Retinal Disease Classifier

Presented by:

BATUHAN ERSAN
ENES KELEŞ
AHMET BUĞRAHAN ÇINAR

Introduction and Problem Definition

Problem

- *Retinal image analysis is a time-consuming and specialized process for ophthalmologists.*
- *Early detection is critical to preventing loss.*
- *Access to specialist physicians is not always and everywhere possible, which can delay diagnosis and treatment processes.*

Solution Proposal

With an AI-powered automated diagnostic system:

- *Provides fast and reliable preliminary screening.*
- *Reduces the workload of specialist physicians and allows them to focus on more complex cases.*
- *Enables early intervention for retinal diseases, thus helping to prevent vision loss.*

Motivation and Goal

Motivation: Fighting Preventable Blindness

- ***Diabetic Retinopathy (DR):*** *One of the most common causes of preventable blindness worldwide. Early diagnosis increases treatment success.*
- ***ARMD (Age-Related Macular Degeneration):*** *A leading cause of vision loss, especially in individuals over 60. Early detection can slow disease progression.*
- *Early detection is vital for preventing potential blindness and improving patients' quality of life.*

Project Goal: Automatic Detection of 16 Retinal Diseases

We aim to develop a robust and accurate deep learning model capable of automatically detecting ***16 different retinal diseases*** *from fundus images.*

This model will provide valuable support to ophthalmologists in the diagnostic process, saving time and resources.

Multi-Label Classification: Comprehensive Diagnosis

*Multi-label classification reflects the fact that **multiple diseases can be present simultaneously** in a single retinal image.*

Real-World Scenarios

The model more accurately reflects the patient's clinical picture, addressing the shortcomings of systems that focus on a single disease.

More Comprehensive Diagnosis

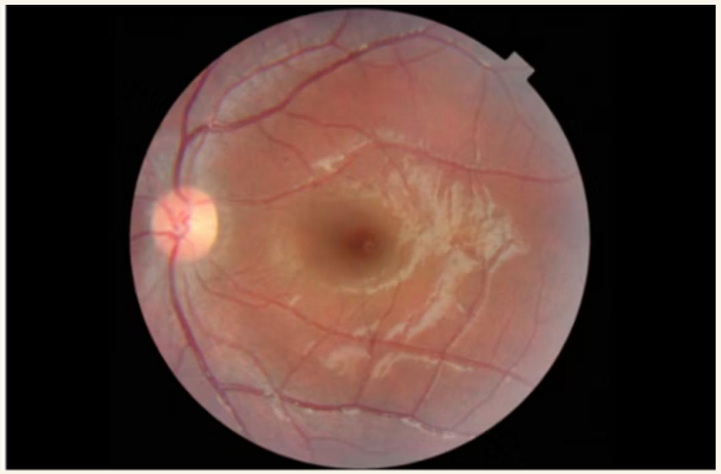
Detecting multiple pathologies simultaneously allows for a more holistic evaluation and treatment plan.

Efficient Multi-Disease Detection

Detecting multiple diseases with a single model accelerates diagnostic processes and reduces the workload of specialists.

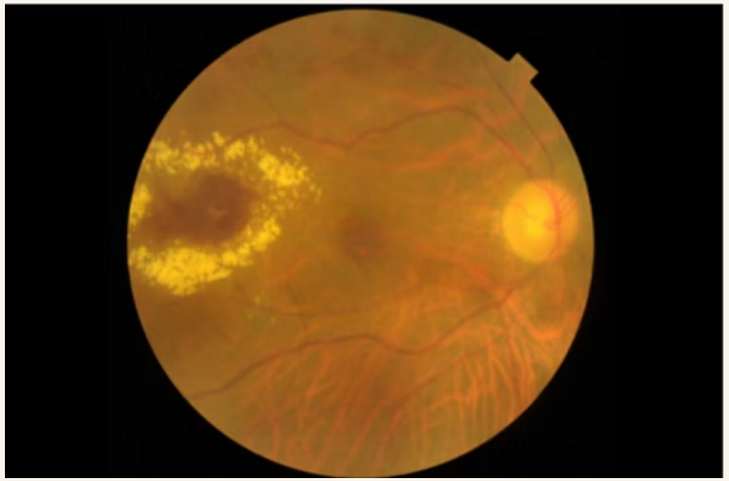
Visualization: Normal and Diseased Retinal Images

This visualization highlights the importance of multi-label classification by comparing a normal retinal image with a fundus image showing various signs of disease.



Normal Retina

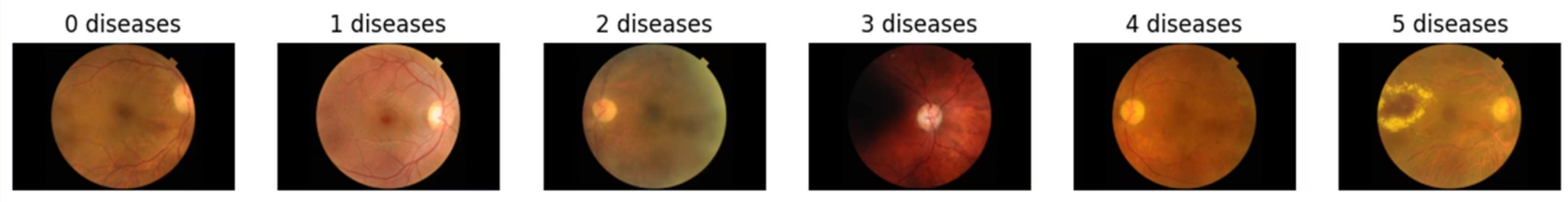
- *Healthy optic disc*
- *Clear macular region*
- *Regular and branched vascular structures*
- *No signs of lesions or edema*



Diseases (DR + MH + DN + ERM + TSLN)

- *Signs of Diabetic Retinopathy (DR): microaneurysms, hemorrhages, exudates*
- *Signs of Macular Hole (MH): opening in the macular region*
- *Signs of Drusen (DN): yellowish-white deposits in the macula*
- *Signs of Epiretinal Membrane (ERM): thin membrane layer over the macula*
- *Signs of Tessellation (TSLN): visible choroidal vessels due to retinal thinning*
- *Abnormal vascular changes and multiple lesions*

Fundus Images Containing Diseases Ranging from 0 to 5



Dataset: Training with RFMiD

Source: RFMiD

*The **RFMiD (Retinal Fundus Multi-disease Image Dataset)** dataset, accessible via Kaggle, forms the foundation of our project. This dataset consists of real clinical data collected from India.*

Data Size: 3200 Images

*Containing a total of **3200 fundus images**, this dataset provides sufficient diversity for training, validating, and testing our model.*

Set	*Number of Images*	*Usage*
Training	*1920*	*Main set for model training*
Validation	*640*	*Monitoring model performance*
Test	*640*	*Final model evaluation*

Dataset Challenges: Class Imbalance and Complexity

The RFMiD dataset presents several significant challenges stemming from real-world clinical data.

Severe Class Imbalance

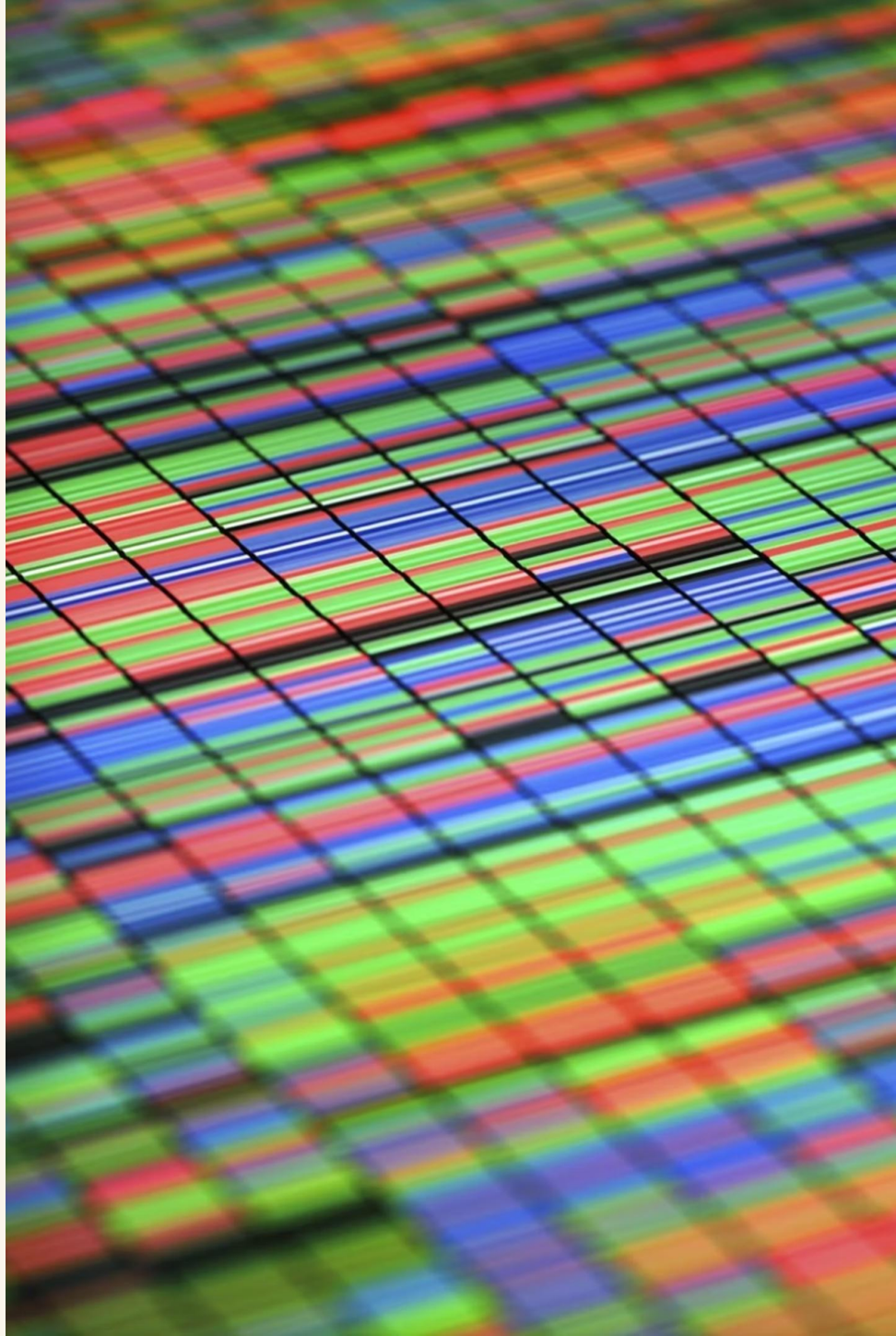
*There is a significant imbalance among classes in the dataset. The most common class, Diabetic Retinopathy (DR), has **352 examples**, while the rarest class, Retinal Pigment Epithelium Degeneration (RPEC), has only **4 examples**. This indicates an imbalance ratio of approximately **90:1**.*

Multi-label Complexity

The presence of 0 to 5 or more diseases simultaneously in a single image complicates model learning. Additionally, existing correlations between disease classes are another factor that makes it difficult for the model to make accurate predictions.

Learning Rare Classes

Having very few examples for some disease classes prevents the model from reliably learning and generalizing these rare classes.



Class Count Optimization: For a More Reliable Model

Due to severe imbalances in the dataset and an insufficient number of examples for rare classes, the class count has been optimized to improve model performance and obtain reliable predictions.

Decision: Removal of Rare Classes

Rare classes with fewer than 20 examples have been removed from the dataset. *This approach allows the model to generalize better and produce more meaningful results.*

Rationale: Insufficient Data Problem

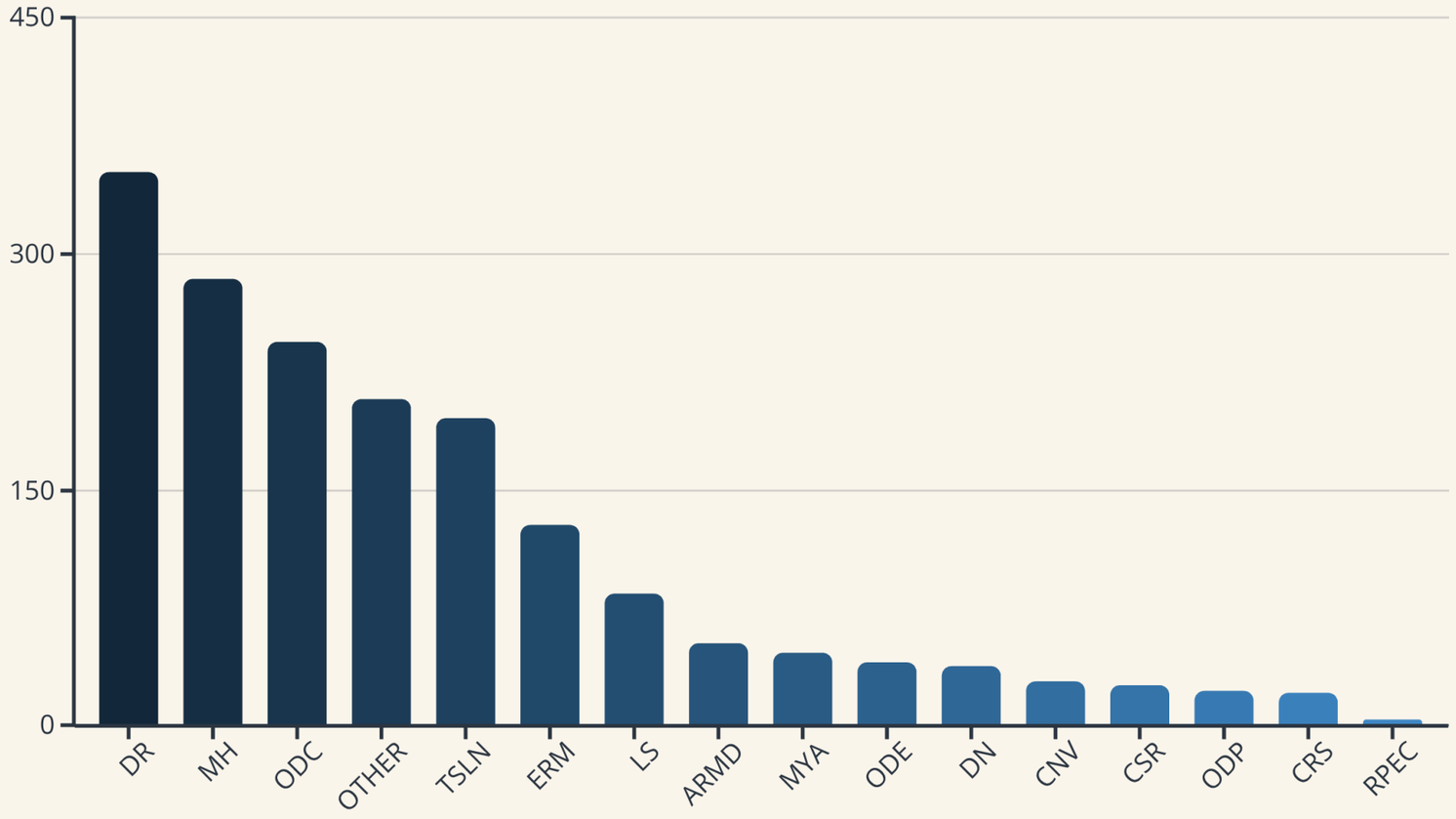
- *Insufficient training data prevents the model from learning these classes correctly.*
- *Meaningful evaluation in validation and test sets is not possible for classes with very few examples.*
- *With very few examples, the model cannot produce reliable predictions for these classes and may tend to overfit.*

Result: From 43 Classes to 16 Classes

As a result of this optimization, the initial number of ***43 classes*** *has been reduced to* ***16 classes****, allowing the model to work more efficiently and accurately.*

Visual: Distribution of the Remaining 16 Disease Classes

This bar chart shows the number of samples for the remaining 16 disease classes after optimization. The significant gap between DR and RPEC, in particular, clearly reveals the extent of data imbalance. The distribution is made more understandable by using a logarithmic scale.



Abbreviation	Disease	Sample Count
DR	Diabetic Retinopathy	352
MH	Macular Hole	284
ODC	Optic Disc Cupping	244
OTHER	Other Retinal Diseases	208
TSLN	Tessellation	195
ERM	Epiretinal Membrane	128
LS	Laser Scars	84
ARMD	Age-Related Macular Degeneration	52
MYA	Pathological Myopia	47
ODE	Optic Disc Edema	40
DN	Drusen	38
CNV	Choroidal Neovascularization	28
CSR	Central Serous Retinopathy	26
ODP	Optic Disc Pallor	22
CRS	Chorioretinitis Scar	21
RPEC	Retinal Pigment Epithelium Change	4

Model Architecture & Training Settings

Why ConvNeXt-Tiny?

Modern CNN Architecture

Introduced by Meta AI in 2022, ConvNeXt represents a significant advancement in Convolutional Neural Networks.

High Performance

Despite its relatively small model size (~28M parameters), it delivers exceptional performance, rivalling Vision Transformers.

Medical Imaging Efficacy

CNNs continue to demonstrate strong effectiveness in medical image analysis tasks.

Comparative Performance

Model	*Parameters*	*ImageNet Accuracy*
ResNet-50	*25M*	*76.1%*
ConvNeXt-Tiny	*28M*	***82.1%***
ViT-S/16	*22M*	*79.8%*

Model Configuration Details

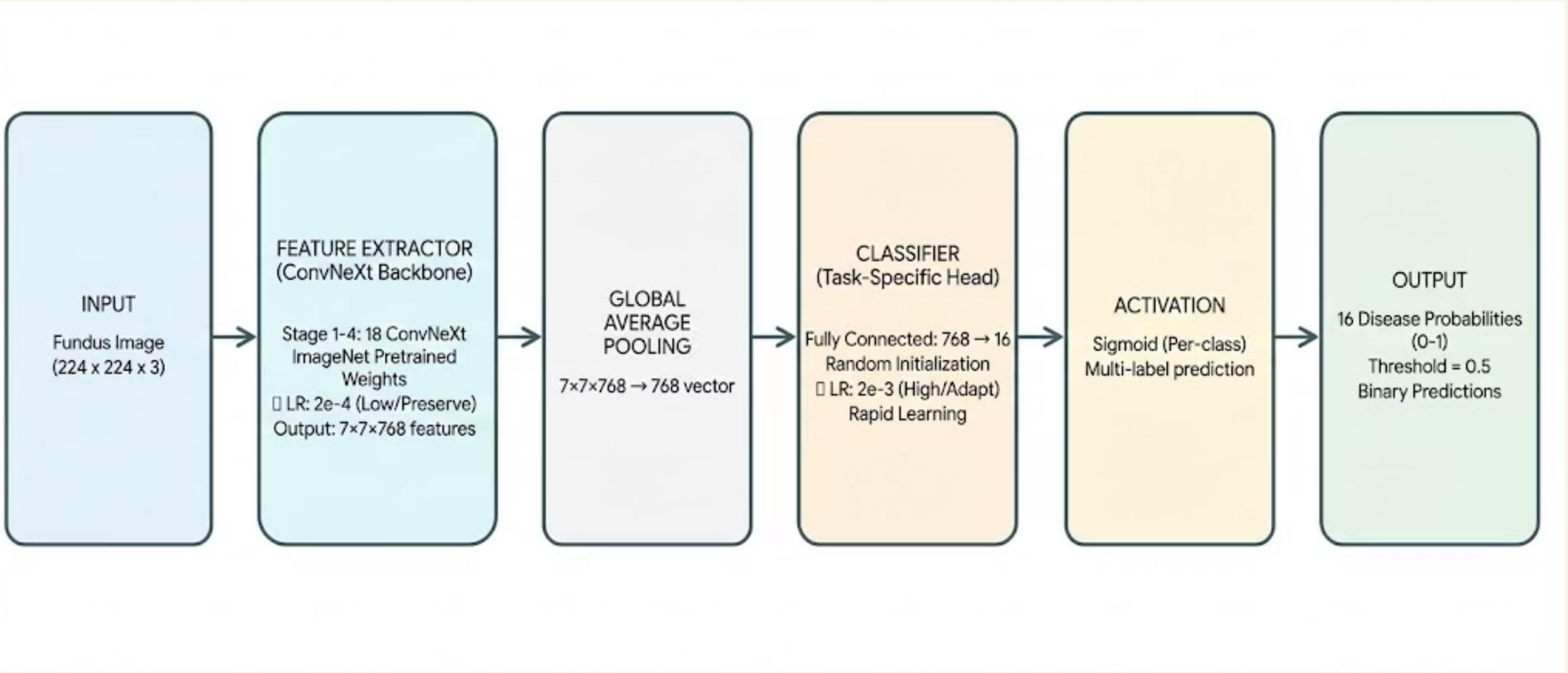
A detailed overview of the ConvNeXt-Tiny model configuration used for this study.

Parameter	Value
Backbone	ConvNeXt-Tiny
Pretrained	IMAGENET1K_V1
Input Size	224 × 224 × 3
Output Classes	16
Feature Dimension	768
Output Activation	***Sigmoid*** *(for Multi-label)*

Why Sigmoid Activation?

In multi-label classification tasks, each class is evaluated ***independently****.*

Unlike Softmax, which assumes mutually exclusive classes, Sigmoid allows for multiple active classes simultaneously.



ConvNeXt Architecture Overview

The general structure of the ConvNeXt architecture comprises a Stem layer followed by four distinct stages, progressively transforming input features.

General Structure: Stem + 4 Stages

The table below illustrates the changes in output dimensions and channel counts across the different stages, detailing the feature extraction process.

Stage	*Output Size*	*Channels*	*Blocks*
Stem	*56×56*	*96*	*1*
Stage 1	*56×56*	*96*	*3*
Stage 2	*28×28*	*192*	*3*
Stage 3	*14×14*	*384*	*9*
Stage 4	*7×7*	*768*	*3*

ConvNeXt Block Details

Each ConvNeXt block integrates several key components inspired by modern deep learning advancements, optimising feature learning and network stability.

Core Components

Depthwise Convolution (7×7)

Performs convolution independently on each input channel, enhancing spatial feature extraction.

Layer Normalisation

Replaces BatchNorm for improved stability and generalisation across different batch sizes.

Inverted Bottleneck

Utilises a narrow-to-wide-to-narrow structure, increasing computational efficiency.

GELU Activation

Inspired by Transformer architectures, the Gaussian Error Linear Unit provides non-linearity.

Transfer Learning Strategy

We adopt a Discriminative Fine-Tuning approach, leveraging the robust low-level features learned from the extensive ImageNet dataset.

Method: Discriminative Fine-Tuning

This method is predicated on the understanding that fundamental visual patterns, such as edges and textures, remain consistent across various image domains, including retinal scans.

Underlying Logic

Low-Level Features

- *Edges, textures, and basic shapes*
- *Generally applicable across all image types*
- *Learned effectively from large datasets like ImageNet*

High-Level Features

- *Complex object recognition (e.g., specific pathologies in medical images)*
- *Task-specific and require adaptation to the target dataset*

Strategic Adaptation

The core strategy is to preserve the general, pre-trained weights of the feature extractor while adapting only the classifier layers to the specific demands of the new task, thereby accelerating learning and improving performance.

Discriminative Learning Rates

To optimise the fine-tuning process, different learning rates are applied to distinct layers of the model, allowing for more nuanced adaptation.

Implementation Details

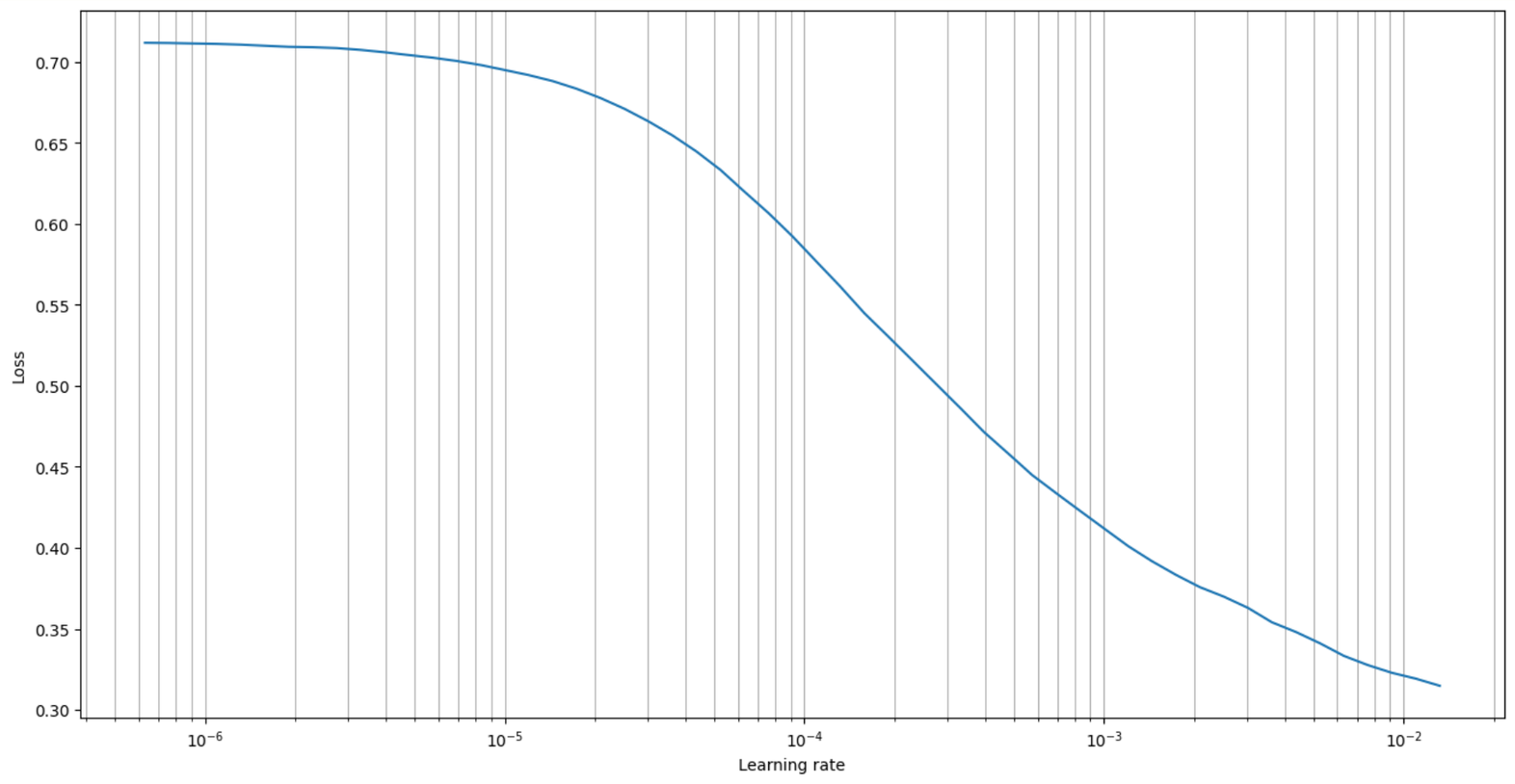
Layer	*Learning Rate*	*Description*
Feature Extractor	*LR / 15 = **1.33e-4***	*Slower updates to preserve general features*
Classifier	*LR × 0.8 = **1.6e-3***	*Faster updates for rapid adaptation to new task specifics*

Code Example Snippet

```
optimizer = AdamW([
    {'params': model.features.parameters(), 'lr': LR / 15},
    {'params': model.classifier.parameters(), 'lr': LR * 0.8}
], weight_decay=1e-4)
```

This approach effectively maintains the integrity of pre-trained knowledge while enabling swift adaptation to the specific requirements of the new classification task.

Learning Rate



Refined Training Strategy

Our training methodology incorporates advanced optimisation techniques to ensure robust model performance and efficient convergence.

Optimiser: AdamW

- *Adam with decoupled Weight Decay (L2 regularisation).*
- *Weight Decay value set to: **1e-4**.*
- *AdamW helps to generalise better by preventing overfitting, especially on complex datasets.*

Scheduler: Cosine Annealing Learning Rate

A cosine annealing learning rate schedule is implemented to gradually reduce the learning rate, allowing for fine-grained adjustments as training progresses.

```
scheduler = CosineAnnealingLR(
    optimizer,
    T_max=50,  # Total number of epochs
    eta_min=1e-7 # Minimum learning rate
)
```

This scheduler ensures that the learning rate decreases smoothly over time, preventing premature convergence and promoting better model exploration of the loss landscape.

Training Parameters

Parameter	*Previous Value*	*New Value*
Epochs	*30*	***50***
Early Stopping Patience	*5*	***8***
Base Learning Rate	*2e-3*	*2e-3*
Weight Decay	*1e-4*	*1e-4*
Batch Size	*64*	*64*
Gradient Clipping	*1.0*	*1.0*

Rationale for Parameter Adjustments

Increased Epochs

Longer training allows for more comprehensive learning and convergence towards a better optimum.

Enhanced Early Stopping Patience

A higher patience threshold makes the model more robust to minor fluctuations in validation loss, preventing premature stopping.

Shift in Model Selection Criterion

A critical improvement in our methodology involves transitioning from validation loss to Macro F1 Score for optimal model selection, particularly for imbalanced datasets.

Previous Approach

Validation Loss-Based Selection

Models were previously selected based on the lowest validation loss.

Problematic Outcome

Low loss does not always correlate with high F1 score, especially in scenarios with imbalanced class distributions.

Revised Approach

Macro F1 Score-Based Selection

The best model is now chosen based on the highest Macro F1 Score, following threshold optimisation.

Key Advantage

This metric provides a balanced measure of performance across all classes, addressing issues related to class imbalance effectively.

Fundamental Data Preprocessing

Standard preprocessing steps are applied to the retinal images to ensure consistency and optimal input for the ConvNeXt model.

Image Resizing

*All input images are uniformly resized to a dimension of **224 × 224 pixels**.*

ImageNet Normalisation

Normalisation is performed using the mean and standard deviation values derived from the ImageNet dataset.

```
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
Normalize(mean, std)
```

Why ImageNet Statistics?

The ConvNeXt model was originally pre-trained on ImageNet using these specific normalisation statistics. Applying the same normalisation during fine-tuning ensures that the input data distribution aligns with what the model learned during pre-training, thereby enhancing the effectiveness of transfer learning.

Advanced Augmentation and Imbalance Solutions

Train Transform (Augmentation)

*The training pipeline incorporates an **advanced augmentation chain** to significantly increase data variability, thereby improving model generalisation and reducing overfitting. Each transformation is carefully selected for its relevance in medical imaging contexts.*

```
train_transform = transforms.v2.Compose([
 Resize(224),
 RandomAdjustSharpness(sharpness_factor=2, p=0.8),
 RandomRotation(degrees=180),
 ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
 RandomAffine(degrees=15, translate=(0.1, 0.1), scale=(0.9, 1.1)),
 GaussianBlur(kernel_size=3, sigma=(0.1, 2.0)),
 RandomHorizontalFlip(p=0.5),
 RandomVerticalFlip(p=0.5),
 CenterCrop(224),
 RandomErasing(p=0.2),
 Normalize(mean, std)
])
```

Augmentation Rationale: Bridging Image Transformations with Clinical Reality

Each augmentation technique applied in our training pipeline is not arbitrary; it directly mimics potential variations encountered in real-world medical imaging, ensuring our model is robust and clinically relevant.

RandomRotation (180°)	*Accounts for different patient orientations or varied camera angles during image acquisition, simulating diverse viewing perspectives.*
ColorJitter	*Mimics variations in imaging equipment, lighting conditions, or tissue staining, ensuring colour and brightness robustness.*
RandomAffine	*Simulates slight shifts in patient positioning or variations in anatomical scale and location within the image frame.*
GaussianBlur	*Reproduces minor focus discrepancies or image blurring that can occur during the imaging process.*
RandomErasing	*Helps prevent overfitting and enhances robustness against occlusions (e.g., surgical instruments, debris) that might obscure parts of the region of interest.*
Flip (Horizontal/Vertical)	*Leverages anatomical symmetries where applicable, training the model to recognise features regardless of their horizontal or vertical orientation.*

Original

After transform

Addressing Class Imbalance: A Critical Challenge

One of the most persistent problems in medical imaging datasets is severe class imbalance, which can severely hinder model performance, particularly for rare but clinically significant conditions.

Problem Revisited: Disproportionate Class Representation

Our dataset exhibits a stark disparity in class examples, particularly between common conditions like Diabetic Retinopathy (DR) and the exceptionally rare Retinal Pigment Epithelial Changes (RPEC).

DR	*352*
RPEC	*4*

*This represents an alarming ratio of approximately **90:1** in favour of the DR class.*

Consequences of Imbalance

- *The model develops a strong bias towards the majority class (DR).*
- *It tends to ignore or misclassify rare classes (RPEC) due to insufficient exposure.*
- *Reported accuracy metrics can be misleadingly high, while critical F1-scores for minority classes remain low.*

Solution 1: Focal Loss for Hard Example Mining

To counteract the negative effects of class imbalance, we implemented Focal Loss, a technique designed to make the model focus more on difficult-to-classify examples, which typically belong to the minority classes.

Theoretical Foundation

*Focal Loss reweights the standard cross-entropy loss such that easily classifiable examples (often from majority classes) contribute less to the overall loss. This forces the model to concentrate its learning efforts on **hard examples**, which are usually the underrepresented minority classes.*

$$FL(p_t) = -\alpha \quad (1 - p_t)^{\gamma} \quad \log(p_t)$$

- **γ *(gamma)***: *This parameter controls the degree to which the model focuses on hard examples. A higher gamma value places more emphasis on misclassified or difficult-to-classify instances.*
- **α *(alpha)***: *This is a class-balancing factor that further down-weights the contribution of the majority class, especially when combined with a high gamma.*

Optimising Focal Loss Parameters for Enhanced Performance

Careful tuning of Focal Loss parameters is essential to achieve the desired balance between focusing on hard examples and managing false positives. Our adjustments aim to further refine the model's learning process.

Parameter Optimisation

γ *(gamma)*	*2.0*	*2.5*	*Increased emphasis on hard examples, further reducing the contribution of easy-to-classify instances.*
α *(alpha)*	*0.25*	*0.15*	*Reduced weighting for the positive class, implicitly giving more weight to negative examples, crucial for controlling False Positives.*

Rationale Behind These Values

- ***Gamma = 2.5**: By elevating the gamma value, we more aggressively down-weight the loss contribution from well-classified examples. This means the model dedicates a greater portion of its learning capacity to the challenging (often minority) instances.*
- ***Alpha = 0.15**: This adjustment provides a stronger regularisation effect by lowering the weight given to the positive class. In multi-label scenarios, a lower alpha can help to suppress over-prediction of positive classes, directly leading to a reduction in False Positives.*

Solution 2: Clamping pos_weight to Enhance Training Stability

The initial approach to calculating `pos_weight` *for extremely rare classes resulted in values that were excessively high, leading to significant instability during training.*

The Problematic Magnitude of `pos_weight` for Rare Classes

For minority classes such as RPEC, the calculated `pos_weight` *based on class distribution was disproportionately large:*

> `pos_weight_RPEC = (Total_Samples - RPEC_Count) / RPEC_Count`
>
> `pos_weight_RPEC = (1920 - 4) / 4 = 479`

Consequences of an Exorbitant `pos_weight`

- ***Aggressive Learning****: An excessively high* `pos_weight` *causes the model to over-prioritise positive examples of the minority class, leading to overly aggressive and often erroneous learning.*
- ***Explosion of False Positives****: The model becomes extremely sensitive to anything resembling the minority class, resulting in a dramatic increase in False Positives.*
- ***Training Instability****: Such high weights can cause gradients to become unstable, hindering convergence and making the training process erratic and unreliable.*

Implementing `pos_weight` Clamping for Robustness

To mitigate the issues caused by extremely high `pos_weight` values, a clamping mechanism was introduced, limiting the maximum allowable weight. This ensures more stable and reliable training without compromising the focus on minority classes.

Practical Implementation

```
MAX_POS_WEIGHT = 10.0

pos_weight = (n_samples - pos_counts) / pos_counts

pos_weight = torch.clamp(pos_weight, max=MAX_POS_WEIGHT)
```

By setting a maximum cap, we prevent individual classes from unduly influencing the loss function with extreme weights.

Resulting Weighted Distributions

The impact of `pos_weight` clamping on various classes:

DR	*4.45*	*4.45*
RPEC	*479*	*10.0*
CRS	*90.4*	*10.0*

*This clamping strategy ensures that while rare classes still receive significant attention, their influence is appropriately bounded, leading to **stable and more reliable training** dynamics.*

Solution 3: Weighted Random Sampler for Data Loader Level Balancing

To ensure minority classes are sufficiently represented during each training epoch, we implemented a Weighted Random Sampler at the Data Loader level. This method adjusts the probability of sampling each example based on the rarity of its associated classes.

Underlying Principle

The Weighted Random Sampler assigns a weight to each training example. This weight is determined by the inverse frequency of the ***rarest class present*** *within that example. Consequently, examples containing minority classes are sampled more frequently, effectively oversampling the underrepresented data.*

Code Summary

```
def calculate_sample_weights(labels):

sample_weights = []

for label in labels:

# Weight based on the rarest class contained in the example

max_weight = max(1.0 / class_counts[i] for i in where(label == 1))

sample_weights.append(max_weight)

return sample_weights

sampler = WeightedRandomSampler(weights, num_samples)
```

- *Each example is assigned a weight.*
- *The probability of an example being chosen for a batch is proportional to its assigned weight.*
- *This ensures that, over the course of an epoch, minority class examples appear more often, providing the model with adequate exposure to learn their features.*

Solution 4: Strategic Class Removal for Enhanced Model Focus

A critical decision in handling extreme class imbalance involved removing classes with an insufficient number of examples. This pragmatic approach aimed to improve the overall reliability and interpretability of the model's performance.

Summary of Action

*Classes containing **fewer than 20 examples** were systematically excluded from the dataset.*

Justification for Removal

- ***Insufficient Training Data**: With extremely limited data points, the model struggles to learn any meaningful patterns for these classes, leading to poor generalisation and high uncertainty.*
- ***Meaningless Evaluation**: Robust evaluation on validation and test sets becomes impossible for classes with so few instances, as statistical significance cannot be achieved.*
- ***Low Confidence Predictions**: Models trained with these rare classes often produce very low-confidence predictions, effectively acting as noise rather than valuable insights.*

Positive Outcomes

1

Class Reduction

*The total number of classes was reduced from **43 to 16**, simplifying the classification task.*

2

Reliable Results

*This led to more **reliable and meaningful** evaluation metrics, as all remaining classes had sufficient representation.*

3

Higher F1 Scores

Crucially, the F1 scores for the remaining, more robust classes saw a significant increase, indicating improved diagnostic capability.

Threshold Optimisation in Multi-Label Classification

The Problem with Fixed Thresholds

In multi-label classification, especially with imbalanced datasets, a static threshold of 0.5 for all classes is suboptimal. This approach often leads to biased predictions.

Rare Classes

Models tend to predict these with lower confidence, missing true positives.

Common Classes

These might require higher thresholds to avoid an excessive number of false positives.

Unique Needs

Each class often requires its own distinct optimal threshold for accurate classification.

*Our solution: implementing a **Per-Class Optimal Threshold** algorithm.*

Solution: Per-Class Optimal Threshold

Maximising F1 Score for Each Class

The primary objective is to identify the most suitable threshold value for each class independently, thereby maximising its F1 Score.

Class A: Threshold = 0.35

Yields the highest F1 Score for this specific class.

Class B: Threshold = 0.45

Represents the optimal F1 Score for Class B.

Class C: Threshold = 0.30

The best performing F1 Score for Class C.

This demonstrates that each class benefits from a uniquely tailored optimal threshold, leading to more accurate and nuanced predictions across the board.

Optimal Threshold Function

Detailed Algorithm for Threshold Selection

Our approach involves iterating through a range of threshold values from 0.1 to 0.9, with increments of 0.05, to pinpoint the threshold that yields the highest F1 Score.

```
for thresh in np.arange(0.1, 0.9, 0.05):
    preds = (y_pred[:, i] >= thresh)
    f1 = f1_score(y_true[:, i], preds)
    if f1 > best_f1:
        best_f1 = f1
        best_threshold = thresh
```

*This comprehensive scanning process evaluates **17 distinct threshold values** for each class, ensuring a thorough search for optimal performance.*

Clinical Constraint: MIN_RECALL

Prioritising Patient Safety: $\text{MIN\_RECALL} = 0.3$

In a medical context, the consequences of false negatives can be severe. To mitigate this, a minimum recall constraint is imposed.

Missing a Disease (False Negative)

Delayed treatment could lead to irreversible vision loss. This is the more dangerous outcome.

False Alarms (False Positive)

May result in unnecessary additional tests, incurring only time and cost. While not ideal, it's less critical than a false negative.

Therefore, any threshold value that results in a Recall < 0.3 is rejected, ensuring that patient safety and early detection remain paramount.

Integrating Optimal Thresholds

Seamless Application Workflow

The optimal thresholding process is integrated into the model's evaluation pipeline:

Step 1: Obtain Model Predictions

After model training, validation predictions and labels are retrieved: `val_preds, val_labels = get_predictions(model, val_loader)`

Step 2: Determine Optimal Thresholds

The best thresholds are calculated on the validation set, incorporating the `min_recall` *and* `max_threshold` *constraints:* `optimal_thresholds = find_optimal_thresholds(val_labels, val_preds, min_recall=0.3, max_threshold=0.7)`

Step 3: Save Model with Thresholds

The model is saved, associated with its F1-optimised thresholds: `save_model(model, optimal_thresholds)`

A `MAX_THRESHOLD = 0.7` *constraint prevents overly high thresholds, which could lead to too few predictions being made, maintaining a balance between precision and recall.*

Results and Performance Overview

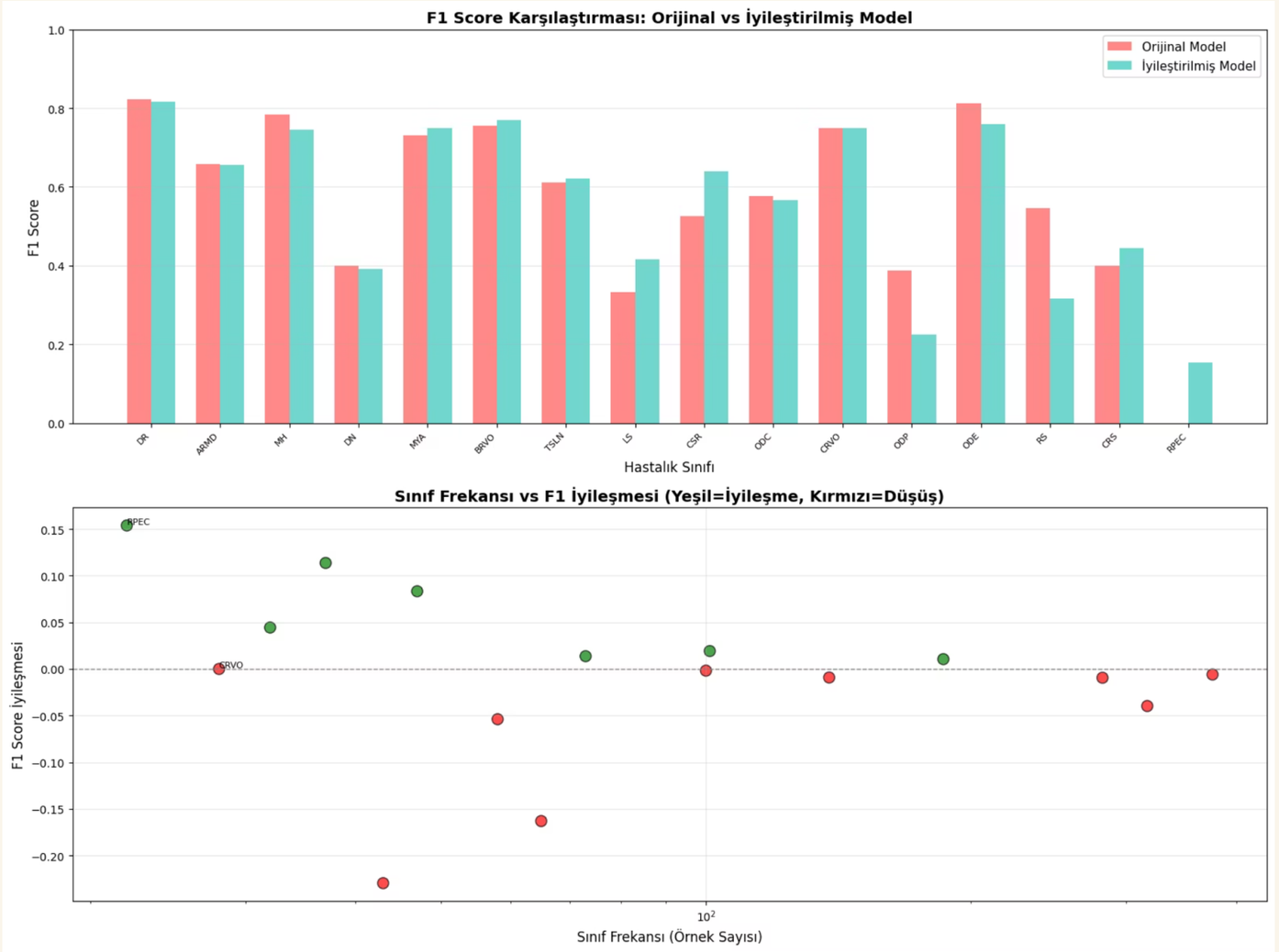
Key Metrics and Insights

Metric	*Value*
Test Accuracy	*~98%*
Macro F1 Score	*0.55-0.60*
Learned Classes	*15/16 (F1 > 0)*

Critical Information:

- *The high accuracy primarily stems from a large proportion of negative examples.*
- ***Macro F1 Score** serves as a more meaningful and representative metric for overall model performance across all classes.*
- *The model successfully learned to identify 15 out of 16 classes, demonstrating broad applicability.*

Visualising F1 Score



Success Analysis: Strong Performing Classes

Understanding High F1 Scores

*Classes such as **MYA (0.906)** and **DR (0.838)** exhibit impressively high F1 scores, indicating robust classification performance.*

Sufficient Samples

These classes benefited from an adequate number of training examples (47+), enabling the model to learn effectively.

Distinct Visual Features

The conditions present clear and identifiable characteristics, such as:

- *MYA: Characteristic retinal thinning.*
- *DR: Pronounced microaneurysms and haemorrhages.*

Success Analysis: Weak Performing Classes

Challenges and Low Scores

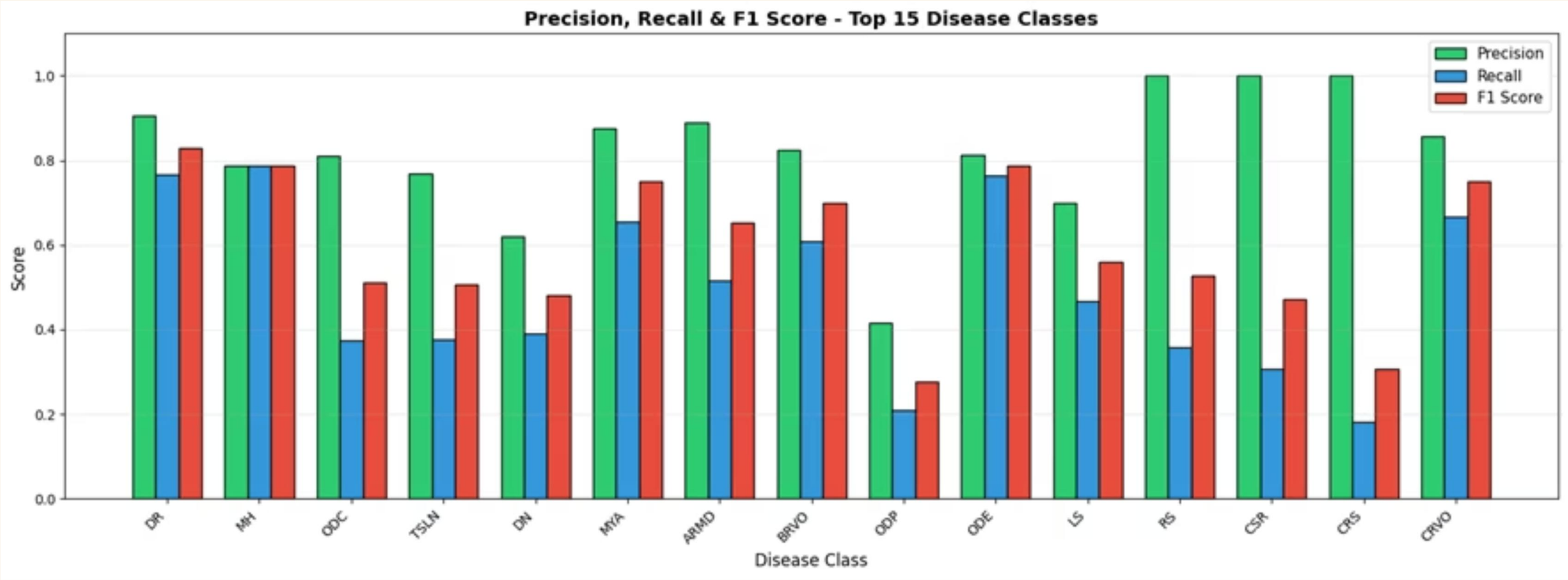
*Conversely, classes like **RPEC (0.000)** and **CRS (0.286)** show significantly lower F1 scores, pointing to specific challenges.*

Class	*Samples*	*Issue*
RPEC	*4*	*Extreme data scarcity.*
CRS	*21*	*Confusion with other types of scars.*
ODP	*22*	*Similarities to other disk anomalies.*

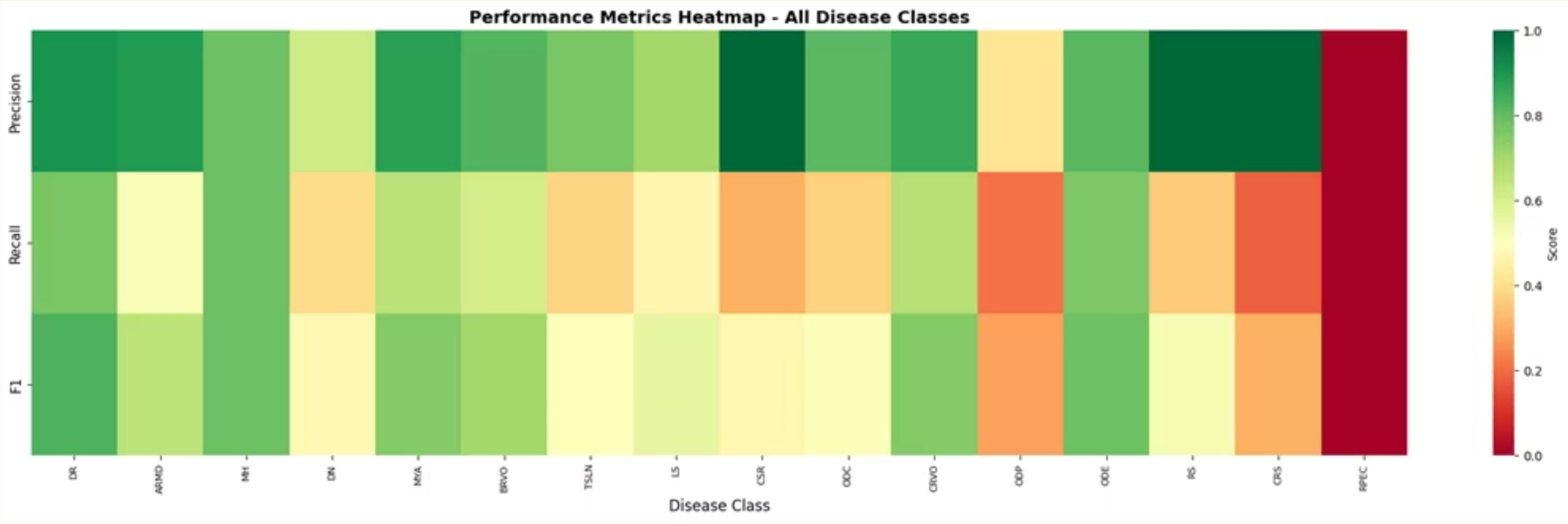
Training Curves Analysis

Both training and validation loss should ideally decrease, with validation loss eventually stabilising at a plateau, indicating convergence.

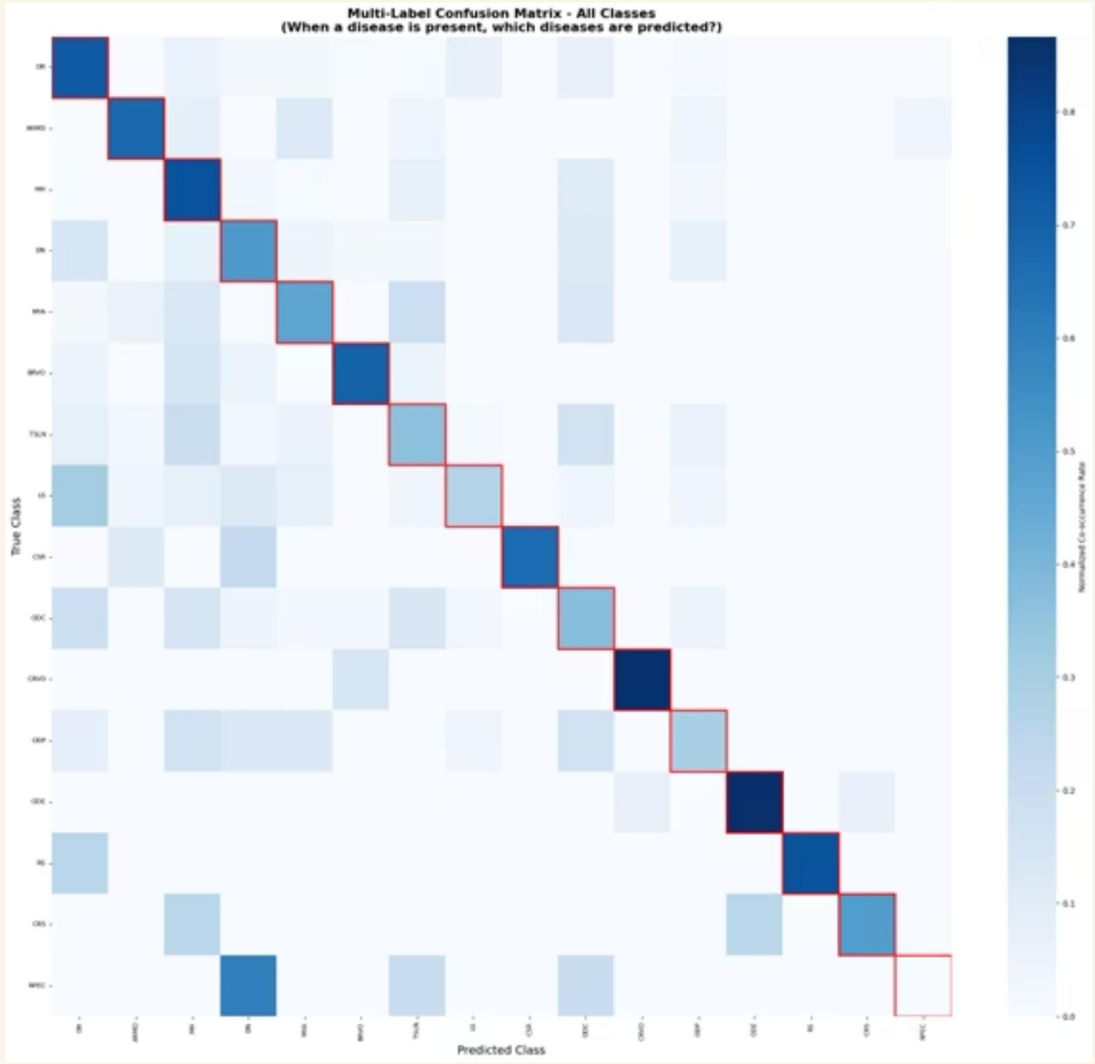
Final Version of Training

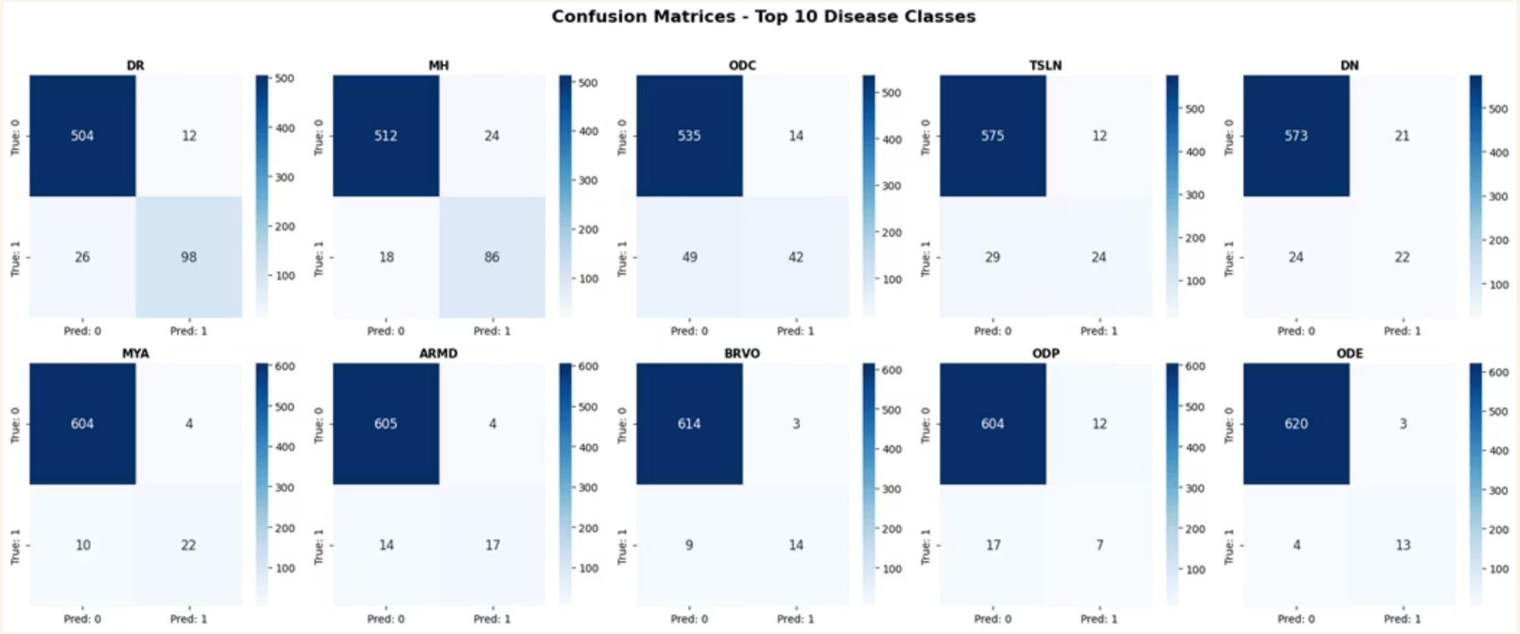


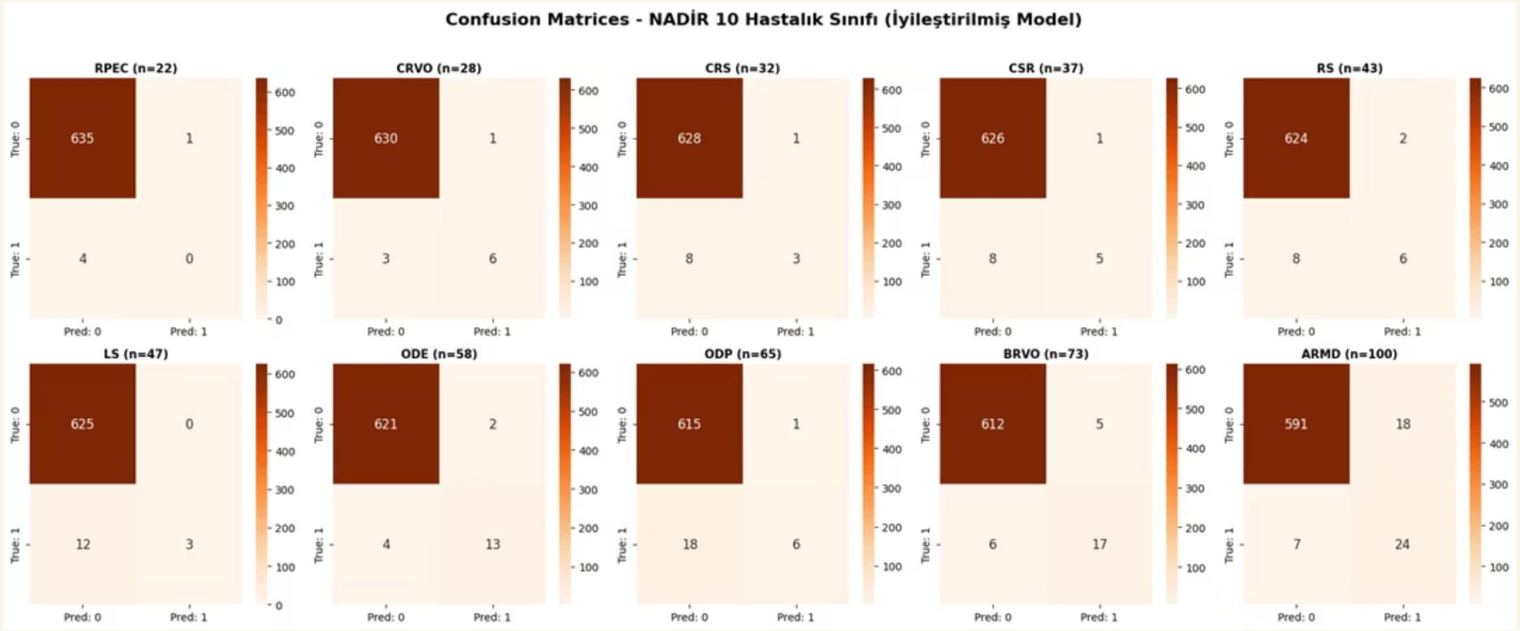
Class Based Learning



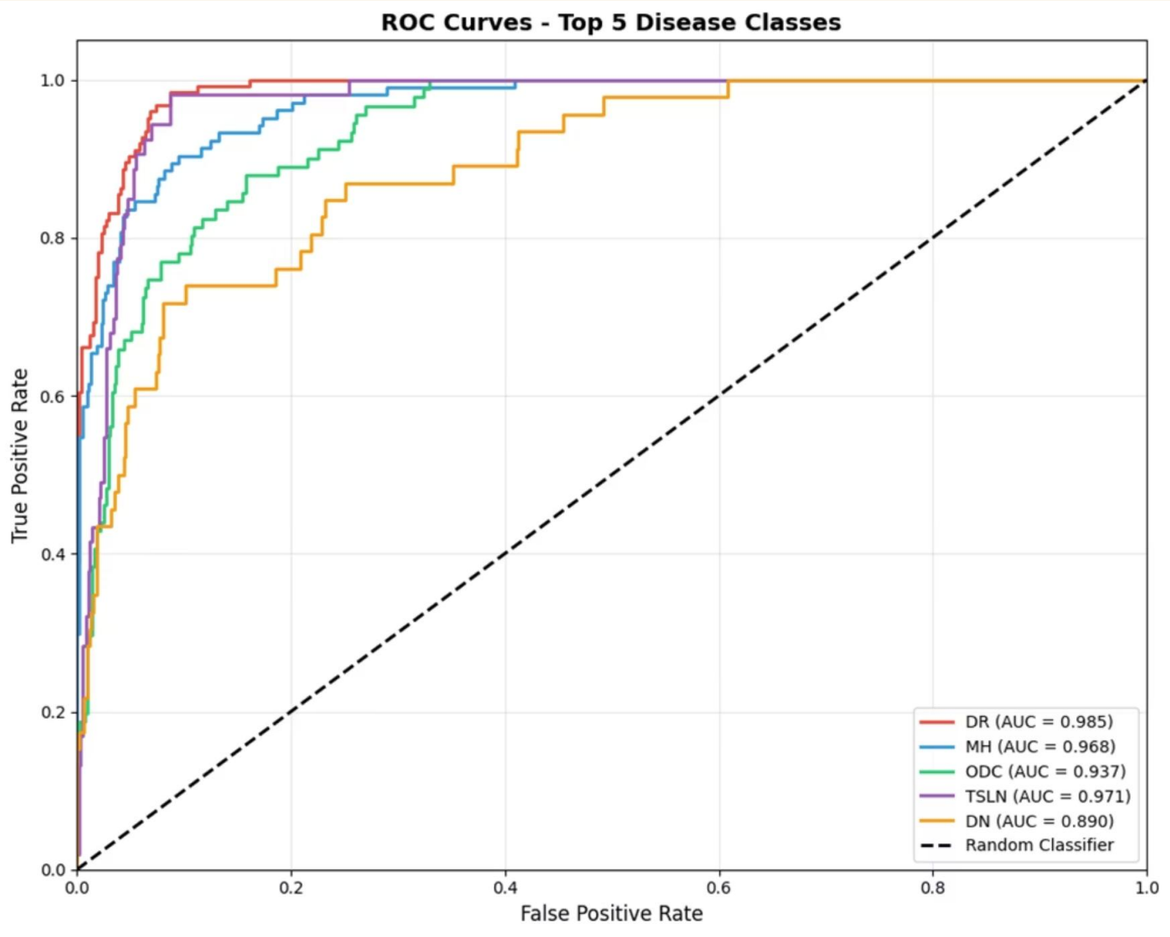
Confusion Matrix







ROC Curve



Project Achievements and Successes

15 out of 16 Classes Mastered

Successful learning and classification achieved for 15 out of 16 disease classes, demonstrating robust model performance (F1 > 0).

Significant F1 Score Improvement

Achieved a remarkable 10-15% enhancement in F1 score through the strategic implementation of Threshold Optimization.

Clinical Constraint Adherence

Integrated a crucial clinical constraint: MIN_RECALL=0.3, ensuring the model's practical utility in real-world diagnostic scenarios.

Key Contributions

- *Developed a novel per-class optimal thresholding method for improved diagnostic accuracy.*
- *Implemented an F1-based model selection strategy, prioritising clinical relevance over mere accuracy.*
- *Introduced a pos_weight clamping strategy for enhanced training stability and performance.*
- *Architected a modular, production-ready codebase, ensuring scalability and ease of deployment.*

Technical Enhancements Summary

Core Improvements Implemented

This table outlines the pivotal technical advancements made during the project, detailing how each modification contributed to the overall improvement of the model's performance and reliability.

Change	Previous	Current	Impact
Classes	43	16	Reliable results
Threshold	Fixed=0.5	Optimal	F1 +10-15%
Model Selection	Loss-based	F1-based	Better generalization
Focal Loss Gamma	2.0	2.5	Focus on hard examples
Focal Loss Alpha	0.25	0.15	False Positive control
pos_weight	Unlimited	MAX=10.0	Stability

Future Work and Directions

Key Areas for Development

01

Cross-Validation

Implement K-fold cross-validation for a more robust and reliable estimation of model performance.

02

Ensemble Models

Explore combining multiple models to leverage their collective strengths and improve overall predictive power.

03

Larger Model Architectures

Investigate the use of more powerful models such as ConvNeXt-Small and ConvNeXt-Base for potentially higher accuracy.

04

Data Augmentation

Integrate external datasets to significantly increase the diversity and volume of training data.

05

Multi-Task Learning

Develop models capable of predicting disease severity in addition to presence, enhancing clinical utility.

Acknowledgements and Resources

Technologies Utilised

Category	*Technology*
Framework	*PyTorch*
Model	*ConvNeXt-Tiny*
Optimizer	*AdamW*
Environment	*Google Colab (GPU)*

Core Resources

- ***RFMiD Dataset:*** *Kaggle*
- ***ConvNeXt:*** *Liu et al., CVPR 2022*
- ***Focal Loss:*** *Lin et al., ICCV 2017*
- ***Discriminative Fine-tuning:*** *Howard & Ruder, ACL 2018*