Vision Encoders for Automated Dermatology Screening

Section 1: Industry Context and Business Problem

Industry: Digital Health -- AI-Assisted Dermatology

Company Profile: DermaScan AI

DermaScan AI is a Series B digital health startup (founded 2022, 85 employees) that provides AI-powered skin lesion analysis to dermatology clinics and primary care practices across the United States and Europe. Their core product is a cloud-based diagnostic support tool that analyzes dermoscopic images (high-magnification photographs of skin lesions) and provides clinicians with risk assessments for melanoma and other skin cancers.

The Business Challenge

Skin cancer is the most common cancer worldwide, with over 5 million cases diagnosed annually in the United States alone. Early detection of melanoma -- the deadliest form -- improves 5-year survival from 30% (late-stage) to 99% (early-stage). However, dermatologist availability is severely limited: there are approximately 12,000 practicing dermatologists for 330 million Americans, creating wait times of 4-8 weeks in many regions.

DermaScan's current system uses a ResNet-50 encoder (pretrained on ImageNet) fine-tuned on 25,000 labeled dermoscopic images. The system classifies images into 7 diagnostic categories (melanoma, basal cell carcinoma, benign keratosis, dermatofibroma, melanocytic nevus, vascular lesion, actinic keratosis).

The problem: The current system achieves 82% overall accuracy, but only 74% sensitivity on melanoma -- meaning 26% of melanoma cases are missed. The clinical requirement is >90% melanoma sensitivity (to serve as a reliable screening tool) with a false positive rate below 15% (to avoid unnecessary biopsies).

Stakes

- **Clinical impact:** Missing a melanoma diagnosis can be fatal. Every percentage point of sensitivity improvement translates to hundreds of saved lives annually across their 200+ clinic partners.
- **Revenue impact:** DermaScan operates on a per-scan pricing model (\$4.50/scan). With 500,000 scans/month, achieving the clinical accuracy threshold unlocks regulatory approval (FDA 510(k) pathway) and a projected 3x expansion in clinic adoption.
- **Competitive pressure:** Three competitors are developing similar tools. First to achieve FDA clearance captures the market.

Constraints

- **Labeled data:** 25,000 dermoscopic images with histopathology-confirmed labels. Acquiring new labeled data costs \$85 per image (requires biopsy confirmation).
- **Inference latency:** Clinicians expect results within 3 seconds per image (including upload, processing, response).
- **Compute budget:** Inference must run on a single NVIDIA T4 GPU (clinic-deployed edge devices).
- **Regulatory:** The model must provide interpretability (attention maps or saliency) to satisfy FDA requirements for clinical decision support.

Section 2: Technical Problem Formulation

Problem Type: Multi-Class Image Classification with Class-Weighted Sensitivity Optimization

Justification: The problem is fundamentally a classification task (assigning one of 7 diagnostic categories to each dermoscopic image), but the objective is not simply maximizing overall accuracy. The clinical utility depends critically on the **sensitivity for melanoma** (the positive class in a one-vs-rest framing), which requires a loss function and evaluation strategy that explicitly prioritizes melanoma detection.

Input/Output Specifications

Input: RGB dermoscopic image, resized to 224x224 pixels, normalized to ImageNet statistics. - Raw capture: 1024x768 or higher, center-cropped and resized - Color space: sRGB - Augmentation at training time: random rotation, horizontal/vertical flip, color jitter, random erasing

Output: A probability distribution over 7 classes, $\hat{y} \in \mathbb{R}^7$, where $\sum_i \hat{y}_i = 1$. - Primary diagnostic: $\arg\max_i \hat{y}_i$ - Melanoma risk score: $\hat{y}_{\text{melanoma}} \in [0, 1]$

Mathematical Foundation

The core architecture is a vision encoder that maps an input image $x \in \mathbb{R}^{3\times224\times224}$ to a feature representation $z \in \mathbb{R}^D$, followed by a classification head:

$$z = f_{\text{encoder}}(x), \quad \hat{y} = \text{softmax}(Wz + b)$$

where f_{encoder} is either a CNN (e.g., ResNet-50) or a Vision Transformer (ViT-B/16), $W \in \mathbb{R}^{7\times D}$ is the classification weight matrix, and $b \in \mathbb{R}^7$ is the bias vector.

CNN encoder (ResNet-50): The encoder applies a sequence of residual blocks with skip connections:

$$z_l = \mathcal{F}(z_{l-1}, W_l) + z_{l-1}$$

where $\mathcal{F}$ denotes the convolutional transformation within each block, and the skip connection z_{l-1} enables gradient flow through deep networks. The final representation z is obtained via global average pooling over the spatial dimensions.

ViT encoder (ViT-B/16): The encoder splits the image into 196 patches of size 16x16, projects each to dimension $D = 768$, prepends a [CLS] token, adds positional embeddings, and processes through 12 Transformer encoder blocks. The [CLS] token output serves as z.

Loss Function

$$\mathcal{L} = \underbrace{-\sum_{i=1}^{7} w_i \cdot y_i \cdot \log(\hat{y}_i)}_{\text{Weighted Cross-Entropy}} + \underbrace{\lambda_1 \cdot \text{FocalLoss}(\hat{y}_{\text{mel}}, y_{\text{mel}})}_{\text{Melanoma-Focused Focal Loss}} + \underbrace{\lambda_2 \cdot \|W\|_F^2}_{\text{Weight Decay}}$$

Term 1 -- Weighted Cross-Entropy: The standard cross-entropy loss, but each class receives a weight w_i inversely proportional to its frequency. This addresses class imbalance: melanoma represents only 8% of the training set. Without class weighting, the model could achieve 92% accuracy by never predicting melanoma.

Term 2 -- Focal Loss (Melanoma-Specific): The focal loss $\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t)$ with $\gamma = 2$ down-weights easy-to-classify samples and focuses learning on hard melanoma examples. This is applied specifically to the melanoma class to improve sensitivity on borderline cases. $\lambda_1 = 0.3$ controls the contribution relative to the main loss.

Term 3 -- L2 Regularization (Weight Decay): Prevents overfitting on the limited training set. Particularly important for ViT, which has 86M parameters but only 25K training images. $\lambda_2 = 0.01$.

Evaluation Metrics

1. **Melanoma Sensitivity (Primary):** $\text{Sensitivity} = \frac{\text{TP}_{\text{mel}}}{\text{TP}_{\text{mel}} + \text{FN}_{\text{mel}}}$ -- must exceed 90%.
2. **Melanoma Specificity:** $\text{Specificity} = \frac{\text{TN}_{\text{mel}}}{\text{TN}_{\text{mel}} + \text{FP}_{\text{mel}}}$ -- target >85% (false positive rate <15%).
3. **Overall Accuracy:** Fraction of correctly classified images across all 7 classes.
4. **AUROC (Melanoma):** Area under the ROC curve for the melanoma vs. rest binary classification.
5. **Per-Class F1 Score:** Harmonic mean of precision and recall for each class.

Baseline

The current baseline is ResNet-50 pretrained on ImageNet, fine-tuned on the 25K dermoscopic images with standard cross-entropy loss and uniform class weights: - Overall accuracy: 82% - Melanoma sensitivity: 74% - Melanoma specificity: 88% - Melanoma AUROC: 0.85

Why Vision Encoders Are the Right Technical Solution

1. **The problem is fundamentally visual.** Dermoscopic features (asymmetry, border irregularity, color variation, diameter, evolution -- the ABCDE criteria) are spatial patterns that require hierarchical feature extraction.
2. **Transfer learning is critical.** With only 25K labeled images, training from scratch is insufficient. Both CNNs and ViTs pretrained on ImageNet provide strong initializations that transfer well to medical imaging.
3. **Attention-based interpretability.** ViTs provide built-in attention maps that highlight which regions of the lesion influence the diagnosis, satisfying FDA interpretability requirements without requiring post-hoc explanation methods (e.g., GradCAM).
4. **The CNN-vs-ViT trade-off is directly relevant.** The limited dataset size (25K) creates a tension between CNN inductive biases (beneficial for small data) and ViT flexibility (beneficial for capturing global lesion patterns). This case study requires the student to evaluate both approaches empirically.

Section 3: Implementation Notebook Structure

3.1 Data Loading and Preprocessing

Load the ISIC 2018 dermoscopic image dataset (or a synthetic subset), apply preprocessing (resize, normalize), and implement class-weighted sampling.

```
def load_dermascan_dataset(data_dir: str, img_size: int = 224, augment: bool = True):
    """
    Load and preprocess the dermoscopic image dataset.

    Args:
        data_dir: Path to the image directory
        img_size: Target image size (default 224)
        augment: Whether to apply training augmentation

    Returns:
        train_loader, val_loader, test_loader, class_weights

    TODO:
        1. Define training transforms with augmentation (RandomRotation, ColorJitter, RandomErasing)
        2. Define validation/test transforms (only resize and normalize)
        3. Compute class weights inversely proportional to class frequency
        4. Create WeightedRandomSampler for training

    Hint: Use torchvision.datasets.ImageFolder and torch.utils.data.WeightedRandomSampler.
          Class weights should be: w_i = N_total / (N_classes * N_i)
    """
    pass
```

3.2 Exploratory Data Analysis (EDA)

Visualize the class distribution, sample images from each class, and analyze image statistics.

```
def run_eda(dataset, class_names: list):
    """
    Exploratory data analysis on the dermoscopic dataset.
```

```
    TODO:
        1. Plot class distribution (bar chart)
        2. Display 3 sample images per class in a grid
        3. Compute and plot per-channel pixel intensity histograms
        4. Identify class imbalance ratio (max_count / min_count)

    Hint: Melanoma is typically the minority class (5-10% of samples).
    """
    pass
```

3.3 Baseline Model

Implement and train the ResNet-50 baseline with standard cross-entropy loss.

```
def create_baseline_model(num_classes: int = 7, pretrained: bool = True):
    """
    Create a ResNet-50 baseline model with ImageNet pretrained weights.

    Args:
        num_classes: Number of output classes
        pretrained: Whether to use ImageNet pretrained weights

    Returns:
        model: ResNet-50 with modified final classification layer

    TODO:
        1. Load ResNet-50 with pretrained ImageNet weights
        2. Replace the final fc layer: model.fc = nn.Linear(2048, num_classes)
        3. Freeze the first 6 layers (fine-tune only deeper layers + classifier)

    Hint: Use torchvision.models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V2)
    """
    pass
```

3.4 Core Model Architecture

Implement both the CNN (ResNet-50) and ViT (ViT-B/16) encoders with the custom classification head.

```
def create_vit_model(num_classes: int = 7, pretrained: bool = True):
    """
    Create a Vision Transformer (ViT-B/16) model with ImageNet pretrained weights.

    Args:
        num_classes: Number of output classes
        pretrained: Whether to use ImageNet pretrained weights

    Returns:
        model: ViT-B/16 with modified classification head

    TODO:
        1. Load ViT-B/16 with pretrained ImageNet weights
        2. Replace the classification head: model.heads.head = nn.Linear(768, num_classes)
        3. Optionally: freeze the patch embedding layer

    Hint: Use torchvision.models.vit_b_16(weights=ViT_B_16_Weights.IMAGENET1K_V1)
    """
    pass
```

```
def create_loss_function(class_weights: torch.Tensor, focal_gamma: float = 2.0,
                         focal_weight: float = 0.3, melanoma_idx: int = 0):
    """
    Create the combined loss function (weighted CE + focal loss on melanoma).

    Args:
        class_weights: Tensor of per-class weights
        focal_gamma: Focal loss gamma parameter
```

```
        focal_weight: Weight for the focal loss term (lambda_1)
        melanoma_idx: Index of the melanoma class

    Returns:
        loss_fn: Callable that takes (predictions, targets) and returns loss

    TODO:
        1. Implement weighted cross-entropy: nn.CrossEntropyLoss(weight=class_weights)
        2. Implement focal loss for the melanoma class:
           - Extract melanoma probability: p_mel = softmax(logits)[:, melanoma_idx]
           - For melanoma samples: FL = -(1 - p_mel)^gamma * log(p_mel)
           - For non-melanoma samples: FL = -(p_mel)^gamma * log(1 - p_mel)
        3. Return weighted sum: CE + focal_weight * FL

    Hint: The focal loss modulates the cross-entropy by (1 - p_t)^gamma,
          focusing on hard examples where p_t is small.
    """
    pass
```

3.5 Training Pipeline

Implement the training loop with learning rate scheduling, early stopping, and model checkpointing.

```
def train_model(model, train_loader, val_loader, loss_fn,
                num_epochs: int = 30, lr: float = 1e-4,
                patience: int = 5, model_name: str = "model"):
    """
    Train the model with validation monitoring and early stopping.

    Args:
        model: The vision encoder model
        train_loader: Training data loader
        val_loader: Validation data loader
        loss_fn: The combined loss function
        num_epochs: Maximum training epochs
        lr: Learning rate
        patience: Early stopping patience
        model_name: Name for checkpointing

    Returns:
        history: Dict with training/validation metrics per epoch
        best_model_state: State dict of the best model

    TODO:
        1. Use AdamW optimizer with weight_decay=0.01
        2. Use CosineAnnealingLR scheduler
        3. Track: train_loss, val_loss, val_accuracy, melanoma_sensitivity
        4. Implement early stopping on melanoma_sensitivity (primary metric)
        5. Save best model checkpoint

    Hint: Melanoma sensitivity = TP / (TP + FN) for the melanoma class only.
    """
    pass
```

3.6 Evaluation

Compute all metrics and generate diagnostic visualizations.

```
def evaluate_model(model, test_loader, class_names: list, melanoma_idx: int = 0):
    """
    Comprehensive evaluation of the trained model.

    Args:
        model: Trained model
        test_loader: Test data loader
        class_names: List of class names
        melanoma_idx: Index of the melanoma class
```

```
    Returns:
        metrics: Dict with all evaluation metrics

    TODO:
        1. Compute confusion matrix
        2. Compute per-class precision, recall, F1
        3. Compute melanoma sensitivity and specificity
        4. Compute melanoma AUROC
        5. Plot ROC curve for melanoma
        6. Plot confusion matrix heatmap

    Hint: Use sklearn.metrics for confusion_matrix, roc_curve, roc_auc_score.
    """
    pass
```

3.7 Error Analysis

Analyze failure cases, particularly missed melanomas.

```
def error_analysis(model, test_loader, class_names: list, melanoma_idx: int = 0):
    """
    Analyze model errors, focusing on missed melanoma cases.

    TODO:
        1. Collect all false negative melanoma cases (predicted non-melanoma, true melanoma)
        2. Display the top 10 most confident false negatives with predicted probabilities
        3. For ViT models: extract and display attention maps for these cases
        4. Analyze what visual features these cases share
        5. Compute error rate by lesion size (small/medium/large)

    Hint: For ViT attention maps, extract the attention weights from the last
          Transformer block and visualize the CLS token's attention to patches.
    """
    pass
```

3.8 Deployment Considerations

Optimize the model for clinic-deployed edge inference.

```
def optimize_for_deployment(model, sample_input, target_latency_ms: float = 3000):
    """
    Optimize the model for deployment on T4 GPU edge devices.

    TODO:
        1. Measure baseline inference latency (forward pass time)
        2. Apply torch.compile() for graph optimization
        3. Convert to TorchScript via torch.jit.trace
        4. Test FP16 inference (model.half())
        5. Benchmark: measure throughput (images/second) and latency
        6. Verify accuracy is preserved after optimization

    Hint: Target is <3 seconds per image including preprocessing.
          FP16 typically provides 2x speedup on T4 with <0.1% accuracy loss.
    """
    pass
```

3.9 Ethical Considerations

Address bias, fairness, and responsible deployment.

```
def fairness_audit(model, test_loader, demographic_labels: dict):
    """
    Audit model performance across demographic subgroups.

    TODO:
```

```
        1. Compute melanoma sensitivity by skin type (Fitzpatrick I-VI)
        2. Compute melanoma sensitivity by patient age group
        3. Identify performance disparities (>5% gap is concerning)
        4. Propose mitigation strategies for any identified biases
        5. Generate a fairness report summarizing findings

    Hint: Dermoscopic imaging performance varies significantly with skin pigmentation.
          Models trained predominantly on Fitzpatrick I-III skin types often
          underperform on Fitzpatrick IV-VI. This is a known and serious problem
          in dermatology AI.
    """
    pass
```

Section 4: Production and System Design Extension

4.1 System Architecture

The production system consists of three tiers:

1. **Edge Tier (Clinic Devices):** NVIDIA Jetson or T4-equipped workstations in each clinic capture dermoscopic images, run preprocessing, and perform inference using the optimized TorchScript model. Results are returned in <3 seconds.
2. **Cloud Tier (Central Platform):** A Kubernetes cluster on GCP hosts the training pipeline, model registry (MLflow), data warehouse (BigQuery), and the clinician-facing web dashboard. New images flagged for review are stored in Cloud Storage.
3. **Feedback Tier:** Clinician confirmations/corrections flow back to the training pipeline for continuous model improvement. Histopathology results (ground truth) are linked to predictions for validation.

4.2 API Design

```
POST /api/v1/analyze
  Input: { "image": base64_encoded_png, "clinic_id": str, "patient_id": str }
  Output: {
    "diagnosis": {
      "primary": "melanocytic_nevus",
      "confidence": 0.87,
      "melanoma_risk": 0.03,
      "all_classes": { ... }
    },
    "attention_map": base64_encoded_heatmap,
    "recommendation": "Low risk. Routine follow-up.",
    "model_version": "v2.4.1",
    "latency_ms": 1840
  }
```

4.3 Model Serving

- **Primary:** TorchServe with custom handler for preprocessing + inference + attention map extraction
- **Optimization:** FP16 inference, TorchScript compiled model, batch processing for multiple lesions per visit

- **Fallback:** If edge device is unavailable, images are uploaded to the cloud tier for inference (adds 1-2 seconds for upload)

4.4 Monitoring

- **Model quality metrics:** Track melanoma sensitivity, specificity, and AUROC daily on incoming production data. Alert if melanoma sensitivity drops below 88% (2% margin below the 90% threshold).
- **System metrics:** Inference latency (p50, p95, p99), GPU utilization, memory usage, queue depth.
- **Data quality:** Monitor image resolution, brightness distribution, color calibration drift (different dermoscopes produce different color profiles).

4.5 Drift Detection

- **Input drift:** Monitor the distribution of image features (mean pixel intensity, color histogram) using Population Stability Index (PSI). Trigger retraining if PSI > 0.25.
- **Prediction drift:** Track the running distribution of predicted class probabilities. If the melanoma prediction rate shifts by >2 standard deviations from the training distribution, trigger investigation.
- **Concept drift:** Compare model predictions against clinician feedback and histopathology results. If disagreement rate exceeds 20%, trigger full model retraining.

4.6 A/B Testing

- **Protocol:** When deploying a new model version, run both old and new models on 10% of incoming scans. Both predictions are generated but only the old model's result is shown to clinicians.
- **Evaluation window:** 30 days minimum, or until 500 confirmed melanoma cases are processed.
- **Promotion criteria:** New model must achieve statistically significant improvement in melanoma sensitivity ($p < 0.05$, binomial test) without degrading specificity by more than 2%.

4.7 CI/CD Pipeline

1. **Data versioning:** DVC tracks training datasets. New labeled images trigger a pipeline run.
2. **Training:** Automated training on GCP Vertex AI with hyperparameter sweeps.
3. **Validation:** Automated evaluation against the held-out test set. Model must pass:
4. Melanoma sensitivity > 90%
5. Overall accuracy > 84%
6. Inference latency < 2500ms on T4
7. **Staging:** Approved models deployed to a staging environment for clinician review.

8. **Production:** After 1-week staging approval, model promoted to production with gradual rollout (10% -> 50% -> 100% over 72 hours).

4.8 Cost Analysis

Component	Monthly Cost
Edge GPU (200 clinics x T4)	\$48,000
Cloud training (4x A100, 20 hrs/month)	\$3,200
Cloud storage (500K images, 5TB)	\$115
MLflow / monitoring	\$500
Data labeling (1000 new images/month)	\$85,000
Total	**\$136,815**

Revenue at 500K scans/month x \$4.50/scan = \$2.25M/month, yielding a healthy margin.

The largest cost is data labeling (\$85K/month). Active learning strategies -- where the model identifies the most informative images for labeling -- can reduce this by 40-60% while maintaining model improvement rates.