NVIDIA NCA-GENM Practice Questions

Shared by Moody on 21-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

You are developing a ML model for image classification. You have a dataset with 10,000 images of cats, dogs and birds. Which of the following ML models would be the most appropriate choice for this task?

Options:

A- Logistic Regression
B- K-Means Clustering
C- Linear Regression
D- Convolutional Neural Network (CNN)

Answer:

D

Explanation:

CNNs are the standard architecture for image classification because their convolutional layers exploit the spatial locality and translation invariance inherent to image data: learned filters detect local patterns (edges, textures, shapes) that compose hierarchically into higher-level features (parts, objects) as depth increases, without requiring the manual feature engineering that traditional models would need to reach comparable accuracy on raw pixel data. Pooling layers further provide a degree of spatial invariance, and parameter sharing across the image keeps the model tractable relative to a fully connected network operating on raw pixels.

Logistic Regression (A) is a linear classifier that operates on flattened feature vectors; applied directly to raw pixels of a 3-class image problem, it cannot capture the non-linear spatial structure needed to separate cats, dogs, and birds reliably, though it could serve as a baseline or as the final classification head atop CNN-extracted features. K-Means (B) is an unsupervised clustering algorithm --- inappropriate here because the task is supervised classification with labeled classes. Linear Regression (C) predicts continuous outputs and is not designed for categorical class prediction at all.

For 10,000 labeled images, a CNN (potentially fine-tuned from a pretrained backbone via transfer learning, given the modest dataset size) is the appropriate and industry-standard choice.

Question 2

Question Type: MultipleChoice

What are some methods to overcome limited throughput between CPU and GPU?

Options:

A- Increase the clock speed of the CPU.
B- Increase the number of CPU cores.
C- Using techniques like memory pooling.
D- Upgrade the GPU to a higher-end model.

Answer:

C

Explanation:

CPU-GPU data transfer over the PCIe (or NVLink) bus is frequently a throughput bottleneck in ML pipelines, particularly when small, frequent transfers dominate rather than large batched ones --- each transfer incurs fixed overhead independent of data size, so many small transfers waste a disproportionate amount of time on overhead rather than useful data movement. Memory pooling techniques --- pre-allocating and reusing pinned (page-locked) host memory buffers rather than repeatedly allocating and freeing memory for each transfer --- reduce this overhead and enable faster, more predictable DMA transfers between host and device. Related software-level techniques include using CUDA streams to overlap data transfer with computation (so the GPU keeps computing while the next batch transfers in the background), and batching transfers to amortize fixed per-transfer overhead across more data.

Options A, B, and D each propose hardware upgrades that address a different bottleneck than the one described: increasing CPU clock speed (A) or core count (B) improves CPU-side compute throughput, not the data-transfer bandwidth or latency between CPU and GPU specifically. Upgrading the GPU (D) increases GPU compute capability but does nothing to address a PCIe/interconnect bandwidth limitation --- a faster GPU sitting idle waiting for data across the same bottlenecked bus would not see meaningfully improved end-to-end throughput. The question specifically asks about *throughput between* CPU and GPU, which points to interconnect/transfer-management optimization rather than raw compute upgrades on either side.

Question 3

Question Type: MultipleChoice

Hyperparameter tuning is used for what purpose in machine learning experimentation?

Options:

A- Adjusting the weights and biases of a neural network to optimize its performance.
B- Selecting the best ML algorithm for a given task.
C- Collecting and preprocessing data to improve the accuracy of the model.
D- Selecting the optimal values for non-trainable parameters, such as learning rate or batch size.

Answer:

D

Explanation:

Hyperparameters are configuration values set *before* training begins and are not updated by the optimization process itself --- learning rate, batch size, number of layers, regularization strength, and number of training epochs are canonical examples. Hyperparameter tuning is the systematic search for the combination of these values that yields the best model performance on a validation set, using strategies such as grid search, random search, or more sample-efficient approaches like Bayesian optimization and population-based training.

This is explicitly distinct from option A, which describes the *training* process itself --- weights and biases are trainable parameters, updated automatically via backpropagation and gradient descent, not selected through hyperparameter search. Option B describes algorithm selection, a higher-level modeling decision that may precede hyperparameter tuning but is not what tuning itself accomplishes (you tune hyperparameters *within* a chosen algorithm/architecture). Option C describes data engineering work that happens upstream of model training entirely, unrelated to parameter search.

In practice, hyperparameter tuning requires careful experimental design to avoid overfitting to the validation set --- techniques like k-fold cross-validation, held-out test sets, and tracking tools (e.g., experiment trackers logging each trial's configuration and resulting metric) are standard practice, connecting this topic directly to the Experimentation domain's broader emphasis on rigorous, reproducible model evaluation.

Question 4

Question Type: MultipleChoice

What is the significance of using a U-Net like architecture in denoising diffusion probabilistic models?

Options:

A- To generate new images from pure noise.
B- To classify input images as noisy or clean.
C- To detect noisy objects in input images.
D- To segment noisy patches in input images.

Answer:

A

Explanation:

In a denoising diffusion probabilistic model (DDPM), the U-Net serves as the noise-prediction network at the core of the iterative generation process: at each reverse-diffusion timestep, the U-Net takes the current noisy image (and typically a timestep embedding, plus conditioning information like a CLIP text embedding in text-to-image models) as input and predicts the noise component present at that step. Subtracting this predicted noise incrementally, over many timesteps starting from pure Gaussian noise, progressively denoises the input into a coherent image --- the mechanism by which DDPMs generate new images from pure noise. U-Net's architecture --- a contracting encoder path paired with an expanding decoder path, connected by skip connections at matching resolutions --- is well suited to this role because the skip connections preserve fine-grained spatial detail that would otherwise be lost through the network's downsampling bottleneck, which matters for producing sharp, high-fidelity denoised output at each step.

Options B, C, and D describe discriminative tasks --- classification, detection, and segmentation --- that describe *other* legitimate applications of U-Net-style architectures (originally developed for biomedical image segmentation) but do not describe its function specifically *within* the diffusion generative process. Within a DDPM pipeline specifically, U-Net's role is generative noise prediction supporting image synthesis from noise, not classification or detection of any kind.

Question 5

Question Type: MultipleChoice

What is the purpose of a kernel in a Convolutional Neural Network (CNN)?

Options:

A- To perform convolution operations on input data.
B- To calculate the loss function.
C- To classify the data into different categories.
D- To normalize the input data.

Answer:

A

Explanation:

A kernel (or filter) in a CNN is a small matrix of learnable weights that slides across the input (an image, feature map, or intermediate activation) computing a dot product at each spatial position --- the convolution operation. Each kernel is trained to detect a specific local pattern: early-layer kernels typically learn to detect low-level features like edges and color gradients, while kernels in deeper layers combine these into detectors for more complex, higher-level patterns (textures, object parts, and eventually whole-object representations as receptive fields grow with depth). A convolutional layer typically applies many kernels in parallel, each producing its own output channel, collectively forming the layer's feature map.

The other options describe separate CNN components with distinct responsibilities: the loss function (B) is computed at the network's output based on the difference between predictions and ground truth, entirely separate from the kernel's role in feature extraction. Classification (C) is typically performed by fully connected (dense) layers --- often with a softmax activation --- placed after the convolutional feature-extraction stack, not by the kernels themselves. Normalization (D) is handled by dedicated layers such as batch normalization or layer normalization, inserted between convolutional layers to stabilize activations, again a separate mechanism from the convolution operation itself.

Question 6

Question Type: MultipleChoice

Which visualization technique is suitable for representing the distribution of performance scores for different multimodal ML models over different modalities?

Options:

A- Heatmap
B- Histogram
C- Box plot

D- Pie chart

Answer:

C

Explanation:

A box plot (box-and-whisker plot) summarizes the distribution of a numeric variable --- median, interquartile range, and outliers --- as a single compact glyph, and critically, multiple box plots can be placed side by side to compare distributions across categorical groupings. This makes it well suited to the scenario described: comparing the spread and central tendency of performance scores across several models, further faceted by modality, in one readable figure. Box plots make skew, variance, and outlier prevalence immediately comparable across groups in a way a single summary statistic (like mean accuracy) cannot.

A histogram (B) shows the distribution of a single variable well but does not scale cleanly to side-by-side comparison across many model/modality combinations without becoming visually cluttered. A heatmap (A) is excellent for showing a matrix of values (e.g., mean score per model modality pair) but represents point estimates, not distributions --- it cannot convey variance or spread. A pie chart (D) is inappropriate for any continuous performance metric.

In practice, a violin plot --- which overlays a kernel density estimate on the box plot's summary statistics --- is often preferred when the underlying distribution's shape (e.g., bimodality) matters, but among the given options, the box plot is the correct choice for distributional comparison across groups.

Question 7

Question Type: MultipleChoice

What is the significance of A/B testing in ML software engineering?

Options:

A- A/B testing is used to measure the impact of changes in the user interface of a ML application.
B- A/B testing helps in optimizing the hyperparameters of a machine learning model.
C- A/B testing is irrelevant in ML software engineering.
D- A/B testing helps in evaluating the performance and effectiveness of different machine learning models.

Answer:

D

Explanation:

A/B testing in an ML engineering context is a controlled experimental methodology where two (or more) model variants --- for example, a current production model (A) and a candidate replacement (B) --- are deployed simultaneously to randomly assigned, statistically comparable segments of live traffic, and their real-world performance is compared on business or task-relevant metrics (conversion rate, click-through rate, task accuracy, latency-adjusted engagement). This provides causal evidence of which model performs better under actual production conditions, which offline evaluation on static held-out datasets cannot fully capture, since production data distributions shift and downstream user behavior interacts with model outputs in ways offline metrics miss.

Option A narrows A/B testing incorrectly to UI changes alone; while A/B testing originated in and remains common for UI/UX experimentation, its application in ML engineering explicitly extends to comparing model versions, algorithms, and feature sets --- not just interface elements. Option B misattributes a distinct methodology (systematic hyperparameter search, covered in the previous question) to A/B testing, which evaluates already-trained model variants rather than searching a hyperparameter space. Option C is simply incorrect --- A/B testing is a cornerstone of responsible ML deployment and MLOps practice, gating rollout decisions before full production release.