Computer Vision

Eye Disease Detection Using Self-Supervised Learning and CNNs

January 2025

Professor: Marco Raoul Marini

Members:

Name: Arman
Surname: Feili
Matricola: 2101835
Email:
feili.2101835@studenti.uniroma1.it

Name: Andrea Melissa
Surname: Almeida Ortega
Matricola: 2179438
Email:
almeidaortega.2179438@studenti.uniroma1.it

Name: Milad
Surname: Torabi
Matricola: 2103454
Email:
torabi.2103454@studenti.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Papers

Eye Disease Classification Using Deep Learning Techniques (link)

Authors:
- Tareq Babaqi, Manar Jaradat, Ayse Erdem Yildirim, Saif H. Al-Nimer, and Daehan Won.

Institution:
- State University of New York at Binghamton.

What They Did:
- Used the same dataset as ours, containing 4200 images resized to 224x224 pixels.
- Trained two CNN models:
  - One CNN model from scratch.
  - One pre-trained EfficientNet model fine-tuned using transfer learning.

Results:
- CNN Model: Achieved **84% accuracy**.
- Transfer Learning (**EfficientNet**): Achieved **94% accuracy**, significantly outperforming the CNN model.
- CNN: Achieved 100% F1-Score, perfectly classifying this disease.

Papers

Why we chose this paper and what are we trying to improve?

- Since this paper uses only **one CNN from scratch** and **one pre-trained EfficientNet model**,

  we aimed to take it a step further by:

  - applying **three pre-trained CNN models** to the dataset
  - Plus two SSL models (self-supervised learning).

  to explore these advanced techniques for learning features from eye images.

Table 1: Evaluation metrics for the used classification algorithms

Model	Class	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
CNN	cataract	87	96	91	84
	Diabetic retinopathy	100	100	100	
	Glaucoma	67	85	75	
	Normal	86	56	68	
Transfer Learning	cataract	97	96	96	94
	Diabetic retinopathy	100	99	99	
	Glaucoma	94	85	89	
	Normal	86	96	91	

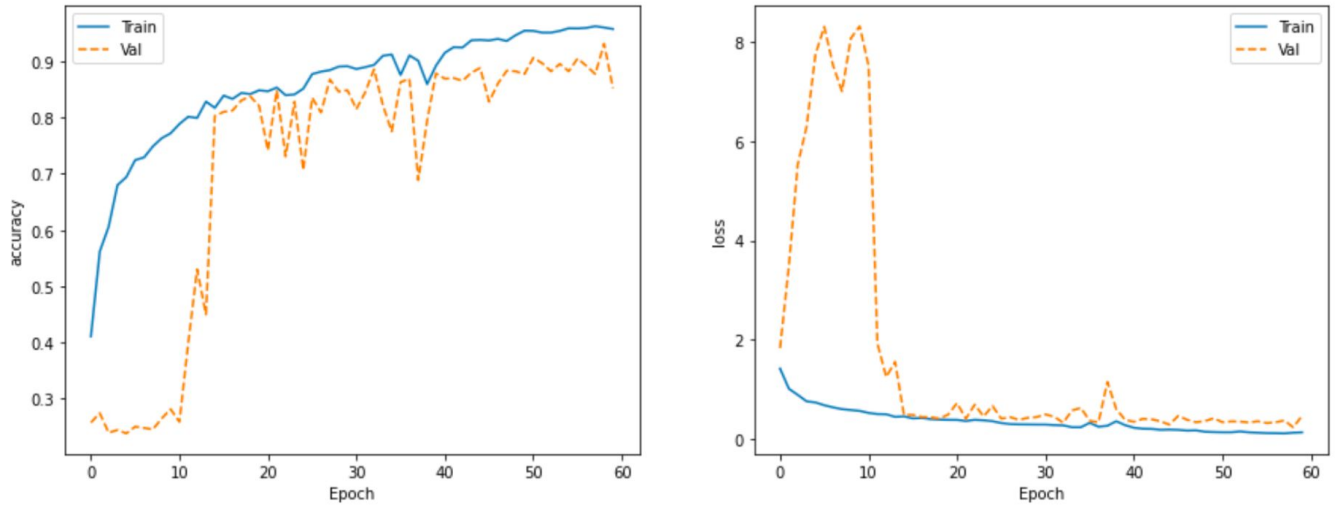


Figure 3: Training accuracy and loss for CNN

Introduction

What is Eye Disease Detection?

- A process that uses medical imaging to **classify eye-related conditions**.
- We can analyze retinal images to **detect diseases** like **diabetic retinopathy, glaucoma, cataracts,** and **macular degeneration**.
- Modern techniques include **deep learning models** that **automate** and improve **detection** accuracy.

Why is it Important?

- **Early detection** can **prevent vision loss** and improve quality of life.
- Helps in **treatment of** potentially **blinding diseases**.
- Automating detection **reduces reliance on manual diagnosis**, saving time and resources.
- **Makes healthcare more accessible**, especially in regions with limited medical expertise.

Right

Left

/eye_diseases_classification/Normal/2389_right.jpg 2389_left

/eye_diseases_classification/diabetic_retinopathy/100_right.jpg 100_left.jpg

Goal of the project

Question:

- How can SSL models be used over CNN models in eye disease detection to make use of unlabeled data and improve adaptability for analyzing diverse eye images?

Approach:

- We prepared the environment and **preprocessed** the dataset, resizing images, splitting data, and applying augmentations to improve robustness.
- We combined original, mildly **augmented**, and heavily augmented datasets into one diverse and unified dataset.
- We **trained** two self-supervised learning models, **DINO** and **SimCLR**, to extract meaningful feature representations from the data.
- We **fine-tuned three pre-trained CNN models** (**ResNet18, EfficientNet-B0,** and **DenseNet121**) using a **two-stage** training process: training the classifier first and then fine-tuning deeper layers.

This approach combined self-supervised learning with fine-tuned CNNs to tackle the challenge of accurate eye disease detection effectively.

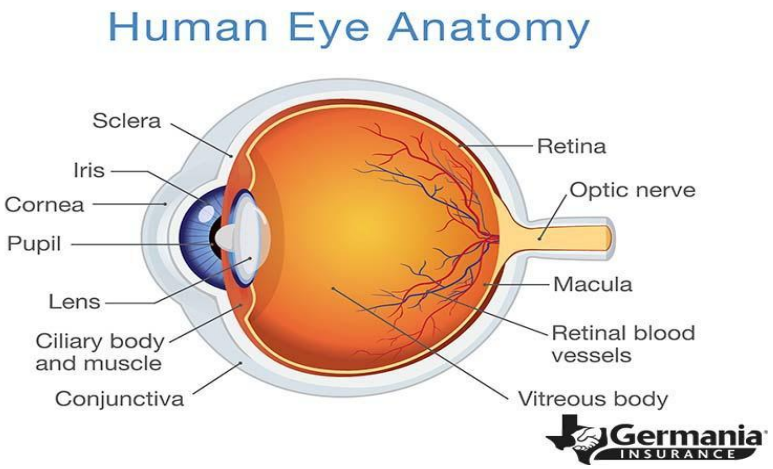
/eye_diseases_classification/glaucoma/_55_10273

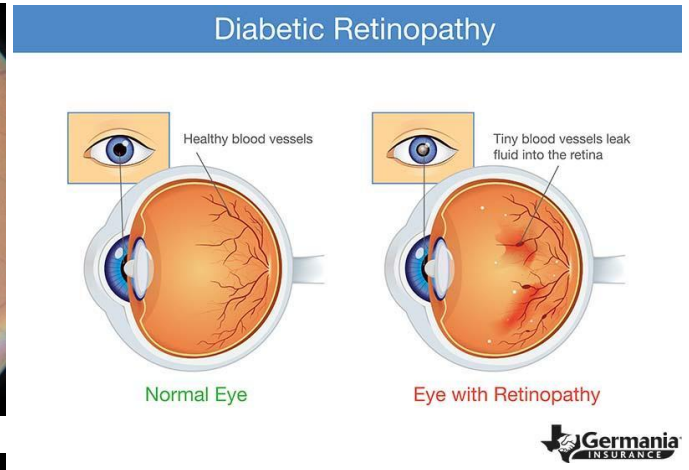
/eye_diseases_classification/cataract/_195_1750450.jpg

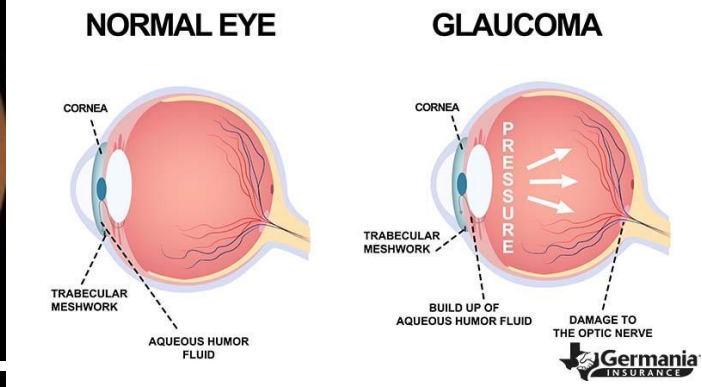
Dataset Overview:

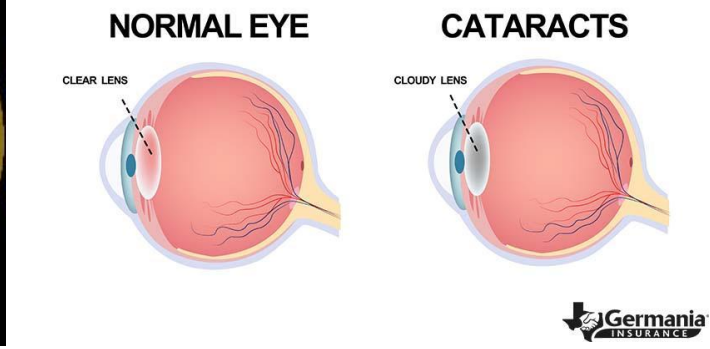
The project uses a Kaggle dataset with approximately **1000 images per class**, divided into four categories:

- **Normal Eyes:** Images with **no disease**, serving as a baseline control group.
- **Cataract:** Characterized by a **hazy appearance** in the images, indicating clouding of the lens that leads to **blurred vision**.
- **Glaucoma:** Features an enlarged optic nerve cup, representing **damage** often **caused by high eye pressure**.
- **Diabetic Retinopathy:** Shows signs like microaneurysms or hemorrhages, resulting from **retinal damage due to diabetes**.



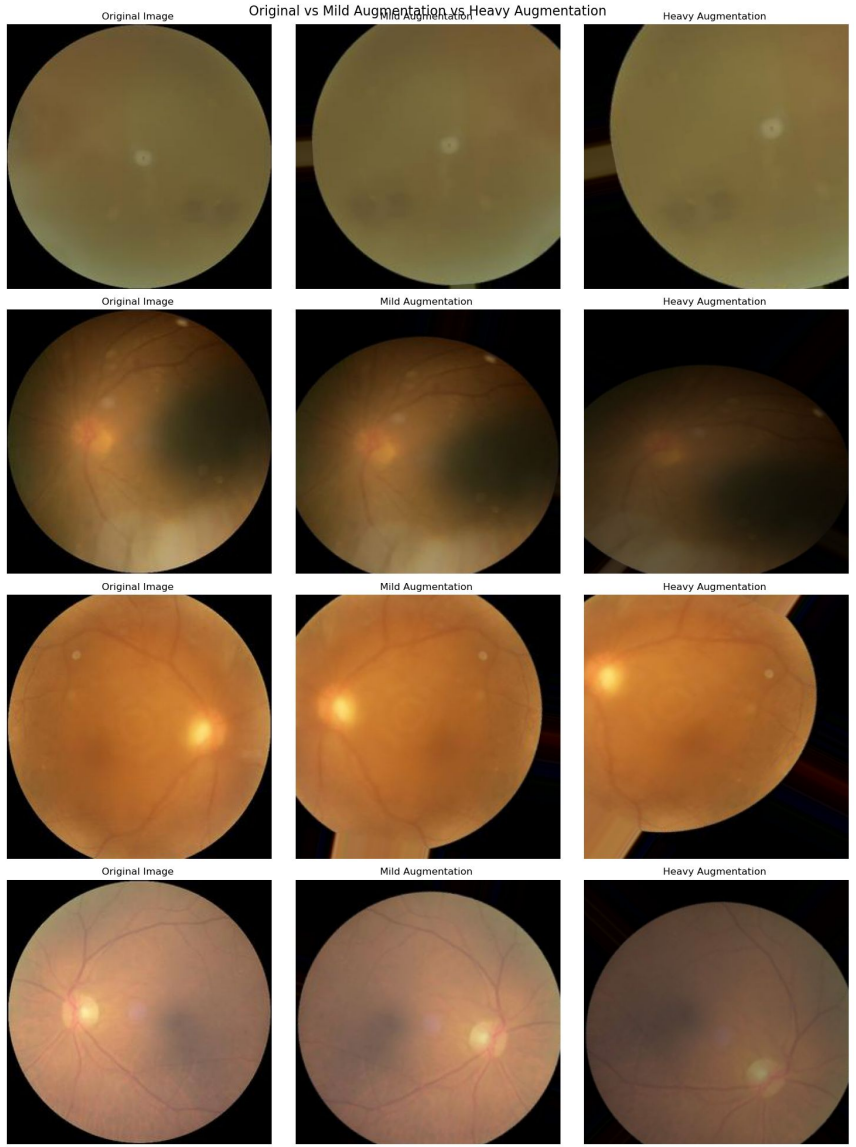






Preprocessing Data

- Set up the project environment by **installing necessary libraries** for data processing, visualization, and machine learning, including PyTorch and torchvision.
- Define and **organize directory** structures for datasets, outputs, and preprocessed data.
- **Manage metadata** to ensure an efficient, repeatable process for tracking dataset details, augmentations, and splits.
- **Resize all images to** a consistent size of **256x256** pixels for uniformity.
- Split the dataset into **training (80%)** and **validation (20%)** subsets, ensuring a balanced distribution of classes.
- **Apply data augmentations**, creating mildly and heavily transformed versions of the images to improve the model's ability to generalize.
- **Validate image dimensions** for **consistency** across the dataset.
- **Visualize the original and augmented images side-by-side** to confirm that the transformations are meaningful and diverse.
- **Save augmentation details** and dataset splits in metadata to avoid redundant processing.
- **Combine original, mildly augmented, and heavily augmented** images into a single unified dataset containing **10116 images** across four classes, ready for training.



Self-Supervised Learning (SSL)

What is SSL?

- SSL means Self-Supervised Learning, a way for machines to learn patterns from data **without needing labels**.
- It creates its own tasks, like **comparing different parts** of data, to teach itself **useful features**.
- These features can then **be used** for tasks like **classifying or predicting**.

Why Use SSL in This Project?

- SSL lets us use all the data, even without labels, making the most of our dataset.
- It helps create strong, useful features that make models more accurate.
- It **saves time** and effort for **avoiding to create a huge labeled database**.

Challenges with SSL We Had in This Project:

- Training SSL models like **DINO and SimCLR** needed a lot of **computer power** and **time**.
- Finding the right settings to make the models work well was tricky. (**fine tuning hyperparameters**)
- access to low computation power of Colab with **limitations on** using **GPU**.
- Understanding how each method worked and **choosing the best hyperparameters** took extra time.

Self-Supervised Learning (SSL)

SimCLR (Simple Contrastive Learning of Representations):

- A method that teaches itself by **comparing different versions of the same image**.
- It focuses on **making features** from **similar images closer** and **from different images further apart**.
- Pull representations of the same image closer while pushing representations of different images apart.

Why We Used SimCLR and How Does It Help Us?

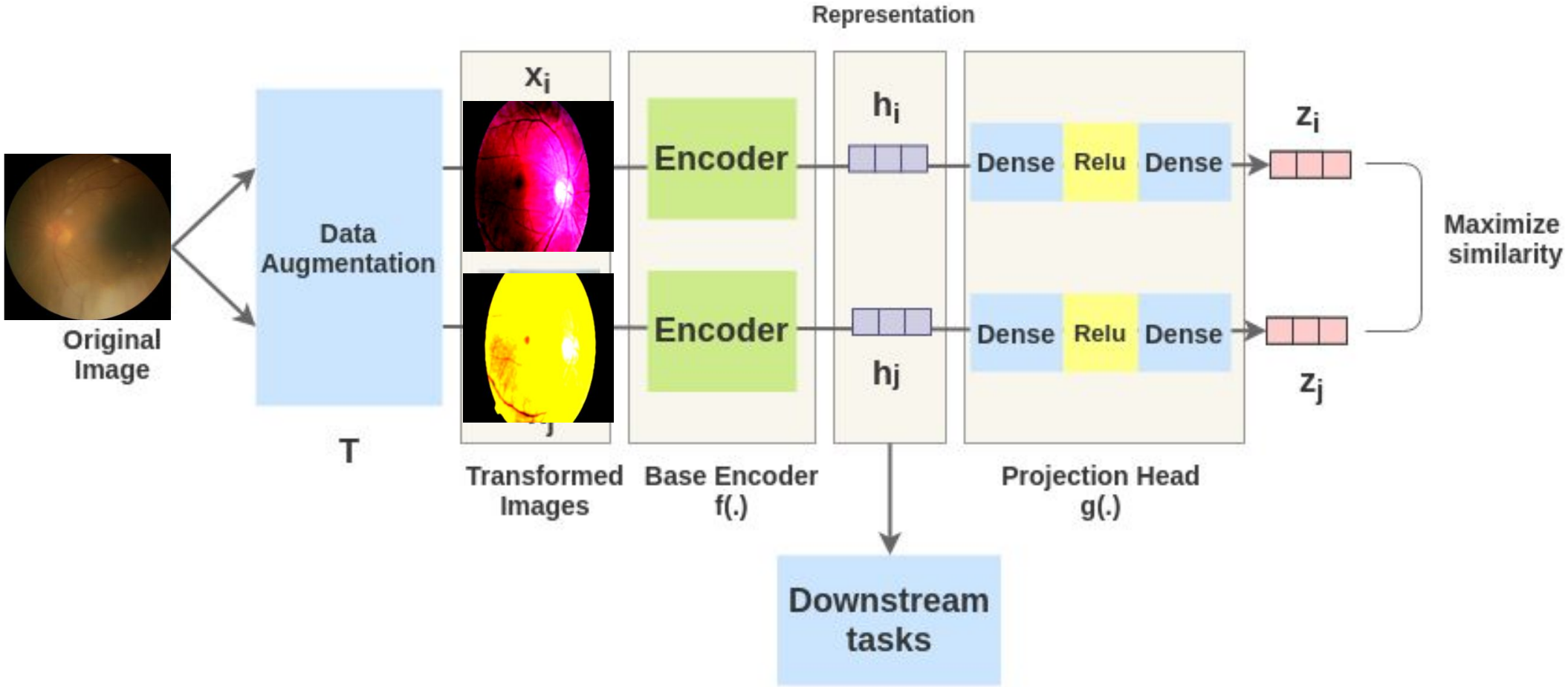
- SimCLR is simple and learns by **comparing different views of the same image**.
- It helps models **start with better features**, making them **perform better later**.

How SimCLR Fits in Our Workflow:

- SimCLR pre-trained a neural network encoder using our aggregated dataset (**10116 images** across four classes).
- The training process spanned **10 epochs**, with the training **loss decreasing** from **2.32** to **2.28**.
- Output: **A feature-rich encoder** that served as a foundation for downstream classification models.

Self-Supervised Learning (SSL)

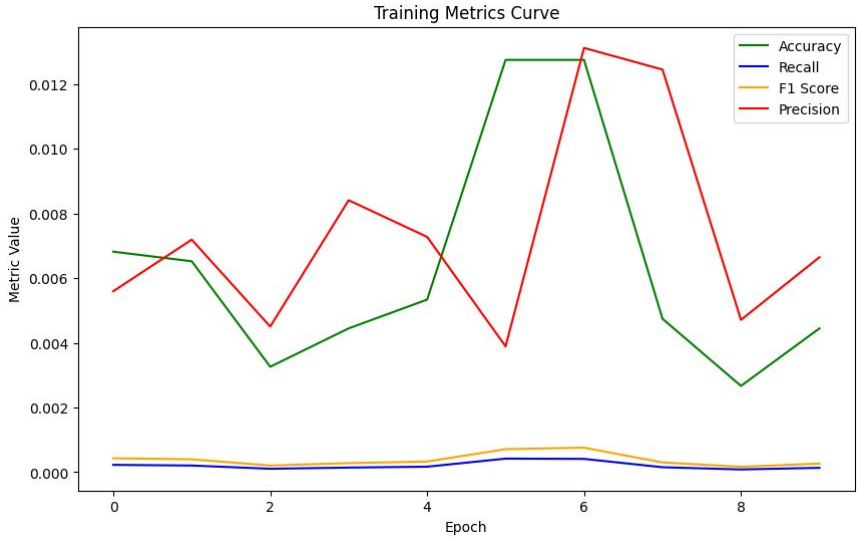
SimCLR Framework:

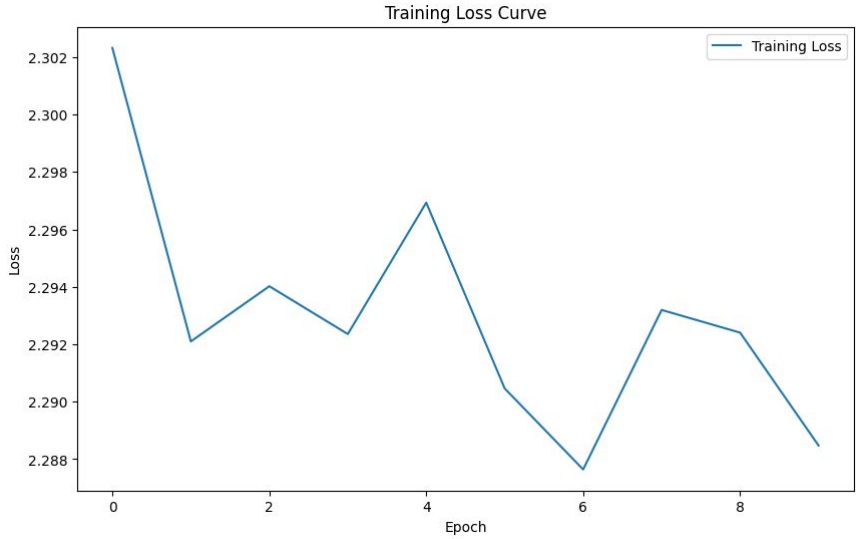


Self-Supervised Learning (SSL)

SimCLR Results:

- The SimCLR model was trained on around **10,116 eye images** for **10 epochs** using a l**earning rate of 0.001** and the **NT-Xent loss function** with a temperature of 0.5.
- The **Adam optimize**r was used with a small **weight** decay of **1e-6 to prevent overfitting** and improve regularization.
- **Training loss** steadily **decreased from 2.3183** in the first epoch to **2.2831** by the **tenth epoch**, indicating gradual learning.
- Training metrics such as **accuracy, recall, precision, and F1 score remained low**, as expected, since **SimCLR focuses on learning feature** representations **rather** than **direct classification**.
- The model consistently learned representations, but the small improvements in metrics reflect the nature of the pretraining stage, which does not yet optimize for final classification performance.





Self-Supervised Learning (SSL)

DINO (Distillation with No Labels):

- A method that **uses a special teacher-student system** to **learn features without needing labels**.
- It's great for **finding useful details** in complex images like eye scans.

Why We Used DINO and How Does It Help Us?

- DINO creates very good features for complex images using advanced techniques.
- It works well without needing labels and helps in improving the final model's performance.
- It **sets** a strong **base** for **fine-tuning with labeled data**.

How we used Dino:

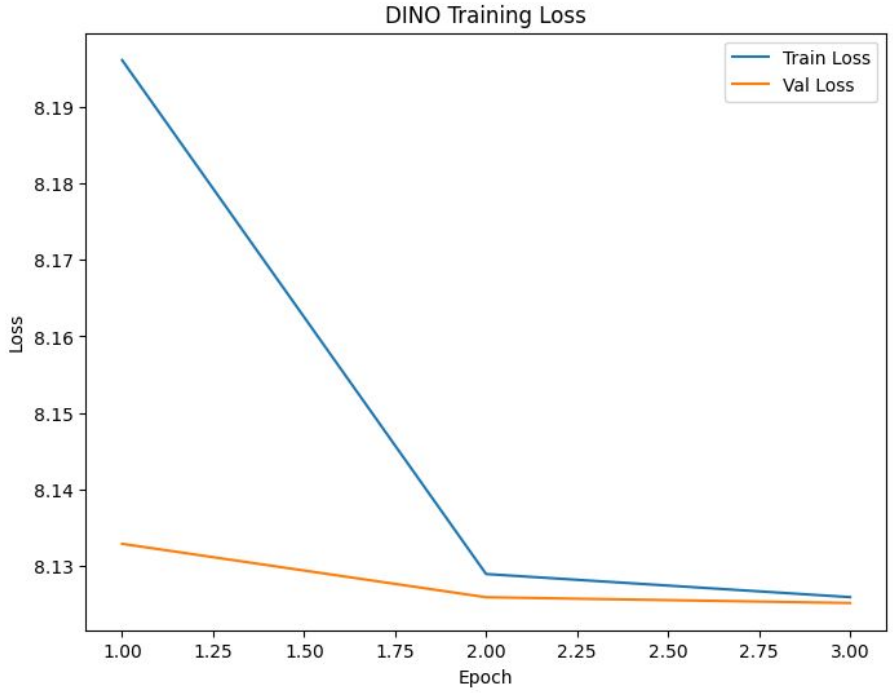
- We Trained the **self-supervised Vision Transformer (ViT) model** using **DINO** to **learn meaningful feature** representations from the dataset.
- Monitor the training progress by plotting training and validation losses **over three epochs**.
- **Adjust the learning rate using a cosine decay schedule, gradually reducing it to nearly zero by the final epoch** for optimal training.
- Evaluate the performance of the DINO model by observing a **steady reduction** in **both training and validation losses**, indicating effective learning.
- Use metrics like accuracy, precision, recall, and F1 score to measure the classifier's performance.

Self-Supervised Learning (SSL)

Dino Results:

- The DINO model was trained on **10,116 eye images** using a **batch size of 16** and an **image size of 256x256**.
- Training was run for **3 epochs** with a **learning rate starting at 0.0005 and gradually reduced to zero** using a cosine decay schedule.
- The learning rate was reduced to near zero by the final epoch, ensuring stability in the final stages of training.
- These results confirm that the **DINO model effectively learned meaningful feature** representations during training.

Epoch	Avg SSL Loss (Train)	Avg SSL Loss (Validation)	Learning Rate (Final)
1	8.1961	8.1329	0.0000422
2	8.1290	8.1259	0.0000152
3	8.1259	8.1252	0.0000000



Self-Supervised Learning (SSL)

Compare DINO Vs. SimCLR:

- **Loss Performance: DINO achieved more significant loss reduction in fewer epochs**, while SimCLR showed **slower** improvement but over a longer period.
- **Focus:** DINO's **ViT-based approach** excels at **high-quality feature learning**, while SimCLR focuses on contrastive learning through augmentations.
- **Efficiency: DINO required fewer epochs** and achieved comparable feature representation performance faster than SimCLR.
- **Metrics: SimCLR's training metrics** (accuracy, F1, precision, recall) **were very low,** reflecting its focus on pre-training rather than immediate classification performance, while DINO's performance isn't directly measured in classification but is better optimized for downstream tasks.

Area	DINO Model	SimCLR Model
Epochs	3	10
Batch size	16	16
Learning rate	0.0005 (cosine decay)	0.001 (constant)
Loss Performance	Loss reduced from 8.1961 to 8.1259 (train), 8.1329 to 8.1252 (val)	Loss reduced from 2.3183 to 2.2831 (train)
Focus	High-quality feature extraction using self-supervised ViT	Contrastive learning of representations via augmentations
Efficiency	**Fewer epochs, faster learning**	Slower, requires more epochs

Convolutional Neural Networks (CNNs)

What are CNNs?

- CNNs (Convolutional Neural Networks) are a **type of deep learning model** designed to **process visual data**, such as images. making them highly effective for image recognition and **classification tasks**.
- CNNs **use convolutional layers** to scan images for patterns and hierarchies, like shapes or textures, that help in identifying objects or features.

Why CNNs for Eye Disease Detection?

- CNNs can **automatically detect features** that indicate eye diseases.
- They are **highly accurate for medical image classification** due to their ability to focus on **relevant features** while ignoring unnecessary details.
- CNNs can process large **image datasets efficiently**, making them suitable for handling thousands of eye images.

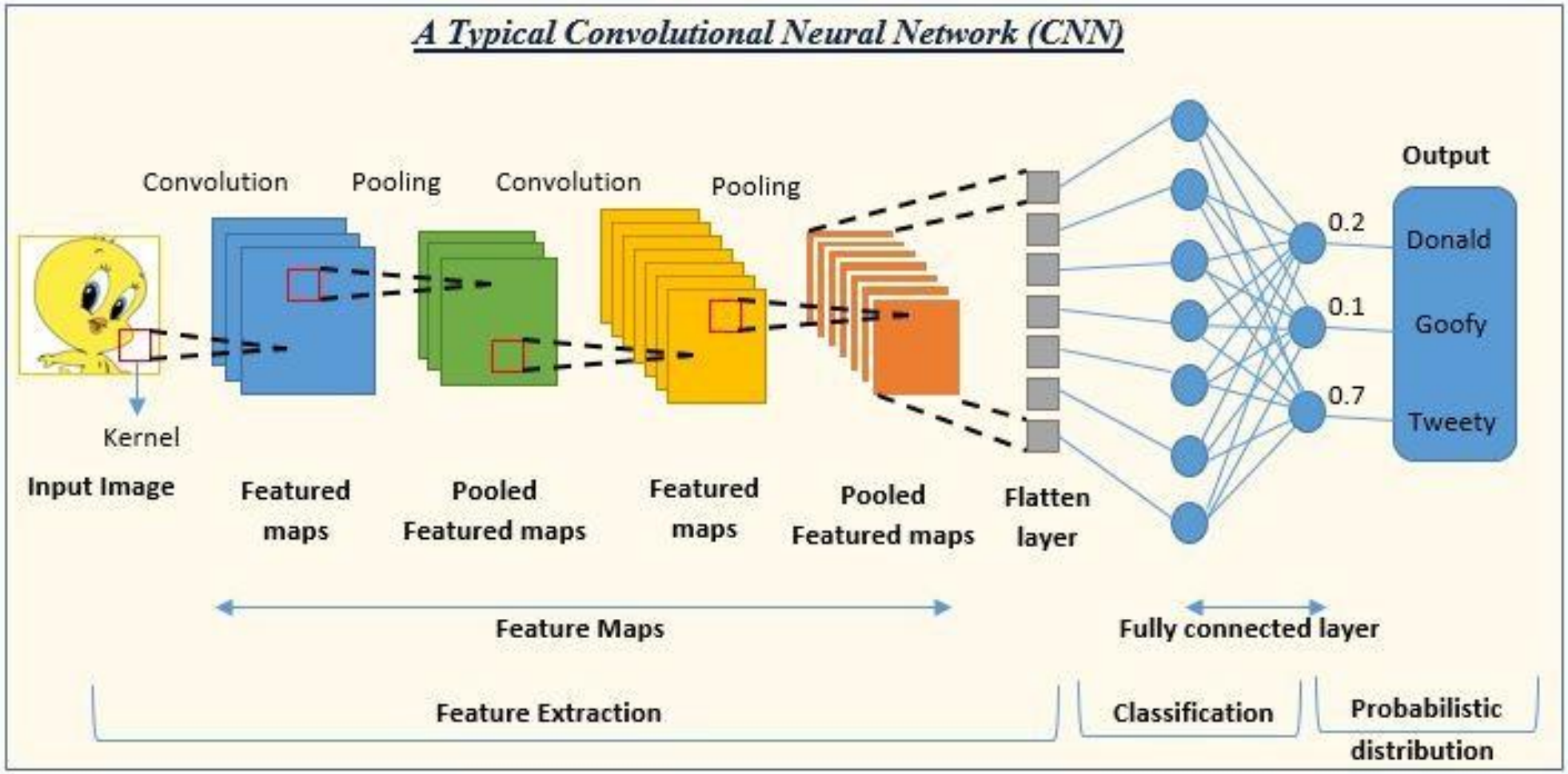
CNN Architecture Basics:

- **Convolutional Layer: Detects features** in the input image, such as edges or textures, **by applying filters**.
- **Pooling Layer: Reduces the size of feature** maps, retaining essential information while making computations faster.
- **Fully Connected Layer: Combines the extracted features** to classify the image into categories.
- **Activation Functions: Apply non-linear transformations** to help the model learn complex patterns.

The architecture processes raw image pixels into features step-by-step, gradually moving from simple to complex representations for final classification.

Convolutional Neural Networks (CNNs)

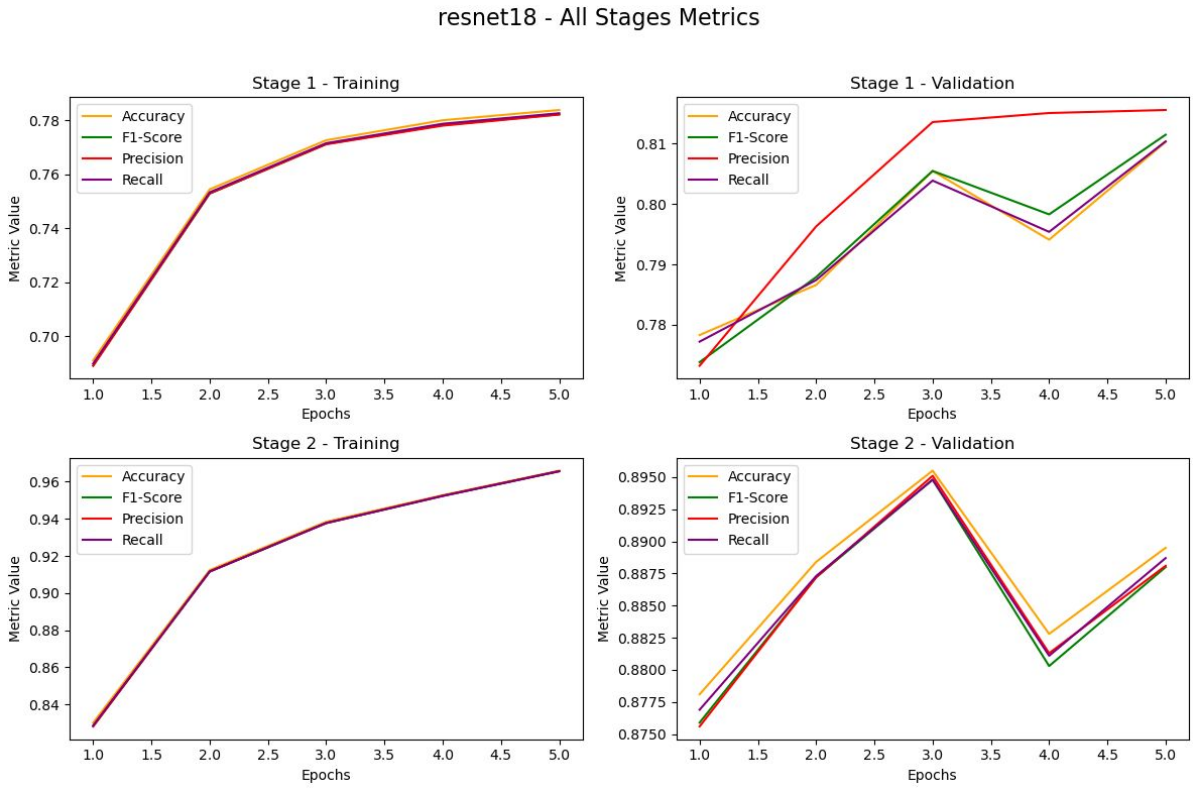
CNN Diagram:



Convolutional Neural Networks (CNNs)

ResNet18 Model:

- **ResNet18** is a convolutional neural network with **18 layers**, designed for efficient feature learning using **residual connections to avoid vanishing gradient issues**.
- It was fine-tuned in our project in two stages:
  - **Stage 1:** Early layers were frozen, and only the final classifier was trained.
    - Train loss reduced from **0.7608** to **0.5611** over **5 epochs**, showing consistent learning.
    - Accuracy improved from **69.11%** to **78.39%**, indicating better classification of training data.
  - **Stage 2:** Deeper layers were unfrozen to fine-tune the entire network.
    - Train loss dropped significantly from **0.4509** to **0.0985**, indicating the model's ability to capture complex patterns in the data.
    - Accuracy improved dramatically from **83.02%** to **96.59%**, reflecting strong learning.
- Metrics: High precision, recall, and F1 scores (**more than 88%**) in later epochs demonstrate the model's robust classification ability.



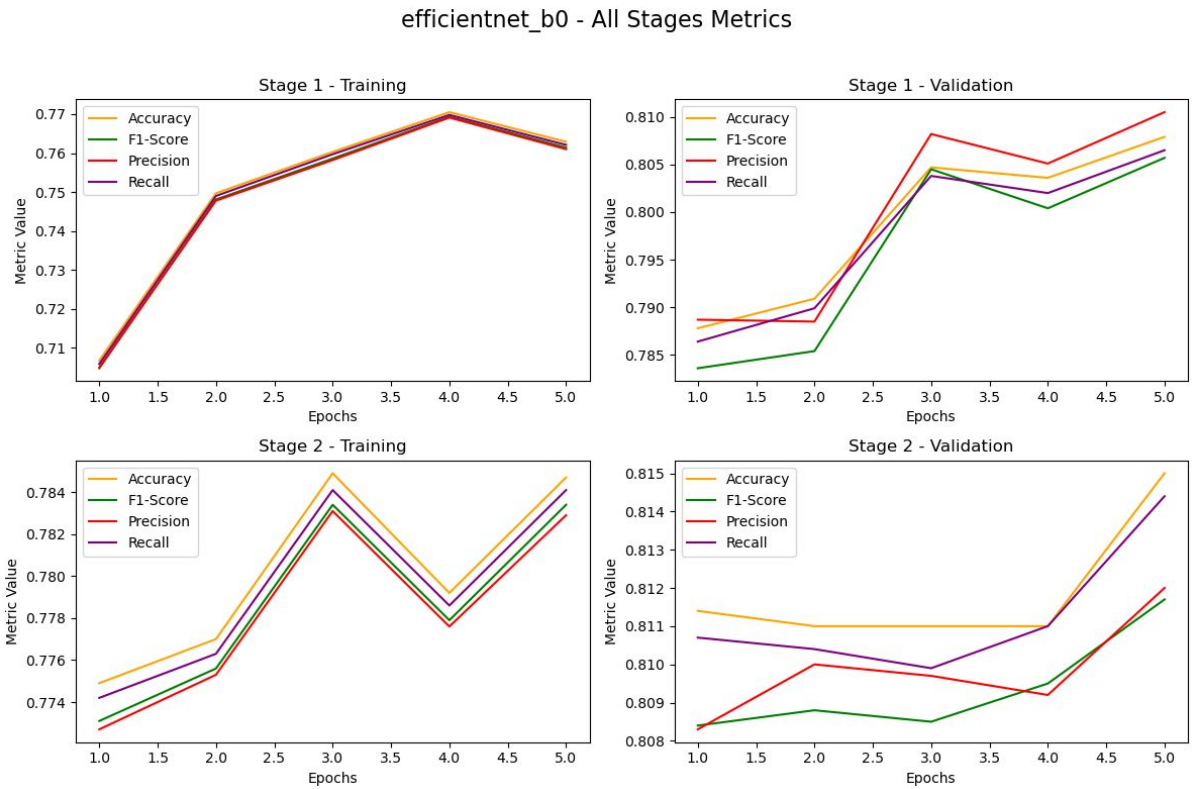
Convolutional Neural Networks (CNNs)

Efficientnet-B0 Model:

- **EfficientNet-B0** is an efficient and computationally **lightweight CNN**, making it **suitable for large datasets**.
- It is known for **achieving high accuracy with fewer parameters** by **scaling the depth, width, and resolution of CNNs** in a balanced way.
- It is **well-suited** for **limited computational resources**, while still maintaining good performance without significant overfitting in complex datasets like eye disease classification.

- It was fine-tuned in our project in two stages:
  - **Stage 1** trained only the classifier:
    - Validation loss reduced from **0.5633** to **0.5014**, with accuracy increasing from **78.78%** to **80.79%**.
  - **Stage 2** fine-tuned deeper layers for enhanced learning:
    - Validation loss remained stable, ending at 0.5036, with final accuracy of **81.50%**.

- Achieved validation accuracy of **81.50%** and F1 score of **81.17%**, demonstrating solid results.
- EfficientNet-B0 showed consistent learning but plateaued in later epochs, possibly due to its lightweight nature.

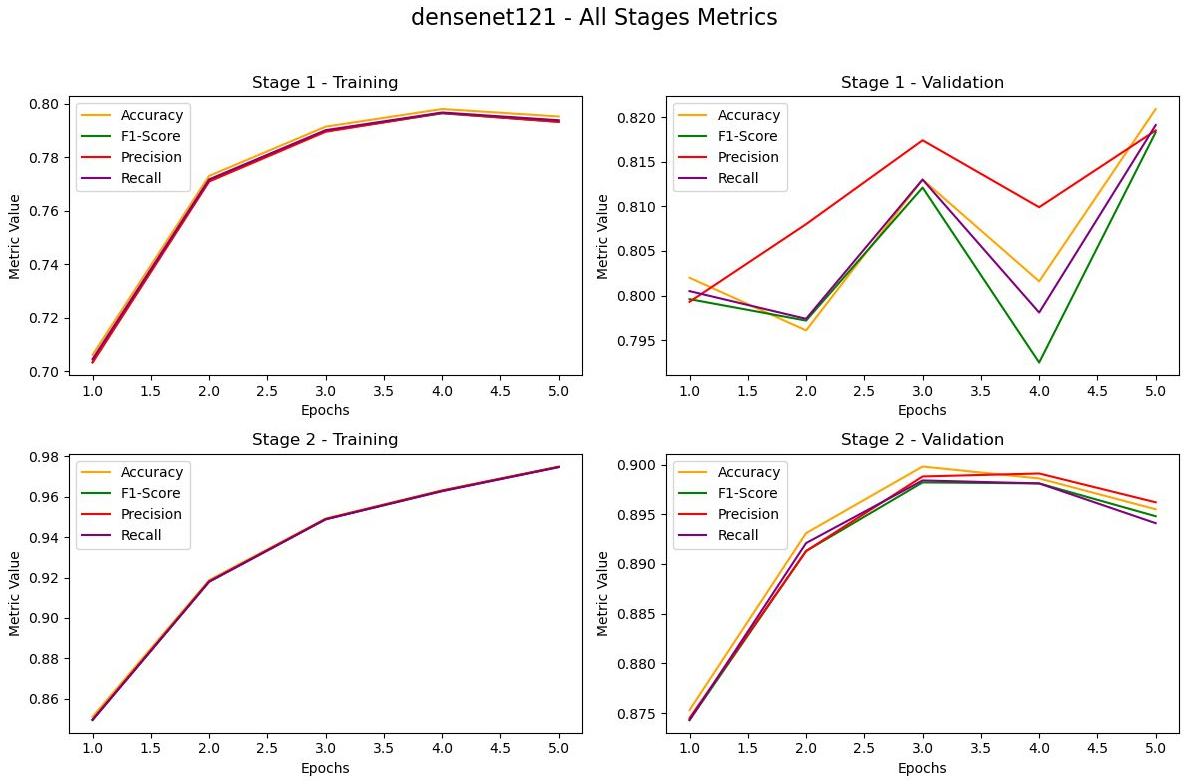


Convolutional Neural Networks (CNNs)

Densenet121 Model:

- **DenseNet121** is a CNN with **121 layers**, which **connects each layer to every other layer in a "dense" manner**.
- It is focused on **reusing features**, which **reduces redundancy** and **improves gradient flow** during backpropagation.
- **Each layer receives inputs from all previous layers**, leading to better feature reuse and reduced parameter count.
- It **efficiently learns compact representations without sacrificing accuracy**.
- It was fine-tuned in our project in two stages:
  - **Stage 1** trained only the classifier:
    - Validation loss dropped from **0.5266** to **0.4536**, and accuracy improved from **80.20%** to **82.09%**.
  - **Stage 2** fine-tuned deeper layers for enhanced learning:
    - Validation accuracy peaked at 89.98% in Epoch 3 and ended at **89.55%**.

- Achieved validation accuracy of **89.55%** and F1 score of **89.48%**, making DenseNet121 the best-performing model in our project.
- Dense connections allowed efficient learning and feature reuse, resulting in high performance across all metrics.



Conclusion

Comparing All models:

- **DINO and SimCLR** are excellent for self-supervised feature learning but require fine-tuning for classification tasks.
- **ResNet18 offers fast training** and solid performance, making it a strong baseline.
- **EfficientNet-B0 is lightweight and efficient** but slightly **underperforms** compared to ResNet18 and DenseNet121.
- **DenseNet121** achieves the **best accuracy (89.55%)** and **F1 score (89.48%)** due to efficient feature reuse and robust learning capabilities.

Conclusion

How can SSL models be used over CNN models in eye disease detection to make use of unlabeled data and improve adaptability for analyzing diverse eye images?

In case of different datasets

- **CNN models** are only suitable for **labeled datasets**, which **cost a lot** and **takes time to generate**. However, by using SSL models we are not depended only on labeled data,
- **SSL models** use **unlabeled** eye images to **learn meaningful features** out of images, and once they learned the features, they can be used on any sort of similar datasets. For classification however, they need well-structured fine tuning.

In case of limited data:

- **CNN models trained on limited labeled data** which may **lead to overfitting**, especially in small medical datasets like ours.
- However **SSL models** can handle this by learning generalized features from unlabeled data, and **reducing the chance of overfitting during fine-tuning**.

Comparable Metrics of SSL models

- After fine-tuning, our SSL models achieved competitive results.
- While **DenseNet121 achieved** the highest validation **accuracy** (**89.55%**) among CNNs, our **SSL models** demonstrated comparable metrics while offering the **added advantage of reduced dependency on labeled data**.

In case of using new approach:

- **Our project went beyond simple classification by adopting self-supervised learning**, a cutting-edge technique, into the workflow. This represents a significant step forward compared to traditional CNN-based approaches.

References

Dataset:

- **Kaggle Dataset:** Eye Diseases Classification Dataset(link)

Research Papers:

- **Eye Disease Classification:** Babaqi, T., Jaradat, M., Yildirim, A. E., Al-Nimer, S. H., & Won, D. (2023). Eye Disease Classification Using Deep Learning Techniques. Proceedings of the IISE Annual Conference & Expo. (Link)

SSL Methods:

- **DINO:** Emerging Properties in Self-Supervised Vision Transformers. (Link)
- **SimCLR:** A Simple Framework for Contrastive Learning of Visual Representations. (Link)

CNNs:

- **ResNet:** Deep Residual Learning for Image Recognition. (link)
- **EfficientNet:** Rethinking Model Scaling for Convolutional Neural Networks. (link)
- **DenseNet:** Densely Connected Convolutional Networks. (link)

References

Tools and Frameworks:

- numpy: Numerical computing for arrays and matrices.
- pandas: Data manipulation and analysis.
- matplotlib: Data visualization and plotting.
- PyTorch: Deep learning framework for building and training models.
- torchvision: Utilities for vision tasks (datasets, models, and transforms).
- vit_pytorch: Implementation of Vision Transformer (ViT) and DINO.
- PIL: Image loading and basic processing.
- tensorflow.keras.preprocessing.image.ImageDataGenerator: Image augmentation for training.
- sklearn.metrics: Evaluation metrics like accuracy, precision, recall, and F1 score.
- Kaggle: Platform for dataset access and experiment tracking.

Acknowledgments:

- **Course Instructor:** Professor Marco Raoul Marini.
- **Kaggle Contributor:** Guna Venkat Doddi, for providing the dataset.

Thank You