Part1 (Multiple Choice)

1. Consider three scalar losses on a labeled example (x, y):

(i) $\frac{1}{2}(y - w^\top x)^2$

(ii) $\max(0, 1 - yw^\top x)$

(iii) $\log(1 + \exp(-yw^\top x))$

Which of these functions is convex?

(a) Only (i) and (iii)

(b) All three

(c) Only (i)

(d) None of the above

Answer: (b) All three**.**

2. Imagine fitting polynomials of increasing degree to noisy data. What trend would you expect as degree increases, before any regularization?

(a) Both bias and variance decrease

(b) Bias decreases, variance increases

(c) Both bias and variance increase

(d) Bias increases, variance decreases

Answer: (b) Bias decreases, variance increases.

3. A function f(x) = $x^T Ax$ is defined for x ∈ R^d and a symmetric matrix x ∈ R^{dxd}. This function is guaranteed to be convex if and only if:

(a) A is invertible.

(b) A has all non-negative eigenvalues (is positive semi-definite).

(c) A is a diagonal matrix.

(d) The trace of A is positive.

Answer: (b) A has all non-negative eigenvalues (is positive semi-definite).

4. In a Markov Decision Process (MDP), the ”Markov property” states that the transition dynamics P(s', r|s, a) depend:

(a) Only on the current state s and action a.

(b) On the entire history of states and actions (s0, a0, . . . , s, a).

(c) Only on the agent's policy π(a|s).

(d) Only on the reward function R(s, a).

Answer: (a) Only on the current state $s$and action a.

5. In gradient descent on a convex differentiable function f(w), which condition guarantees convergence to a global minimum?

(a) f is Lipschitz-continuous

(b) f has a unique critical point

(c) f is strongly convex and has Lipschitz-continuous gradients

(d) Step size ηt = 1/t

Answer: (c) f is strongly convex and has Lipschitz-continuous gradients.

6. When an agent's actions influence data collection, standard supervised-learning risk minimization fails because:

(a) Training vs. test distributions differ (distribution shift)

(b) The model is under-parameterized

(c) The loss is non-convex

(d) The stochastic gradient estimates are biased

Answer: (a) Training vs. test distributions differ (distribution shift).

7. The self-attention operation in Transformers scales with:

(a) O(n) in sequence length

(b) O(n log n)

(c) O(n^2)

(d) O(1)

Answer: **(c) O(n^2)**

8. Linear separability in the penultimate layer of a neural network classifier implies that:

(a) The network's final linear classifier can achieve zero empirical error

(b) The earlier layers must be invertible

(c) The representation preserves Euclidean distances

(d) The features are orthogonal

Answer: **(a) The network's final linear classifier can achieve zero empirical error.**

9. In Bayesian inference, the evidence term P(E) in P(H|E) = $\frac{P(E|H)P(H)}{P(E)}$ is often computationally intractable. Why?

(a) It is a subjective probability that must be set by a human expert

(b) It requires summing or integrating over all possible hypotheses

H′: P(E) = $\int P(E|H')P(H')dH'$

(c) It requires knowing the future, as P(E) is the probability of an event that has not yet occurred

(d) It is an unstable value that must be re-computed at every time step using MCMC

Answer: **(b) It requires summing or integrating over all possible hypotheses H′: P(E) =** $\boldsymbol{\int P(E|H')P(H')dH'}$

10. Aleatoric uncertainty arises from:

(a) Noise inherent in the data-generation process

(b) Insufficient model capacity

(c) Poor initialization

(d) Overfitting

Answer: **(a) Noise inherent in the data-generation process**

11. Why are GPUs so much more effective than CPUs for training and running most deep learning models?

(a) GPUs have a significantly faster clock speed (GHz) for serial computation

(b) GPUs have thousands of simple cores designed for high-throughput parallel computation (SIMD)

(c) GPUs have specialized hardware for computing the non-linear 'ReLU' activation function

(d) CPUs use a different instruction set that is mathematically incompatible with backpropagation

Answer: **(b) GPUs have thousands of simple cores designed for high-throughput parallel computation (SIMD)**

12. In mechanism design (e.g. designing an auction), what is typically the fundamental goal of the Principal?

(a) To accurately estimate the posterior probability distribution of the agents' private values

(b) To find the optimal action for a single agent

(c) To design the rules of the game such that self-interested agents, acting rationally, will produce an outcome optimal for the designer

(d) To make all agents communicate with each other to reach a consensus

Answer: **(c) To design the rules of the game such that self-interested agents, acting rationally, will produce an outcome optimal for the designer.**

13. Consider the following two-player coordination game payoff matrix. Player 1 chooses Row; Player 2 chooses Column.

	Col A	**Col B**
Row A	(10, 10)	(0, 0)
Row B	(0, 0)	(5, 5)

Which of the following statements is true about this game?

(a) (Row A, Col A) is the only pure strategy Nash Equilibrium

(b) This game has no pure strategy Nash Equilibria

(c) (Row B, Col B) is a dominated strategy for both players

(d) This game has two pure strategy Nash Equilibria

Answer: **(d) This game has two pure strategy Nash Equilibria**

14. You run an A/B test for a new "agentic" feature and find a 3% increase in user retention(your key metric) with p = 0.03. What is the correct interpretation of this p-value?

(a) There is a 3% probability that the new feature is better.

(b) There is a 97% probability that the new feature is better.

(c) If the new feature had no effect there would be only a 3% chance of observing an increase at least this large by chance

(d) The new feature is approximately 3% better than the old one on average, but the reliability of this depends on the sample size

Answer: (c) If the new feature had no effect, there would be only a 3% chance of observing an increase at least this large by chance.

15. An agent can choose to "bet" or "pass.". If it passes, it gets a guaranteed payoff of $5. If it bets, it has a 20% chance of winning $100 and an 80% chance of losing $10. A risk-neutral agent maximizing its expected utility should:

(a) Pass

(b) Bet

Answer: (b) Bet

16. What is the infinite sum of the geometric series $\sum_{k=0}^{\infty}(0.9)^k$?

(a) 9.0

(b) 1.0

(c) 10.0

(d) ∞ (the series diverges)

Answer: (c) 10.0

17. In a multi-agent system, an agent must send its state (a ≈1KB message) to 100 other agents every 100ms. The network is unreliable. The most critical requirement is that the latest state message arrives with minimal delay; old, lost messages are irrelevant. Which of the following network protocols is the appropriate choice?

(a) TCP

(b) UDP

(c) HTTP

(d) FTP

Answer: (b) UDP

18. In Variational Inference (VI), the goal is to approximate an intractable posterior P(Z|X) with a tractable distribution Q(Z; λ). How is this approximation typically optimized?

(a) Maximize the ELBO L(λ) which is equivalent to minimizing the KL divergence D_{KL}(Q(Z; λ) || P(Z|X)).

(b) Minimizing the reverse KL divergence D_{KL} (P(Z|X) || Q(Z; λ))

(c) Running an MCMC sampler until the distribution of Q(Z; λ) converges to the stationary distribution of P(Z|X)

(d) Finding the parameters λ that correspond to the mode of the posterior P(Z|X) (i.e. the MAP estimate)

Answer: (a) Maximize the ELBO $\mathcal{L}(\lambda)$, which is equivalent to minimizing the KL divergence $D_{\mathrm{KL}}(Q(Z;\lambda) \parallel P(Z \mid X))$.

19. In a standard Pre-LN Transformer block, which statement best describes how attention heads interact with the residual stream?

(a) Queries are computed from a head-specific recurrent state that does not depend on the residual stream

(b) Q, K, and V are linear projections of the residual stream; each head forms a weighted sum of value vectors, concatenated head outputs are linearly projected, and the result is added back into the residual via a skip connection

(c) Heads write into a head-local memory that is concatenated with, but not added to, the residual stream (no skip connection)

(d) LayerNorm disables the residual connection during attention, so the attention output overwrites the residual stream

Answer: (b) Q, K, and V are linear projections of the residual stream; each head forms a weighted sum of value vectors, concatenated head outputs are linearly projected, and the result is added back into the residual via a skip connection

20. Which of the following is not an empirically observed scaling regularity for LLMs?

(a) Pretraining perplexity follows approximate power-law improvement with increases in model size, dataset size, and total training compute over wide ranges.

(b) At fixed training compute, there is a compute-optimal trade-off between parameters and tokens ("Chinchilla"-style scaling), so training longer on more data with a somewhat smaller model can outperform an undertrained larger model.

(c) Downstream task performance often improves roughly as a power law with additional pretraining compute, albeit with task-dependent exponents.

(d) Inference FLOPs per token decrease as a power law as model parameter count increases.

Answer: (d) Inference FLOPs per token decrease as a power law as model parameter count increases.

Part2 (Multiple Choice)

1. In a discounted-reward MDP, assume the discount factor is γ = 0.8. An agent receives a constant reward r = 5 at every time step forever. What is the total discounted return $V = \sum_{k=0}^{\infty} \gamma^k r$?
Answer: $V = 25$

2. An MDP has two states, sA and sB, and a discount factor γ = 0.9. The current value estimates are Vk(sA) = 10 and Vk(sB) = 20. An agent in sA takes an action a with the following dynamics:

- Transition to s_A with P = 0.5 and r = 0
- Transition to s_B with P = 0.5 and r = 10

Using the Bellman equation $V_{k+1}(s) = \sum_{s',r} \mathrm{P}(\mathrm{s}', \mathrm{r}|\mathrm{s}, \mathrm{a})[\mathrm{r} + \gamma \mathrm{V}_k(\mathrm{s}')]$, compute the updated value $V_{k+1}(s_A)$.

Given:

- $\gamma = 0.9$
- $V_k(s_A) = 10$
- $V_k(s_B) = 20$
- From $s_A$with action a:
  - to s_A: $P = 0.5, r = 0$

- to s_B: $P = 0.5, r = 10$

Compute:

$$\begin{aligned} V_{k+1}(s_A) &= 0.5\,[0 + 0.9V_k(s_A)] + 0.5\,[10 + 0.9V_k(s_B)] \\ &= 0.5\,[0 + 0.9 \cdot 10] + 0.5\,[10 + 0.9 \cdot 20] \\ &= 0.5 \cdot 9 + 0.5 \cdot 28 \\ &= 4.5 + 14 \\ &= 18.5 \end{aligned}$$

Answer: $V_{k+1}(s_A) = 18.5$

3. In an attention head, the output z_i for token i is $z_i = \sum_{j=1}^{n} a_{ij} v_j$, where $v_j = W_V x_j$ (x_j is the input token j). This output is added to the residual stream: $x'_i = x_i + z_i$. For a sequence of length n = 2, write the full expression for the updated token x'_1 in terms of x_1, x_2, the attention weights α_{11}, α_{12}, and the value matrix W_V .
   For sequence length $n = 2$,

$$z_1 = \sum_{j=1}^{2} \alpha_{1j}\, v_j = \alpha_{11} v_1 + \alpha_{12} v_2$$

   with

$$v_1 = W_V x_1, v_2 = W_V x_2.$$

   So

$$z_1 = \alpha_{11} W_V x_1 + \alpha_{12} W_V x_2.$$

   The updated token is

$$x'_1 = x_1 + z_1 = x_1 + \alpha_{11} W_V x_1 + \alpha_{12} W_V x_2.$$

   (Equivalently: $x'_1 = x_1 + W_V(\alpha_{11} x_1 + \alpha_{12} x_2)$.)

Answer: $x'_1 = x_1 + z_1 = x_1 + \alpha_{11} W_V x_1 + \alpha_{12} W_V x_2$.

4. You are training a large neural network. You observe that your GPU utilization is low (e.g. < 60%), but your CPU cores are at ≈100%. This strongly suggests a bottleneck in what part of your training pipeline?

Answer: It suggests a bottleneck in the **data input / preprocessing pipeline on the CPU** — e.g., data loading, decoding, and augmentation are too slow, so the GPU is often waiting idle for the next batch.

5.Let us assume you are using quantization-aware training, wherein the forward pass simulates quantization (e.g. r_out = round(r_in)). This has zero or undefined gradients. What is a simple method to allow gradients to flow through the backward pass?
Answer: Use a **straight-through estimator**, i.e., in the forward pass apply the rounding/quantization operation, but in the backward pass **pretend it was the identity function** so the gradient of round(·) is approximated as 1 (or passed through unchanged).

Part3 (Multiple Choice)

1. This exercise explores the representational power of untrained convolutional networks.
- Using PyTorch, JAX, or similar, define a simple CNN-type model with 2–4 hidden layers. For example:
  - Conv (e.g. 16 filters, 3×3 kernel) → ReLU → MaxPool
  - Conv (e.g. 32 filters, 3×3 kernel) → ReLU → MaxPool
  - Linear (e.g. 128 units)
- Load the MNIST dataset.
- Initialize your network. Do not train it. The weights should remain at their random initialization values.
- Perform a full forward pass for the entire MNIST training set through the network. Extract the activations from the last layer (e.g. the 128-unit fully-connected layer). These are your "random features."
- Using scikit-learn, train a linear support vector classifier (you may use sklearn.svm)
- Deliverables:
  (a) Report the final classification accuracy of your SVM on the test set's features
  (b) Supply your full code, packaged with everything needed to run from scratch and reproduce your results (fixed random seeds, Python environment spec and/or requirements.txt)
  (c) Answer the following questions:
    i. What do you observe about the results?
    ii. Why does this work at all? What role does the architecture play?
    iii. What support can you find in the research literature (either theoretical or empirical) that connects to these observations? (hint: think kernels)

2. Implement a small, decoder-only Transformer for character-level text generation.
You will use a provided text file (e.g. shakespeare.txt) as your corpus.
- Constraint: You must implement the model using only framework primitives (e.g. torch.nn.Linear, torch.nn.LayerNorm, torch.nn.Embedding, torch.matmul, etc.). You may not use high-level canned modules like torch.nn.TransformerDecoder, or torch.nn.MultiheadAttention.
- Implement a MultiHeadAttention module.
This module should:
  – Take Q,K, V projections.
  – Split them into multiple heads.
  – Compute scaled dot-product attention: Attention(Q, K, V) = softmax($\frac{QK^T}{\sqrt{d_k}}$)V.
  – Include a causal (look-ahead) mask for auto-regressive decoding.
  – Concatenate head outputs and pass them through a final linear layer.
- Implement a TransformerBlock (either Pre-LN or Post-LN) that combines your MultiHeadAttention module, a position-wise FFN (two linear layers with a ReLU), residual connections, and layer norm.
- Build your final model by stacking a character embedding layer, positional encoding, several TransformerBlocks, etc.

• Train this model on the shakespeare.txt corpus to predict the next character. You should be able to train it pretty quickly on your laptop.
• Deliverables:
(a) After training, show 10-15 lines of text generated (sampled) from your model by providing it with a brief prompt (e.g. “JULIET:”).
(b) Supply your full code, packaged with everything needed to run from scratch and reproduce your results (fixed random seeds, Python environment spec and/or requirements.txt)

3. Implement the Byte Pair Encoding (BPE) algorithm from scratch to create a subword tokenizer.
• Use the same shakespeare.txt corpus.
• Initialize your vocabulary to be the set of all unique characters (bytes) in the text.
• Write a function that counts the frequency of all adjacent pairs of tokens in the corpus (which is initially split by character).
• Write a merge loop:
– Find the most frequent adjacent pair (e.g. (’t’, ’h’)).
– Create a new token for this pair (e.g. ’th’).
– Add this new token to your vocabulary (and merge rules).
– Re-tokenize the corpus by replacing all occurrences of the frequent pair with the new merged token.
• Run this merge loop for a fixed number of iterations (e.g. k = 1000 merges).
• Deliverables:
(a) Show the final merge rules your algorithm learned
(b) Show how your trained tokenizer encodes the string: “Alas, poor Yorick! I knew him, Horatio:”
(c) Supply your full code, packaged with everything needed to run from scratch and reproduce your results. You should not need any external packages for this portion