses an environment model for planning.	Can improve sample efficiency.
Model-free RL	Learns behavior without explicitly modeling environment dynamics.	Often simpler but data-hungry.
Multi-agent reinforcement learning	Multiple agents learn and interact in shared environments.	Challenges include coordination and competition.

Chapter 13

Time Series Deep Learning

Chapter Introduction

Model ordered data such as sensors, finance, demand, logs, and events.

Topic	Simple Definition	Exam / Interview Tip
Time series forecasting	Predicting future values from historical ordered observations.	Know horizon, seasonality, lag features.
Sequence modeling	Learning patterns across ordered inputs.	Applies to text, audio, time series.
LSTM forecasting	Using LSTM networks to model temporal dependencies.	Useful for smaller sequential datasets.
GRU forecasting	Using GRU networks for efficient sequence forecasting.	Simpler alternative to LSTM.
Temporal convolutional networks	Convolutional networks designed for sequence/time series data.	Parallel and stable for long sequences.
Transformer forecasting	Using attention-based models for time series prediction.	Good for long-range dependencies.
Informer	Efficient Transformer architecture for long sequence forecasting.	Designed for long time series.
Autoformer	Forecasting model using decomposition and auto-correlation mechanisms.	Handles trends and seasonality.
Prophet with neural networks	Combining classical decomposable forecasting ideas with neural approaches.	Useful hybrid idea.
Anomaly detection	Finding unusual patterns or outliers in time-dependent data.	Common in monitoring and fraud.
Financial forecasting	Predicting financial variables such as prices, volatility, or risk.	Very noisy; avoid overclaiming.
Sensor data modeling	Learning from IoT, wearable, industrial, or device sensor streams.	Need noise handling and drift monitoring.

Chapter 14

Graph Deep Learning

Chapter Introduction

Apply neural networks to connected data such as networks, molecules, and knowledge graphs.

Topic	Simple Definition	Exam / Interview Tip
Graph neural networks	Neural networks that learn from nodes, edges, and graph structure.	Use message passing.
Graph convolutional networks	GNNs that aggregate information from neighboring nodes.	Good for node classification.
Graph attention networks	GNNs that use attention to weight neighbor importance.	Adds adaptive aggregation.
GraphSAGE	GNN method that samples and aggregates neighbors for scalable learning.	Useful for large graphs.
Message passing neural networks	Framework where nodes exchange messages through edges and update states.	General GNN view.
Node classification	Predicting labels for graph nodes.	Example: user, paper, product labels.
Link prediction	Predicting whether edges should exist between nodes.	Used in recommendations and fraud.
Graph classification	Predicting labels for entire graphs.	Used for molecules and documents.
Knowledge graphs	Structured facts represented as entities and relationships.	Useful for reasoning and retrieval.
Graph embeddings	Vector representations of nodes, edges, or graphs.	Used in search and recommendation.
GraphRAG	Retrieval approach that uses graph relationships to improve context selection and reasoning.	Good for connected enterprise knowledge.

Chapter 15

Multimodal Deep Learning

Chapter Introduction

Build models that combine text, images, audio, video, and structured data.

Topic	Simple Definition	Exam / Interview Tip
Vision-language models	Models that connect visual and textual representations.	Used for captioning, VQA, image search.
Text-image models	Models that generate or retrieve images from text and vice versa.	CLIP and diffusion are key examples.
Audio-text models	Models linking speech/audio with text.	Used in ASR, TTS, audio QA.
Video understanding	Modeling spatial and temporal information in videos.	Needs frame and motion reasoning.
CLIP	Model aligning images and text through contrastive learning.	Enables zero-shot classification and search.
BLIP	Vision-language model family for captioning and image-text understanding.	Strong VLM concept.
Flamingo-style models	Few-shot multimodal models combining vision encoders with language models.	Useful architecture pattern.
Visual question answering	Answering questions about images or videos.	Requires grounding in visual content.
Multimodal transformers	Transformer architectures that process multiple data types together.	Used in frontier AI systems.
Multimodal RAG	Retrieval-augmented generation over text, images, tables, audio, or video.	Important enterprise AI pattern.

Chapter 16

Audio and Speech Deep Learning

Chapter Introduction

Process speech, sound, and audio signals with neural models.

Topic	Simple Definition	Exam / Interview Tip
Speech recognition	Converting spoken audio into text.	Also called ASR.
Text-to-speech	Generating spoken audio from text.	Used in assistants and accessibility.
Speaker recognition	Identifying or verifying a speaker from voice.	Uses voice embeddings.
Audio classification	Classifying sound clips into categories.	Examples: music, alarms, events.
Keyword spotting	Detecting specific wake words or commands.	Edge-friendly task.
WaveNet	Autoregressive neural audio generation model.	Important TTS milestone.
Tacotron	Neural TTS architecture mapping text to spectrograms.	Often paired with vocoders.
DeepSpeech	End-to-end speech recognition model.	Classic ASR system.
Whisper-style models	Transformer-based speech recognition systems trained on large-scale audio.	Robust general ASR concept.
Spectrograms	Time-frequency representation of audio signal energy.	Common input for audio models.
Mel-frequency cepstral coefficients	Compact audio features based on human hearing scale.	Classic speech feature representation.

Chapter 17

Model Evaluation

Chapter Introduction

Measure whether models are accurate, reliable, safe, and useful.

Topic	Simple Definition	Exam / Interview Tip
Accuracy	Fraction of correct predictions.	Misleading for imbalanced data.
Precision	Of predicted positives, how many were actually positive.	Important when false positives are costly.
Recall	Of actual positives, how many were found.	Important when false negatives are costly.
F1-score	Harmonic mean of precision and recall.	Good for imbalanced classification.
ROC-AUC	Measures ranking ability across thresholds.	Useful for binary classifiers.
Confusion matrix	Table showing true positives, false positives, true negatives, and false negatives.	Great for error analysis.
MAE	Mean absolute error for regression.	Easy to interpret.
MSE	Mean squared error for regression.	Penalizes large errors strongly.
RMSE	Square root of MSE, in target units.	Common regression metric.
R-squared	Proportion of variance explained by the model.	Can be misleading outside context.
Loss curves	Plots of training and validation loss over time.	Diagnose overfitting/underfitting.
Validation accuracy	Accuracy on validation data used for tuning.	Do not confuse with test accuracy.
Generalization gap	Difference between training and validation/test performance.	Large gap suggests overfitting.
Calibration	How well predicted probabilities match real outcome frequencies.	Important in risk decisions.

Topic	Simple Definition	Exam / Interview Tip
Robustness testing	Testing model behavior under noise, shifts, attacks, and edge cases.	Production requirement.
Error analysis	Systematic review of failures to identify patterns and fixes.	Best way to improve models.
Ablation studies	Removing components to measure their contribution.	Common in research and interviews.
Benchmarking	Comparing models against standard datasets or baselines.	Always include a baseline.
Faithfulness	Whether generated answer is supported by provided context.	Critical for RAG.
Groundedness	Degree to which answer relies on retrieved or source material.	Measure hallucination risk.
Relevance	How well response addresses user question.	High relevance is not always high correctness.
Helpfulness	Practical usefulness and completeness of the output.	Often judged by users or LLM-as-judge.
Toxicity	Presence of harmful, abusive, or unsafe language.	Safety metric.
Bias	Systematic unfairness across groups or contexts.	Requires dataset and metric design.
Hallucination rate	Frequency of unsupported or fabricated claims.	Track especially in knowledge tasks.
Context precision	How much retrieved context is actually relevant.	Retrieval quality metric.
Context recall	How much needed information was retrieved.	Low recall causes missing answers.
Answer correctness	Whether final answer is factually or task-wise correct.	Needs labels, rubrics, or expert review.

Chapter 18

Data Engineering for Deep Learning

Chapter Introduction

Prepare high-quality data and reliable pipelines for model training and inference.

Topic	Simple Definition	Exam / Interview Tip
Data collection	Gathering raw examples from sources such as databases, logs, sensors, or users.	Define consent, quality, and coverage.
Data cleaning	Removing or fixing duplicates, errors, missing values, and inconsistent formats.	Usually consumes most project time.
Data labeling	Creating target annotations for supervised learning.	Label quality limits model quality.
Data preprocessing	Transforming raw data into model-ready inputs.	Includes resizing, tokenizing, scaling.
Data normalization	Scaling data to consistent ranges or distributions.	Improves optimization stability.
Dataset versioning	Tracking data changes across experiments.	Needed for reproducibility.
Synthetic data	Artificially generated data used for training or testing.	Validate against real-world distribution.
Data drift	Change in input distribution over time.	Monitor after deployment.
Feature stores	Systems for managing reusable features for training and serving.	Important in ML platforms.
Data quality checks	Automated tests for schema, missing values, ranges, and anomalies.	Prevent bad training runs.

Chapter 19

MLOps for Deep Learning

Chapter Introduction

Operationalize models with repeatable, monitored, and deployable workflows.

Topic	Simple Definition	Exam / Interview Tip
Experiment tracking	Recording parameters, metrics, artifacts, and results from runs.	Tools: MLflow, W&B, TensorBoard.
Model registry	Central place to version, approve, and manage models.	Supports governance and rollback.
Model versioning	Tracking model artifacts and metadata across releases.	Needed for auditability.
CI/CD for ML	Automated testing, building, and deployment for ML code and artifacts.	Includes data and model tests.
Model deployment	Making a model available for batch or real-time use.	Choose serving pattern by latency needs.
Model monitoring	Tracking performance, drift, latency, errors, and usage after deployment.	Production never ends at deploy.
Model rollback	Returning to a previous stable model version.	Plan before incidents happen.
Model serving	Infrastructure that receives requests and returns predictions.	Examples: TorchServe, TF Serving, BentoML.
Batch inference	Running predictions periodically on many records.	Good for offline workflows.
Real-time inference	Serving predictions immediately per request.	Requires latency and reliability design.
A/B testing	Comparing models or features with controlled user groups.	Use statistical significance and guardrail metrics.
Shadow deployment	Running new model silently beside production model.	Reduces deployment risk.

Topic	Simple Definition	Exam / Interview Tip
Canary deployment	Gradually releasing a model to a small traffic slice.	Detect issues early.
Model observability	Deep visibility into model inputs, outputs, traces, and system health.	Critical for incident response.
Concept drift	Change in relationship between inputs and targets over time.	Can degrade accuracy even if data looks similar.
MLflow	Tool for experiment tracking, model registry, and packaging.	Common MLOps interview tool.
Weights & Biases	Experiment tracking and model development platform.	Popular for deep learning teams.
Kubeflow	Kubernetes-based ML workflow platform.	Used for scalable pipelines.
Airflow	Workflow orchestrator for scheduled data and ML pipelines.	Great for DAGs and batch workflows.
Docker	Containerization tool for packaging code and dependencies.	Solves environment consistency.
Kubernetes	Container orchestration platform for scaling services and jobs.	Common for production ML.
Ray	Distributed computing framework for ML workloads.	Useful for training, tuning, serving.
DVC	Data version control tool for datasets and ML pipelines.	Git-like workflows for data.
BentoML	Framework for packaging and serving ML models.	Good for model APIs.
TorchServe	PyTorch model serving framework.	Production serving option.

Chapter 20

Deployment and Production

Chapter Introduction

Optimize models for real users, infrastructure, cost, and latency.

Topic	Simple Definition	Exam / Interview Tip
REST API deployment	Serving model predictions through HTTP endpoints.	Common backend pattern.
FastAPI	Python web framework often used for ML inference APIs.	Fast and easy for production prototypes.
Flask	Lightweight Python web framework for APIs and web apps.	Simple but less structured than FastAPI.
Model containers	Docker images that package model code, dependencies, and runtime.	Improves portability.
GPU inference	Using GPUs to accelerate prediction workloads.	Best for large models/high throughput.
CPU inference	Running predictions on CPUs.	Cheaper for small models or low traffic.
ONNX	Open model exchange format for portability across runtimes.	Useful for deployment optimization.
TensorRT	NVIDIA inference optimizer and runtime.	Improves GPU inference speed.
Quantization	Reducing numerical precision of weights/activations.	Improves speed and memory usage.
Model compression	Reducing model size through pruning, distillation, or quantization.	Important for edge and cost.
Pruning	Removing weights or structures that contribute little.	Can speed inference with right hardware.
Knowledge distillation	Training a smaller student model to mimic a larger teacher.	Common compression method.

Topic	Simple Definition	Exam / Interview Tip
Edge deployment	Running models on devices near users or sensors.	Requires small, efficient models.
Mobile deployment	Running models on phones or tablets.	Use TFLite, Core ML, ONNX Runtime Mobile.
Cloud deployment	Serving or training models using cloud infrastructure.	Know managed services and GPUs.
Serverless inference	Running prediction functions on demand without managing servers.	Good for intermittent workloads.

Chapter 21

Hardware for Deep Learning

Chapter Introduction

Understand compute systems that make training and inference possible.

Topic	Simple Definition	Exam / Interview Tip
CPU	General-purpose processor suitable for preprocessing and smaller inference.	Not ideal for large matrix-heavy training.
GPU	Parallel processor optimized for matrix operations.	Main accelerator for deep learning.
TPU	Specialized accelerator designed for tensor operations.	Used in large-scale training.
CUDA	NVIDIA parallel computing platform for GPU programming.	Most deep learning GPU stack uses it.
cuDNN	NVIDIA library accelerating neural network primitives.	Used under frameworks.
VRAM	GPU memory used for model weights, activations, and batches.	Limits batch size and model size.
Distributed training	Training across multiple devices or machines.	Needed for large models/data.
Multi-GPU training	Using multiple GPUs to speed or scale training.	Know data parallelism.
Data parallelism	Each device trains on different data batches and syncs gradients.	Most common scaling method.
Model parallelism	Splitting model layers or tensors across devices.	Used for very large models.
Pipeline parallelism	Splitting model stages across devices with microbatches.	Improves utilization for huge models.
Mixed precision training	Using lower precision where safe to improve speed and memory.	FP16/BF16 common.
FP32	32-bit floating point precision.	Stable but memory-heavy.

Topic	Simple Definition	Exam / Interview Tip
FP16	16-bit floating point precision.	Fast but can need loss scaling.
BF16	Brain floating point 16-bit format with wider exponent range.	Stable for large models.
INT8 inference	8-bit integer inference for faster, smaller models.	Requires quantization calibration or aware training.

Chapter 22

Advanced Deep Learning Topics

Chapter Introduction

Explore specialized topics often asked in advanced interviews or research discussions.

Topic	Simple Definition	Exam / Interview Tip
Memory networks	Architectures with explicit memory components for reasoning over stored information.	Conceptually related to long-context and retrieval.
Neural architecture search	Automated search for effective model architectures.	Balances performance and compute.
Meta-learning	Learning how to learn quickly across tasks.	Useful for few-shot scenarios.
Few-shot learning	Learning or adapting from a small number of examples.	LLMs often show this in-context.
Zero-shot learning	Performing tasks without task-specific examples.	Often enabled by pretrained models.
Continual learning	Learning new tasks over time without forgetting old ones.	Challenge: catastrophic forgetting.
Federated learning	Training across decentralized devices without centralizing raw data.	Privacy-aware approach.
Causal representation learning	Learning features that capture cause-effect structure.	Useful for robust generalization.
Bayesian deep learning	Combines neural networks with uncertainty estimation.	Important in high-risk decisions.
Neural ordinary differential equations	Models continuous transformations using differential equations.	Advanced research topic.
Spiking neural networks	Neural models inspired by discrete biological spikes.	Relevant to neuromorphic computing.
Capsule networks	Networks designed to preserve part-whole relationships.	Interesting but less mainstream.

Topic	Simple Definition	Exam / Interview Tip
Differentiable programming	Writing programs where components are differentiable and trainable.	Broad view of modern ML systems.

Chapter 23

AI Safety, Security, and Ethics

Chapter Introduction

Design models that are safe, fair, private, and trustworthy.

Topic	Simple Definition	Exam / Interview Tip
Bias and fairness	Measuring and reducing unfair outcomes across groups.	Requires careful data and metric design.
Explainability	Making model decisions understandable to humans.	Important for trust and regulation.
Interpretability	Understanding internal model behavior and learned features.	Harder for deep networks.
Adversarial attacks	Inputs intentionally crafted to fool models.	Important in vision and security.
Model poisoning	Attacking training process or model parameters.	Supply-chain risk.
Data poisoning	Injecting malicious or misleading training data.	Can create backdoors.
Prompt injection	Attack that manipulates LLM instructions through user or retrieved text.	Major LLM app threat.
Privacy-preserving ML	Methods that reduce exposure of sensitive data.	Includes federated learning and differential privacy.
Differential privacy	Mathematical privacy guarantee limiting information leakage about individuals.	Often adds noise.
Secure model deployment	Protecting model APIs, data, prompts, outputs, and infrastructure.	Use auth, logging, rate limits, validation.
Model governance	Policies and processes for approving, monitoring, and auditing models.	Enterprise requirement.
Responsible AI	Designing AI systems aligned with safety, fairness, transparency, and accountability.	Broader than technical metrics.
AI compliance	Meeting legal, regulatory, and organizational AI requirements.	Document decisions and controls.

Topic	Simple Definition	Exam / Interview Tip
Human-in-the-loop systems	Including human review or approval in model workflows.	Useful for high-risk decisions.

Chapter 24

Explainable AI

Chapter Introduction

Techniques that help humans inspect and trust model behavior.

Topic	Simple Definition	Exam / Interview Tip
Feature importance	Ranking input features by contribution to predictions.	Simple explanation method.
Saliency maps	Visual maps showing image pixels influential to prediction.	Useful but can be noisy.
Grad-CAM	Vision explanation method using gradients to highlight important regions.	Common CNN interpretability tool.
SHAP	Explanation method based on Shapley values from game theory.	Strong but can be expensive.
LIME	Local explanation method using simple surrogate models around one prediction.	Good for local interpretability.
Integrated gradients	Attribution method accumulating gradients from baseline to input.	Popular for deep networks.
Attention visualization	Inspecting attention weights or patterns.	Helpful but not always faithful explanation.
Counterfactual explanations	Explaining what minimal changes would alter a prediction.	Useful for actionable insights.

Chapter 25

Deep Learning Project Topics

Chapter Introduction

Project ideas to practice skills and build interview stories.

Topic	Simple Definition	Exam / Interview Tip
MNIST digit classifier	Beginner project classifying handwritten digits.	Practice basic CNN or MLP training.
Cats vs dogs classifier	Image classification project for binary vision tasks.	Practice augmentation and transfer learning.
Sentiment analysis	Classifies text sentiment as positive, negative, or neutral.	Good NLP beginner project.
Movie review classifier	Text classification using reviews and labels.	Compare TF-IDF baseline vs neural model.
House price prediction	Regression project predicting property prices.	Good for preprocessing and metrics.
Simple chatbot	Basic conversational agent using rules, retrieval, or small LLM patterns.	Focus on evaluation and safety.
Object detection app	App that detects objects in images or video.	Use YOLO or Faster R-CNN.
Face recognition system	System for face verification or identification.	Discuss privacy and ethics.
Resume classifier	NLP system classifying resumes by role or skill match.	Watch bias and compliance risks.
Text summarizer	Model or LLM app that condenses long documents.	Evaluate faithfulness.
Product recommendation system	System recommending items based on behavior or similarity.	Use embeddings and ranking.
Time series forecasting system	Predicts future demand, traffic, or sensor values.	Monitor drift.

Topic	Simple Definition	Exam / Interview Tip
Image captioning model	Generates descriptions for images.	Multimodal project.
RAG chatbot	Chatbot grounded in private documents using retrieval.	Strong AI engineer portfolio project.
Fine-tuned LLM assistant	Domain-specific assistant adapted with supervised data or PEFT.	Compare to prompting and RAG.
Diffusion image generator	Model or workflow for text-to-image generation.	Learn diffusion concepts.
Multi-agent AI system	Multiple agents solving tasks with tools and memory.	Evaluate reliability carefully.
Medical image segmentation	Segments anatomical or pathology regions in images.	High-stakes evaluation and compliance.
Fraud detection model	Detects suspicious transactions or behavior.	Focus on precision-recall and drift.
Real-time speech-to-text system	Transcribes audio streams into text.	Latency and robustness matter.
Multimodal search engine	Search over text, images, or documents using embeddings.	Great production AI project.
Graph neural network recommender	Recommendation system using graph relationships.	Demonstrates advanced modeling.

Chapter 26

Best Specializations for 2026

Chapter Introduction

These specializations are strong choices for portfolio building, AI engineering interviews, and production-oriented career growth.

Specialization	Why it matters
LLM engineering	Build, evaluate, and optimize applications powered by language models.
RAG systems	Ground answers in external knowledge using retrieval, reranking, and citations.
Agentic AI	Design AI systems that plan, use tools, keep state, and complete tasks.
Computer vision	Build systems for images, objects, segmentation, and visual reasoning.
Multimodal AI	Combine text, image, audio, video, and structured data.
AI model evaluation	Create reliable test sets, rubrics, metrics, and monitoring.
Deep learning deployment	Serve models with low latency, scalability, and cost control.
MLOps	Automate reproducible training, deployment, monitoring, and governance.
AI safety and security	Protect AI systems from unsafe behavior, attacks, and compliance risks.
Edge AI and model optimization	Compress and deploy models to mobile, IoT, or local devices.

Chapter 27

High-Yield Interview Questions

Chapter Introduction

Practice these answers until you can explain them naturally. Interviewers reward clarity, tradeoff awareness, and practical examples.

What is deep learning?

Deep learning is machine learning using neural networks with many layers to learn representations from data. It is powerful because it can learn features automatically from raw inputs such as images, text, audio, and sensor data.

Why do neural networks need activation functions?

Without activation functions, stacked layers collapse into one linear transformation. Activations add non-linearity, allowing the network to model complex relationships.

Explain backpropagation in simple words.

Backpropagation calculates how much each weight contributed to the error, then uses gradients to update the weights so future predictions become better.

How do you handle overfitting?

Use more data, data augmentation, dropout, L1/L2 regularization, early stopping, simpler models, cross-validation, and better evaluation splits.

When would you use CNNs?

Use CNNs when local spatial patterns matter, especially in images, video frames, medical scans, and some signal-processing tasks.

Why did Transformers become popular?

Transformers use attention to model long-range dependencies and train in parallel more effectively than RNNs, making them scalable for large language and multimodal models.

What is the difference between fine-tuning and RAG?

Fine-tuning changes model weights to adapt behavior or domain style. RAG retrieves external context at runtime so the model can answer using current or private knowledge without changing weights.

How do you evaluate a deep learning model?

Use task metrics, validation/test splits, loss curves, error analysis, robustness tests, calibration, and production monitoring. For LLMs, also measure groundedness, relevance, hallucination rate, safety, latency, and cost.

What is vanishing gradient?

It happens when gradients become very small as they flow backward, causing early layers to learn slowly. ReLU, residual connections, normalization, and better initialization help.

What makes a model production-ready?

It must be accurate, reliable, monitored, versioned, secure, scalable, cost-aware, explainable enough for the domain, and supported by rollback and incident-response processes.

Chapter 28

Final Exam and Interview Checklist

Chapter Introduction

Use this final page before an exam, mock interview, or technical screening.

- Explain forward propagation and backpropagation clearly.
- Know common activation functions and when they fail.
- Compare SGD, Adam, and AdamW.
- Diagnose overfitting from train/validation curves.
- Explain CNNs using filters, feature maps, stride, and padding.
- Explain Transformers using attention, tokens, and positional encoding.
- Compare RNN, LSTM, CNN, and Transformer approaches.
- Explain RAG versus fine-tuning.
- Choose metrics based on error cost and task type.
- Discuss deployment, monitoring, drift, latency, and cost.
- Mention safety, bias, privacy, and human review for high-risk systems.
- Use simple examples when explaining complex concepts.

Best answer formula for interviews

Definition → Why it matters → When to use it → Tradeoff → Real project example.

Free Resources and Premium Guides

Continue Learning with AI Engineering Insider

Use this ebook as your quick-reference deep learning map. For more free resources, newsletters, and updates, subscribe here:

Free resources: aiengineeringinsider.substack.com/subscribe
Premium and individual guides: beacons.ai/aiengineeringinsider

Closing Note

Deep learning becomes easier when you connect every topic back to one core idea: the model learns representations from data by minimizing error. Master the basics, build small projects, evaluate carefully, and then specialize in the areas that match your career goal.