Infyma Hackathon Report

Team: We3bears
Members: Moawiz, Areeba, and Sarim

March 10, 2025

Abstract

This report details the approach and experimental results of our project for the Infyma Hackathon. We developed a multi-model classification system with rigorous training, logging, and inference pipelines. Two distinct models were trained with individual hyperparameter tuning, and their performances were logged using Weights & Biases. An ensemble of these models was also evaluated. For inference, we built an interactive Gradio interface that displays classification scores, predicted classes, and Grad-CAM visualizations.

1 Introduction

The objective of our project is to develop a robust classification system that leverages multiple models and ensemble learning for Infyma Hackathon. Detailed guidelines can be found on the project GitHub page (`https://github.com/ZayanRashid295/Infyma/`).

2 Dataset Overview

Our dataset consists of multiple image classes, used for training and evaluating our models. Below are some key visualizations of the dataset:

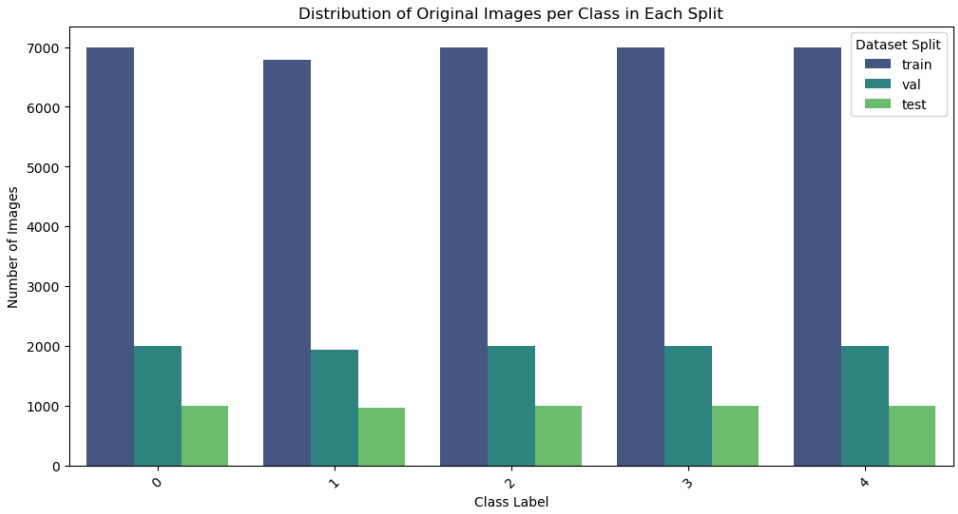

Figure 1: Sample Images from Different Classes in the Dataset

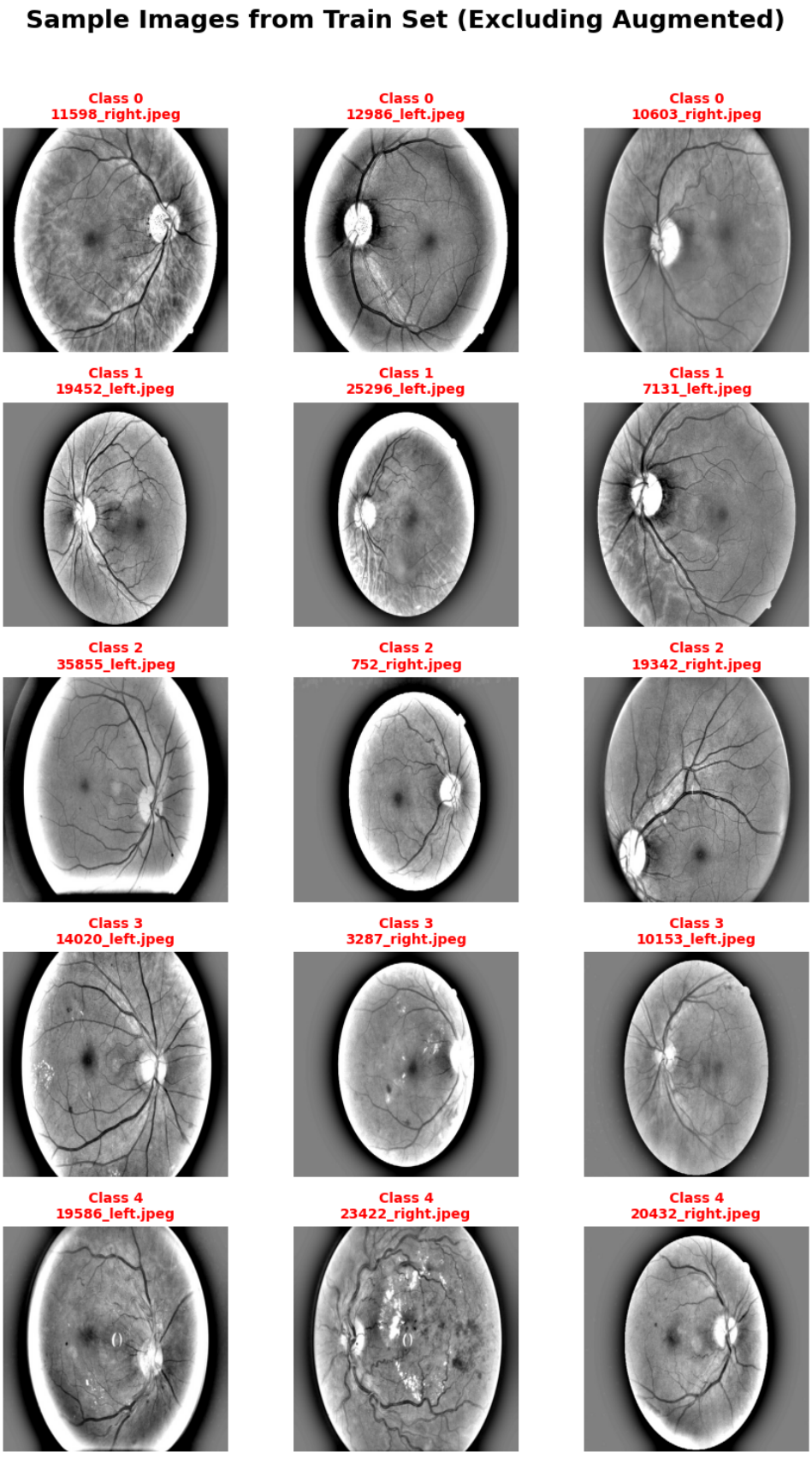

Figure 2: Class Distribution across Train, Validation, and Test Sets

3 Methodology

3.1 Model Training

We trained two separate models with independent hyperparameter tuning. Our training pipeline incorporates the following key components:

- **Metric Logging:** We used wandb to log training and validation metrics such as loss, accuracy, precision, and F1-score.
- **Model Checkpointing and Early Stopping:** The best performing model was saved based on validation performance with early stopping implemented to avoid overfitting.
- **Regularization:** To fulfill the evaluation criteria, our classification layer includes dropout and batch normalization.
- **Normalization:** Input images were preprocessed using normalization transformations.
- **Ensemble Metrics:** Performance metrics for the ensemble model were logged, providing additional insights.

Additionally, we trained a standalone Vision Transformer (ViT) and a standalone DenseNet201 model. These models can also be used in the inference interface. Ultimately, our best-performing model was the ensemble model.

3.2 Hyperparameter Tuning

We experimented with different hyperparameter configurations during training, as shown in Table 1. After tuning, our best model had a **learning rate** of **1e-4** and a **dropout rate** of **0.5**.

Hyperparameter	Values
Learning Rate	{1e-3, 1e-4, 1e-5}
Dropout Rate	{0.3, 0.5}

Table 1: Hyperparameter Tuning Configurations

3.3 Model Architecture

The following diagram illustrates the architecture of our best-performing ensemble model:

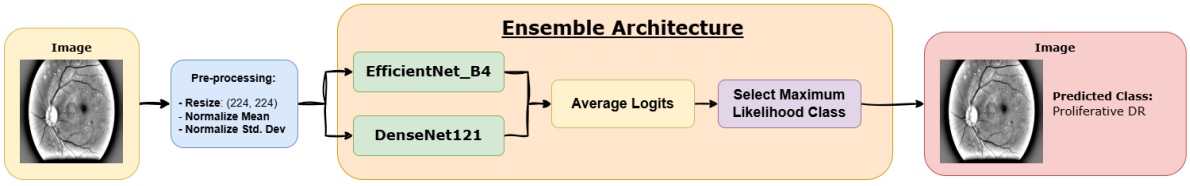

Figure 3: Model Architecture Diagram

3.4 Inference Pipeline

For the inference stage, we built a Gradio interface that offers:

- The ability for users to upload their own images.
- Display of classification scores and the predicted class.
- **Grad-CAM** visualization for model interpretability.
- **Real-time inference speed measurement** to provide users with an estimate of the model's processing time per image.

4 Results

4.1 Evaluation Metrics

During training, several key visualizations were recorded for our best ensemble model:

- Confusion matrices displaying the performance of the ensemble model on train, validation, and test sets.
- Graphs showing training/validation loss, accuracy, and F1 trends of each model present in the ensemble system.

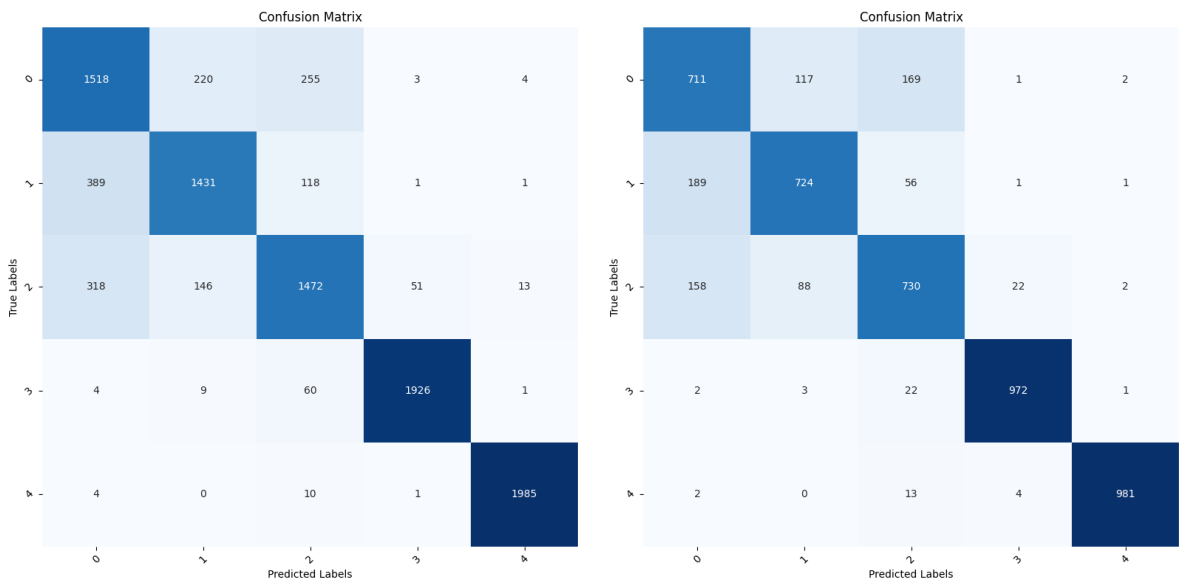

Figure 4: Confusion Matrices for Ensemble Validation and Test Sets

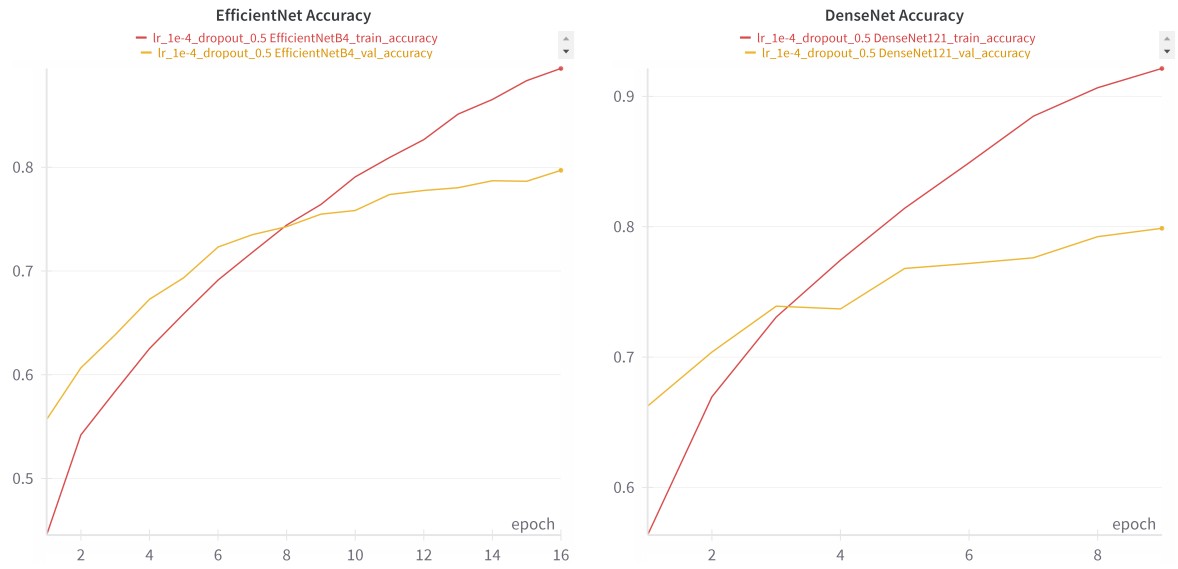

Figure 5: Accuracy trend for Train and Validation on EfficientNet and DenseNet (Early Stopping on DenseNet)

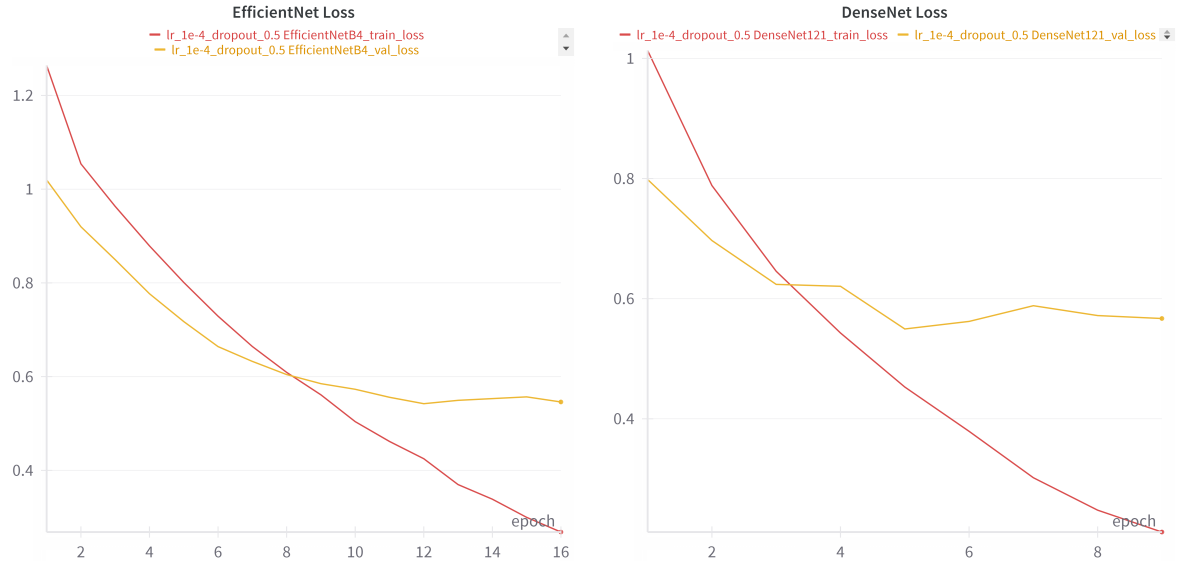

Figure 6: Loss for Train and Validation on EfficientNet and DenseNet (Early Stopping on DenseNet)

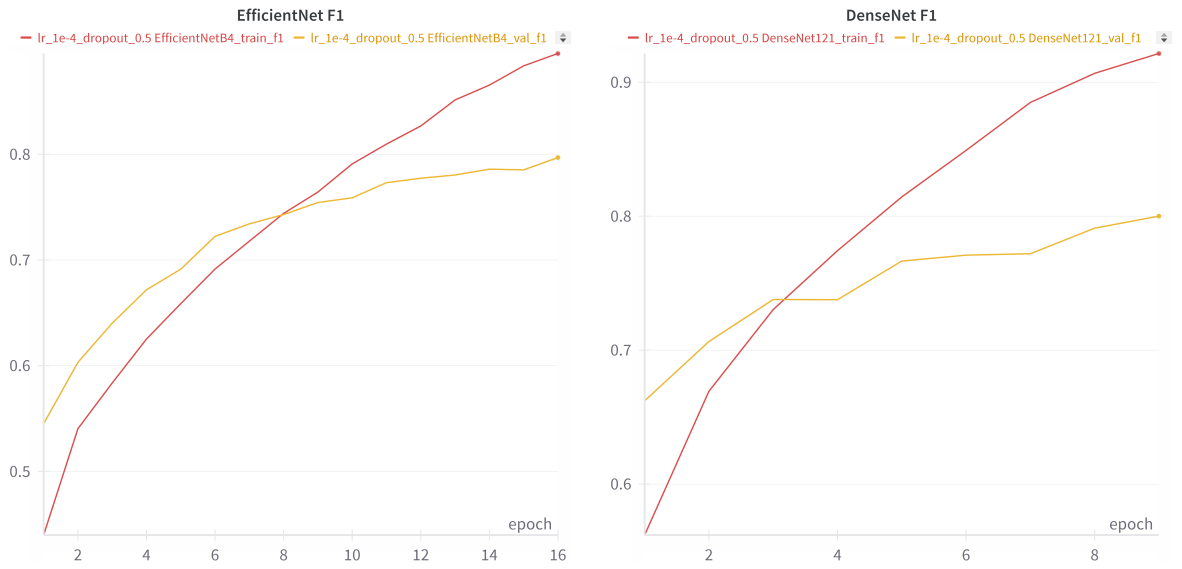

Figure 7: F1 Score trend for Train and Validation on EfficientNet and DenseNet (Early Stopping on DenseNet)

	precision	recall	f1-score	support
No DR (Healthy)	0.96	0.99	0.98	7000.0
Mild DR	0.99	0.97	0.98	6792.0
Moderate DR	0.99	0.99	0.99	7000.0
Severe DR	1.0	1.0	1.0	7000.0
Proliferative DR	1.0	1.0	1.0	7000.0
accuracy	0.99	0.99	0.99	0.99
macro avg	0.99	0.99	0.99	34792.0
weighted avg	0.99	0.99	0.99	34792.0

Figure 8: Classification Report for Ensemble Model on **Training Set**

	precision	recall	f1-score	support
No DR (Healthy)	0.68	0.76	0.72	2000.0
Mild DR	0.79	0.74	0.76	1940.0
Moderate DR	0.77	0.74	0.75	2000.0
Severe DR	0.97	0.96	0.97	2000.0
Proliferative DR	0.99	0.99	0.99	2000.0
accuracy	0.84	0.84	0.84	0.84
macro avg	0.84	0.84	0.84	9940.0
weighted avg	0.84	0.84	0.84	9940.0

Figure 9: Classification Report for Ensemble Model on **Validation Set**

	precision	recall	f1-score	support
No DR (Healthy)	0.67	0.71	0.69	1000.0
Mild DR	0.78	0.75	0.76	971.0
Moderate DR	0.74	0.73	0.73	1000.0
Severe DR	0.97	0.97	0.97	1000.0
Proliferative DR	0.99	0.98	0.99	1000.0
accuracy	0.83	0.83	0.83	0.83
macro avg	0.83	0.83	0.83	4971.0
weighted avg	0.83	0.83	0.83	4971.0

Figure 10: Classification Report for Ensemble Model on **Test Set**

4.2 Model Comparison

We also compared our ensemble model performance against our other individual standalone models (ViT and DenseNet201). The ensemble model outperformed both models in terms of **accuracy, precision, F1-score, and recall**, demonstrating the advantages of combining multiple models. However, **DenseNet201 achieved the fastest inference speed**, making it more suitable for real-time applications requiring lower latency.

Model	Accuracy	F1 Score	Precision	Recall	Inference Speed (ms)
ViT	0.742	0.742	0.745	0.742	3900
DenseNet201	0.755	0.753	0.752	0.755	**2500**
Ensemble	**0.828**	**0.829**	**0.830**	**0.828**	2900

Table 2: Comparison of ViT, DenseNet201, and Ensemble Model Performance

4.3 Inference Results

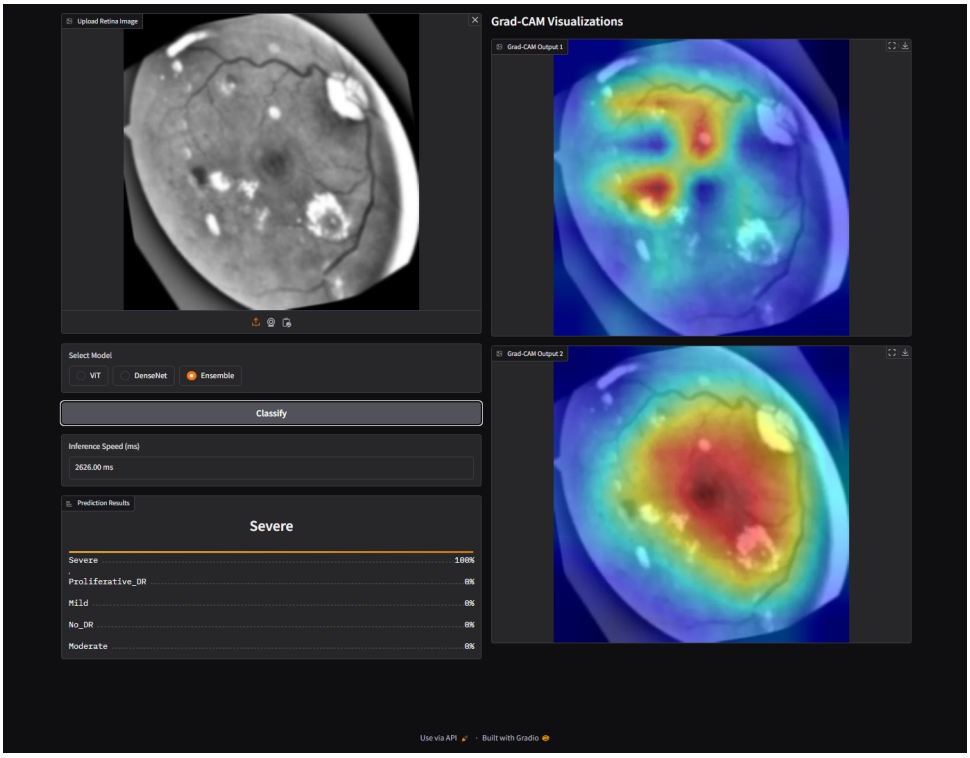

Figure 11: Gradio Inference Interface

The results confirm that our ensemble approach provides **superior classification performance**, making it the best choice when accuracy is the primary concern. However, DenseNet201 remains a strong option for applications where inference speed is a priority.

5 Conclusion

Our project successfully combines multiple deep-learning models, extensive metric logging, and an interactive inference interface. The use of dropout, batch normalization, early stopping, and normalization transformations contributed to robust model performance. Ultimately, our best-performing model was the ensemble model. Thank you for your time in reading this :).