ISIC 2024 — Skin Cancer Detection

Interview Study Guide

Multi-modal deep learning pipeline combining Vision Transformers, ConvNeXt, and Gradient Boosting for clinical-grade melanoma screening

Alexy Louis

14 Questions · Key Concepts · Quick Reference Glossary

Questions & Answers

Q1 — The Clinical Stakes — pAUC@80TPR

Melanoma is the deadliest form of skin cancer. In a screening system, the consequences of getting it wrong are asymmetric — missing a cancer is catastrophic, but flagging a benign mole just means an extra biopsy. The ISIC 2024 competition evaluates models using partial AUC above 80% True Positive Rate.

■ **What does the pAUC@80TPR metric enforce about the model?**

- ■ **A)** The model must detect at least 80% of cancers
- ■ **B)** The model must be 80% accurate overall
- ■ **C)** The model must run within 80% of the time limit
- ■ **D)** 80% of predictions must be confident (>0.8 probability)

■ **Explanation**

The metric throws away all operating points where the model catches fewer than 80% of malignant lesions. It only measures discriminative ability in the high-sensitivity zone. Standard AUC gives equal credit to a model that catches 30% of cancers with zero false positives, but that's clinically useless. pAUC@80TPR says "show me how well you separate classes while catching most of the cancers." This directly mirrors clinical screening where missing a melanoma is unacceptable. Bonus: with a 1000:1 imbalance, a model predicting "benign" for everything gets 99.9% accuracy — meaningless.

Q2 — The Data Challenge — Balanced Batch Sampling

The ISIC 2024 dataset has ~401,000 samples with only 393 malignant — a 1000:1 imbalance. If we fed all data in random batches, most mini-batches would contain zero positive examples. We solve this with a BalancedBatchSampler.

■ In a training batch of 32 images, what does the BalancedBatchSampler do?

- ■ **A)** 16 malignant + 16 benign per batch — ensures balanced gradient signal
- ■ **B)** Duplicates malignant images 1000× before training
- ■ **C)** Removes 99.9% of benign samples to match the 393 malignant
- ■ **D)** Weights the loss function by 1000× for malignant samples

■ Explanation

The sampler controls which indices get selected for each mini-batch. For every batch of 32, it picks 16 from the malignant pool and 16 from the benign pool, shuffles them together, and feeds that to the DataLoader. The malignant samples get reused across batches (only 393 exist), but they aren't pre-duplicated — the sampler keeps drawing from that small pool. The model sees a balanced class signal in every gradient update, so the loss never becomes dominated by "everything is benign." Think of it as dealing from two separate decks rather than one lopsided deck.

Q3 — The Backbone Concept

We use two backbone models: EVA02 (Vision Transformer) and ConvNeXtV2 (modern CNN). Both are loaded from the timm library with pretrained=True and num_classes=0.

■ What is a 'backbone' and what does num_classes=0 do?

- ■ **A)** Feature extractor with classification head removed — outputs raw embeddings
- ■ **B)** Loss function switched to binary mode
- ■ **C)** Data loading pipeline with no class filtering
- ■ **D)** Frozen layers with no fine-tuning allowed

■ Explanation

A backbone is a pretrained network (trained on millions of ImageNet images) that extracts meaningful visual features. Setting num_classes=0 strips off the final classification head (a linear layer mapping to 1000 ImageNet classes) and gives the raw feature vector (e.g., 768-dimensional for EVA02). We attach our own head — nn.Linear(768, 1) — to map those features to malignant/benign. This is transfer learning: the backbone already knows textures, color gradients, and shapes from ImageNet; we teach the head what those mean for skin lesions.

Q4 — ViT vs ConvNeXt — Ensemble Diversity

Our ensemble uses EVA02 (a Vision Transformer using self-attention over image patches) and ConvNeXtV2 (a modern CNN with sliding convolutional filters). EVA02 generally scores slightly higher individually.

■ Why use both architectures instead of two copies of EVA02?

- ■ **A)** CNNs are always more accurate than ViTs
- ■ **B)** Architectural diversity reduces correlated errors
- ■ **C)** It doubles the training data
- ■ **D)** Kaggle requires multiple architectures

■ Explanation

Diversity matters more than individual accuracy in ensembles. ViTs use self-attention to relate every patch globally — great for overall shape and symmetry. ConvNeXt applies small sliding filters — strong at local textures and border patterns. When both agree, confidence is high. When they disagree, the ensemble can still succeed if one model caught what the other missed. Two EVA02 models would make very similar mistakes. This is called error decorrelation — the less correlated the errors, the more the ensemble improves over any single model.

Q5 — Convolutional Layers

ConvNeXt uses convolutional layers as its fundamental building block. A convolutional layer slides a small filter (e.g., 3×3) across the image.

■ What does a convolutional filter compute, and why is it suited for images?

- ■ **A)** Weighted sum over a local region — captures local patterns with translation invariance
- ■ **B)** Average pixel intensity in a region — produces a blurred image
- ■ **C)** Maximum pixel value in a region — highlights strongest features
- ■ **D)** Pixel-by-pixel comparison to a stored template — binary match

■ Explanation

A 3×3 filter has 9 learnable weights. At each position, it multiplies each pixel by the corresponding weight, sums them, and produces one output. Different filters detect different features: vertical edges, horizontal edges, corners. The critical property is translation invariance — the same filter works everywhere in the image, so an edge-detector functions regardless of where the edge appears. Early layers detect simple edges and colors; deeper layers combine those into complex patterns like "irregular border" or "multi-colored lesion."

Q6 — ReLU and Activation Functions

Between layers in both ViTs and CNNs, we apply activation functions. The most common is ReLU: max(0, x). Without activations, the network would only contain linear operations.

■ What would happen if we removed all activation functions?

- ■ **A)** Training would be faster
- ■ **B)** The network collapses to a single linear layer — can't learn complex boundaries
- ■ **C)** The model would overfit more
- ■ **D)** Only the last layer would receive gradients

■ Explanation

Key math: if layer 1 computes y = W■x and layer 2 computes z = W■y, then z = W■(W■x) = (W■W■)x. The product of two matrices is just another matrix — so any number of stacked linear layers equals one linear layer. The network literally cannot learn anything more complex than a linear separator. ReLU (max(0, x)) breaks this linearity by zeroing negatives — a simple operation that lets deep networks approximate arbitrarily complex functions. Depth only helps when combined with non-linearity.

Q7 — Focal Loss

We use Focal Loss ($\alpha=0.25$, $\gamma=2.0$) instead of standard Binary Cross-Entropy. Even with balanced batch sampling, Focal Loss adds another layer of imbalance handling by modifying how individual samples contribute to the loss.

■ What are 'easy' examples in Focal Loss, and why do we down-weight them?

- ■ **A)** Benign samples the model is already confident about — their loss drowns out hard cases
- ■ **B)** Malignant samples that look obviously cancerous
- ■ **C)** Duplicate or near-duplicate images
- ■ **D)** Low-resolution or blurry images

■ Explanation

In standard BCE, all samples contribute equally. After a few epochs, the model confidently predicts most benign samples (>95% confidence). Each produces tiny loss, but 400,000 tiny losses add up and dominate, drowning out the 393 hard cases. Focal Loss adds a multiplier: $(1 - p_t)^\gamma$. When $\gamma=2$: a 95% confident sample gets multiplier $(0.05)^2 = 0.0025$ (nearly zero); a 50% uncertain sample gets $(0.50)^2 = 0.25$ (still contributes). The model's weight updates focus on ambiguous cases. It's like a teacher who stops drilling material you've mastered and focuses on what you're struggling with.

Q8 — Data Augmentation

Before each image enters the model during training, we apply random transformations using Albumentations: flips, rotations, color jitter, noise. The model never sees the exact same image twice.

■ **With only 393 malignant samples that get reused many times, why is augmentation critical?**

- 🟩 **A)** Prevents memorizing the 393 images — forces learning general features
- ■ **B)** Makes training converge faster
- ■ **C)** Increases image resolution
- ■ **D)** Replaces the need for balanced sampling

■ **Explanation**

This prevents overfitting. The balanced sampler reuses those 393 malignant images hundreds of times per epoch. Without augmentation, the model would learn "malignant = that exact pixel pattern" rather than "malignant = irregular borders, color heterogeneity, asymmetric shape." Random transforms force the model to learn the concept of melanoma rather than memorize specific images. A melanoma flipped horizontally is still a melanoma — the model needs to learn that invariance. In effect, 393 images with augmentation behaves like thousands of unique images.

Q9 — Cross-Validation Design

We use 5-fold StratifiedGroupKFold with patient_id as the group. Some patients have multiple lesions photographed.

■ **Why must we group cross-validation splits by patient_id?**

- 🟩 **A)** Prevents data leakage from patient-specific cues
- ■ **B)** Reduces computation time
- ■ **C)** Kaggle requires patient grouping
- ■ **D)** Ensures equal fold sizes

■ **Explanation**

Cross-validation splits data into K folds, rotating which fold is held out for validation. 5-fold means 80/20 train/val split, repeated 5 times. "Stratified" ensures each fold has the same proportion of malignant samples (~78-79 each). "GroupKFold" with patient_id ensures ALL of a patient's lesions go into the same fold. Without this, the model could learn patient-specific cues (skin tone, lighting, body characteristics) rather than lesion-specific features — this is data leakage. Validation scores would look great but real-world performance would be terrible. 5 folds is the standard tradeoff: enough training data (80%), reasonable compute, stable estimates.

Q10 — Patient-Normalized Features — The Ugly Duckling

Our PatientFeatureTransformer creates z-scores relative to each patient's own lesions. Example: a patient has 12 lesions, one with area 25mm². Patient average is 10mm², std is 5mm².

■ **What is the z-score and why is it more useful than the raw 25mm² value?**

■ **A)** Z = 3.0 — it's unusually large FOR THIS PATIENT (ugly duckling sign)
■ **B)** Z = 2.5 — it normalizes across the entire dataset
■ **C)** Z = 15.0 — it's the raw difference from mean
■ **D)** Z = 0.5 — it's a percentile rank

■ **Explanation**

Z = (25 - 10) / 5 = 3.0. This captures "how abnormal is this lesion for this person" — the dermatological "ugly duckling" sign. A 25mm² lesion on a patient averaging 10mm² (z=3.0) is alarming; the same 25mm² on a patient averaging 22mm² (z=0.6) is normal. The raw value carries no patient context. This was the single most impactful feature engineering technique in top Kaggle solutions (~+0.02 pAUC). It bridges domain knowledge (dermatology) with ML (feature engineering) — a powerful interview talking point.

Q11 — GBDT Stacking — The Final Combiner

After training both image models, we extract embeddings (768-d from EVA02, 640-d from ConvNeXt) and combine them with engineered tabular features (~200-d). A LightGBM GBDT produces the final prediction.

■ **Why use a GBDT instead of simply averaging the two neural network outputs?**

■ **A)** GBDT learns which features matter and captures cross-modal interactions
■ **B)** GBDT is faster at inference
■ **C)** GBDT prevents overfitting inherently
■ **D)** GBDT handles missing values natively

■ **Explanation**

Embeddings are abstract, compressed representations — 768 numbers summarizing everything EVA02 "sees" in an image. The GBDT receives ~1,600 features: 768 (ViT) + 640 (ConvNeXt) + ~200 (tabular). It can learn cross-modal interactions like "when ViT embedding dim 42 is high AND the patient z-score for color variation > 2.0, this is very likely malignant." Simple averaging treats all signals equally and can't combine image + tabular modalities. This stacking approach is why every top Kaggle solution uses GBDT as the final combiner.

Q12 — EDA — Data-Driven Design

Our first notebook (01_eda.ipynb) doesn't train any model. It creates visualizations: class distributions, correlation heatmaps, feature distributions, patient-level analysis.

■ Why is the EDA notebook valuable when it produces no predictions?

- 🟩 **A)** It informs every downstream design decision
- ■ **B)** Kaggle requires an EDA submission
- ■ **C)** It pre-processes and cleans data for training
- ■ **D)** It proves we have data access

■ Explanation

Every major design decision traces back to an EDA discovery: the 1000:1 imbalance → balanced sampling + focal loss; multi-lesion patients → GroupKFold; strong separability of tbp_lv_deltaLBnorm → prioritized in features; mel_thick_mm 99.98% missing → excluded. In an interview, walk through: "Here's the problem I discovered, here's how I quantified it, here's the design decision it led to." That data → insight → decision narrative separates strong candidates from those who copy kernels.

Q13 — Mixed Precision Training (FP16)

Our training scripts use torch.cuda.amp.autocast and GradScaler for FP16 mixed precision. This was critical for fitting within Kaggle's GPU time limits.

■ What does FP16 mixed precision change, and why is GradScaler needed?

- 🟩 **A)** Halves memory and doubles speed; GradScaler prevents gradient underflow to zero
- ■ **B)** Compresses model weights permanently to reduce file size
- ■ **C)** Enables CPU half-precision for small batches
- ■ **D)** Runs forward and backward passes on separate GPU cores

■ Explanation

FP32 represents numbers as small as ~$10^{■3■}$, but FP16 bottoms out at ~$10^{■■}$. During backpropagation in deep networks (86M params for EVA02), gradients can be extremely small and round to zero in FP16. GradScaler multiplies the loss by ~1024 before backprop so gradients stay in FP16's range, then divides back afterward. "Mixed" means weights stay FP32 for precision, but forward/backward use FP16 for speed. Modern GPUs have Tensor Cores doing FP16 at 2× speed — giving us the speedup needed for Kaggle's 12h limit.

Q14 — The Full Pipeline — End to End

The complete multi-modal ensemble pipeline processes both images and clinical metadata to produce a final malignant/benign prediction.

■ What is the correct end-to-end order of our pipeline?

- 🟩 **A)** Images → ViT/ConvNeXt → embeddings, Metadata → features, Both → GBDT → prediction
- ■ **B)** Metadata → GBDT → filter, then filtered images → ViT → prediction
- ■ **C)** Images + Metadata → single ViT → prediction
- ■ **D)** Images → ViT → ConvNeXt → GBDT → prediction (sequential chain)

■ Explanation

The two image models run in parallel (not sequentially), each producing embeddings from the same image. Independently, metadata goes through feature engineering (z-scores, ABCDE criteria, aggregates). Everything is concatenated into ~1,600 features and fed to LightGBM. This is the "multi-modal" part — fusing visual and clinical information so the GBDT learns cross-modal interactions. This architecture diagram is powerful to draw on a whiteboard in an interview.

Quick Reference Glossary

Backbone	Pretrained feature extractor — the body of the network without the classification head. Outputs a fixed-size embedding vector.
Embedding	Compressed numerical representation of an input (e.g., 768 floats summarizing a 224×224 image). Captures learned abstract features.
Transfer Learning	Using a model pretrained on one task (ImageNet) and adapting it to another (skin cancer). The backbone keeps its learned features; only the head is retrained.
Focal Loss	Loss function that down-weights easy/confident examples via (1-p_t)^γ multiplier. Focuses learning on hard, ambiguous cases.
ReLU	Activation function max(0, x). Introduces non-linearity so stacked layers can learn complex patterns instead of collapsing to a single linear transform.
Convolution	Sliding weighted sum over local image regions using learned filters. Detects local patterns with translation invariance.
Self-Attention (ViT)	Mechanism that relates every image patch to every other patch, capturing global dependencies. Core operation in Vision Transformers.
pAUC@80TPR	Partial AUC evaluated only where TPR ≥ 80%. Ensures the model detects at least 80% of cancers — the clinically relevant sensitivity zone.
AUC (Full)	Area Under the ROC Curve. Measures overall discriminative ability across all operating points. Treats all sensitivity/specificity tradeoffs equally.
BalancedBatchSampler	Custom sampler that creates 50/50 class-balanced mini-batches by drawing equally from malignant and benign index pools.
Data Augmentation	Random image transforms (flip, rotate, color jitter, noise) applied during training. Prevents memorization, simulates data variety.
Cross-Validation (CV)	Train/validate on rotating subsets. K-fold: data split into K parts, each takes a turn as validation. Gives robust performance estimate.
StratifiedGroupKFold	CV that preserves class proportions (Stratified) while keeping all of a patient's samples in the same fold (Group). Prevents data leakage.
Data Leakage	When information from the validation/test set leaks into training — inflates metrics but destroys generalization. Patient grouping prevents this.
Z-score (Patient)	Feature = (value - patient_mean) / patient_std. Captures how abnormal a lesion is relative to that patient — the 'ugly duckling' sign.
ABCDE Criteria	Clinical melanoma checklist: Asymmetry, Border irregularity, Color variation, Diameter >6mm, Evolution. We compute A-D as engineered features.
GBDT (LightGBM)	Gradient Boosted Decision Trees. Ensemble of shallow trees trained sequentially, each correcting the previous one's errors. Excels on tabular data.

Stacking	Using one model's outputs as inputs to another. Here: ViT/ConvNeXt embeddings + tabular features → GBDT for the final prediction.
Mixed Precision (FP16)	Using 16-bit floats for forward/backward passes (2× speed), 32-bit for weights (precision). GradScaler prevents gradient underflow.
GradScaler	Scales the loss up before backprop so small gradients stay in FP16 range, then scales gradients back down before weight updates.
Early Stopping	Halt training when validation metric stops improving for N epochs (patience). Prevents overfitting and wasted compute.
Differential Learning Rate	Lower LR for pretrained backbone (1e-5), higher for new head (1e-3). Backbone needs gentle tuning; head learns from scratch.
Out-of-Fold (OOF)	Predictions made on each sample when it was in the held-out validation fold. Provides unbiased performance estimate for the full dataset.
EDA	Exploratory Data Analysis. Visualize and understand data before modeling. Informs every downstream design decision.

Your Score

12 / 14 correct on first attempt (86%)

Missed: **Q6** (ReLU / linearity collapse) and **Q7** (Focal Loss definition — answered with a great conceptual explanation of loss functions that showed strong foundations). Both are now covered in the explanations above.