MSc Business Analytics

Full Time

Machine Learning and Content Analytics

Professor: Haris Papageorgiou

Assistant: Giorgos Perakis

Early Detection of Diabetic Retinopathy

A Deep Learning Approach

Members	Student Number
Koutsogianni Agapi	f2822316
Merlou Anna	f2822307
Ziaragkalis Stavros	f2822304
Xanalatou Athina	f2822406

September 2024, Athens

CONTENTS

TABLE OF FIGURES

Abstract

The efficacy of DenseNet-121 for early detection of diabetic retinopathy (DR) from fundus camera images is critical, given the condition's prevalence as a leading cause of blindness among diabetics. This study aims to evaluate the effectiveness of the DenseNet-121 architecture in identifying varying stages of DR, leveraging its dense connectivity to enhance feature propagation and mitigate gradient vanishing, thereby improving diagnostic accuracy. To address the ordinal nature of DR severity levels, we implemented DenseNet-121 with a specific design consideration, allowing the model to learn effective representations of the multilabeled classification task, considering the ordinal nature of DR severity levels. This investigation was a controlled, comparative analysis utilizing a dataset of fundus images annotated for DR severity. Image preprocessing included normalization, resizing, and enhancement techniques such as Gaussian blurring and contrast adjustment, and we employed data augmentation techniques, including rotation, flipping, and color jittering to optimize model input quality. The study enrolled images classified across multiple severity stages of DR, stratified by the degree of retinal damage. Models were trained using DenseNet-121 and compared against InceptionV3 and traditional convolutional neural network (CNN) architectures. Performance metrics such as accuracy, sensitivity, and specificity were calculated to assess the efficacy of each model, with a primary focus on the Quadratic Weighted Kappa (QWK) metric, which is well-suited for evaluating the ordinal nature of the severity levels of DR.

A total of 3,662 fundus images were processed, with a majority presenting with no to moderate DR. The DenseNet-121 model outperformed other architectures, demonstrating superior sensitivity (82%) and specificity (82%) in detecting early stages of DR. Notably, DenseNet-121 achieved a stable behavior with 89% quadratic weighted kappa, compared to 89% and 73% for InceptionV3 and CNNs, respectively, which both showed a more unstable performance in the validation dataset. The model was particularly effective in identifying non-proliferative DR stages, critical for timely therapeutic intervention. By leveraging the dense connectivity of the model, we enabled the learning of effective representations that capture the subtle differences between consecutive DR stages. This approach allowed the model to better generalize across the different severity levels, leading to improved performance. DenseNet-121 significantly enhances the detection and classification of DR from fundus images, outperforming traditional methods and other advanced models. This study confirms the potential of advanced machine learning techniques to revolutionize DR screening and diagnosis, particularly enhancing early detection rates and accuracy. Such advancements are crucial for implementing effective treatment strategies, potentially reducing the incidence of diabetes-related vision loss globally.

Introduction

The escalation of diabetes globally heightens the urgency for efficient DR screening techniques that can operate effectively even in resource-constrained settings. Traditional diagnostic methods, while reliable, often demand substantial healthcare resources and specialized skills not readily accessible in many regions. Substantial research has been directed towards the automation of DR detection, leveraging advances in medical imaging and machine learning. Several studies have demonstrated the potential of convolutional neural networks (CNNs) and other deep learning frameworks to enhance DR screening's accuracy and efficiency. Among these, DenseNet-121 has emerged as a particularly promising architecture due to its unique structure that facilitates deep feature propagation and reuse, mitigating the common issue of vanishing gradients in deep networks. This study introduces a customized DenseNet-121 model specifically adapted for the early detection and severity classification of diabetic retinopathy from fundus camera images. Our approach distinguishes itself by focusing on the model's capacity to handle the nuanced features of retinal images, which are often varied and complex due to the nature of DR's progression. By integrating advanced image preprocessing techniques and a robust training methodology, we aim to enhance the model's diagnostic accuracy and reliability.

The primary contributions of this research are threefold: First, we demonstrate the efficacy of DenseNet-121 in a novel application for DR, highlighting its advantages over other architectures like InceptionV3 in terms of feature utilization and depth. Second, we detail a comprehensive preprocessing strategy that improves model training and performance, addressing common challenges in medical image analysis such as variable image quality and class imbalance. Lastly, the study provides insights into the integration of such models into clinical workflows, suggesting a scalable solution that could significantly improve DR screening practices. The remainder of the paper is organized as follows: Section II reviews related works and situates our research within the context of current technological advancements in DR detection. Section III describes the methodology, including data preprocessing, model architecture, and model implementation. Section IV presents the experimental results and discusses the model's performance in comparison to less complex models. Section V concludes with the implications of our findings for future research directions.

A. Core Concept

Diabetic Retinopathy

Definition

Diabetes is a chronic disease affecting over 400 million people worldwide. People with diabetes face complications, one of which is diabetic retinopathy. Diabetic retinopathy affects the blood vessels of the retina, and the light-sensitive tissue at the back of the eye. High blood sugar levels associated with diabetes can cause damage to the small blood vessels in the retina, leading to fluid leakage, swelling, and the abnormal growth of new blood vessels. Over time, these changes impair the retina's ability to function properly, which can lead to vision loss or even blindness. Over 537 million people globally are diagnosed with diabetes and the prevalence of DR is expected to rise, leading to a surge in demand for retinal screening.

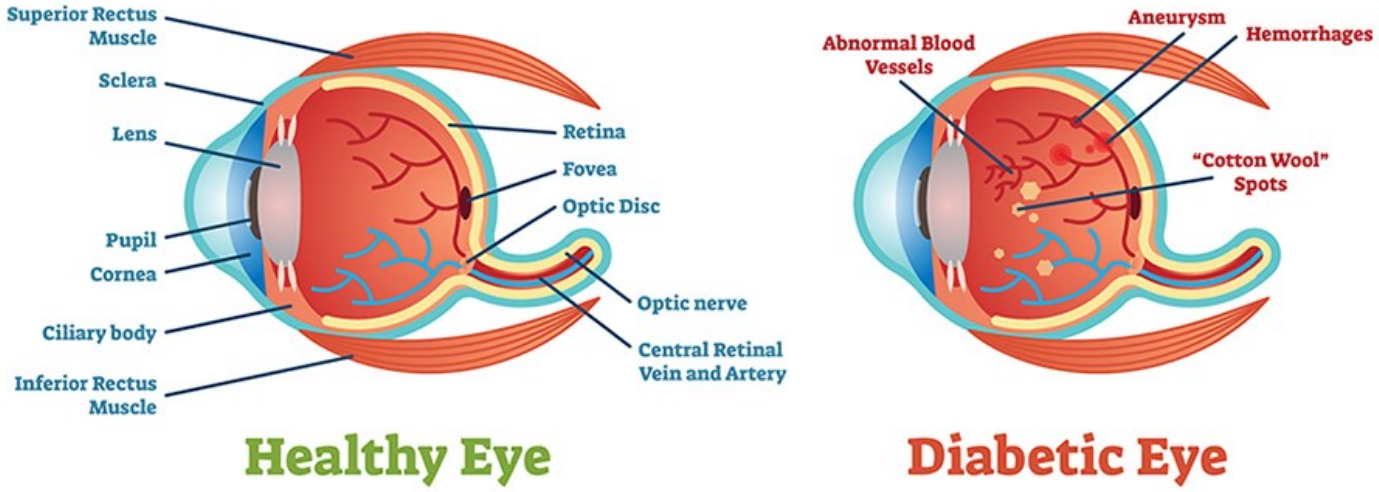


Figure 1 - Healthy vs Diabetic Eye

There are two main stages of DR, Non-Proliferative Diabetic Retinopathy (NPDR) and Proliferative Diabetic Retinopathy (PDR) (See Figure 2):

- **NPDR** corresponds to the **early stages** of the disease, including **mild**, **moderate**, and **severe** forms, where damage to the retinal blood vessels occurs without the formation of new abnormal vessels. The blood vessels in the retina weaken and may leak fluid or tiny amounts of blood, causing the retina to swell. These stages align with the first three stages of DR, where symptoms like microaneurysms, blocked vessels, and retinal hemorrhages begin to appear.
- As the disease progresses, it transitions into **PDR**, which corresponds to the **later stages** of DR. In PDR, abnormal new blood vessels (neovascularization) form in response to ischemia, leading to more severe complications such as vitreous hemorrhage and retinal detachment. This **proliferative stage** represents the most **advanced form of DR** and carries the highest risk of vision loss or blindness.

Early detection through regular eye screenings is crucial for preventing the progression of DR and minimizing the risk of vision impairment.

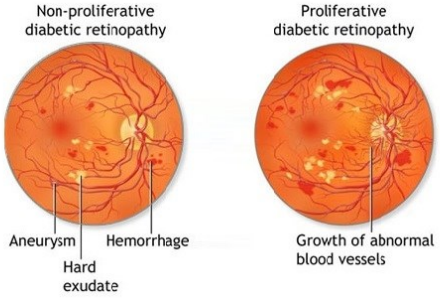


Figure 2 - Non-Proliferative and Proliferative DR eye

The progression from NPDR to PDR underscores the importance of early detection and intervention. While NPDR might remain asymptomatic for an extended period, regular eye screenings are vital for catching early signs before they advance to the proliferative stage, where the risk of irreversible vision damage is much higher.

Stages of DR

DR progresses through various stages, each marked by worsening retinal damage, primarily caused by weakened or blocked blood vessels. These stages are categorized based on the degree of retinal changes, such as swelling, hemorrhaging, and abnormal blood vessel growth. Early detection through regular eye examinations is crucial for preventing vision loss. The disease can remain asymptomatic in its early stages, making it especially dangerous as the retina incurs damage without obvious signs.

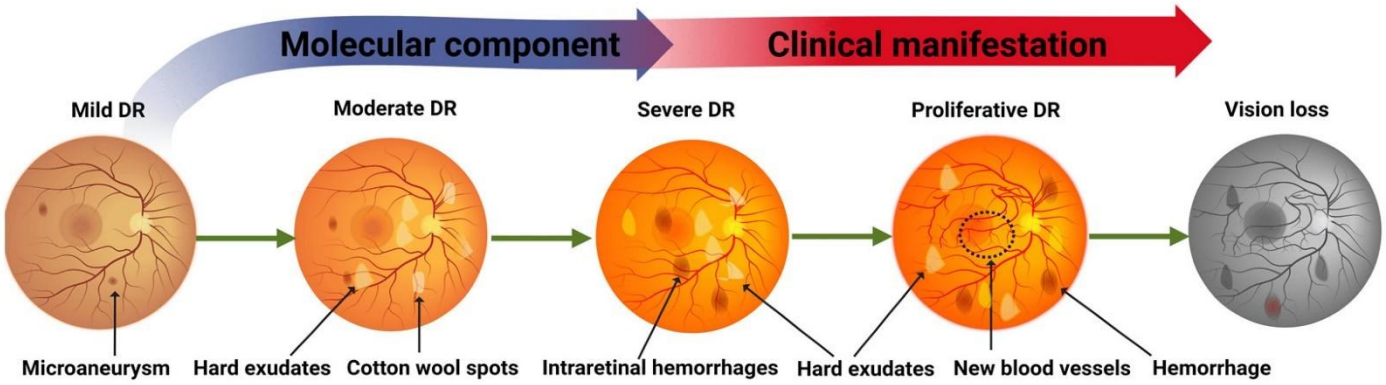


Figure 3 - Stages of transaction from NPDR to PDR

The following section outlines the distinct stages of diabetic retinopathy, beginning with no visible disease and advancing through increasingly severe damage until the proliferative stage (See Figure 3):

1. **No Disease Visible (No DR, Stage I)**
   The retina appears healthy, there are no abnormalities visible and there are no signs of diabetic retinopathy.

2. **Mild Non-Proliferative Diabetic Retinopathy (Mild NPDR, Stage II)**
   The first observable signs of DR occur at this stage, with the presence of microaneurysms, which are tiny bulges in the retinal blood vessels. This stage is considered mild, and while there may be no noticeable vision changes, it's an early indicator of potential retinal damage.
3. **Moderate Non-Proliferative Diabetic Retinopathy (Moderate NPDR, Stage III)**
   At this stage, more significant retinal damage is evident, including microaneurysms, retinal dot and blot hemorrhages (small bleeding spots in the retina), and hard exudates or cotton wool spots (areas of retinal swelling and ischemia). These changes may start to impair vision.
4. **Severe Non-Proliferative Diabetic Retinopathy (Severe NPDR, Stage IV)**
   The retinal damage becomes more extensive in severe NPDR. Signs include more than 20 intraretinal hemorrhages in each of the four retinal quadrants, venous beading (irregularities in the retinal veins) in two or more quadrants, and prominent intraretinal microvascular abnormalities (IRMA) in one or more quadrants. Although there are no signs of proliferative changes, this stage indicates a high risk of progressing to the more dangerous proliferative form of DR.
5. **Proliferative Diabetic Retinopathy (PDR, Stage V)**
   This is the most advanced and severe stage of DR. It is characterized by the growth of neovascularization in the retina. These fragile vessels can easily bleed, leading to vitreous or pre-retinal hemorrhages. Abnormal vessels can also cause retinal detachment, significantly increasing the risk of severe vision loss, or result in neovascular glaucoma, which can lead to blindness if not treated promptly.

Grading protocol ICDR and ETDRS Severity Levels:

Stages	Score	Observable Findings
No apparent retinopathy	0	No abnormalities (Level 10 ETDRS[1])
Mild non-proliferative DR	1	Microaneurysm(s) only (Level 20 ETDRS)
Moderate non-proliferative DR	2	More than just microaneurysm(s) but less than severe non-proliferative diabetic retinopathy (Level 35, 43, 47 ETDRS)
Severe non-proliferative DR	3	Any of the following: > 20 intra-retinal hemorrhages in each of 4 quadrants, definite venous beading in ≥2 quadrants, prominent intra-retinal microvascular abnormalities in ≥1 quadrant, or no signs of proliferative retinopathy. (Level 53 ETDRS: 4-2-1 rule)
Proliferative DR	4	One or more of the following: neovascularization and/or vitreous or preretinal hemorrhages. (Levels 61, 65, 71, 75, 81, 85 ETDRS)

Figure 4 - Severity Levels

[1] Early Treatment Diabetic Retinopathy Study

Common symptoms of DR

The most well-known noticeable symptoms, including:

- **Blurry or fluctuating vision:** The swelling of the retina can cause blurred vision or changes in how well you can see.
- **Dark spots or floaters:** Small spots or strings that float in your field of vision may occur because of blood leakage into the eye.
- **Difficulty seeing at night:** Night vision can be compromised, making it hard to drive or move around in low-light conditions.
- **Color vision impairment:** Some individuals may notice difficulty in distinguishing colors due to damage to the retinal tissue.
- **Sudden vision loss:** In advanced stages, sudden vision loss can occur, often due to severe bleeding from abnormal blood vessels or retinal detachment.

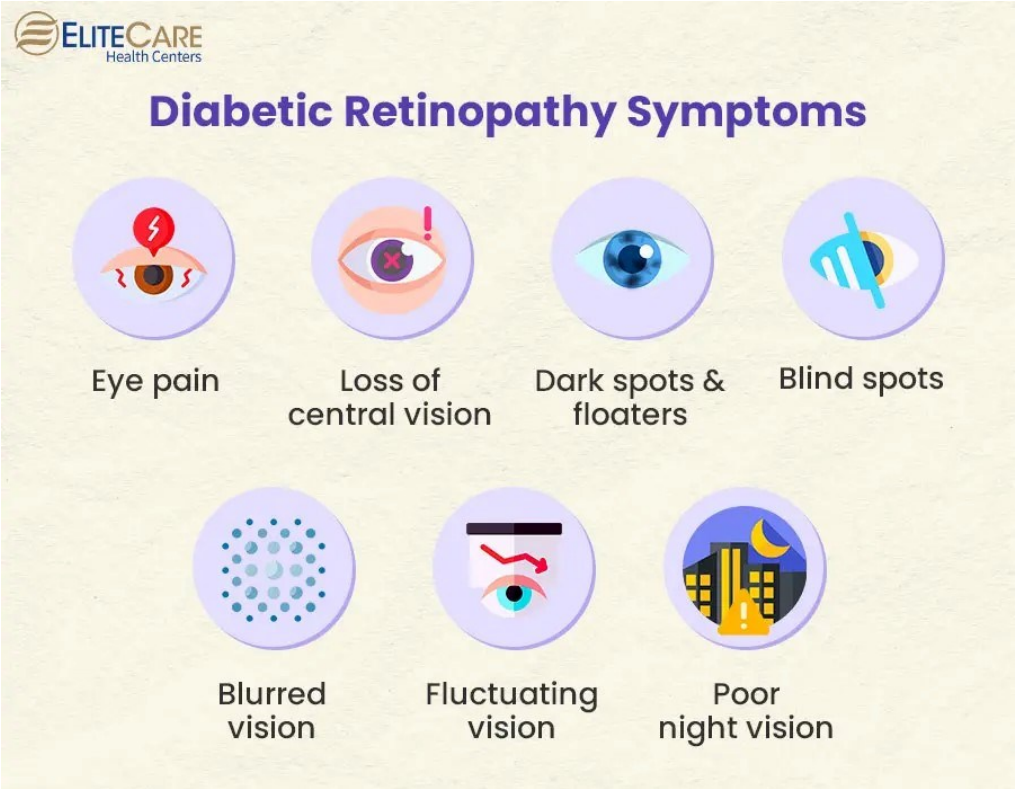


Figure 5 - Symptoms of DR

Further Complications

Diabetic Retinopathy not only threatens vision but can also lead to other serious eye complications if left untreated. These include:

Figure 6 - Macular Edema eye

Macular Edema: This occurs when fluid leaks into the macula, the part of the retina responsible for sharp, central vision, causing it to swell and leading to distorted or blurred vision. Macular edema is a common cause of vision loss in people with DR.

Figure 7 - Retinal Detachment eye

Retinal Detachment: In advanced cases, the formation of scar tissue from abnormal blood vessel growth can cause the retina to pull away from the back of the eye, resulting in a retinal detachment. This is a medical emergency that can lead to permanent blindness if not treated promptly.

Figure 8 - Glaucoma eye

Glaucoma: Diabetic Retinopathy can increase the risk of developing glaucoma, a condition that damages the optic nerve due to increased pressure inside the eye. Over time, this pressure can lead to vision loss.

Figure 9 - Vitreous Hemorrhage eye

Vitreous Hemorrhage: The abnormal blood vessels in Proliferative Diabetic Retinopathy can rupture and bleed into the vitreous, the clear, gel-like substance that fills the center of the eye. This bleeding can block vision and may require surgical treatment.

Suffering Population

In 2020, an estimated 103 million people were affected by DR, and projections indicate that this number could exceed 160 million by 2045. The burden of DR is particularly heavy in low- and middle-income countries (LMICs), where diabetes rates are rapidly increasing, and healthcare systems often lack adequate resources for early detection and treatment. In these regions, DR presents a significant public health challenge due to limited access to specialized eye care and screening services. Consequently, individuals are more likely to develop advanced stages of DR.

The prevalence of DR is influenced by several factors, including the duration of diabetes. Approximately one-third of all people with diabetes are expected to develop some form of DR during their lifetime. On a global scale, it is estimated that 5 to 10% of all cases of blindness can be attributed to DR. Globally, the prevalence of DR among diabetic patients is estimated to be 27.0%, which leads to 400 thousand blindness in the world.

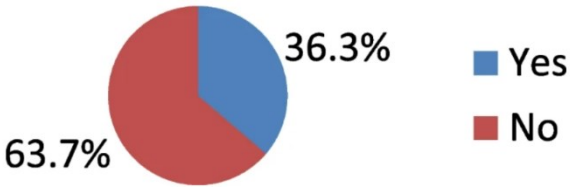


Figure 10 - Prevalence of DR and associated factors among type 2 diabetic patients

Research from 2021 estimated that 36.3% of diabetic patients are likely to develop diabetic retinopathy. People with Type 1 diabetes tend to experience retinopathy earlier and more frequently because they are often diagnosed at a younger age and live with diabetes for a longer time. In contrast, people with Type 2 diabetes may already have retinopathy at the time of diagnosis, as their diabetes can go undetected for years before being identified (See Figure 10).

Today, there is a great need for technologies that provide solutions for screening large populations, particularly in LMICs, where healthcare infrastructures are often overstretched. The map in Figure 11 illustrates the global distribution of studies focused on DR in LMICs. Research efforts are concentrated primarily in Asia, with China (31 studies) and India (27 studies) leading the way. These countries face a significant DR burden due to their large diabetic populations and constrained healthcare resources. In contrast, countries such as Mexico, Brazil, South Africa, and a few African nations have fewer studies (1-4 studies). This disparity highlights the need for more research and intervention efforts, particularly in underrepresented regions like Africa and Latin America. This map emphasizes the uneven distribution of research on diabetic retinopathy, with significant gaps in regions where the disease burden is on the rise.

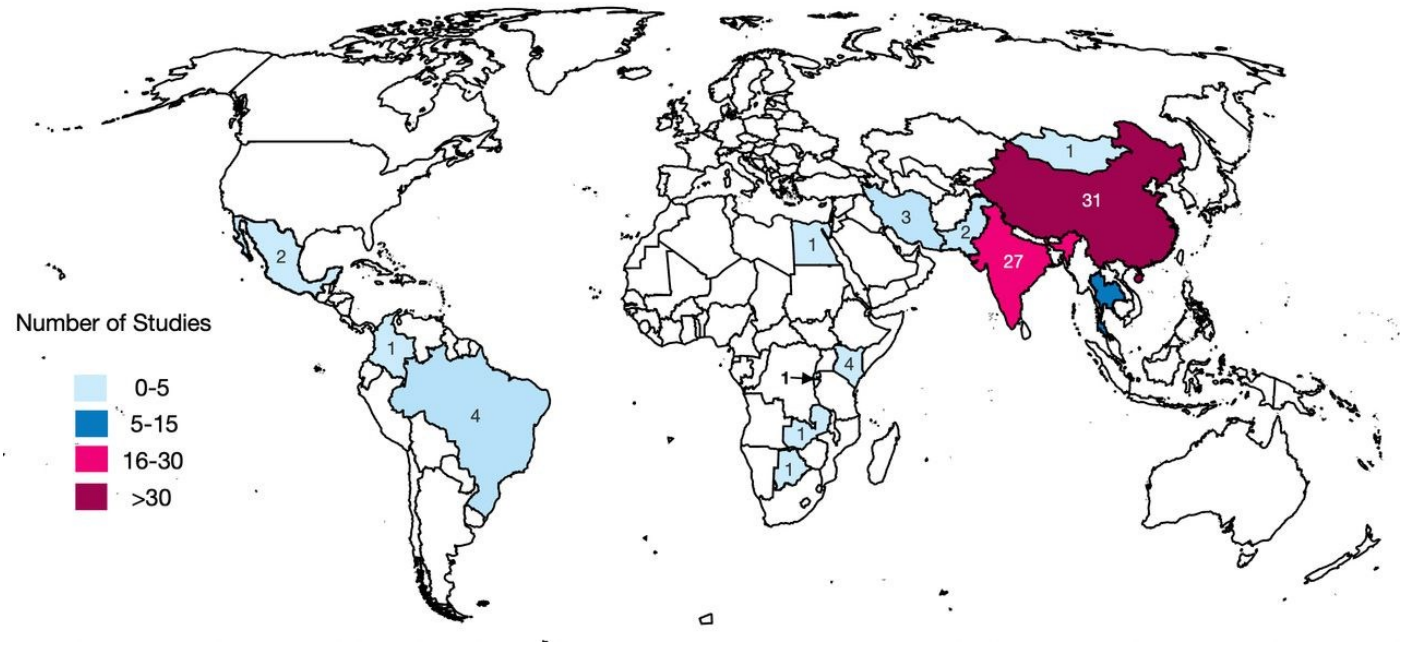


Figure 11 - World map showing the distribution of AI research for DR in LMICs

The graph in Figure 12 compares the performance of various AI systems used for diabetic retinopathy screening across different countries. The x-axis represents specificity (%), while the y-axis represents sensitivity (%). The highest-performing AI systems, such as "Remidio" and "SELENA+", developed in India and Thailand, demonstrate both high sensitivity (close to 100%) and high specificity (above 90%). This indicates their strong ability to accurately identify both positive and negative cases.

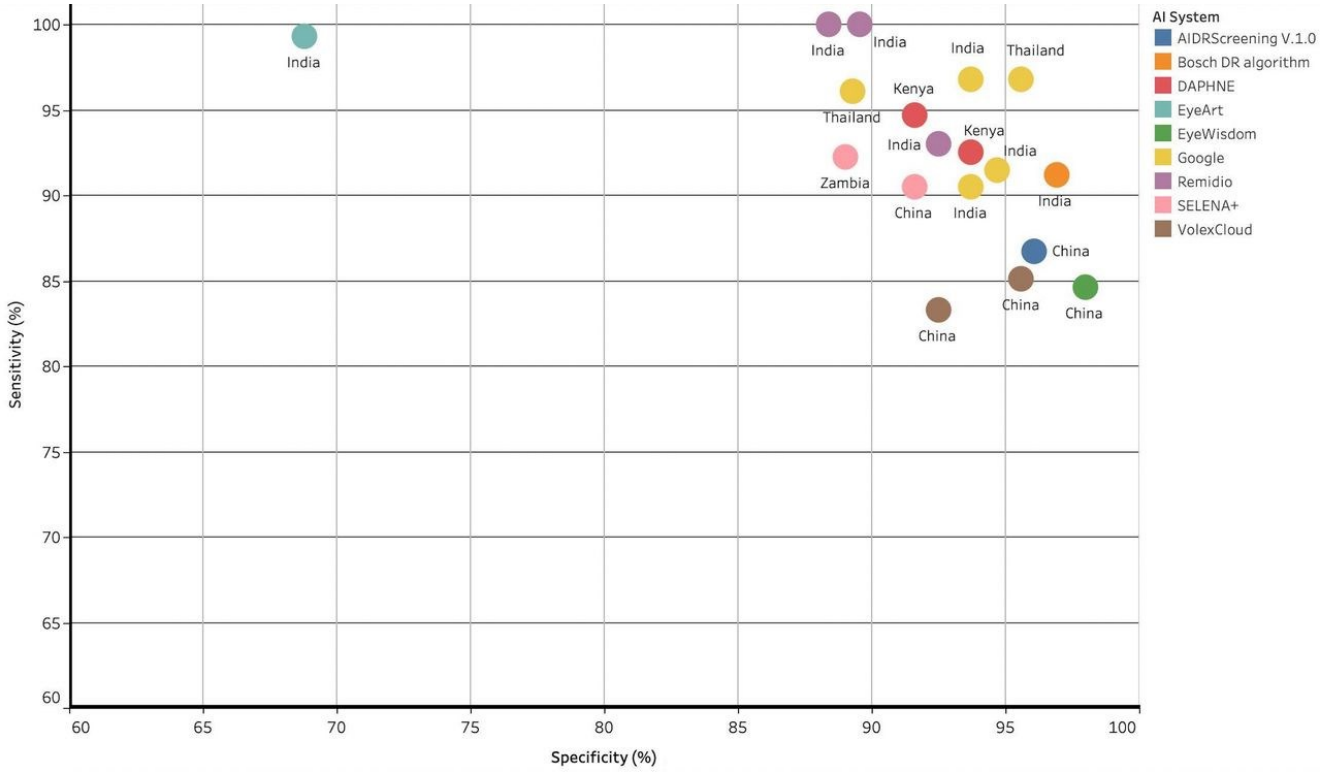


Figure 12 - Performance of AI systems for diabetic retinopathy prediction

India and China are prominently represented, reflecting their active engagement in AI research for DR. Other countries, such as Zambia, Kenya, and Thailand, also have relevant studies, albeit fewer. AI systems like EyeArt, Google, and Bosch's DR algorithms show some variation in performance, but many achieve sensitivity and specificity rates between 80% and 95%. This graph underscores the role AI can play in detecting DR, especially in LMICs. These tools can significantly improve early diagnosis and treatment, potentially reducing the prevalence of blindness or others severe eye illnesses.

Value of DR Research

Clinical Excellence

Clinical excellence in diabetic retinopathy management is paramount for preventing vision loss in diabetic patients, as the disease remains one of the leading causes of blindness globally. The burden of DR is significant, with an increasing prevalence driven by rising rates of diabetes, particularly in aging populations. Early detection and accurate classification of the disease stages, such as non-proliferative and proliferative DR, are essential for guiding appropriate treatment decisions.

The use of advanced retinal imaging techniques, such as **Optical Coherence Tomography (OCT)** and **fundus photography**, allows clinicians to detect even the earliest signs of DR, including microaneurysms, retinal hemorrhages, and retinal swelling.

OCT is a non-invasive imaging test that uses light waves to take cross-sectional images of the retina, the light-sensitive tissue at the back of the eye. It provides detailed, high-resolution images of the retina's layers, allowing doctors to measure their thickness and detect abnormalities that may indicate diseases like age-related macular degeneration, diabetic retinopathy, glaucoma, and macular holes or edema. This technology is crucial for diagnosing and monitoring these conditions because it can detect changes in the retina that are not visible through regular eye exams, making it a critical tool in DR management. OCT is often used to assess structural details like retinal thickness, which is particularly important in identifying and managing macular edema in DR patients.

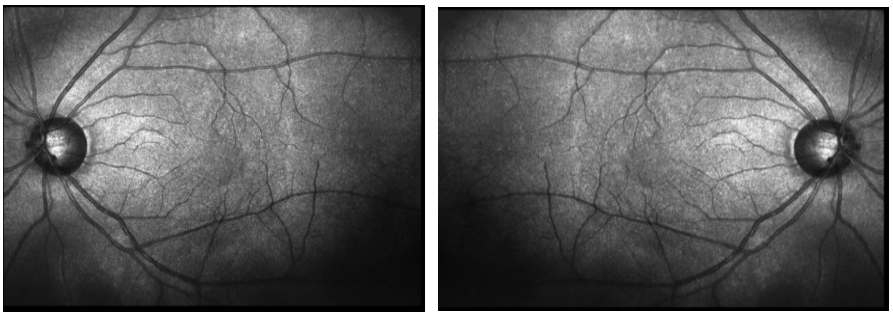


Figure 14 - Optical Coherence Tomography image

Figure 13 - Fundus Tomography image

On the other hand, fundus photography captures a picture of the interior surface of the eye, specifically the retina, optic disc, macula, and posterior pole, referring to the back of the eye. The images are typically captured using a specialized camera (fundus camera), which can record color images or perform fluorescein angiography, a type of imaging that uses a fluorescent dye to highlight blood vessels in the retina. Fundus photography is often used to document the appearance of the retina over time, detect conditions like diabetic retinopathy, glaucoma, and macular degeneration, and assess the optic nerve for signs of glaucoma.

This method allows clinicians to monitor disease progression by capturing surface features such as hemorrhages or changes in the optic disc (See Figure 14).

While OCT provides a detailed cross-sectional view of the retina's layers, giving depth information, fundus photography captures a surface view of the retina in a single, two-dimensional image. Thus, OCT is crucial for assessing structural retinal details, while fundus photography is primarily used for documenting surface abnormalities. Both imaging modalities complement each other, providing a comprehensive view of retinal health and aiding in accurate diagnosis and disease management.

Furthermore, automated systems using artificial intelligence and machine learning can enhance clinical decision-making by providing screening methods, reducing diagnostic errors, and ensuring consistency in patient evaluations across healthcare systems. As reported in multiple DR research studies worldwide, the integration of AI into diagnostic workflows has enabled improved sensitivity and specificity in DR detection compared to traditional manual methods. For instance, in large-scale screenings, AI-based algorithms have demonstrated the ability to classify DR stages with accuracy comparable to that of experienced ophthalmologists. These advancements contribute to clinical excellence by reducing the burden on ophthalmologists, especially in regions with limited healthcare resources, while improving patient outcomes by enabling timely intervention. As healthcare systems increasingly adopt these technologies, the hope is that they will help reduce the global burden of vision loss associated with diabetic retinopathy.

Technological Advancements

In recent years, significant technological advancements have been made in the early detection of Diabetic Retinopathy using deep learning techniques. These approaches primarily rely on machine learning architectures to analyze retinal images for signs of DR, facilitating early intervention and treatment. Various studies have employed diverse datasets and methods to optimize detection performance. The outline of some common methods may include:

1. **Convolutional Neural Networks (CNNs)**: CNNs are the most widely used deep learning models for DR detection. They can automatically extract features from images, making them ideal for medical image classification tasks. Variants of CNNs, such as VGG-16, ResNet, and DenseNet, have been commonly applied to improve classification accuracy and specificity.
2. **Transfer Learning**: Pre-trained networks like AlexNet, GoogleLeNet, and ResNet have been fine-tuned using DR-specific datasets. Transfer learning allows models to leverage previously learned features, reducing the need for large training datasets and improving performance on small datasets.
3. **Ensemble Learning**: Ensemble learning combines multiple models to improve performance by leveraging the strengths of different architectures to enhance robustness and accuracy.
4. **Preprocessing Techniques**: Preprocessing steps, such as Contrast Limited Adaptive Histogram Equalization (CLAHE), are used to enhance the quality of retinal images, making it easier for models to detect subtle signs of DR.
5. **Binary vs. Multiclass Classification**: Many approaches simplify DR detection into binary classification (DR vs. No DR). However, multiclass classification, which identifies different stages of DR, is crucial for early diagnosis and appropriate intervention.
6. **Evaluation Metrics**: Common evaluation metrics include accuracy, sensitivity, specificity, and the area under the curve (AUC). These metrics help measure how well the model performs in identifying DR, especially at early stages.

The bellow matrix summarizes some key studies along with their methods, datasets, and results:

Study	Method	Dataset	Results	Notes and Challenges
Pratt et al. (2016)	CNN with Stochastic Gradient Descent	Kaggle (80,000 images)	Accuracy: 75%, Sensitivity: 30%	High false negatives in binary classification
Soniya et al. (2016)	Single and heterogeneous CNN	DIARETDB0 (130 images)	Accuracy: 95%	Small dataset size
Gargeya & Leng (2017)	CNN	MESSIDOR2, E-Ophtha	AUC: 0.94, Sensitivity: 93%	No accuracy measure reported
Lam et al. (2017)	GoogleLeNet-v1, AlexNet, VGG-16	Kaggle EyePACS (243 images)	Accuracy: 74%-95%	Binary classification
Khalifa et al. (2019)	AlexNet, ResNet18, VGG16, etc.	APTOS 2019	Best Accuracy: 97.9% (AlexNet)	High accuracy, limited dataset diversity
Nguyen et al. (2020)	CNN, VGG-16, VGG-19	Kaggle 2015 competition	Accuracy: 82%	No ensemble or fusion used
Tymchenko et al. (2020)	Three-headed CNN	APTOS 2019	Accuracy: 99.3%	Focused only on detecting blindness
Pour et al. (2020)	EfficientNet B5, CLAHE	MESSIDOR, IDRiD	AUC: 0.93-0.94	Binary classification
Mushtaq & Siddiqui (2021)	DenseNet-169	Diabetic Retinopathy Detection 2015, APTOS 2019	Accuracy: 90%	No fusion/ensemble methods applied
Karki & Kulkarni (2021)	EfficientNet	Kaggle APTOS	Kappa: 0.924	Lacks evaluation using standard metrics
Parthasharathi et al. (2022)	CNN	Kaggle (1000 images)	Accuracy: 91.5%	Binary classification
Oulhadj et al. (2022)	DenseNet, InceptionV3, ResNet-50	Kaggle APTOS	Accuracy: 85.28%	Moderate accuracy
Lahmar & Idri (2022)	VGG16, VGG19, InceptionV3, etc.	Kaggle DR, MESSIDOR-2	Accuracy: 84.01%-88%	Moderate accuracy across multiple models
Gundluru et al. (2022)	DNN, PCA, Harris Hawks Optimization	UCI Diabetic Retinopathy Debrecen Dataset	Accuracy: 96.7%	Risk of overfitting

Figure 15 - Key studies for DR prediction

Key Observations

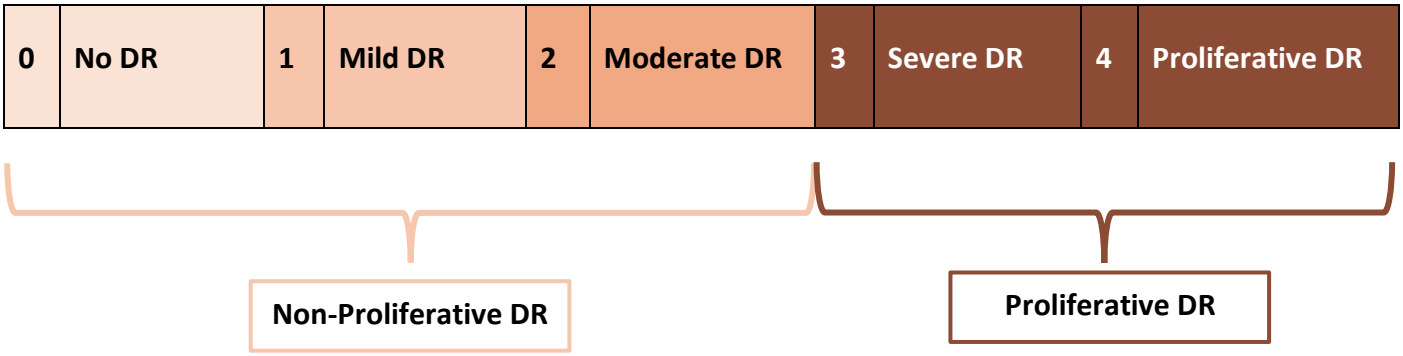
- High Accuracy Models: Approaches using AlexNet, ResNet, and DenseNet consistently achieve high accuracy, with some studies reporting over 95% accuracy. However, these models often rely on binary classification, which may overlook early or intermediate stages of DR.
- Small Dataset Limitations: Studies such as those using the DIARETDB0 dataset face challenges due to small sample sizes, which limit the model's ability to generalize to larger, more diverse populations.
- Evaluation Metrics: While accuracy is a commonly used metric, sensitivity and specificity are critical in medical applications to ensure that false negatives are minimized. Some studies, like Tymchenko et al., report impressive accuracy but lack sensitivity to early DR stages.
- Future Directions: There is room for improvement in developing multiclass classifiers that can distinguish between various stages of DR. Moreover, using ensemble learning techniques could reduce the likelihood of overfitting, especially when working with smaller datasets.

APTOS Research and Competition

The detection and classification of DR have undergone significant advancements, driven largely by research efforts and innovation in the fields of machine learning, deep learning, and medical imaging.

One such prominent effort is exemplified by the **Asia Pacific Tele-Ophthalmology Society (APTOS) 2019 Blindness Detection** competition. The APTOS Blindness Detection competition is a data science competition that was organized by APTOS and hosted on Kaggle in 2019. The goal of the competition was to develop machine learning and deep learning models that could detect and classify different levels of DR.

The APTOS 2019 Blindness Detection competition provided a dataset of 3,662 retinal images labeled according to the severity of diabetic retinopathy. Participants were tasked with building machine learning models that could classify these images into five categories:



This multiclass classification problem posed a challenge for researchers and developers, as it required models not only to distinguish between healthy and unhealthy retinal images but also to accurately differentiate between varying levels of disease progression. The competition fostered innovation by encouraging the use of state-of-the-art deep learning techniques, which included:

- Transfer Learning: Many participants employed pre-trained models such as EfficientNet, ResNet, and DenseNet. These models, which were initially trained on large-scale image datasets like ImageNet, were fine-tuned on the APTOS dataset. Transfer learning allowed participants to benefit from previously learned visual features, which helped improve performance even with a relatively small dataset.
- Ensemble Learning: To achieve higher accuracy, many competitors combined the predictions of multiple models (ensembles) to create more reliable classifiers. By taking advantage of the strengths

of different architectures, ensemble models helped reduce the variance and bias that can arise in single-model approaches.

- Advanced Preprocessing: Effective preprocessing of retinal images played a key role in improving model performance. Techniques like CLAHE, which enhances the contrast of images, were widely adopted to make subtle features in the images more detectable by the models. This was crucial in distinguishing between mild and moderate stages of DR.

Innovation Outcomes

The APTOS 2019 competition served as a platform for participants to experiment with and refine various machine learning and image processing techniques. Some of the key advancements include:

- Use of EfficientNet: One of the most popular models used during the competition was EfficientNet. Its unique scaling strategy, which balances model depth, width, and resolution, proved highly effective in capturing fine details in retinal images. This model achieved some of the highest accuracy scores in the competition and has since become a go-to architecture in medical imaging tasks.
- Multiclass Classification: While many previous works focused on binary classification (i.e., DR vs. No DR), the APTOS 2019 competition pushed forward the notion of multiclass classification. This approach is critical for early detection and timely intervention, as it allows for a more granular understanding of the progression of DR. Early detection can significantly reduce the risk of vision loss, making this an important innovation in the field.

Impact on Medical Practice

The advancements showcased in the APTOS 2019 competition have strong implications for real-world medical practice. Automated DR screening tools that can accurately classify the severity of DR have the potential to revolutionize eye care, especially in regions with limited access to ophthalmologists. By integrating these models into telemedicine platforms, healthcare providers can ensure that at-risk patients receive timely referrals for further examination and treatment, preventing the progression of DR to more severe stages and ultimately reducing blindness rates.

Ongoing Research Directions

The research and innovations emerging from the APTOS 2019 competition continue to influence the broader field of medical image analysis. Ongoing research is focusing on:

- Improving Generalizability: One of the key research goals is to develop models that can generalize across different datasets and population groups. This includes expanding datasets to include more diverse populations and employing techniques like domain adaptation to ensure that models trained on one dataset perform well on others.
- Multimodal Learning: Integrating data from multiple sources (e.g. combining retinal images with patient history or genetic data) is a promising direction that could improve the accuracy of DR detection and prediction.

- Explainability and Transparency: While deep learning models can achieve high accuracy, there is a growing emphasis on ensuring that these models are interpretable. Research in explainable AI (XAI) aims to provide insights into how models make decisions, which is essential for gaining the trust of clinicians and patients.

B. Business Use Case

Our company is committed to revolutionizing the diagnosis and treatment of diabetic retinopathy with AI-powered image analysis. By harnessing the power of machine learning and computer vision, we aim to provide a more accurate, efficient, and accessible diagnostic solution for healthcare providers and patients alike. Our system is designed to support the early detection and prevention of diabetic retinopathy, enabling timely interventions and improving patient outcomes. Ultimately, our vision is to make advanced ophthalmic diagnostics accessible to all healthcare facilities, from major hospitals to local clinics, and to equip healthcare providers worldwide with the tools they need to deliver fast, accurate, and early diagnoses of diabetic retinopathy. We envision a future where our AI system is used in telemedicine platforms, integrated into mobile screening units, and embedded in primary care facilities, making early DR detection available even in the most remote regions, and significantly reducing the global burden of vision loss caused by diabetic retinopathy.

Why: Problem Statement

Diabetic retinopathy is a growing concern worldwide, with millions of people suffering from vision loss and blindness due to late detection and inadequate treatment. Current diagnostic methods are often time-consuming, expensive, and inaccessible to underserved communities. Our company aims to address this issue by developing an AI-powered diabetic retinopathy detection system that enables early detection and prevention of vision loss.

What: Business Propositions

Our business propositions are centered around the following key areas:

- **Early Detection and Prevention**: Our system enables early detection of Diabetic Retinopathy, allowing healthcare providers to identify patients at risk of vision loss and initiate timely interventions to prevent or slow disease progression.
- **Increased Accessibility**: It increases accessibility to Diabetic Retinopathy diagnosis, particularly in underserved communities, by providing a cost-effective and efficient diagnostic solution that can be used in a variety of settings, including primary care clinics and community health centers.
- **Enhanced Patient Engagement**: Our platform empowers patients to take a more active role in their eye health by providing them with personalized information about their Diabetic Retinopathy risk and severity, enabling them to work more effectively with their healthcare providers to manage their condition.
- **Data-Driven Insights**: Our system provides healthcare providers with predictions of the level of Diabetic Retinopathy (among 5 ordinal levels) and the corresponding probability scores, enabling them to make informed decisions about patient care and treatment plans.

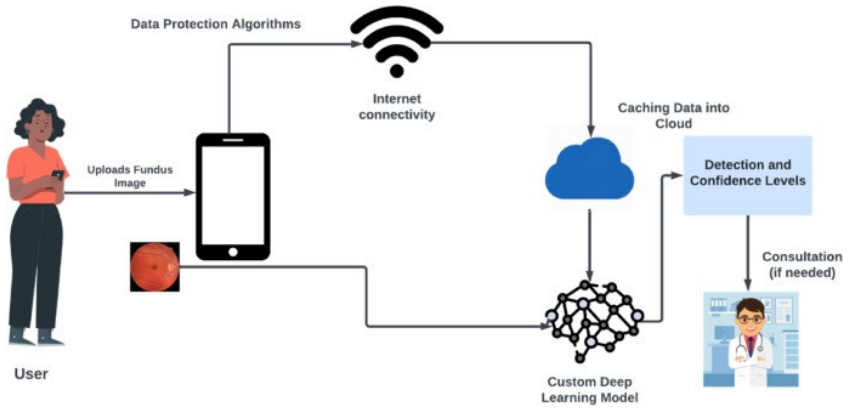


Figure 16 - A Secure Cloud-Based Detection System

How: Objectives and Strategies

To achieve our business goals, we have set the following objectives:

- **Improve Diagnostic Accuracy**: Achieve high accuracy in detecting diabetic retinopathy and its stages using image analysis powered by AI.
- **Increase Early Detection Rates**: Increase the number of patients diagnosed with diabetic retinopathy at an early stage, enabling timely interventions and preventing vision loss.
- **Reduce Healthcare Costs**: Reduce healthcare costs associated with diabetic retinopathy by minimizing the need for invasive treatments and hospitalizations.
- **Enhance Patient Experience**: Provide a seamless and user-friendly experience for patients, enabling them to easily access and manage their eye health.

Our strategies for achieving our objectives include integrating our system with Electronic Health Records (EHRs) to enable seamless data exchange and streamline clinical workflows. We also plan to enable remote image analysis and reporting, which will facilitate telemedicine consultations and reduce the need for in-person visits. Additionally, we will implement real-time communication tools to facilitate communication between patients, healthcare providers, and our system, ensuring that all parties are informed and up to date on patient care. Furthermore, we will provide personalized recommendations and treatment plans based on individual patient needs and medical histories, allowing for tailored care that addresses the unique needs of each patient.

Our system has various applications in the healthcare landscape, including:

- **Primary Care**: Integrate our system into primary care settings to enable early detection and prevention of diabetic retinopathy.
- **Specialty Care**: Collaborate with ophthalmologists and optometrists to provide specialized care and treatment plans for patients with diabetic retinopathy.
- **Population Health Management**: Partner with healthcare organizations to implement population health management strategies and reduce the burden of diabetic retinopathy on healthcare systems.
- **Clinical Research**: Collaborate with researchers to advance the understanding of diabetic retinopathy and develop new treatments and therapies.

Monetization Strategies for Sustainable Healthcare

Our monetization strategies are designed to be flexible and adaptable to the needs of healthcare providers and organizations. We plan to offer a subscription-based model that allows healthcare providers to access our diagnostic system for a flat fee, as well as a pay-per-use model that enables doctors to use our system on a per-patient basis. Additionally, we will partner with healthcare organizations to implement value-based care models that incentivize early detection and prevention of diabetic retinopathy, aligning our revenue streams with the value we bring to patients and healthcare systems. We will also offer data analytics services to healthcare organizations and pharmaceutical companies, providing valuable insights that can inform treatment plans and research initiatives, and generating additional revenue streams for our business.

Potential Future Applications and Expansion

As our diabetic retinopathy detection system continues to evolve, we vision expanding its capabilities to improve predictions and patient outcomes. One potential future direction is to enhance the tool's ability to detect and predict specific abnormalities associated with diabetic retinopathy, such as microaneurysms, soft exudates, hard exudates, and hemorrhages. By incorporating this level of detail, our system could provide even more accurate and nuanced predictions of diabetic retinopathy, enabling healthcare providers to tailor treatment plans to individual patients' needs. Furthermore, this information could be used to develop personalized risk profiles and monitoring plans, allowing for earlier interventions and more effective prevention of vision loss. By continually refining and expanding our system's capabilities, we aim to stay at the forefront of diabetic retinopathy diagnosis and treatment, ultimately improving the lives of millions of people worldwide.

Data privacy concerns

As we develop and deploy our AI-powered diabetic retinopathy detection system, we recognize the importance of protecting sensitive patient data. We are committed to ensuring the confidentiality, integrity, and availability of all patient information, and will implement robust data protection measures to safeguard against unauthorized access, use, or disclosure. Our system will be designed to comply with relevant data protection regulations, including the GDPR and Greek Law 4624/2019 on the Protection of Personal Data, and will employ state-of-the-art encryption, secure data storage, and access controls to protect patient data. We will also establish clear policies and procedures for data handling, sharing, and retention, and will provide transparency to patients and healthcare providers about how their data is being used and protected. By prioritizing data privacy and security, we aim to build trust with our users and stakeholders, and to ensure that our system is used in a way that respects the rights and dignity of patients.

C. Implementation

Data Collection

Our model's development and evaluation relied on two prominent datasets in the field of diabetic retinopathy research: the APTOS 2019 and Messidor-2 datasets. These datasets provided a diverse collection of retinal images for building a robust deep learning model capable of detecting diabetic retinopathy (DR) at various stages.

The APTOS 2019 dataset was sourced from the Asia Pacific Tele-Ophthalmology Society (APTOS) 2019 Blindness Detection competition. This dataset comprises 3,662 high-resolution fundus images captured under different conditions. Each image in the dataset is annotated with a severity grade ranging from 0 to 4, representing the following stages of diabetic retinopathy:

- **0:** **No** diabetic retinopathy
- **1:** **Mild** diabetic retinopathy
- **2:** **Moderate** diabetic retinopathy
- **3:** **Severe** diabetic retinopathy
- **4:** **Proliferative** diabetic retinopathy

The images in the APTOS 2019 dataset vary in resolution and quality, reflecting real-world clinical conditions, which makes them suitable for training our model in our effort to build the ability of generalizing well to diverse inputs. The annotations provided by ophthalmologists ensure that the dataset captures different DR stages.

To ensure the robustness and reliability of our deep learning model for diabetic retinopathy detection, we employed a two-stage splitting strategy to divide the APTOS 2019 dataset into training, validation, and test sets. Initially, we split the dataset into two subsets, with 70% of the images allocated for training and 30% reserved for a temporary set that would be further divided into validation and test sets. This initial split allowed us to allocate enough images for training, which is crucial for the model to learn the complex patterns and relationships in the data. A large training set enables the model to capture the variability in the data, reducing the risk of overfitting, which occurs when a model is too complex and learns the noise in the training data, resulting in poor performance on new, unseen data. On the other hand, a model that is too simple may suffer from underfitting, where it fails to capture the underlying patterns in the data, leading to poor performance on both the training and test sets. Good generalization, which is the ability of a model to perform well on new, unseen data, is achieved when a model strikes a balance between complexity and simplicity, capturing the underlying patterns in the data without overfitting or underfitting. With enough training data, the model can learn to recognize the underlying structures and features that are indicative of diabetic retinopathy, rather than simply memorizing the training examples. We then split the temporary set into two equal subsets, with 50% allocated for validation and 50% for testing. This secondary split enabled us to evaluate the model's performance on a separate validation set during training, allowing for hyperparameter tuning and model selection, while also maintaining a completely independent test set for final evaluation.

By using stratification based on the diagnosis column, we ensured that the same proportion of each class was maintained across all three sets, preventing class imbalance issues and ensuring that the model was trained and evaluated on a representative sample of the data.

This careful splitting strategy allowed us to optimize our model's performance, while also providing a realistic estimate of its generalizability to new, unseen data. Good generalization is critical in medical image analysis, as it ensures that the model can accurately diagnose diabetic retinopathy in new patients, rather than simply recognizing patterns in the training data. By prioritizing generalization, we can increase the model's reliability and trustworthiness, ultimately leading to better patient outcomes.

Summary Statistics	
Count	3,662
Mean	1.1
Standard Deviation	1.3
Minimum	0
25%	0
50%	1
75%	2
Maximum	4
Median	1

Figure 17 - Descriptive statistics measures of diagnostic labels

Dataset Overview

The dataset under consideration consists of both image data and associated diagnostic labels, providing a source of information for building predictive models. To ensure a thorough understanding of the dataset's structure and inherent characteristics, we begin with an Exploratory Data Analysis (EDA). This step is critical not only for identifying potential issues such as class imbalances, missing values, or outliers but also for uncovering patterns that can inform feature engineering and model selection.

We will explore the dataset using various statistical metrics, including mean, standard deviation, and median. These metrics offer insights into the distribution and variability of the data, helping us to identify any irregularities that might affect model performance. Additionally, we will utilize visualizations such as class distribution plots, histograms of pixel intensities, and blurriness distributions to visually assess the quality and characteristics of the data. These plots provide a more intuitive understanding of the dataset and highlight any patterns or anomalies that could impact the effectiveness of our machine learning models.

The EDA both facilitates the initial data exploration and sets the stage for informed decision-making in the data preprocessing and modeling phases. By understanding the data's nuances, we can tailor our approach to feature extraction and model development, ultimately enhancing the accuracy and robustness of the final solution.

Table 1 presents an extended summary of the initial dataset, highlighting essential statistical metrics that will guide our subsequent analyses and visual explorations.

- **Dataset Size:** The dataset comprises 3,662 observations across various stages of diabetic retinopathy. While this is a relatively adequate sample size for machine learning, careful data augmentation and validation strategies will be essential to avoid challenges in training complex models and to maximize the utility of the dataset.
- **Distribution of Severity Levels:** The mean value of 1.1 and the median of 1 indicate a skew towards the lower end of the severity scale, with a significant portion of images labeled as no or mild retinopathy (stages 0 and 1). This class imbalance may lead to a bias in model predictions, favoring less severe stages. Addressing this with effective data augmentation techniques will help the model learn from underrepresented severe stages.
- **Severity Spread:** The standard deviation of 1.3 suggests a moderate spread in severity levels, with most samples concentrated in the lower stages. To ensure that the model can accurately detect more severe stages, applying targeted data augmentation strategies will be crucial for generating a more balanced representation of the severity spectrum.

- **Range of Severity:** The dataset includes the full range of diabetic retinopathy severity, from no apparent disease (minimum value 0) to severe proliferative retinopathy (maximum value 4). This diversity is beneficial for training a comprehensive model. However, enhancing this representation through preprocessing and augmentation will be necessary to prevent the model from overfitting to the more prevalent mild cases.
- **Class Imbalance**: With 50% of the data concentrated in stages 0 and 1 and only 25% in stages 2 and higher, there is an imbalance. This could hinder the model's ability to generalize to more severe cases. Focusing on advanced data augmentation and preprocessing techniques will be key to mitigating this imbalance and ensuring the model's robustness across all severity levels.

Explanatory Data Analysis

To better understand the dataset's composition, we will visualize the distribution of diabetic retinopathy stages. This plot will show the frequency of each class, highlighting any imbalances. Identifying such imbalances is essential for developing effective strategies, such as data augmentation and resampling, to ensure robust model performance across all severity levels.

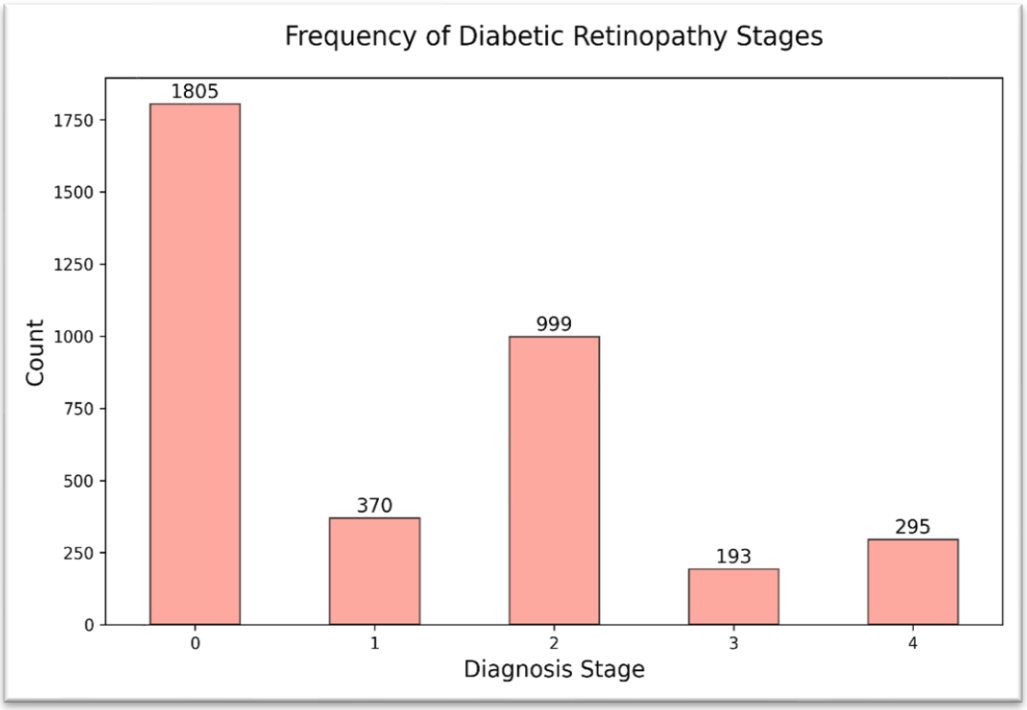


Figure 18 - Descriptive statistics measures of diagnostic labels

The class distribution plot confirms (see Figure 18), as expected, the inferences from the summary statistics table and reveals an imbalance, with most images labeled as class 0 (no diabetic retinopathy), comprising nearly half of the dataset. Classes 1, 3, and 4 are underrepresented, which could lead to a model biased towards predicting no or mild retinopathy. This imbalance poses a challenge for the model's ability to accurately detect and classify the more severe stages. Class 2, while slightly better represented, still requires attention to ensure balanced learning. To address these disparities and enhance model performance across all stages, targeted data augmentation and preprocessing techniques will be essential.

In addition to understanding the class distribution of diabetic retinopathy stages, it is crucial to analyze the dimensions of the retinal images themselves. The quality and consistency of image dimensions can significantly impact the performance of machine learning models. This plot displays the distribution of image heights and widths, providing insights into the variability of the dataset's resolution.

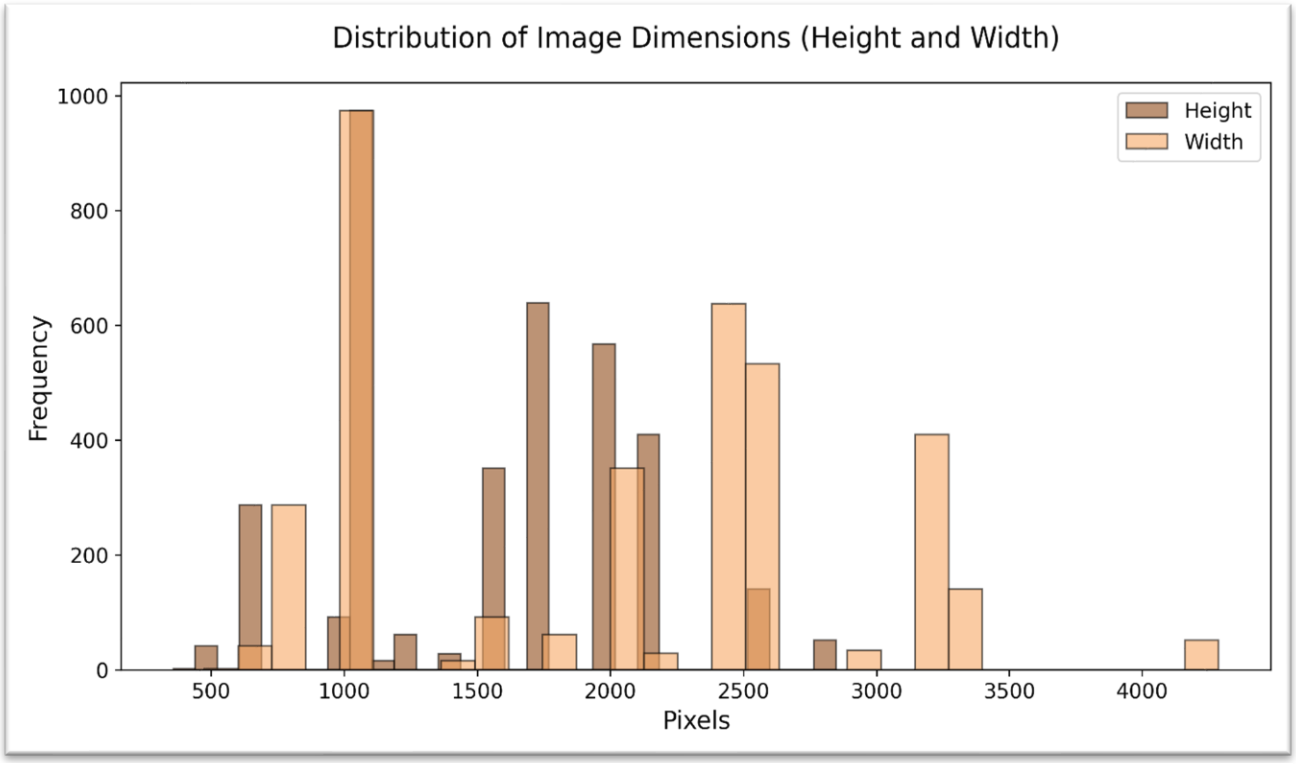


Figure 19 - Distribution of Image Heights and Widths in the Dataset.

The plot in Figure 19 reveals distinct peaks in both height and width distributions, indicating the presence of a few standard image sizes. The most common image height and width are around 1000 pixels, suggesting that a significant portion of the dataset shares similar dimensions. However, there is also a noticeable spread, particularly in the width distribution, with images ranging from around 500 to over 4000 pixels.

This variability in image dimensions suggests that preprocessing steps, such as resizing to a common resolution, will be necessary to ensure uniformity in model input. The peaks in the distribution indicate that resizing to one of the more common dimensions could minimize the loss of detail. By standardizing image sizes, we can improve the model's performance and consistency, as it will be trained on data with uniform characteristics. Another critical factor influencing model performance is the quality of the images, specifically their sharpness. Blurry images can obscure important features needed for accurate classification. Therefore, it is essential to evaluate the distribution of image blurriness across the dataset to identify potential issues that could impact model training. The next plot displays the distribution of image blurriness, measured as the variance of the Laplacian, which quantifies the sharpness of the images. Analyzing this distribution will help determine whether additional preprocessing, such as image sharpening or exclusion of excessively blurry images, is needed.

The distribution of image blurriness reveals that a significant number of images have low variance values, indicating a considerable level of blurriness. Specifically, many images fall below a variance of 20, suggesting that these images may lack sufficient detail and sharpness. This could potentially obscure critical features needed for accurate classification of diabetic retinopathy stages.

Images with blurriness values below 20 might require additional preprocessing, such as image sharpening, to enhance their clarity. On the other hand, a good portion of the dataset shows variance values between 20 and 60, which indicates acceptable sharpness for feature extraction. Ensuring that most images in the dataset fall within this range will likely improve the model's ability to detect and classify different stages of diabetic retinopathy accurately.

After analyzing the sharpness of the images, it is important to assess the color characteristics across different classes of diabetic retinopathy. The distribution of pixel intensities in the red, green, and blue channels can provide valuable insights into the overall color composition and any variations between classes. Understanding the color distribution helps identify patterns or anomalies that may aid in distinguishing between different severity levels of the disease. The following plots show the average color distribution for each class, helping to highlight any unique color characteristics that may contribute to the model's ability to accurately classify the images.

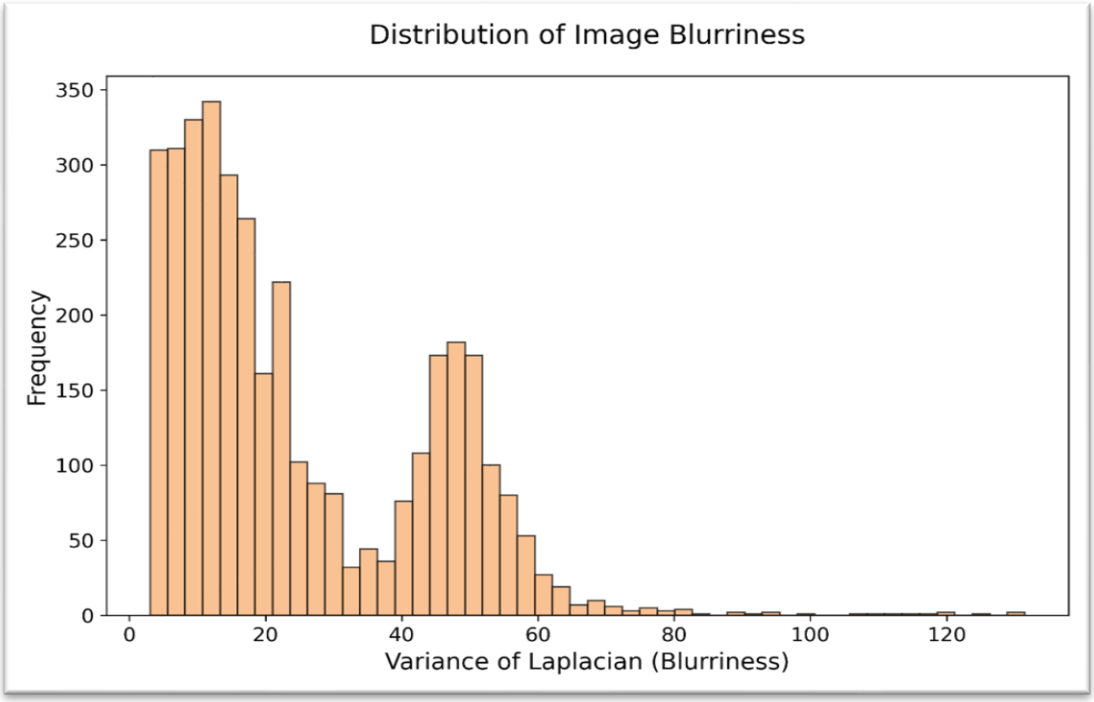


Figure 20 - Distribution of image blurriness

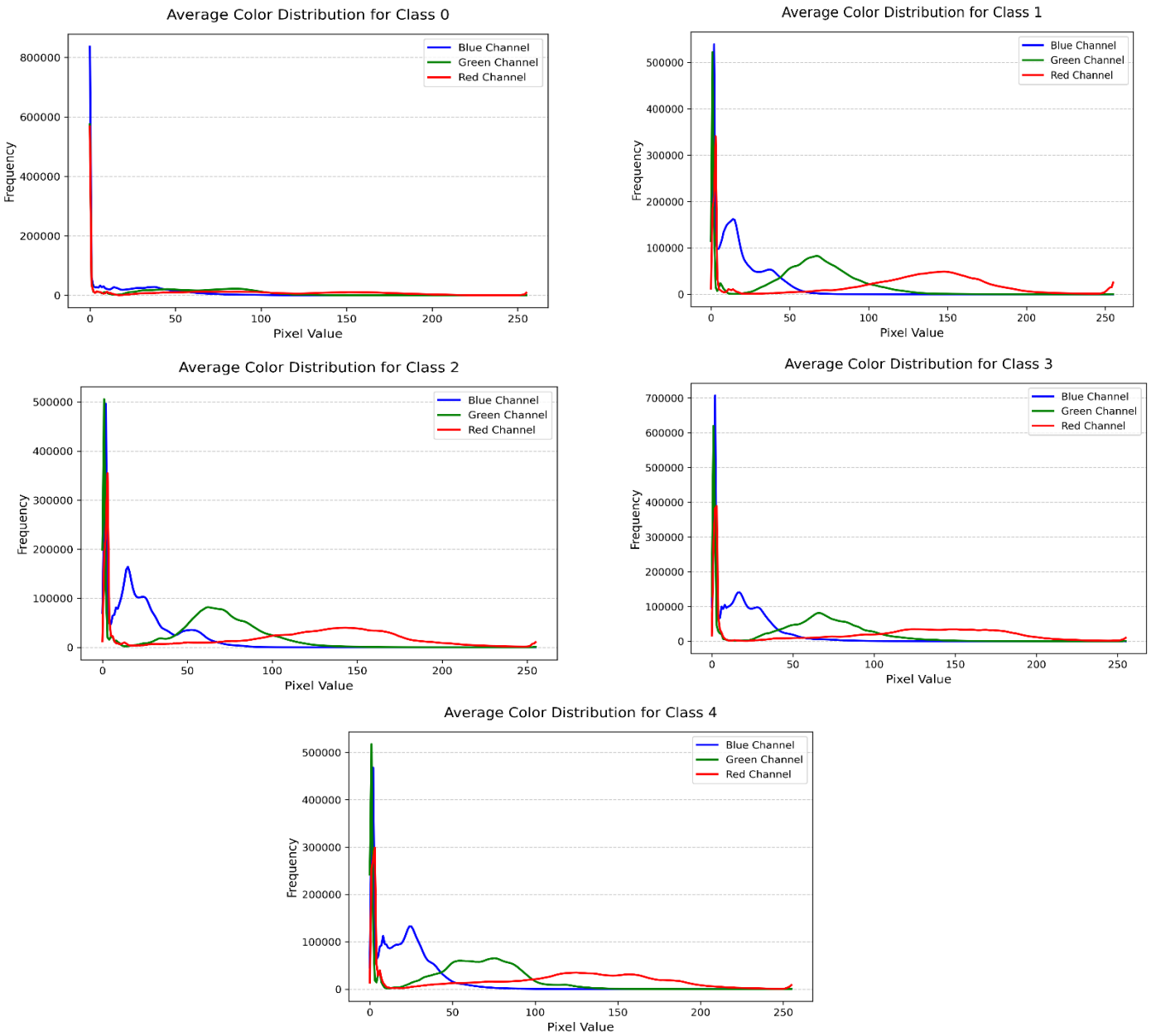


Figure 21 - Color Distribution over Classes of images

The color distribution (see Figure 21) shows that Class 0 (no diabetic retinopathy) has a distinct color profile with sharp peaks at low pixel values across all channels, indicating darker and more uniform images typical of healthy retinas. In contrast, Classes 1 to 4, representing various stages of diabetic retinopathy, exhibit broader and more variable distributions in the red, green, and blue channels. This suggests increased color variability may be due to retinal abnormalities. While these classes differ slightly, they share a common pattern of increased intensity and spread compared to Class 0, reflecting the progression of the disease and the presence of features like hemorrhages and exudates. This indicates that Class 0 is visually distinct, while Classes 1 to 4 show similar color characteristics, aligning with the severity of diabetic retinopathy.

Following the analysis of color distribution, we examine the overall pixel intensity of the retinal images in grayscale. Analyzing the histogram of pixel intensities can reveal patterns of light and dark regions within the images, offering insights into the contrast and distribution of features like lesions or hemorrhages. This can help assess the quality and uniformity of the dataset, which is crucial for ensuring consistent model performance.

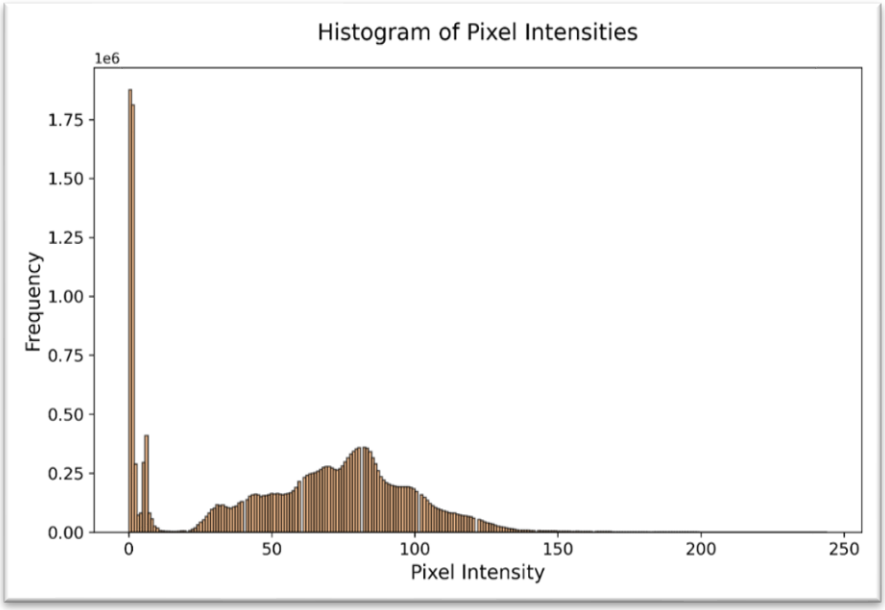


Figure 22 - Histogram of pixel intensities

The histogram of pixel intensities (see Figure 22) reveals a clear pattern with two prominent features. First, there is a substantial peak at the lowest intensity values up to a **pixel value of 7**, indicating a significant portion of the images contain large dark areas. This likely represents the background or peripheral regions of the retinal images, which are uniformly dark across the dataset.

Second, there is a broader peak around the mid-range pixel values (approximately 50 to 100), which corresponds to the primary features of the retina, such as blood vessels, optic disc, and other anatomical structures. The relatively low frequency of high-intensity values suggests that bright lesions or exudates, typical in advanced stages of diabetic retinopathy, are less prevalent.

This distribution highlights the need for preprocessing, such as contrast cropping, enhancement to ensure that all features, especially those in the mid to high intensity range, are adequately represented and can be effectively utilized by the model. Addressing these variations in intensity will help in creating a more balanced and informative input for training, potentially improving the model's ability to detect and classify different stages of diabetic retinopathy.

Image Pre-processing

The preprocessing phase is a critical component of the diabetic retinopathy detection pipeline, as it significantly enhances the quality of the retinal images, allowing for more effective feature extraction and classification. The primary objective of this phase is to ensure that the retinal images are accurately **cropped, resized, and filtered** to highlight the important regions of interest while minimizing noise and irrelevant information.

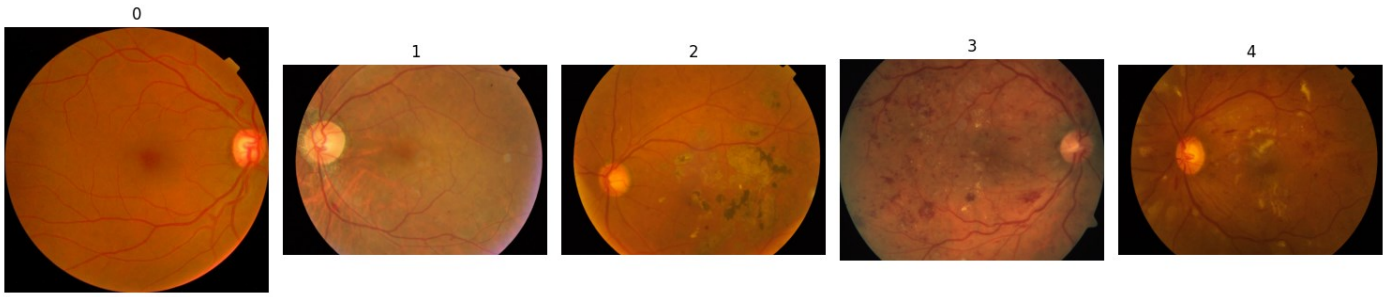


Figure 23 - Initial phase of the input images

Image Cropping

Retinal images often contain non-informative black or dark regions that can reduce the model's effectiveness. To address this, a customized cropping function is employed to remove irrelevant sections of the image. This function operates on both grayscale and RGB images, using a predefined tolerance level to generate a mask that filters out pixels below the tolerance. The image is then cropped on the mask, ensuring that only the relevant sections are retained.

1. **Grayscale Image Processing:** For images in grayscale format, the cropping function applies a simple yet effective masking technique. It creates a binary mask by comparing each pixel value to the predefined tolerance level. Pixels with values greater than tolerance are considered relevant and are retained, while those with values below the tolerance are discarded. The resulting mask is then used to crop the image, effectively removing any non-informative regions.
2. **RGB Image Processing:** In the case of RGB images, the cropping function first converts the image to grayscale to simplify the identification of irrelevant regions. This is done using a color conversion technique that transforms the RGB image into a grayscale representation. The same masking technique used for grayscale images is then applied to the grayscale version of the RGB image. The resulting mask is used to crop each color channel of the original RGB image, ensuring that only the relevant sections are retained.

To prevent the loss of important data, the function checks if the entire image is too dark, indicating that it does not contain useful information. If this is the case, the function reverts to the original image, avoiding any potential loss of critical information. Otherwise, the cropped color channels are stacked together to form the final preprocessed image.

By applying this customized cropping function, the algorithm can effectively remove non-informative regions from retinal images, improving the quality of the input data and enhancing the model's ability to detect diabetic retinopathy.

Image Resizing and Standardization

To maintain consistency across the dataset, each image is resized to a fixed dimension. This ensures uniformity in the input data, which is essential for deep learning models that require standardized input sizes. Resizing not only aids in computational efficiency but also guarantees that each image is represented on the same scale, allowing the model to better generalize across varying image resolutions.

Image Enhancement for Improved Clarity

Following the cropping and resizing of the images, a critical step is taken to enhance the clarity of the retinal images. This is essential because retinal images can often be noisy or lack sufficient contrast, which can hinder the detection of early-stage diabetic retinopathy. To address this challenge, a sophisticated sharpening technique is applied to the images.

- **Noise Reduction through Gaussian Blurring**: The first step in the image enhancement process is to reduce the noise present in the image. This is achieved by applying a Gaussian blur to the image. The Gaussian blur is a widely used image processing technique that reduces the noise in an image by averaging the pixel values with their neighboring pixels. In this case, the Gaussian blur is applied with a kernel size of (0,0) and a standard deviation of IMG_SIZE/10. This effectively reduces the noise in the image, but also has the unintended consequence of reducing the sharpness of key features.
- **Contrast Enhancement through Sharpening**: To counteract the loss of sharpness caused by the Gaussian blur, a weighted sum of the original and blurred images is calculated. This sharpening technique is designed to increase the contrast of fine details in the image, such as blood vessels and microaneurysms, which are key indicators of diabetic retinopathy. The weighted sum is calculated using the cv2.addWeighted function, which takes the original image, the blurred image, and a set of weights as input. In this case, the weights are set to 4 for the original image and -4 for the blurred image, with a constant value of 128 added to the result. This has the effect of amplifying the fine details in the image, while reducing the impact of noise.

The result of the image (see Figure 22) enhancement process is a clearer, more focused image that enables the model to detect subtle patterns that may not have been visible in the original input. The enhanced image has improved contrast and reduced noise, making it easier for the model to identify the key features that are indicative of diabetic retinopathy. By applying this sophisticated image enhancement technique, the accuracy of the model is significantly improved, allowing for more effective detection and diagnosis of diabetic retinopathy.

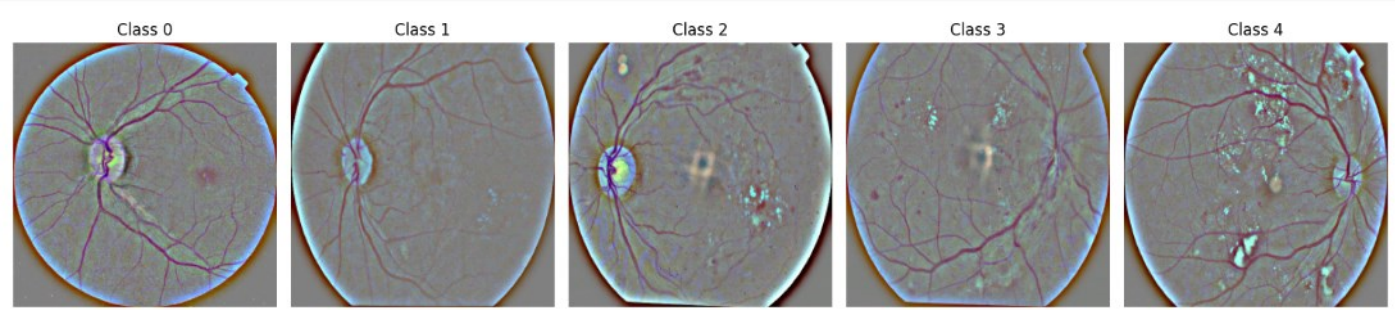


Figure 24 - Final phase of the input preprocessed images

Methodology

A detailed implementation Roadmap

- **Data Preprocessing** We employed a preprocessing pipeline to prepare the retinal images for training. The primary objective of this phase was to ensure that the images were accurately cropped, resized, and filtered to highlight the important regions of interest while minimizing noise and irrelevant information.
- **Data Augmentation** To further augment the training set and improve the model's robustness, we applied the following random transformations to each image in the training set: horizontal flipping with a 50% probability, vertical flipping with a 50% probability, brightness adjustment by a factor of up to 20%, and contrast adjustment by a factor of between 0.8 and 1.2.
- **Model Selection** Our strategy was to try three different models, namely CNN, InceptionV3, and DenseNet121, to choose the best one for our task.
- **Model Evaluation** To evaluate the performance of each model, we used the Quadratic Weighted Kappa (QWK) score, which is a metric that measures the agreement between predicted and true labels while adjusting for the possibility of agreement occurring by chance (elaboration on the metric follows [1]).
- **Callback Implementation** We implemented a custom callback to compute the QWK score at the end of each training epoch. This callback played a crucial role in the training process by providing a more advanced metric for assessing model performance in ordinal classification tasks, where the order of the classes is important. For the InceptionV3 and DenseNet121 models, we also implemented early stopping to prevent overfitting, where the training process was stopped if the validation loss did not improve for 3 consecutive iterations. By prioritizing QWK, the model not only improves classification accuracy but also ensures that the most severe misclassifications, which carry significant real-world implications, are reduced.
- **Model Saving** At the end of each epoch, the QWK score is computed on the validation data by comparing the predicted class labels with the true labels. If the QWK score improved compared to previous epochs, the model was saved automatically, ensuring that the best-performing model, based on the QWK score, was preserved for future use.
- **Prioritizing QWK** In medical prediction tasks, relying on the QWK metric is often more appropriate than simple accuracy, as it accounts for the ordered nature of categories and assigns greater penalties for predictions that deviate significantly from the true class. By prioritizing QWK, the model not only improves classification accuracy but also ensures that the most severe misclassifications, which carry significant real-world implications, are reduced.

Quadratic Weighted Kappa Formula [1]

The QWK metric is computed as follows:

$$\boldsymbol{QWK = 1 - \frac{\sum_{i,j} W_{ij} \cdot O_{ij}}{\sum_{i,j} W_{ij} \cdot E_{ij}}}$$

Where:

- W_{ij} is the weight matrix that penalizes the disagreement between classes i and j, typically defined as: $W_{ij} = \frac{(i-j)^2}{(N-1)^2}$

Here, N is the number of ordinal categories, and i,j represents the class labels.

This matrix assigns penalties based on the distance between the true class and the predicted class. The larger the difference between i and j the more misclassification is penalized. This helps QWK account for the severity of misclassification in ordinal tasks.

- O_{ij} is the observed agreement matrix, which represents the number of times class i was predicted as class j.
- E_{ij} is the expected agreement matrix, which is the number of times class i is expected to be predicted as class j, assuming predictions were random but respect the label distributions.

The ratio of the observed disagreement to the expected disagreement is subtracted from 1 to give a score that ranges from **-1** (complete disagreement) to **1** (perfect agreement). A score of **0** indicates agreement is no better than random guessing, while values close to **1** indicate strong agreement between the true and predicted class distributions, especially considering the ordinal nature of the classification.

Convolutional Neural Network

Architecture of CNN

Convolutional Neural Networks (CNNs) are a type of neural network designed to process data with grid-like topology, such as images. CNN architecture is inspired by the structure and function of the visual cortex in the human brain, which is responsible for processing visual information. A CNN typically consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers:

- **Convolutional layers** are the core building blocks of a CNN, applying filters to small regions of the input image to detect local patterns and features. Each filter computes a dot product between the filter and the local region of the image, producing a feature map that represents the presence of the feature at different locations in the image.
- **Pooling layers** downsample the feature maps produced by the convolutional layers to reduce the spatial dimensions and retain only the most important information. There are two types of pooling layers: max pooling, which takes the maximum value across each window, and average pooling, which takes the average value across each window.
- **Fully connected layers** are used to classify the output of the convolutional and pooling layers into a fixed number of categories.

The input image is fed into the CNN, which is typically a 3D array of pixel values (width, height, and color channels). The convolutional layer applies filters to small regions of the input image, producing feature maps that represent the presence of local patterns and features. The output of the convolutional layer is then passed through an activation function, such as ReLU (Rectified Linear Unit), which introduces non-linearity into the model. ReLU is a simple activation function that maps all negative values to zero and all positive values to the same value. This means that ReLU only allows the positive values to pass through, while suppressing the negative values. By doing so, ReLU helps to:

- Introduce non-linearity into the model, allowing it to learn more complex and abstract representations of the input data.
- Increase the sparsity of the feature maps, which can help to reduce the number of parameters and improve the model's computational efficiency.
- Improve the model's ability to generalize to new, unseen data.

The output of the convolutional layer is then downsampled using a pooling layer to reduce the spatial dimensions and retain only the most important information. The output of the pooling layer is flattened into a 1D array to prepare it for the fully connected layers. The flattened output is fed into one or more fully connected layers, which classify the output into a fixed number of categories. The final output of the CNN is the predicted class label or probability distribution over all classes.

CNNs are prone to overfitting, especially when dealing with large and complex datasets. To prevent overfitting, several techniques are used, including dropout, data augmentation, and regularization. Data augmentation is a technique that involves artificially increasing the size of the training dataset by applying random transformations to the input images. This can include:

- **Rotation**: rotating the image by a certain angle to create new training examples
- **Scaling**: scaling the image up or down to create new training examples

- **Flipping**: flipping the image horizontally or vertically to create new training examples
- **Color jittering**: randomly changing the brightness, contrast, or color balance of the image to create new training examples

By applying these transformations, data augmentation can help to:

- Increase the size of the training dataset, which can improve the model's ability to generalize to new, unseen data
- Reduce overfitting by providing the model with a more diverse range of training examples
- Improve the model's robustness to different types of noise and variations in the input data

Regularization is another technique that can be used to prevent overfitting. Regularization involves adding a penalty term to the loss function to discourage the model from overfitting. This penalty term is typically based on the magnitude of the model's weights and is designed to encourage the model to use smaller weights and simpler solutions. By doing so, regularization can help to:

- Reduce overfitting by preventing the model from using complex solutions that are not supported by the training data
- Improve the model's ability to generalize
- Reduce the risk of model instability and exploding gradients

There are several types of regularization techniques, including:

- **L1 regularization** adds a penalty term to the loss function based on the absolute value of the model's weights
- **L2 regularization** adds a penalty term to the loss function based on the square of the model's weights
- **Dropout regularization**: randomly drops out neurons during training to prevent the model from relying too heavily on any single neuron.

CNNs are typically trained using a stochastic gradient descent (SGD) algorithm, which is a type of optimization algorithm that minimizes the difference between the predicted probabilities and the true labels. The SGD algorithm is a popular choice for training CNNs because it is efficient and effective, especially when dealing with large datasets. The training process involves a forward pass, where the input image is passed through CNN to produce a predicted output. During the forward pass, the input image is propagated through the network, layer by layer, to produce a predicted output. This process involves:

- **Convolutional layers:** applying filters to the input image to produce feature maps
- **Activation functions**: applying non-linear transformations to the feature maps to introduce non-linearity into the model
- **Pooling layers**: downsampling the feature maps to reduce the spatial dimensions and retain only the most important information
- **Fully connected layers**: classifying the output into a fixed number of categories

The softmax function is a type of activation function that is commonly used in the output layer of a neural network when the task is a multi-class classification problem. It takes the output of the fully connected layer and maps it to a probability distribution over all classes.

The softmax function is defined as:

$$\boldsymbol{softmax}(\boldsymbol{x}) = \frac{\mathbf{exp}(\boldsymbol{x})}{\boldsymbol{\Sigma}\,\mathbf{exp}(\boldsymbol{x})}$$

where x is the input to the softmax function, exp is the exponential function, and Σ is the summation over all classes.

The softmax function ensures that the output of the network is a valid probability distribution, meaning that the output values are all between 0 and 1 and sum up to 1. This is useful for multi-class classification problems, where the network needs to predict the probability of each class given the input data. The predicted output is then compared to the true label using a loss function, such as cross-entropy loss. The cross-entropy loss function measures the difference between the predicted probabilities and the true labels, and is defined as:

$$\boldsymbol{L}(\boldsymbol{y}, \boldsymbol{y}') = -\sum_{\mathbf{0}}^{\boldsymbol{n}} (\boldsymbol{y} \cdot \boldsymbol{log}\ (\boldsymbol{y}'))$$

where y is the true label, y' is the predicted output, n is the number of the sample and log is the natural logarithm.

The error is then propagated backwards through the network to calculate the gradients of the loss with respect to each parameter. This process is known as backpropagation, and involves:

- Calculating the error gradient of the loss function with respect to the output of each layer
- Propagating the error gradient backwards through the network, layer by layer, to calculate the gradients of the loss with respect to each parameter
- Using the gradients to update the parameters of the network

The parameters are updated using the gradients and the learning rate, which is a hyperparameter that controls the step size of each update. The learning rate is typically set to a small value, such as 0.01 or 0.001, and is adjusted during training to optimize the convergence of the algorithm.

The update rule for the parameters is typically given by:

$$\boldsymbol{w\_new} = \boldsymbol{w\_old} - \boldsymbol{\alpha} \cdot \boldsymbol{\nabla L}(\boldsymbol{w\_old})$$

where w_new is the updated parameter, w_old is the previous parameter, α is the learning rate, and ***∇L(w_old)*** *is the gradient of the loss with respect to the previous parameter.*

By iteratively updating the parameters using the gradients and the learning rate, the SGD algorithm can converge to a local minimum of the loss function, which corresponds to a good set of parameters for CNN.

In the training process of a CNN, a batch refers to a subset of the training dataset that is used to update the network's parameters. The batch size is a hyperparameter that determines the number of training examples in each batch. For example, if the batch size is 32, the network will process 32 training examples together as a single batch. An epoch, on the other hand, refers to a single pass through the entire training dataset. During each epoch, the network processes all the training examples, one batch at a time.

After processing all the batches in an epoch, the network's performance is evaluated on a validation set, which is a separate dataset that is not used for training. The validation set is used to monitor the network's performance and to prevent overfitting. The model with the best performance on the validation set is typically chosen as the final model. This is done by tracking specific evaluation metrics after each epoch and selecting the model with the best assessment. This process is known as early stopping, and it helps to prevent overfitting by stopping the training process when the network's performance on the validation set starts to degrade. The best model is then used to make predictions on unseen data.

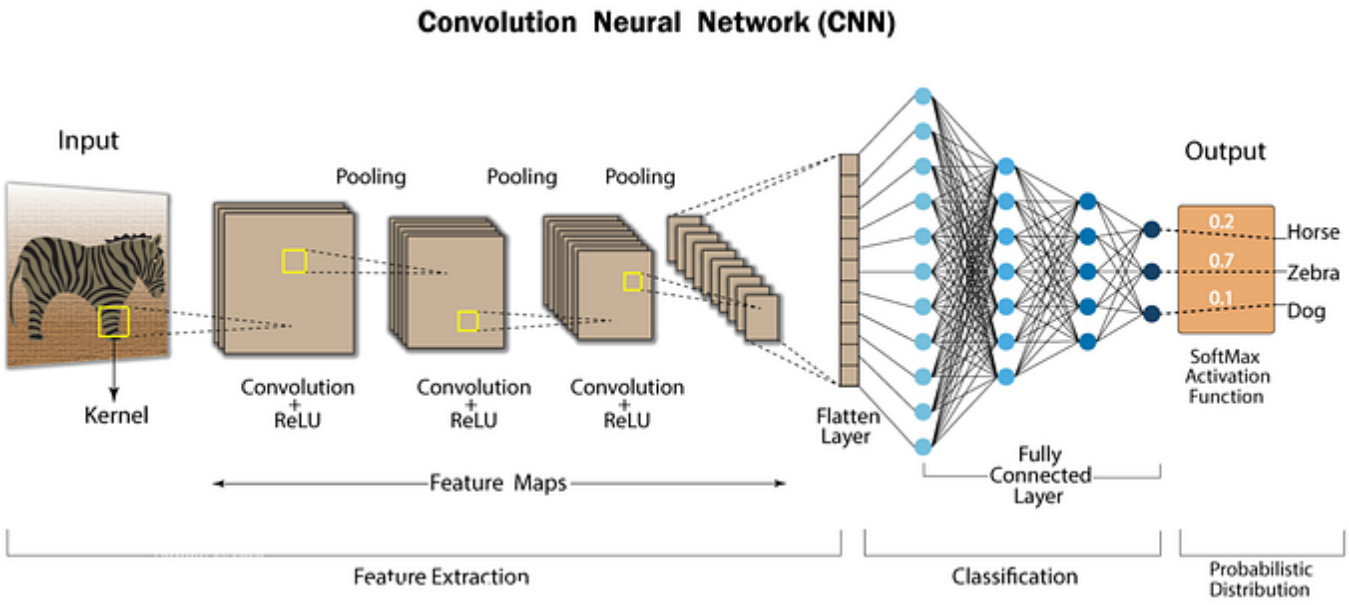


Figure 25 - Architecture of a CNN

Implementation of CNN for Image Classification

- **Load and Preprocess the Dataset**

The preprocessed images, which had undergone prior transformations to enhance their quality and consistency, were loaded from their respective directories and resized to uniform dimensions. The images were then compiled into arrays, which formed the basis of the training and validation datasets used to train and evaluate the model. By working with a systematically prepared dataset, we ensured that the model received a consistent and high-quality input, which is essential for its optimal performance during training. The dataset, consisting of training, validation, and testing sets, was preprocessed to facilitate image classification using a CNN model. To enable numerical representation, the categorical labels were converted to a one-hot encoded format, where each label was transformed into a binary vector. This conversion ensured compatibility with subsequent network operations. In parallel, image normalization was performed by scaling the pixel values to the range [0, 1] through division by the maximum pixels' value (255). This standardization step is crucial for consistent neural network performance, as it ensures that all images are on the same scale. Furthermore, boolean labels were transformed into integers (0 or 1) to align with the expected input format of the neural network.

- **Data Augmentation**

To enhance the robustness and generalization ability of CNN, data augmentation techniques were employed on the training images. Specifically, the training images were subjected to a series of random transformations, including:

- **Horizontal flipping:** Each image was randomly flipped left or right with a 50% probability. This simulates the scenario where an object may be viewed from a different angle or orientation, forcing the model to learn features that are invariant to horizontal reflections.
- **Vertical flipping:** Each image was randomly flipped up or down with a 50% probability. This simulates the scenario where an object may be viewed from a different perspective or orientation, forcing the model to learn features that are invariant to vertical reflections.
- **Brightness adjustment:** The brightness of each image was randomly adjusted by a factor of up to 20%. This simulates the scenario where an image may be taken under different lighting conditions, forcing the model to learn features that are robust to changes in brightness.
- **Contrast adjustment:** The contrast of each image was randomly adjusted by a factor of between 0.8 and 1.2. This simulates the scenario where an image may be taken under different lighting conditions or with different camera settings, forcing the model to learn features that are robust to changes in contrast.

By introducing this variability into the training data, the model is forced to learn more robust and invariant features, which in turn enhances its ability to generalize to new data.

- **Define Custom Metric**

The model's performance was evaluated using a custom metric that combines accuracy and the Kappa score. Accuracy is a widely used metric that calculates the proportion of correctly predicted labels by comparing the predicted labels with the actual labels, resulting in a value between 0 and 1, where 1 represents perfect accuracy and 0 represents complete inaccuracy. However, accuracy alone does not provide a comprehensive understanding of the model's performance, especially when dealing with multi-class classification problems. Therefore, we also employed the Kappa score, also known as quadratic weighted kappa, which measures the agreement between the predicted labels and the actual labels while considering the fact that the model is predicting multiple labels.

To implement this custom metric, we utilized a callback, which is a powerful tool in deep learning frameworks. A callback is a function that is called at specific points during the training process, allowing us to perform custom actions, such as monitoring metrics, saving models, or adjusting hyperparameters. In our case, we created a custom callback, which calculates the Kappa score at the end of each epoch on the validation set. The Kappa score is then stored in a list and tracked in the logs, allowing us to monitor its progress during training. Additionally, the callback saves the model with the best Kappa score achieved during training. The Kappa score is calculated using the quadratic weighted Kappa, which is suitable for multi-class classification problems.

- **Modify the Pre-trained Model**

To tailor the pre-trained model to the specific needs of the task, we designed a customized CNN architecture that leverages the strengths of convolutional and pooling layers. The convolutional layers, with 32 filters and a kernel size of 3, are responsible for capturing spatial hierarchies in the images, allowing the model to extract relevant features at multiple scales.

The subsequent max-pooling layer, with a pool size of 2, reduces the dimensionality of the feature maps, enhancing computational efficiency and mitigating the risk of overfitting. To further combat overfitting, we strategically placed a dropout layer with a dropout rate of 0.2, which randomly omits a proportion of the neurons during training. This encourages the model to develop redundant pathways, improving its generalization capabilities and reducing its reliance on any individual neuron. The output of the convolutional and pooling layers is then flattened and fed into a dense layer with 32 units and ReLU activation, allowing the model to learn complex representations of the input data.

- **Add Output Layer**

The final layer of the model is a dense layer with 5 units and softmax activation, which is specifically designed for multi-class classification tasks. This layer takes the output from the previous layers and produces a discrete probability distribution across the 5 predefined labels, where each class probability is derived and represents the model's prediction for each class.

The softmax activation function is particularly well-suited for this task, as it ensures that the output probabilities are normalized and sum to 1, allowing for direct interpretation as a probability distribution. By using softmax activation, the model can effectively handle the multi-class classification problem and provide a clear indication of its prediction for each class. The output layer is then used to define the complete model, specifying the input and output layers. This setup enables the model to take in input images and produce a probability distribution over the 5 classes, facilitating direct classification and evaluation of the model's performance.

- **Compile the model**

The CNN was compiled using the categorical cross-entropy loss function, which is specifically designed for multi-class classification problems where the labels are one-hot encoded. This loss function measures the difference between the model's predictions and the actual labels, providing a quantitative measure of the model's performance. To optimize the model's weights and minimize the loss, we employed the Adam optimizer, an algorithm that adaptively adjusts the learning rate to expedite convergence. The Adam optimizer uses a combination of two moment estimates, namely the first moment (mean) and the second moment (variance), to adapt the learning rate for each parameter individually. This allows the optimizer to adjust the learning rate based on the magnitude of the gradient, which helps to stabilize the training process and prevent overshooting. The Adam optimizer was configured with a learning rate of 0.001, which was chosen to balance the trade-off between convergence speed and stability. Additionally, we set the beta1 and beta2 hyperparameters to their default values of 0.9 and 0.999, respectively, which control the decay rates of the first and second moment estimates. By using these default values, we allowed the optimizer to adapt the learning rate in a way that balances the need for rapid convergence with the need for stability. Furthermore, we tracked the model's accuracy during training, providing a complementary metric to evaluate its performance. By compiling the model with this configuration, we ensured that it would effectively learn from the training data and accurately adjust its weights to minimize prediction errors.

Evaluation of Convolutional Neural Network

In the model evaluation phase, we shift our focus from model training to analyzing the model's performance over time, using learning curves to visualize key metrics such as accuracy, loss, and the Quadratic Weighted Kappa score across training epochs. These curves allow us to compare the model's performance on both the training and validation datasets, offering valuable insights into its learning behavior and generalization capabilities. By observing these curves, we can infer whether the model is underfitting (performing poorly on both training and validation data), overfitting (performing well on training data but poorly on validation data), or achieving a good balance between the two. A consistent convergence of training and validation curves suggests that the model has learned the underlying patterns effectively without overfitting. This diagnostic tool is crucial for guiding adjustments to model architecture, hyperparameters, and training strategies, ensuring optimal performance on the unseen data.

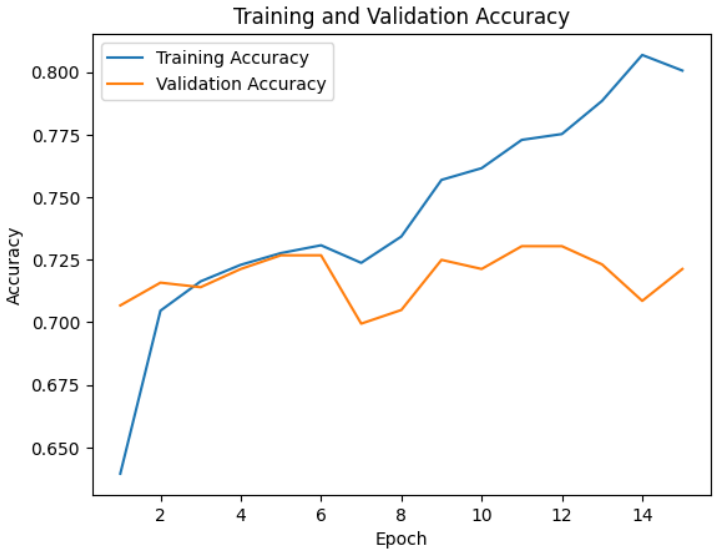


Figure 26 – Training and Validation Accuracy of CNN

Figure 26 illustrates the evolution of accuracy during epochs in training and validation sets:

- Initial Learning: The CNN model's training accuracy improves steadily during the initial epochs, showing that the model is effectively learning from the training data. The validation accuracy also shows improvement initially, indicating that the model is **generalizing well up to a certain point**.
- Stabilization and Fluctuation: The validation accuracy reaches a peak around epoch 6, though **fluctuates around 72%** during the whole training process, suggesting that the model is struggling to achieve further improvement in generalization beyond this point.
- Training vs. Validation Accuracy Gap: While the training accuracy continues to rise, the validation accuracy plateaus and fluctuates, indicating that the **model might be overfitting slightly**. However, the gap is not too wide, suggesting that the overfitting is not severe but rather indicative of the model reaching its maximum capability.

Figure 27 – Loss during epochs on training and validation set of CNN

Figure 27 illustrates the evolution of loss during epochs in training and validation set:

- Initial Loss Reduction: Training loss drops significantly during the first few epochs, indicating that the model is effectively minimizing the error during training, opposed to validation loss.
- Validation Loss Behavior: After the initial decrease, the validation loss fluctuates and **peaks sharply high at the last epoch**. This behavior, along with the fluctuating validation accuracy, indicates that the model has some difficulty generalizing to the validation data, which might be due to the complexity of the task.
- Training Loss vs. Validation Loss: The training loss continues to decrease, ending at a significantly lower value than the validation loss. This difference suggests that while the model is minimizing error on the training set, it does not translate equally well to the validation set, reinforcing the idea of **overfitting**.

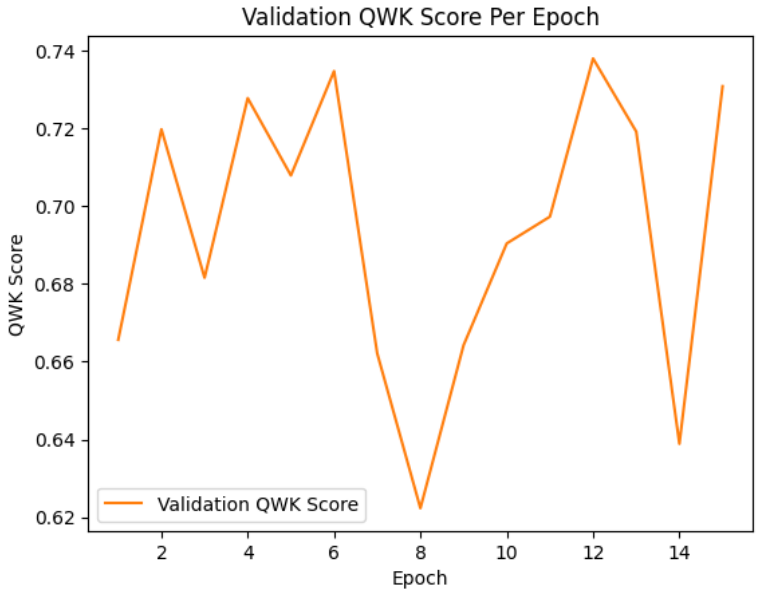


Figure 28 - Training and Validation QWK of CNN

Figure 28 illustrates the evolution of QWK during epochs:

- Fluctuating of QWK: The validation QWK score shows **variability across epochs**, with peaks and troughs indicating that the model's ability to capture the ordinal nature of the classification problem is inconsistent.
- Best Model: The best validation QWK score of 0.74 was achieved at **epoch 12**. This score reflects a **moderate level of agreement** between the predicted and actual classifications, which is an improvement over previous epochs but still highlights the limitations of the CNN model for this task.
- Comparison to Accuracy and Loss: The QWK score provides a more nuanced view of model performance, showing that while the accuracy might not change significantly, the **model's ability to capture the severity levels of diabetic retinopathy fluctuates more significantly**.

Observations Based on Model CNN Summary

Model Performance Limitation: The results suggest that while the CNN model learns effectively from the training data, its **simpler architecture might not be sufficient** to capture the complex patterns required for accurate diabetic retinopathy grading. This is reflected in the fluctuation of the validation QWK score and the plateau in validation accuracy around 72%.

Best Epoch Selection: The model saved at **epoch 12** represents the best compromise between training and validation performance, with a **QWK score of 0.74.** This indicates that continuing training beyond this point did not yield significant improvements and could **potentially lead to overfitting**.

Need for Advanced Models: These findings reinforce the **necessity of using more sophisticated models** like Inception v3 and DenseNet121, which have shown better stability and performance in handling such complex classification tasks due to their deeper architecture and more advanced feature extraction capabilities. The CNN model's **performance limitations**, including the sharp peak in validation loss at the last epoch, suggest that a more robust model is needed to handle the complexities of diabetic retinopathy grading.

Inception V3

Architecture of Inception V3

Inception V3 is a refined version of Inception architecture, pre-trained on the ImageNet dataset, which consists of over a million images spanning a thousand categories. This architecture includes several advanced features, such as factorized convolutions, efficient grid size reduction, and auxiliary classifiers, which contribute to its superior performance in complex image classification tasks. However, training a traditional CNN from scratch on a large dataset can be a resource-intensive process, requiring vast amounts of labeled data and computational resources. For instance, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) model was trained on 1.2 million images, taking 2–3 weeks across several GPUs to achieve state-of-the-art performance. This approach is often impractical for specialized applications with limited data availability, such as medical image analysis. To address these challenges, transfer learning has become a powerful alternative. Transfer learning leverages a pre-trained model, originally developed for a large-scale task, and repurposes it for a new, often smaller dataset. Compared to a traditional CNN trained from scratch on the same medical dataset, Inception V3 offers several advantages. A simple CNN model, with fewer layers and parameters, may struggle to capture intricate features from limited data, resulting in lower accuracy and overfitting. In contrast, Inception V3, through transfer learning, not only accelerates the training process but also enhances model performance by utilizing pre-trained knowledge. This approach reduces the need for extensive computational resources and large datasets, making it a practical and effective solution for specialized medical image classification tasks.

The primary design goal of the Inception V3 model was to maintain a balance between computational cost and performance, addressing the limitations of its predecessors while introducing new strategies for optimizing the convolutional layers. Several key modifications were implemented to achieve this:

- **Input Layer:** The input layer accepts the image as input, which is then processed by the subsequent layers.
- **Convolutional Layers:** Multiple convolutional layers are stacked together, responsible for extracting different features at various scales and orientations. These layers use factorized convolutions, which decompose large convolutional filters into smaller, more efficient ones. For example, a 5x5 convolution is replaced by two consecutive 3x3 convolutions. In addition to factorized convolutions, Inception V3 also uses depthwise separable convolutions, which factorize the standard convolution operation into two separate operations: a depthwise convolution and a pointwise convolution. This allows for a significant reduction in computational costs while maintaining the accuracy of the model.
- **Inception Modules**: The core building blocks of Inception V3, these modules incorporate parallel convolutional operations of different filter sizes (1x1, 3x3, 5x5) within the same layer. These parallel operations allow the network to capture features at different levels of abstraction. The Inception modules also use asymmetric convolutions, which factorize convolutions into smaller, more efficient ones. For example, a 3x3 convolution is broken down into a 1x3 convolution followed by a 3x1 convolution. There are several variations of the Inception module used in Inception V3, including Inception-A, Inception-B, Inception-C, and Inception-D. Each of these modules has a slightly different architecture, with varying numbers of convolutional layers and filter sizes. This allows the model to capture a wide range of features at different scales and orientations.

- **Max Pooling Layers:** These layers perform down-sampling of feature maps by reducing the spatial dimensions while preserving the important information. Inception V3 introduces an efficient grid size reduction technique using parallel strides and pooling operations, which reduces the spatial dimensions of the feature maps without a drastic increase in computational cost. In addition to max pooling, Inception V3 also uses reduction modules to reduce the spatial dimensions of the feature maps. These modules consist of a convolutional layer followed by a max pooling layer and are used to reduce the spatial dimensions of the feature maps while preserving the important information.
- **Auxiliary Classifiers:** Additional branches are included in the architecture where classifiers are applied to intermediate feature maps. These auxiliary classifiers help with gradient flow during training and prevent the vanishing gradient problem. These auxiliary classifiers are used primarily as regularizers, providing additional gradient information to regularize the training process and prevent overfitting. In addition to auxiliary classifiers, Inception V3 also uses dropout regularization to prevent overfitting. This involves randomly dropping out a fraction of the neurons during training, which helps to prevent the model from becoming too specialized to the training data. Inception V3 also uses weight decay (L2 regularization) to prevent overfitting, which adds a penalty term to the loss function to discourage large weights. The auxiliary classifiers in Inception V3 consist of a convolutional layer followed by a fully connected layer and a softmax activation function. These classifiers are used to provide additional gradient information to the model during training, which helps to improve the accuracy and robustness of the model.
- **Fully Connected Layers:** The feature maps are flattened and carried further via layers that are fully connected, which perform the task of classification based on the learned features. Instead of using fully connected layers for classification, Inception V3 uses global average pooling to reduce the spatial dimensions of the feature maps to a single value. This allows for a significant reduction in the number of parameters and computations required, while maintaining the accuracy of the model.

The network training process was optimized using several techniques that improved the stability, speed, and generalization of the model. Specifically, the following techniques were used:

- **RMSProp Optimizer:** type of SGD optimizer that uses a moving average of the squared gradients to normalize the update step. This helps to stabilize the training process by reducing the effect of large gradients and improving the convergence speed and being able to adapt to the changing gradient landscape during training. The learning rate schedule used in Inception V3 is a crucial component of the training process. The learning rate is initially set to a high value and then decays exponentially over time, which helps to converge the model quickly and prevent overfitting.
- **Batch Normalization:** technique that normalizes the input to each layer to have zero mean and unit variance. This helps to improve the stability and speed of training by reducing the effect of internal covariate shift, which occurs when the distribution of the input to a layer change during training. Batch normalization also helps to reduce the need for regularization and improves the generalization of the model. The batch normalization layers in Inception V3 use a momentum value of 0.99 and an epsilon value of 0.001. These values were chosen to provide a good balance between stability and speed of training.
- **Gradient Clipping:** technique that limits the magnitude of the gradients during training. This helps to prevent the gradients from becoming too large, which can cause the model to become unstable or diverge. Gradient clipping is particularly useful for deep learning models, where the gradients can become very large due to the multiple layers of computation. By clipping the gradients, the model can learn more stable and robust features.

- **Label smoothing:** technique that replaces the ground truth label distribution with a weighted mixture of the ground truth and a uniform distribution. This technique discourages the model from becoming overly confident in its predictions and improves generalization by:
    - **Reducing overfitting:** By adding noise to the labels, the model is forced to learn more robust features that are less dependent on the specific labels.
    - **Improving calibration:** Label smoothing helps to improve the calibration of the model, which is the degree to which the model's predictions match the true probabilities.
    - **Encouraging exploration:** By adding noise to the labels, the model is encouraged to explore different parts of the input space, which can help to improve generalization.

The key features of the model are:

- **Factorized Convolutions:** Decompose large convolutional filters into smaller, more efficient ones.
- **Asymmetric Convolutions:** Factorize convolutions into smaller, more efficient ones.
- **Efficient Grid Size Reduction:** Reduces the spatial dimensions of the feature maps without a drastic increase in computational cost.
- **Auxiliary Classifiers as Regularizers:** Provide additional gradient information to regularize the training process and prevent overfitting.
- **Label Smoothing Regularization:** Discourages the model from becoming overly confident in its predictions and improves generalization.
- **Improved Training Methodology:** Stabilizes the training, reduces the likelihood of overfitting, and improves convergence speed.

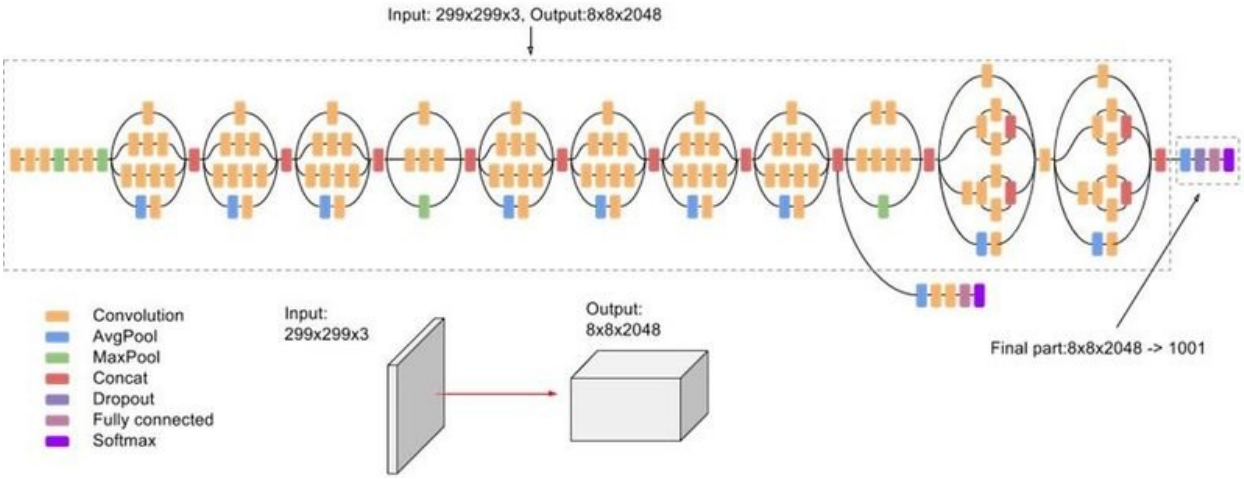


Figure 29 - Architecture of an Inception V3

Implementation of Inception V3 for Image Classification

- **Load and Preprocess the Dataset**

The preprocessed images, which had undergone prior transformations to enhance their quality and consistency, were loaded from their respective directories and resized to uniform dimensions. The images were then loaded and resized to a uniform dimension of 256x256 pixels, a prerequisite for compatibility with the InceptionV3 input specifications. This resizing process was automated to ensure efficiency and consistency. Finally, the labels associated with each image were transformed into a one-hot encoded format, converting categorical labels into a numerical form where each label is represented by a unique binary vector, facilitating their interpretation by the neural network.

- **Apply Data Augmentation**

To enhance the model's robustness and simulate a more diverse range of training scenarios, we employed data augmentation techniques. Specifically, we applied random transformations to the training images, including flipping them horizontally and vertically, adjusting their brightness, and modifying their contrast. The horizontal and vertical flips were applied randomly to simulate different orientations and perspectives, allowing the model to learn features that are invariant to these changes. The brightness adjustments, which randomly increased or decreased the brightness by up to 20%, helped the model to be more robust to variations in lighting conditions. Similarly, the contrast modifications, which randomly increased or decreased the contrast by up to 20%, enabled the model to better handle images with varying levels of contrast.

By dynamically applying these transformations as the images were fed into the model, we increased the model's generalizability and ability to recognize patterns in a wider range of images, without relying on additional data.

- **Define Custom Metric**

To evaluate the model's performance, we meticulously tracked two key metrics: accuracy and Quadratic Weighted Kappa (QWK) score. Accuracy measures the proportion of correct predictions, providing a straightforward metric of performance. However, to gain a more nuanced understanding of the model's strengths and weaknesses, we also employed the QWK score. This metric assesses the degree of agreement between predicted and actual labels, adjusting for random chance, which is particularly valuable in multiclass classification tasks where chance agreement can be a significant factor. To calculate the QWK score, we used the Cohen's kappa coefficient with quadratic weights, which provides a more robust measure of agreement between the predicted and actual labels. We calculated the QWK score after each epoch, allowing us to continuously monitor the model's predictive alignment with the observed data and track its improvement over time. Furthermore, we implemented a callback function to store the QWK scores for each epoch, enabling us to plot the model's performance trajectory and identify the best-performing model based on the highest QWK score achieved. This allowed us to save the best model and use it for future predictions, ensuring that we were using the most accurate and reliable model for our classification task.

- **Modify the Pre-trained Model**

To adapt the InceptionV3 model to our specific classification task, we modified its architecture by removing the top layers and integrating new layers tailored to our needs. We started by loading the pre-trained InceptionV3 model, which was trained on the ImageNet dataset, and excluded its top layers. This allowed us to leverage the model's learned features while avoiding the need to retrain the entire network from scratch. We then specified the input shape of the model to match our image data, which consisted of RGB images with a size of pixels (256). By doing so, we ensured that the model was compatible with our dataset and could effectively process our images. To further reduce the model's complexity and computational demand, we added a global average pooling layer after the base model output. This layer condensed the feature maps into a single vector per map, allowing the model to focus on the most important features and reducing the risk of overfitting. Additionally, we incorporated a dropout layer to randomly disable a fraction of neurons during training, promoting a more robust internal representation and preventing the model from relying too heavily on any single neuron or group of neurons.

- **Add Output Layer**

To enable the model to produce a clear and probabilistic classification output, we added a dense output layer with softmax activation at the culmination of the model. This layer was designed to compute a probability distribution across the five predefined classes for each image, providing a robust and interpretable output that underpins decision-making processes within the model. Specifically, we started by taking the output of the InceptionV3 model and applying a global average pooling layer to reduce the spatial dimensions of the feature maps and retain only the most important information. We then applied a dropout layer with a dropout rate of 0.5 to randomly disable half of the neurons during training, which helped to prevent overfitting and promote a more robust internal representation. Finally, we added the dense output layer with softmax activation, which ensured that the output was a probability distribution over the five classes, allowing for clear and confident classification decisions. By using softmax activation, we ensured that the output was normalized and interpretable, with each class having a probability value between 0 and 1, and the sum of all probabilities equal to 1.

- **Compile the Model**

To prepare the model for training, we compiled it using a combination of components that would enable effective learning from the training data. Specifically, we employed the categorical cross-entropy loss function, which is well-suited for tasks involving multiple categories, as it measures the difference between the model's predictions and the actual labels. This loss function is particularly useful in our case, as we are dealing with a multi-class classification problem. To optimize the model's parameters and minimize the loss, we utilized the Adam optimizer, which is a popular choice for its ability to adaptively adjust the learning rate and accelerate convergence. We set the learning rate to 0.0002, which allowed the model to learn from the data at a moderate pace, avoiding overshooting and undershooting. Additionally, we tracked the model's accuracy during training, which provided us with a clear understanding of its performance and allowed us to adjust as needed. By compiling the model with these components, we established a robust framework for learning from the training data and achieving optimal performance.

- **Train the Model**

The training process was executed over a specified number of epochs utilizing the augmented data to continuously refine the model's parameters against the validation dataset. The dataset was first shuffled to randomize the order of the samples and then divided into **batches**, each containing **16 images**, ensuring that the model processed manageable chunks of data at a time. This batch size allowed for efficient training, balancing memory usage and speed. Additionally, prefetching was employed to prepare the next batch of data in advance, reducing the time spent waiting for data to be loaded.

Custom callbacks were implemented to further enhance the model's performance, including the QWK callback, which computed the QWK score at the conclusion of each epoch, providing real-time feedback on the model's performance. Model checkpoints were also employed, allowing the model to save its state whenever the QWK score on the validation set improved. This ensured that the best-performing model was saved, preventing the loss of optimal weights and allowing for its future use in predictions.

- **Callbacks of the model**
  - **Custom QWK**

To ensure that we retained the best-performing version of the model during training, model checkpoints were implemented, particularly focusing on the validation QWK score.

The QWK score is a valuable metric for multi-class classification tasks, as it evaluates the agreement between the predicted and actual labels, adjusting for random chance. We employed a custom QWKCallback that calculated the QWK score after each epoch using the validation data. Whenever the QWK score improved, the model was saved. This was critical for capturing the best model state during the training process, ensuring that even if the model overfitted in later epochs or validation loss fluctuated, the model with the highest validation QWK score was preserved. The model checkpoints helped avoid the risk of losing the best model due to overfitting or degradation of performance in later epochs. This strategy allowed us to reliably retrieve the best-performing model for evaluation and deployment.

- **Early Stopping**

To prevent overfitting and ensure that the model stops training at the optimal point, we implemented Early Stopping as part of the training process. The model monitored the validation loss (val_loss) during training. If the validation loss did not improve by at least 0.0001 after three consecutive epochs, the training process was automatically halted. This was achieved using the EarlyStopping callback from Keras, with the patience set to 3 epochs and min_delta set to 0.0001. This method of early stopping helped ensure that the model did not overfit the training data by continuing to train when no significant improvement was observed. It also reduced the training time, as the model could stop before reaching the maximum number of epochs if no improvement in performance was detected. The use of early stopping provided a safeguard against overfitting, especially when the model had already converged to an optimal state.

- **Learning Rate Adjustment**

The learning rate plays a critical role in determining the step size for weight updates during the model's optimization process. In our case, the learning rate was initially set to 0.0002, allowing the model to update its parameters at a moderate pace, avoiding both overshooting (due to large learning rates) and underfitting (due to very small learning rates). However, it is essential to adapt the learning rate as the training progresses to fine-tune the model further. We utilized the ReduceLROnPlateau callback from Keras, which reduces the learning rate when the model stops improving. Specifically, the learning rate was reduced by a factor of 0.1 if the val_loss did not improve by at least 0.0004 for two consecutive epochs. This helped to refine the model in the later stages of training when smaller updates were necessary to achieve higher performance. The minimum allowed learning rate was set to 1e-6, ensuring that the learning rate did not drop too low, which could have prevented the model from making meaningful updates. This gradual reduction of the learning rate helped the model to avoid getting stuck in suboptimal solutions and ensured better convergence towards the global minimum of the loss function.

Evaluation of Inception V3

In the model evaluation phase, we shift our focus from model training to analyzing the model's performance over time, using learning curves to visualize key metrics such as accuracy, loss, and the Quadratic Weighted Kappa score across training epochs. These curves allow us to compare the model’s performance on both the training and validation datasets, offering valuable insights into its learning behavior and generalization capabilities. By observing these curves, we can infer whether the model is underfitting, overfitting, or achieving a good balance between the two. A consistent convergence of training and validation curves suggests that the model has learned the underlying patterns effectively without overfitting. This diagnostic tool is crucial for guiding adjustments to model architecture, hyperparameters, and training strategies, ensuring optimal performance on the unseen data.

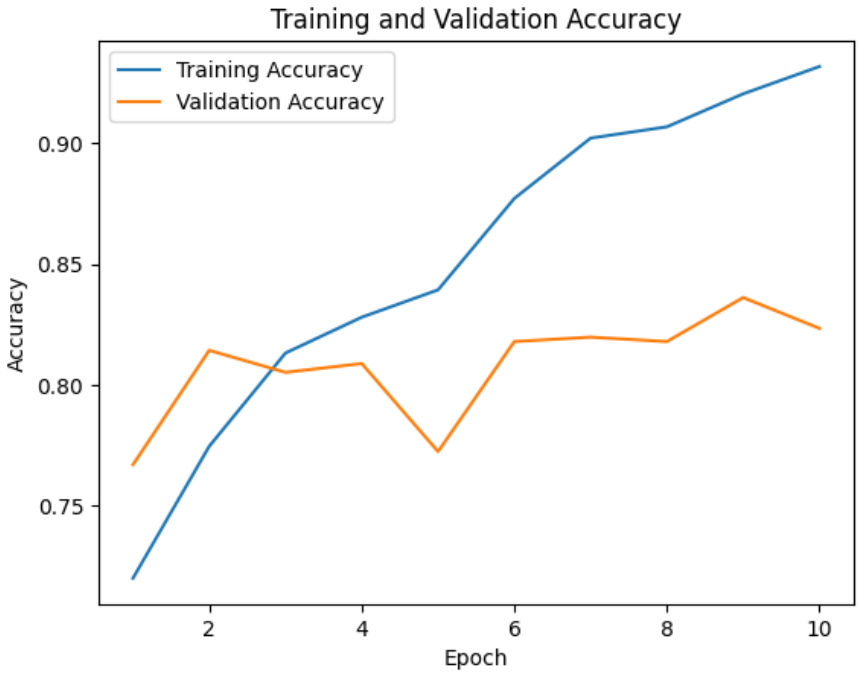


Figure 30 – Training and Validation Accuracy of Inception V3

Figure 30 illustrates the evolution of accuracy over epochs, exhibiting the following behaviors:

- Rapid Improvement in Initial Epochs: The model shows a **rapid increase** in both training and validation accuracy in the initial epochs, indicating effective learning and generalization from the start. During epoch 2, the training accuracy reaches 76.94% while the validation accuracy reaches 81.42%, demonstrating significant progress.
- Plateau in Later Epochs: After the initial improvement, the **validation accuracy stabilizes** around 82%, while the training accuracy continues to increase slightly, reaching 93.78% by epoch 10. This pattern suggests that the model has reached its learning capacity on the validation set, indicating that further training may not yield substantial gains in generalization.
- Overfitting Indications: The slight gap between training and validation accuracy **suggests potential overfitting**. By epoch 10, the training accuracy is considerably higher than the validation accuracy, which remains stable. This discrepancy indicates that the model is better at predicting the training data than the unseen validation data, possibly due to memorization.

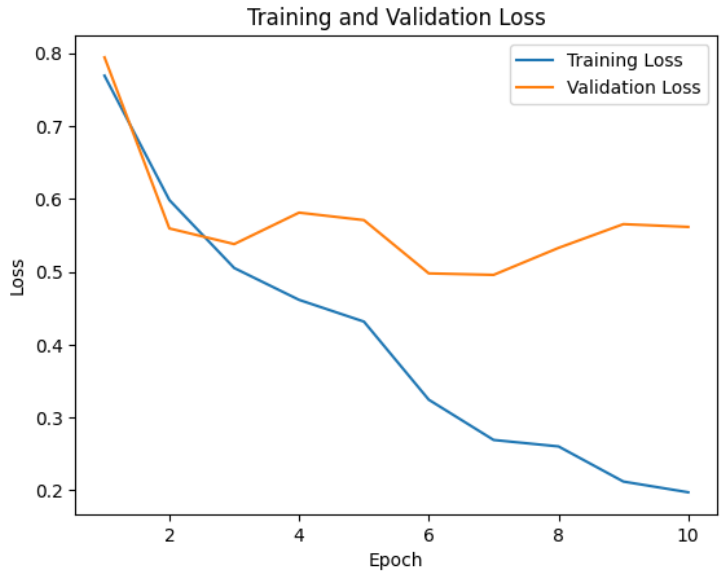


Figure 31 - Loss during epochs on training and validation set of Inception V3

Figure 31 depicts the evolution of the loss during epochs, exhibiting the following behavior:

- Effective Loss Minimization: The **training loss decreases steadily** from 0.89 in epoch 1 to 0.19 in epoch 10, reflecting effective learning and optimization of the model. The **validation loss also shows a decreasing trend**, but it fluctuates, with a minimum of ≈0.50 at epoch 7.
- Stabilization and Divergence: Despite the steady decrease in training loss, the **validation loss** stabilizes and even **slightly increases** after epoch 7. This divergence between training and validation loss, especially after reducing the learning rate, **reinforces the likelihood of overfitting**.
- Impact of Learning Rate Reduction: The learning rate reduction in epoch 5 and epoch 9 helped stabilize the loss, but the fluctuations in validation loss suggest that the model has **hit a performance ceiling** under the current configuration.

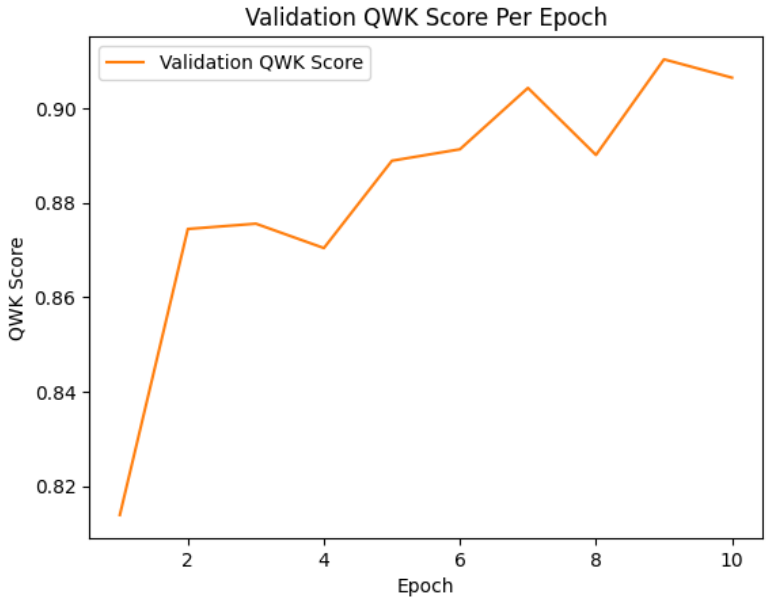


Figure 32 – Training and Validation QWK of Inception V3

Figure 32 illustrates the evolution of Kappa during epochs in the validation set, showing the following behavior:

Steady Improvement: The **QWK score shows consistent improvement** in the initial epochs, with the best score of 0.91 achieved at epoch 9. This score indicates a high level of agreement between predicted and actual classes, reflecting the **model's effective handling** of the ordinal nature of the classification problem.

Stabilization and Slight Decline: After achieving the peak QWK score, subsequent epochs show **slight fluctuations.** The QWK score decreases marginally, indicating that the **model's performance has stabilized** and is not significantly improving with additional training.

Optimal Epoch: The best model, saved at epoch 9, with **a QWK score of 0.91**, represents the point of optimal generalization. Training beyond this point, as seen in the subsequent epochs, did not lead to a substantial improvement, which justifies the use of early stopping.

Observations Based on Model Inception V3 Summary

- Early Stopping Effectiveness: The **early stopping callback was not triggered** because the validation loss continued to improve, albeit marginally, every 3 consecutive epochs, meeting the minimum improvement threshold of 1^{-4}. This suggests that the model was still making progress, albeit slowly, and therefore the early stopping mechanism did not intervene to halt training.

- Model Generalization and Capacity: The Inception v3 model indicates good generalization capabilities, as evidenced by its high QWK score of 0.91. However, the **significant gap** between training and validation accuracy, as well as the stabilization of validation accuracy around 82% despite continued improvement in training accuracy, suggests a **potential risk of overfitting**. This is further reinforced by the divergence between training and validation loss after epoch 7. While the model has demonstrated good generalization capabilities, the risk of overfitting could compromise its ability to generalize well to unseen data and potentially lead to poor performance in future tests.

- Best Performance Metrics: At Epoch 9, the model demonstrated its **best performance with a validation QWK of 0.91**, indicating excellent agreement between predictions and actual labels. However, despite achieving a **high accuracy of 92.33% on the training set**, the **validation accuracy was significantly lower at 83.61%,** suggesting a potential gap between the model's performance on seen and unseen data. Furthermore, the divergence between training and validation loss, especially after reducing the learning rate, reinforces the likelihood of overfitting. Therefore, while the high QWK score suggests that the model's predictions remain robust, the discrepancy between training and validation metrics indicates that the model may be overfitting to the training data.

DenseNet 121

Architecture of DenseNet 121

DenseNet, also known as Densely Connected Convolutional Networks, is a type of convolutional neural network (CNN) architecture that was introduced in 2016 by Gao Huang, Zhuang Liu, and Kilian Weinberger. The primary innovation of DenseNet lies in its unique connectivity pattern, which differs significantly from traditional CNN architectures. In this paragraph, we will delve into the detailed architecture of DenseNet and explore how it works. The DenseNet model is a significant improvement over traditional CNNs in several ways. Traditional CNNs are prone to vanishing gradients, where the gradients of the loss function become

smaller as they are backpropagated through the network, making it difficult to train deeper networks. This is because the gradients are multiplied together as they are backpropagated, causing them to decrease exponentially. Additionally, traditional CNNs often suffer from feature redundancy, where multiple layers learn similar features, leading to inefficient parameter usage and increased risk of overfitting. This is because each layer is only connected to the next layer, causing the features learned by earlier layers to be lost as the data flows through the network. Furthermore, traditional CNNs typically use a sequential architecture, where each layer is only connected to the next layer, which can lead to a loss of feature information as the data flows through the network. In contrast, DenseNet introduces a dense connectivity pattern, where each layer is connected to every other layer in a feed-forward fashion, allowing each layer to receive additional inputs from all preceding layers and pass its own feature-maps to all subsequent layers. This allows the features learned by earlier layers to be preserved and reused throughout the network, reducing the need to relearn redundant features and improving the overall efficiency of the network. By addressing these limitations, DenseNet can achieve state-of-the-art performance on a variety of image classification tasks, including diabetic retinopathy prediction using fundus camera images.

The name "DenseNet" refers to the dense connectivity pattern of the network, where each layer is connected to every other layer in a feed-forward fashion. This architecture has received significant attention in the field of computer vision, with over 2000 citations and a best paper award at the CVPR Conference. The core innovation of DenseNet lies in its unique connectivity pattern, which differs significantly from traditional CNN architectures. The dense connectivity pattern of DenseNet is a key innovation of architecture. In traditional CNNs, each layer is connected only to the previous layer, and the output of each layer is passed to the next layer in a sequential manner. In contrast, DenseNet has a dense connectivity pattern, where each layer is connected to every other layer in a feed-forward fashion. To illustrate this, consider a DenseNet with L layers. Each layer l receives input from all previous layers, i.e., $x_0, x_1, \dots, x_{l-1}$ and produces an output xl. The output xl is then passed to all subsequent layers, i.e., $x_{l+1}, x_{l+2}, \dots, x_L$. This creates a dense connectivity pattern, where each layer has access to the feature maps of all previous layers.

The building block of DenseNet is called a Dense Block, which consists of multiple convolutional layers with a specific connectivity pattern. Each Dense Block has two main components: a convolutional layer and a concatenation layer.

- **Convolutional Layer**: The convolutional layer is a standard convolutional layer with a specific number of filters, kernel size, and stride. The convolutional layer takes the input from all previous layers and produces a feature map.
- **Concatenation Layer**: The concatenation layer takes the output of the convolutional layer and concatenates it with the outputs of all previous layers in the block. This creates a feature map that combines the information from all previous layers.

Each Dense Block typically consists of 3-4 convolutional layers, and the output of each block is passed to the next block. The concatenation layer is used to combine the feature maps from all previous layers, allowing the network to capture a wide range of features.

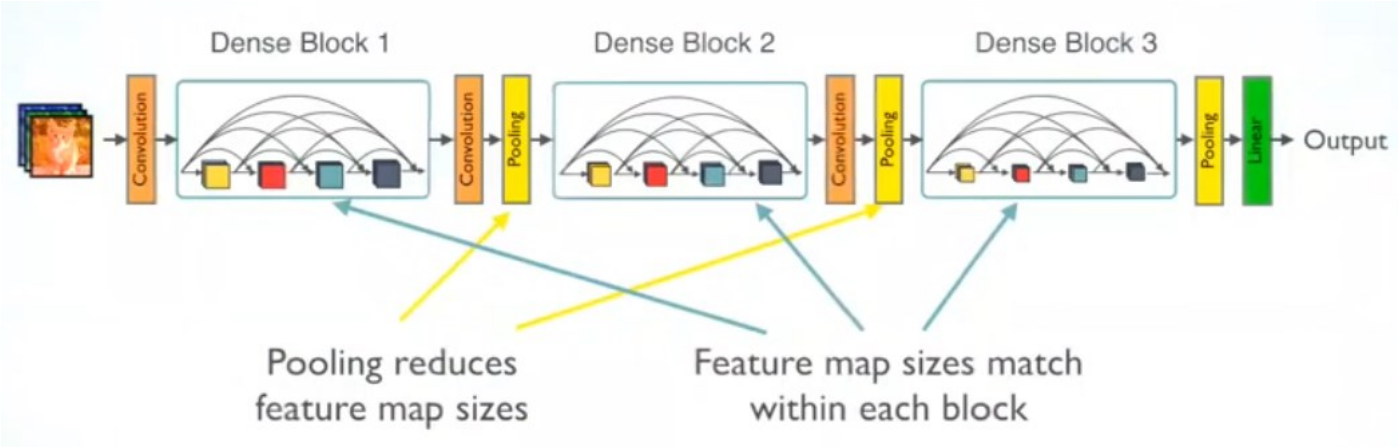


Figure 33 - Multiple Dense Blocks with Transition Layerse

To reduce the spatial dimensions of the feature maps and increase the number of channels, DenseNet uses a transition layer between each Dense Block. This transition layer consists of two main components: a convolutional layer with a kernel size of 1x1, which reduces the number of feature maps, and an average pooling layer with a stride of 2, which reduces the spatial dimensions of the feature maps by half. The transition layer plays a crucial role in reducing the spatial dimensions of the feature maps while increasing the number of channels, allowing the network to capture more abstract features and reduce the number of parameters.

One of the key hyperparameters in DenseNet is the growth rate, which controls the number of new feature maps added to each layer. The growth rate is typically set to a small value, such as 12 or 24, allowing the network to gradually increase the number of feature maps as the depth of the network increases. However, the growth rate is a critical hyperparameter, as a high growth rate can lead to many parameters, making the network prone to overfitting, while a low growth rate can result in a small number of parameters, making the network less expressive.

To reduce the number of parameters and improve the computational efficiency of the network, DenseNet uses a bottleneck layer at the beginning of each Dense Block. The bottleneck layer consists of two main components: a convolutional layer with a kernel size of 1x1, which reduces the number of feature maps, and a convolutional layer with a kernel size of 3x3, which increases the number of feature maps. The bottleneck layer reduces the number of feature maps while increasing the number of channels, allowing the network to capture more abstract features while reducing the number of parameters.

The final output layer of DenseNet consists of two main components: a global average pooling layer and a fully connected layer. The global average pooling layer reduces the spatial dimensions of the feature maps to 1x1, while the fully connected layer produces the final output probabilities. The global average pooling

layer plays a crucial role in reducing the spatial dimensions of the feature maps, and the fully connected layer produces the final output probabilities, allowing the network to make accurate predictions.

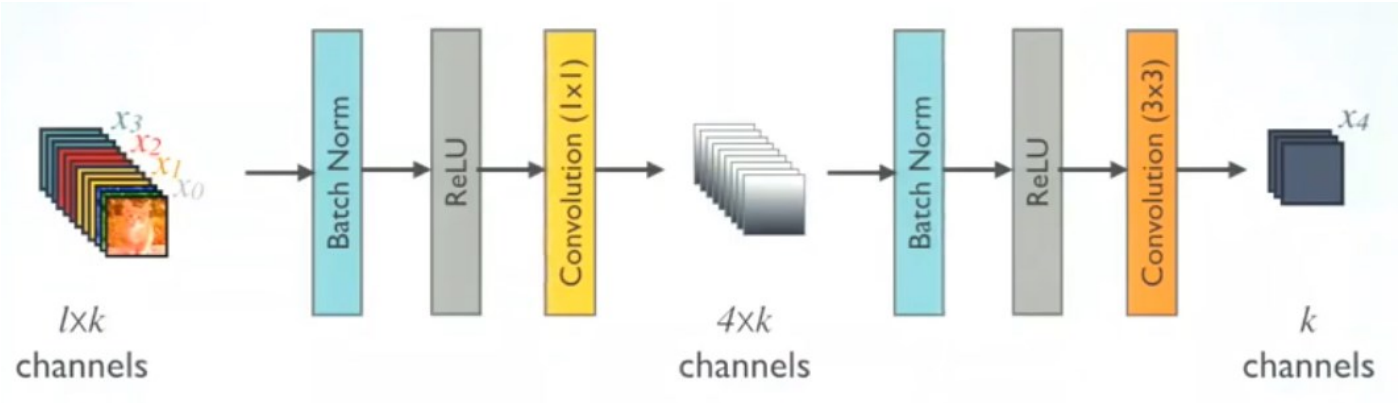


Figure 34 - Bottleneck structure where 1×1 convolutions are used before a 3×3 convolution layer

The DenseNet architecture has several key benefits that make it a highly effective and efficient neural network. One of the primary advantages is its dense connectivity pattern, which allows the network to capture a wide range of features, from low-level edges to high-level semantic information. This is achieved by connecting each layer to every other layer in a feed-forward fashion, enabling the network to learn complex and abstract representations of the input data. Additionally, the use of bottleneck layers and transition layers reduces the number of parameters, making the network more computationally efficient and reducing the risk of overfitting. Furthermore, the use of concatenation layers and transition layers allows for improved feature extraction, as the network can combine and refine features from multiple layers, resulting in a more robust and accurate representation of the input data. Overall, the DenseNet architecture is a powerful and efficient neural network that is well-suited for a wide range of computer vision tasks.

DenseNet121 is a variant of the DenseNet architecture that has been optimized for image classification tasks. One of the key optimizations of DenseNet121 is the reduced number of parameters, which is achieved by using a smaller growth rate of 32 and a smaller number of dense blocks, specifically 4 dense blocks. This reduction in parameters leads to improved computational efficiency, as the network requires fewer convolutional layers and feature maps to process. Additionally, the smaller number of feature maps and dense blocks also results in improved memory efficiency, making it a more practical choice for deployment on devices with limited memory. Furthermore, the reduced number of parameters and convolutional layers also leads to improved training speed, allowing for faster convergence and reduced training times. Despite the reduced number of parameters, DenseNet121 achieves improved accuracy compared to the original DenseNet architecture, thanks to the use of a larger number of dense blocks and a larger growth rate.

The DenseNet121 architecture (see Figure 35) begins with a standard convolutional layer, comprising 64 filters with a kernel size of 7x7 and a stride of 2, followed by a max pooling layer with a kernel size of 3x3 and a stride of 2, which sets the stage for the initial dense block. The network then consists of four dense blocks, each comprising a set number of convolutional blocks (6, 12, 24, and 16, respectively). Within each dense block, every layer receives feature maps from all preceding layers, processes them through batch normalization, ReLU activation, and a 3x3 convolution with a growth rate of 32, and then passes the result to all subsequent layers. The outputs of all layers within each dense block are concatenated together, leading to a growth in the depth of feature maps as data passes through the block. Specifically, the first dense block

consists of 6 convolutional layers, the second dense block consists of 12 convolutional layers, the third dense block consists of 24 convolutional layers, and the fourth dense block consists of 16 convolutional layers.

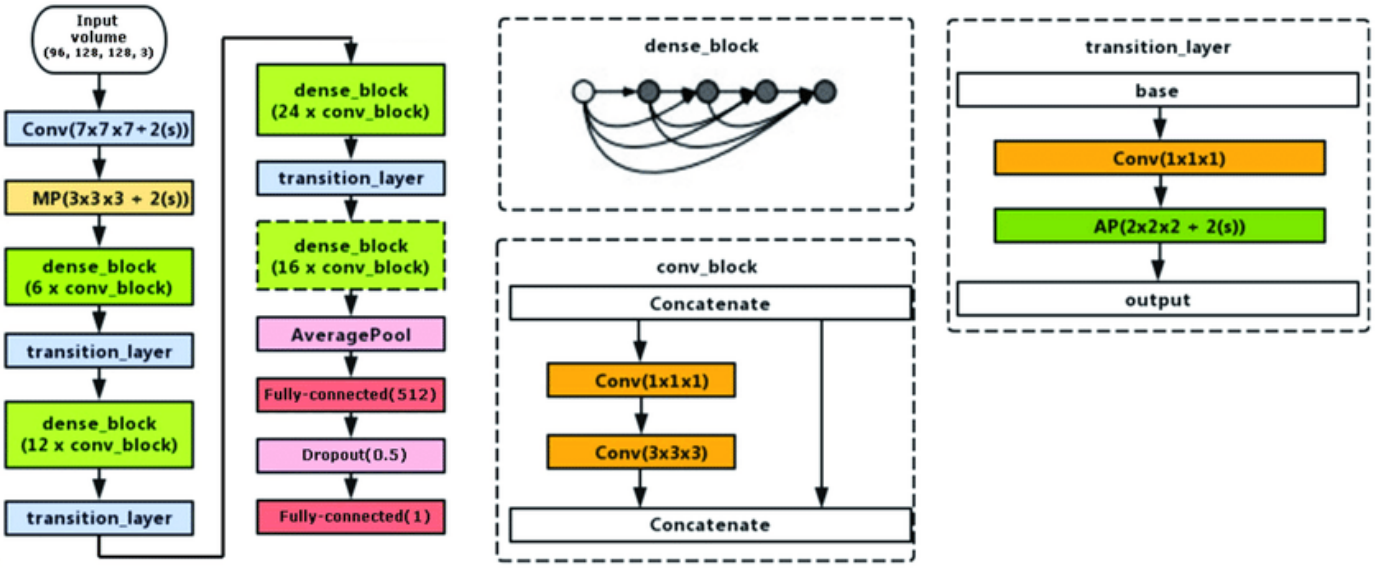


Figure 35 - Dense 121 Architecture

Between the dense blocks, transition layers manage the flow and dimensions of data, consisting of batch normalization, a 1x1 convolutional layer with a compression factor of 0.5, and a 2x2 average pooling layer. After passing through all the dense blocks and transition layers, the network employs an average pooling layer with a kernel size of 7x7, followed by a fully connected layer with 1000 units and dropout for regularization. The final output is then produced after the last fully connected layer, which processes all accumulated and concatenated features to make the final classification or prediction. This architecture is highly effective for complex image recognition tasks due to its efficient use of parameters and its ability to retain important features from the initial to the final layers without significant loss due to depth.

Implementation of DenseNet121 for Multilabel Image Classification

- **Load and Preprocess the Dataset**

The preprocessed images, which had undergone prior transformations, were loaded from their respective directories and resized to uniform dimensions. The dataset of images was utilized for multilabel image classification using the DenseNet121 model. The objective was to classify each image into one or more of the five predefined classes. To ensure uniform dimensions, the images were resized to a fixed size of 224x224 pixels, a requirement for the DenseNet121 model. The image loading and resizing process was parallelized by leveraging multiple CPU cores, thereby reducing the time taken to load and preprocess the images. The labels were converted to one-hot encoding, a technique employed to transform categorical labels into numerical representations that can be processed by machine learning algorithms.

One-hot encoding generates a binary vector for each label, where all elements are 0 except for the element corresponding to the label, which is 1. The one-hot encoded labels were then converted to multilabel format by applying a logical OR operation between each class and the next higher class, enabling the model to predict multiple labels for each image.

- **Apply Data Augmentation**

Data augmentation was applied to the training images to enhance the model's performance. The images were randomly flipped left-right and up-down, and their brightness and contrast were adjusted. Additionally, the images were rotated by 0, 90, 180, or 270 degrees to further increase the diversity of the training data. The augmentation was applied to the dataset, and the images were batched into groups of 32 using the batch method. To improve performance, the dataset was prefetched, allowing the next batch of images to be loaded while the current batch was being processed.

- **Ordinal Classification Approach**

To effectively capture the progressive nature of diabetic retinopathy, an ordinal classification approach is employed, leveraging the inherent ordering of the disease severity levels from No Disease (Class 0) to Proliferative (Class 4). By transforming the target labels into cumulative binary vectors, the model can learn to predict the severity levels in a more informed manner, respecting the natural progression of the disease. For instance, Class 0 is represented as [0, 0, 0, 0, 0], indicating no presence of the disease, while Class 4 is represented as [1, 1, 1, 1, 1], signifying the most severe level of the disease. This strategic decision allows the model to capture the subtle differences between the classes and provide more reliable predictions. During inference, the model outputs a probability for each class using the sigmoid function, and a threshold equal to 0.5 is applied to determine the predicted class. By starting from the least severe class and moving upward, the model can accurately determine the severity level of diabetic retinopathy, ultimately leading to improved performance and more accurate diagnoses. This ordinal classification technique ensures that the predicted categories respect the progression of diabetic retinopathy severity, enabling the model to better capture the underlying relationships between the classes and provide more accurate representations of the disease progression.

- **Define Custom Metric**

The model's performance was evaluated based on accuracy, which is calculated by comparing the predicted labels with the actual labels and returning the mean of the matches across all samples. Accuracy is calculated as the number of true positives plus the number of true negatives divided by the total number of samples, resulting in a value between 0 and 1, where 1 represents perfect accuracy and 0 represents complete inaccuracy. In addition to accuracy, the Kappa score was also defined on the validation set after each epoch, providing a measure of agreement between the predicted labels and the actual labels. The Kappa score, is a statistical measure that considers the fact that the model is predicting multiple labels. It is calculated as the ratio of the number of agreements between the predicted and actual labels to the number of agreements that would be expected by chance, ranging from -1 to 1, where 1 represents perfect agreement, 0 represents no agreement, and -1 represents complete disagreement. A Kappa score of 0.5 or higher is generally considered to be a moderate to strong agreement, while a score of 0.8 or higher is a strong agreement. By utilizing both accuracy and Kappa score, a more comprehensive understanding of the model's performance can be gained.

- **Modify the Pre-trained Model**

The top fully connected layers of the pre-trained model were excluded, as they are specific to the ImageNet dataset, which differs from the dataset used in this project. The pre-trained DenseNet121 model was trained on the ImageNet dataset, comprising over 14 million images from 21,841 categories. Consequently, the top fully connected layers of the model have learned to recognize features specific to the ImageNet dataset, such as the presence of certain objects, scenes, or textures. However, these features are not relevant to the dataset used in this project, which consists of specialized images very specific to our purpose (fundus pictures of eyes). By excluding the top fully connected layers, the model is prevented from relying on these pre-learned features and is instead forced to learn new features specific to our dataset. Custom layers were added on top of the DenseNet121 model to adapt it to the specific requirements of our project. A global average pooling layer was added to reduce the spatial dimensions of the feature maps, which helps to reduce the number of parameters in the model and the number of computations required. This is particularly useful for image classification tasks, where the spatial location of the features is not as important as the presence or absence of the features themselves. The global average pooling layer also helps to capture the overall patterns and structures in the images, rather than just focusing on local features. A dropout layer was also added to prevent overfitting, which occurs when the model is too complex and learns the noise in the training data. Overfitting can result in poor performance on unseen data, as the model is not able to generalize well to new examples. By randomly dropping out neurons during training, the dropout layer forces the model to learn multiple representations of the data, which helps to improve its ability to generalize.

- **Add Output Layer**

A dense layer with sigmoid activation was added to the model to predict the labels, which is particularly well-suited for multi-label classification tasks where the model needs to predict multiple labels for each image. The sigmoid activation function outputs a probability for each label, allowing the likelihood of each label being present in the image to be determined. This enables the model to learn a complex mapping between the input features and the output labels, which is essential for accurate multi-label classification.

By utilizing the sigmoid activation function, the model can effectively handle the complexity of multi-label classification, where each image can have multiple labels, and provide a robust and accurate prediction of the labels for each image.

- **Compile the Model**

The model was compiled using the binary cross entropy loss function, a widely used loss function for multilabel classification tasks, which effectively measures the difference between the predicted probabilities and the true labels. The binary cross entropy loss function provides a smooth and continuous loss function that can be optimized using gradient descent, allowing the model to learn from the training data and make accurate predictions on the test data. The Adam optimizer was employed with a learning rate of 0.00005, which adaptively adjusts the learning rate for each parameter based on the magnitude of the gradient, allowing for efficient and effective optimization of the model's parameters. This compilation setup enabled the model to effectively learn from the training data and make accurate predictions on the test data.

- **Train the Model**

The DenseNet121 model was trained using the Keras fit method over a total of 30 epochs, utilizing both the training and validation datasets.

During training, the data was processed in **batches of 32 images**, which optimizes memory usage by breaking down the dataset into smaller, more manageable chunks, ensuring efficient GPU utilization. The use of batches improves the speed of training while maintaining sufficient gradient updates for model convergence. The DenseNet121 architecture, which contains approximately **8 million parameters** (specific to the variant used), was employed. This number of parameters strikes a balance between model complexity and computational efficiency. The model's reduced parameter count, compared to larger architectures, helps prevent overfitting while the model retains its efficiency. DenseNet's efficient feature reuse, facilitated by its densely connected layers, allowed the model to achieve better performance on the validation dataset with a lower number of parameters, improving computational performance.

- **Callbacks of the model**
  - **Custom QWK**

The QWK metric was used as a custom callback to monitor the model's agreement between the predicted labels and the actual labels on the validation set. This metric provides an additional evaluation of model performance, especially important in multi-label classification problems. It is computed after each epoch, ensuring that the model not only minimizes loss but also maintains good prediction agreement. The QWK score is particularly suited for problems where agreement between predictions and true labels is critical. A QWK score above 0.8 is generally considered excellent, indicating a strong agreement between predicted and actual labels.

  - **Early Stopping**

Early stopping was used to prevent the model from overfitting to the training data. If the validation loss did not improve for **5 consecutive epochs**, the training was halted early. This ensured that the model did not waste time and resources continuing to train when it had already reached its optimal performance on the validation set. It also helped in preventing the model from overfitting to the training data by ceasing training once the model showed signs of stagnation or degradation in validation performance.

  - **Learning Rate Adjustment**

A ReduceLROnPlateau callback was employed to adjust the learning rate during training. Specifically, if the validation loss did not improve for **4 consecutive epochs**, the learning rate was reduced by a factor of 0.1. This gradual reduction in the learning rate helps the model fine-tune its weights as it approaches convergence, allowing for more precise updates to the parameters during the later stages of training. This setup ensured that if the model encountered a plateau in validation loss improvement, the learning rate would be reduced to enable further progress without overshooting the loss landscape.

Evaluation of DenseNet 121

In the model evaluation phase, we shift our focus from model training to analyzing the model's performance over time, using learning curves to visualize key metrics such as accuracy, loss, and the Quadratic Weighted Kappa score across training epochs. These curves allow us to compare the model's performance on both the training and validation datasets, offering valuable insights into its learning behavior and generalization capabilities. By observing these curves, we can infer whether the model is underfitting (performing poorly on both training and validation data), overfitting (performing well on training data but poorly on validation data), or achieving a good balance between the two.

A consistent convergence of training and validation curves suggests that the model has learned the underlying patterns effectively without overfitting. This diagnostic tool is crucial for guiding adjustments to model architecture, hyperparameters, and training strategies, ensuring optimal performance on the unseen data.

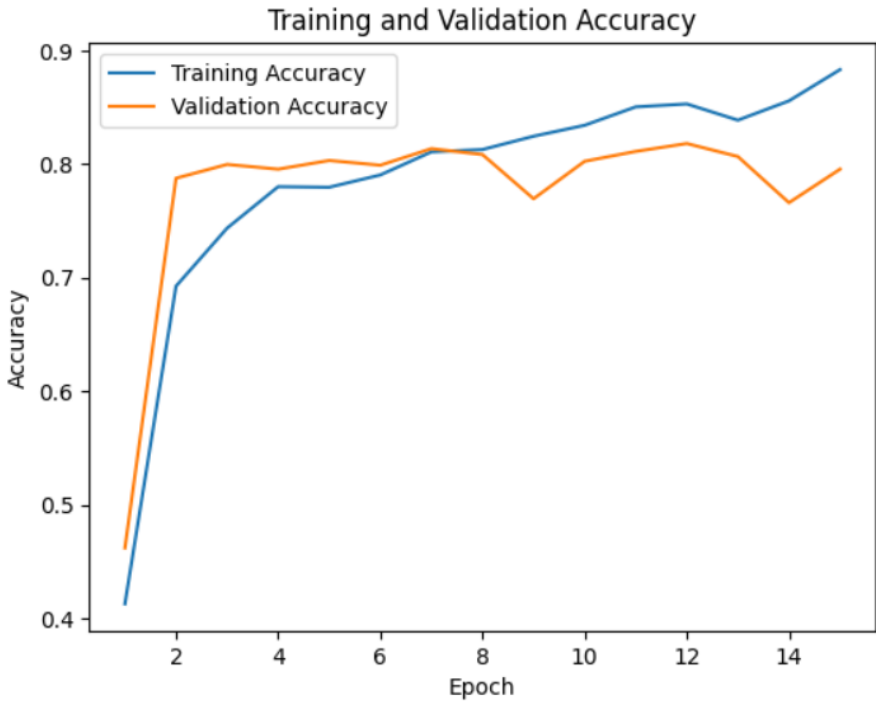


Figure 36 – Training and validation Accuracy of DenseNet121

Figure 36 illustrates the evolution of accuracy during epochs in training and validation sets:

- Initial Convergence: During the initial epochs, both training and validation accuracy show a steep increase, with validation accuracy reaching approximately **0.79 by epoch 3**. This indicates effective initial learning and quick adaptation to the patterns in the training data.
- Validation Accuracy Fluctuation: After reaching the peak, the **validation accuracy** starts to fluctuate around **0.80**, suggesting the model reaches a state of equilibrium. The minor variations indicate that the model's ability to generalize to unseen data is relatively stable, but further improvements are minimal.
- Training Accuracy Increase: The **training accuracy** continues to improve gradually, reaching around **0.88 by epoch 15**, whereas validation accuracy does not exhibit the same trend. This divergence hints at a potential overfitting scenario, where the model performs well on the training data but struggles to generalize to the validation set.

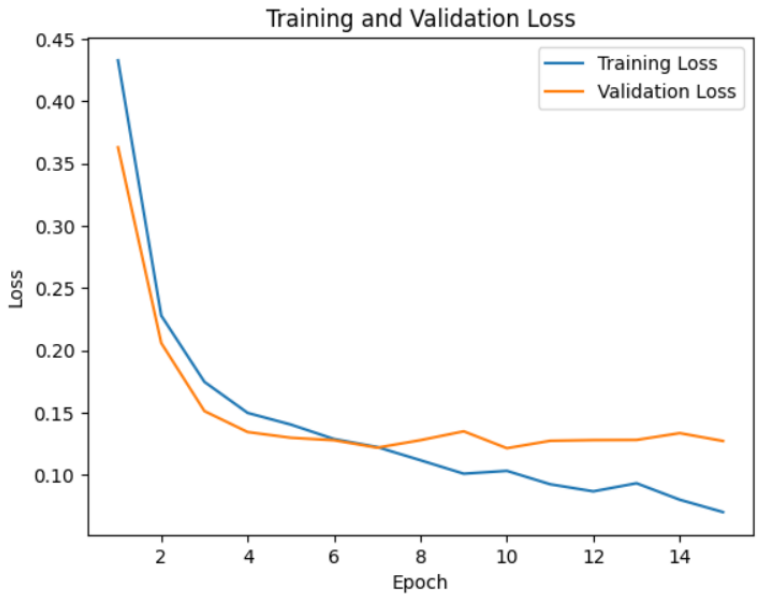


Figure 37 - Loss during epochs on training and validation set of DenseNet121

Figure 37 illustrates the evolution of loss during epochs in training and validation setduring epochs:

- Rapid Loss Reduction: Both training and validation loss decrease sharply during the first few epochs, reflecting efficient learning. **By epoch 3**, the **validation loss** reaches approximately **0.15**, which aligns with the high validation accuracy observed earlier.
- Validation Loss Plateau: From **epoch 5** onwards, the **validation loss** stabilizes around **0.13–0.15**, with minor fluctuations. This plateau, along with a continual decrease in training loss, further supports the notion of slight overfitting. The model minimizes errors on the training set, while the error on the validation set does not reduce further.
- Low Final Loss Values: The **final loss** values are relatively low for both training and validation datasets, with **training loss** around **0.07** and **validation loss** around **0.13** at **epoch 15**. This indicates good overall performance, though the gap between training and validation loss is a point of concern.

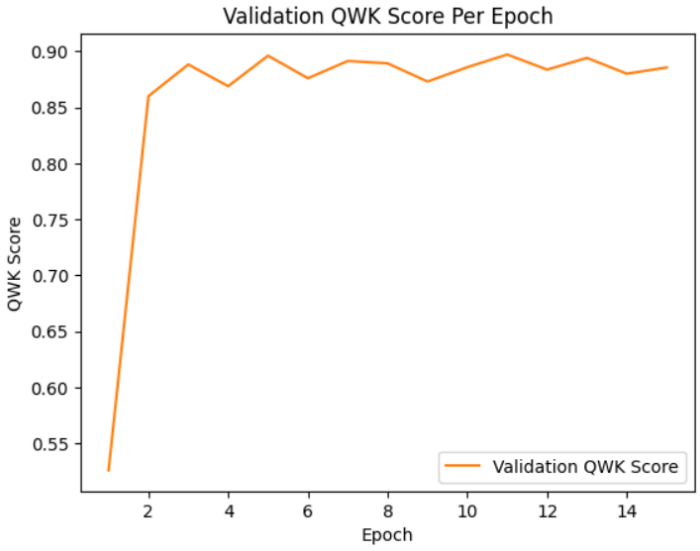


Figure 38 - Training and validation QWK of DenseNet121

Figure 38 illustrates the evolution of QWK:

- Initial Surge: The **QWK score** exhibits a rapid increase in the first few epochs, stabilizing around **0.88 by epoch 3**. This high score indicates a strong initial performance in capturing the severity levels of diabetic retinopathy, a key aspect of the classification task.
- Stability and Minor Fluctuations: The QWK score remains **stable between 0.87 and 0.89** throughout **most epochs**, with slight fluctuations. This consistency suggests that the model maintains a good level of agreement between predicted and true labels, even as validation loss and accuracy fluctuate.
- Optimal Performance Epoch: The **highest QWK score** of **0.90** is observed at **epoch 11**, indicating peak performance. After this point, the QWK score shows minor declines and fluctuations, but overall remains robust, supporting the effectiveness of the model in capturing the ordinal nature of the classification problem.

Observations Based on Model DenseNet 121 Summary

- **Early Stopping Impact:** The early stopping mechanism triggered at epoch 15, with a patience of 5 epochs, prevented further training as no significant improvement in validation loss was observed. This intervention helps avoid overfitting, ensuring that the model does not continue to optimize on the training data at the cost of generalization.
- **Model Generalization:** The stable validation accuracy and QWK score, combined with the early stopping, suggest that the model has achieved a good balance between learning and generalization, though careful monitoring and tuning may still be required to address slight overfitting tendencies.
- **Final Performance Metrics:** The model's best performance was observed with a validation accuracy of 0.81 and a QWK score of 0.90 at epoch 11, indicating a high level of agreement and accuracy.

Comparison of Model Performance Metrics

In this section, we evaluate and compare the performance of the three deep learning models—CNN, Inception V3, and DenseNet121—on the test set, consisting of unseen data. The models were trained and tested on a dataset with the goal of achieving high classification accuracy while minimizing misclassifications. Several key performance metrics were utilized to assess the models, including accuracy, F1-score, sensitivity (recall), specificity, and Quadratic Weighted Kappa. These metrics provide an overview of how well each model handles both positive and negative class predictions, as well as the balance between precision and recall, indicating how well the model generalizes on unseen data. Based on these metrics, **DenseNet121** and **Inception V3 outperformed the CNN model**, achieving the **highest QWK** of about **0.90**, indicating better alignment between predicted and actual labels.

The **optimal model** was **selected** not only **based** on its superior QWK score but **also** on its demonstrated **stability across different epochs during the training process**. This is particularly crucial in medical applications, where variability in performance across epochs may reveal model instability. Relying on a model that shows **fluctuations** in performance **across epochs may indicate overfitting or underfitting**. This is especially important for our application because potential misclassifications can have serious

consequences. The **complexity** of the **DenseNet architecture** is well-suited to capturing the **intricate patterns** from fundus images taken with a fundus camera, allowing it to **produce reliable predictions**. A **model** that **performs consistently across multiple epochs** is more **likely** to **generalize** well to **unseen data**, making it more reliable in critical applications like medical diagnosis, where stable and consistent performance is essential for ensuring patient safety and care.

In addition to stability across epochs, sensitivity (recall) was another crucial metric for evaluating model performance. **Sensitivity measures** the model's ability to **correctly identify true positive cases**—in this context, patients with diabetic retinopathy. On this dataset, better sensitivity is essential because it **helps mitigate the risk of Type II errors**, also known as false negatives. In medical applications, a false negative occurs when the model fails to detect a condition that is present, leading to missed diagnoses. This can have severe consequences, as untreated diabetic retinopathy may lead to vision loss or other complications. **DenseNet121** achieved a **sensitivity of 82.18%,** while Inception V3 had a lower sensitivity of 70.61%. Given the high-stakes nature of this dataset, where accurately identifying patients with diabetic retinopathy is critical, prioritizing models with higher sensitivity ensures that more patients with the condition are correctly identified, improving overall patient outcomes. (See Figure 39)

For these reasons**, DenseNet121** was deemed the **most suitable model**, exhibiting the highest QWK score and sensitivity, along with stable performance across epochs during training. In the following analysis, we present the performance metrics for each model, highlighting their strengths and weaknesses. While the metrics provide a comprehensive view of each model's performance, the QWK score, stability across epochs, and sensitivity were given greater importance in our evaluation, as they directly affect the model's reliability and effectiveness in real-world medical settings. Based on these criteria, DenseNet121 emerged as the top-performing model.

Models	Accuracy	F1-Score	Sensitivity	Specificity	QWK	Time Running
CNN	71.82%	63.91%	37.49%	71.82%	0.74	48.44 sec
Inception V3	85.64%	85.20%	70.61%	85.64%	0.90	295.67 sec
DenseNet 121	82.18%	82.11%	82.18%	82.18%	0.90	307.07 sec

Figure 39 - Performance metrics of all models on test set

Below are the precise definitions of each performance metric considered:

- **Accuracy** measures the overall proportion of correctly classified instances, providing a general sense of the model's performance. However, in a medical context like diabetic retinopathy diagnosis, the costs of Type II errors (false negatives) can be severe, as missing a diagnosis can led to delayed treatment and worse patient outcomes.

$$ACC = \frac{(TP + TN)}{(P + N)}$$

- **Sensitivity** (Recall) is vital, as it measures the proportion of actual positive instances, patients with diabetic retinopathy, that are correctly identified by the model. A high sensitivity ensures that the model is not missing many true cases.

$$Recall = \frac{TP}{(TP + FN)}$$

- **Specificity** measures the proportion of actual negative instances, patients without diabetic retinopathy, that are correctly identified, which is also important to avoid unnecessary treatments or patient anxiety.

$$SPC = \frac{TN}{(FP + TN)}$$

- **Precision** measures the proportion of predicted positive instances that are positive, indicating the model's ability to avoid false positives.

$$Precision = \frac{TP}{(TP + FP)}$$

- **F1 Score** provides a balanced measure of both precision and sensitivity, offering a comprehensive evaluation of the model's performance. In this scenario, a high F1 score would indicate that the model is both sensitive to actual cases of diabetic retinopathy and precise in its predictions, minimizing both Type II errors and unnecessary treatments.

$$F1 = \frac{2TP}{(2TP + FP + FN)} \quad or \quad F1 = \frac{2 \cdot (precision \cdot sensitivity)}{precision + sensitivity}$$

where, TP are True Positives, meaning the number of correctly predicted positive instances, TN are True Negatives, meaning the number of correctly predicted negative instances, FP are False Positives, meaning the number of incorrectly predicted positive instances, and FN are False Negatives, meaning the number of incorrectly predicted negative instances.

The assessment of models as shown in Figure 39 reflects the results in the following confusion matrices:

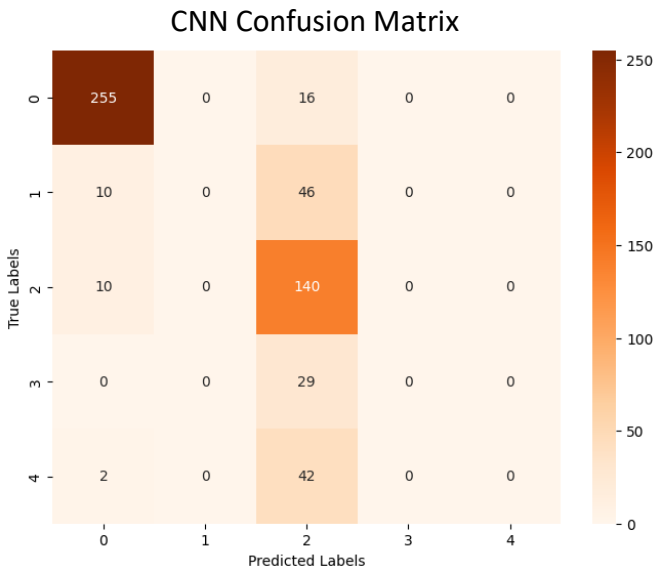


Figure 40 - Confusion matrix of CNN

The CNN model exhibits a notable tendency to classify images into the labels "No Disease" and "Moderate", struggling to discern distinct patterns across more varied categories. This limitation is compounded by the dataset's inherent imbalance, where a substantial majority of samples are concentrated in these two categories. As a result, the model's performance metrics appear deceptively high, masking its actual limitations in accurately classifying images across the full spectrum of categories. This skew in the data distribution poses a significant risk of misclassification, underscoring the need for a more nuanced approach to addressing the model's shortcomings.

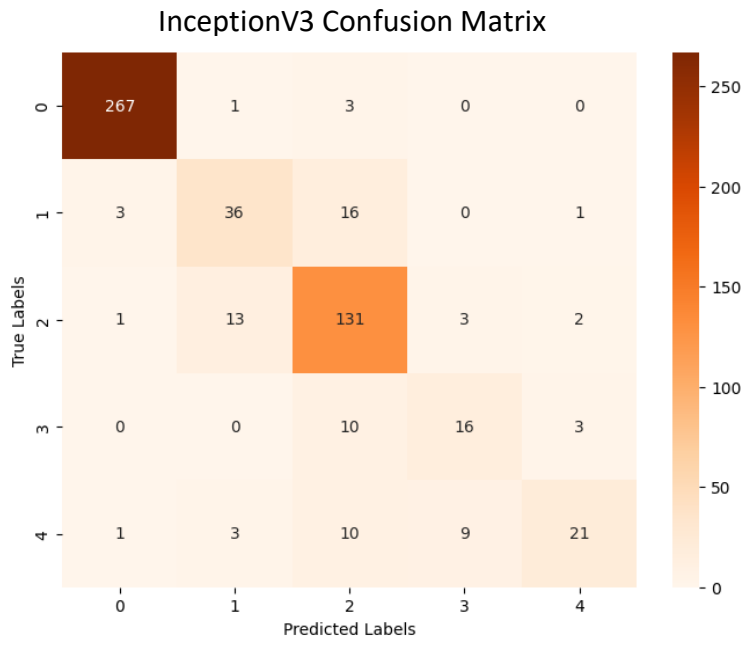


Figure 41 - Confusion matrix of Inception V3

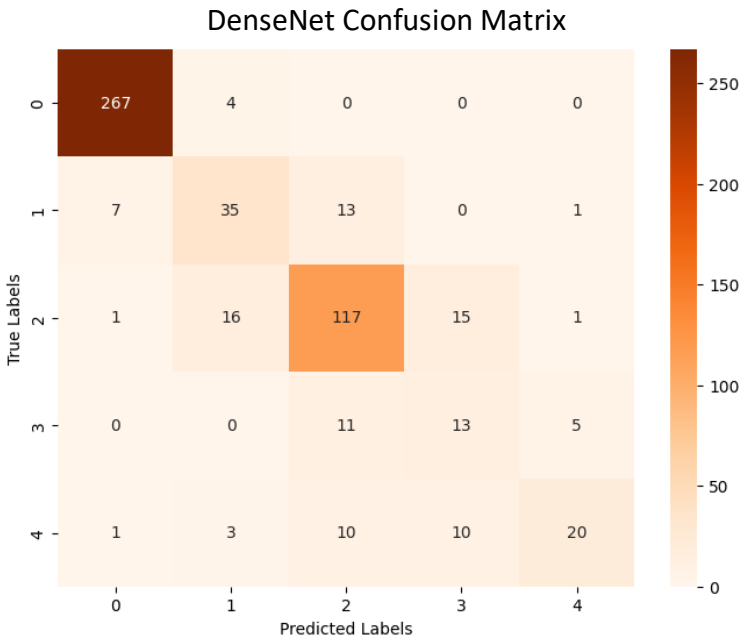


Figure 42 - Confusion matrix of DenseNet 121

When evaluating the performance of the three models, it became apparent that DenseNet121 and InceptionV3 had similar behavior in terms of misclassification patterns and evaluation metrics. However, upon closer inspection, it was noticed that InceptionV3 had a significant gap between its training and validation accuracy, with the training accuracy being considerably higher. This discrepancy suggests that the model is better at predicting the training data than the unseen validation data, which could be due to memorization. This is a concern because it indicates that the model may not generalize well to unseen data. The slight gap between training and validation accuracy also suggests potential overfitting. In contrast, DenseNet121 demonstrated better behavior during training, with the validation loss and accuracy remaining close to the training loss and accuracy throughout the training process. This suggests that the model is generalizing well to the validation data and is less prone to overfitting. The risk of overfitting and the potential for poor performance on unseen data were significant concerns with InceptionV3. In conclusion, while InceptionV3 and DenseNet121 had similar performance metrics, the potential risks associated with InceptionV3's behavior made DenseNet121 the more reliable choice.

Qualitative & Error Analysis

Having selected DenseNet121 as our model, we will now evaluate its performance on the test dataset to assess its predictive accuracy and examine its behavior across different levels of Diabetic Retinopathy.

		Prediction	
		No DR	**Else**
Actual	**No DR**	267	4
	Else	9	270

Accuracy	97.636%
Sensitivity	98.524%
Specificity	96.774%
Precision	96.739%
F1 Score	97.623%

The assessment of the first category, labeled "No Disease," reveals that only 13 images were misclassified among the total evaluated. This result is satisfactory, particularly considering the substantial proportion of the dataset categorized under "No Disease," which provided the model with an ample opportunity to learn and recognize this label effectively. Misclassification at this level represents a Type I error, potentially leading patients to undergo unnecessary diagnostic procedures for a condition they do not have. Such errors, while less harmful than false negatives in this context, highlight the need for further model refinement to minimize unnecessary medical examinations.

		Prediction	
		Mild DR	**Else**
Actual	**Mild DR**	35	21
	Else	23	471

Accuracy	92%
Sensitivity	62.5%
Specificity	95.344%
Precision	60.345%
F1 Score	61.404%

The evaluation of the model's performance in classifying images as "Mild DR" yields mixed results. Out of the total dataset, 23 images were incorrectly predicted as "Mild DR" when the patients were actually healthy, and 21 cases of Mild DR were missed by the model. The model demonstrates a tendency to overestimate the presence of the disease, as evidenced by 14 out of the 21 false positives. This type of error, where the condition is overdiagnosed, can lead patients to undergo unnecessary further testing for a condition that is less severe than predicted. However, this scenario is preferable in medical diagnostics over missing a potential diagnosis, as the risks associated with undetected conditions can be far more severe. In conclusion, while the model exhibits high accuracy overall, the precision and sensitivity for detecting "Mild DR" specifically suggest that there is room for improvement to reduce the rate of false positives and enhance the model's diagnostic reliability.

		Prediction	
		Moderate	**Else**
Actual	**Moderate**	117	33
	Else	34	366

Accuracy	87.818%
Sensitivity	78%
Specificity	91.5%
Precision	77.483%
F1 Score	77.741%

The assessment of the model's capability to accurately classify the "Moderate" level of Diabetic Retinopathy reveals a satisfactory but imperfect performance. Of the 150 patients with Moderate Diabetic Retinopathy, the model correctly identified 117 cases but misdiagnosed 33 cases. Among the misdiagnoses, 16 patients were erroneously categorized as having a more severe condition, and 17 were diagnosed with a less severe condition. Notably, only one patient without the disease was incorrectly classified as having Moderate Diabetic Retinopathy.This classification accuracy is crucial because it predominantly leads to further medical examinations rather than misdiagnoses that could result in no treatment. The model's tendency to err on the side of caution by overdiagnosing rather than underdiagnosing is preferable in medical scenarios where failing to detect an existing condition could have more serious repercussions. Therefore, while the model shows a high degree of specificity (91.5%) and a reasonable sensitivity (78%), its precision and F1 score (77.483% and 77.741%, respectively) suggest there is room for improvement, particularly in reducing false positives and avoiding unnecessary escalation of medical responses.

		Prediction	
		Severe	Else
Actual	**Severe**	13	16
	Else	25	496

Accuracy	92.545%
Sensitivity	44.828%
Specificity	95.202%
Precision	34.211%
F1 Score	38.806%

The performance evaluation of the model in classifying the "Severe" stage of Diabetic Retinopathy indicates a significant challenge. Among the 29 actual "Severe" cases, the model misclassified 16, resulting in a sensitivity of only 44.828%. While the model displayed high specificity (95.202%), suggesting it can accurately identify non-severe cases, the low precision (34.211%) and F1 score (38.806%) highlight its struggle with accurately detecting actual severe cases. 11 of the 29 severe cases were underestimated, classified as less severe rather than more, which implies that while the model recognizes the presence of disease-related patterns, it fails to grasp the severity adequately. This underestimation is somewhat mitigated by the fact that these cases were classified only one level below their actual severity, indicating a partial recognition of the disease's seriousness. This type of error, where severe conditions are not recognized as such, could delay necessary aggressive treatments, impacting patient outcomes. Thus, enhancing the model's ability to discriminate more precisely between severity levels remains a critical area for improvement.

		Prediction	
		Proliferative	**Else**
Actual	**Proliferative**	20	24
	Else	7	499

Accuracy	97.636%
Sensitivity	98.524%
Specificity	96.774%
Precision	96.739%
F1 Score	97.623%

The evaluation of the model's performance in classifying the most severe stage of Diabetic Retinopathy, termed "Proliferative," presents substantial challenges. Out of 44 patients who were actually suffering from this severe condition, the model incorrectly categorized 24 patients, with only one being mistakenly identified as healthy. While this misclassification rate is concerning, it is somewhat mitigated by the model's high sensitivity (98.524%) and specificity (96.774%), indicating a strong ability to recognize non-severe cases correctly. The accuracy of the model remains high at 97.636%, with a precision of 96.739% and an F1 score of 97.623%. These metrics suggest that when the model does identify a case as Proliferative, it is usually correct. However, the high rate of underdiagnosed severe cases underscores a critical need for improvement. The silver lining is that most misclassifications did not downgrade the condition to a non-severe status, which means that patients will still be flagged for further examinations, allowing healthcare providers to detect and address the severity of the condition in subsequent assessments. Nonetheless, enhancing the model's ability to accurately detect truly severe cases is imperative to ensure timely and appropriate medical interventions.

To improve the model's performance on its pain points, we could consider several steps. Firstly, incorporating more diverse datasets could enhance model generalization and robustness, reducing the risk of overfitting and improving its ability to generalize to new unseen data. Additionally, employing ensemble methods that combine the predictions of multiple models, such as DenseNet, ResNet, and InceptionV3, could significantly enhance predictive performance. We could also explore object detection techniques to enable the precise identification and localization of pathological features within the retina. Furthermore, future work could involve data augmentation and enhancement, incorporating patient metadata, and improving explainability through techniques such as multimodal learning and explainable AI. By addressing these areas, we can further improve the model's accuracy and utility in clinical practice, ultimately enhancing its performance on more severe conditions and providing more detailed reports to aid ophthalmologists in diagnosis and treatment planning.

D. Software Integration

Environment Setup

In the software integration phase, Python was selected as the primary programming language for building and deploying the deep learning models aimed at classifying retinal images for the detection of diabetic retinopathy. Several models were implemented, including CNN, InceptionV3, and DenseNet121. Google Colab was utilized as the execution environment due to its powerful computational resources, such as A100 GPUs, and its seamless integration with Google Drive for efficient dataset management. Below is a summary of the technical libraries and tools used throughout the project for data preprocessing, model training, and evaluation.

Libraries for Data Handling and Visualization

To efficiently handle data and create visualizations during the project, we used the following libraries:

- NumPy: For handling large, multi-dimensional arrays and matrices of numeric data.
- Pandas: For data manipulation and analysis, particularly for loading and processing CSV datasets.
- OpenCV (cv2): For image processing, including resizing and augmenting the retinal images.
- Matplotlib and Seaborn: For data visualization, including plotting confusion matrices, learning curves, and statistical visualizations.
- Tqdm: For displaying progress bars during data loading and processing.
- PrettyTable: For presenting tabular data in a visually appealing way.

TensorFlow and Keras for Model Development

The core of this project was developed using TensorFlow and its high-level API, Keras. These libraries were employed to build, train, and fine-tune the models, specifically:

- CNN: To build the image classification model using TensorFlow Keras layers.
- InceptionV3: A pre-trained model with transfer learning capabilities.
- DenseNet121: Another pre-trained model from the DenseNet family used for its high efficiency in extracting features from images.

Scikit-learn for Model Evaluation

Several metrics from Scikit-learn were employed to evaluate model performance:

- Confusion Matrix: To assess classification accuracy.
- Cohen's Kappa Score: To use it in computing the quadratic weighted kappa to measure the agreement between predicted and actual labels.
- Accuracy, F1, Recall, and ROC AUC Scores: To evaluate the precision, recall, and overall model performance.

Concurrent Processing for Optimized Data Loading

To optimize data loading before training, we utilized **ThreadPoolExecutor**, which allowed for concurrent processing and efficient management of input data, particularly when working with large datasets.

Library Versions and Requirements

The following libraries and their specific versions were utilized in the project:

- NumPy 1.26.4: Used for efficient numerical computations, handling large multi-dimensional arrays and matrices, essential for managing image data and tensor operations.
- Pandas 2.1.4: Employed for data manipulation and analysis, particularly for handling CSV files containing image metadata and labels.
- OpenCV (cv2) 4.10.0: Crucial for image processing tasks, including resizing and augmentation of retinal images to ensure they were ready for model input.
- TensorFlow 2.17.0: Served as the core deep learning framework, providing tools for building and training models, including CNN, InceptionV3, and DenseNet121 architectures.
- Matplotlib 3.7.1: Utilized for visualizing data, including generating plots for confusion matrices, learning curves, and model performance metrics.
- Seaborn 0.13.1: Used for advanced statistical data visualizations, particularly for creating detailed heatmaps and graphical representations of data distributions.
- Tqdm 4.66.5: Provided progress bars to monitor the status of various processes, such as data loading, image processing, and model training.
- PrettyTable 3.11.0: Used to display results and performance metrics in a tabular format, offering a clear view of model evaluation outcomes.
- Scikit-learn 1.5.2: Employed for model evaluation using metrics such as accuracy, F1 scores, QWK via Cohen's Kappa, and confusion matrices to assess model performance.
- PIL (Pillow) 10.4.0: Used for image manipulation, particularly for opening, resizing, and transforming images before feeding them into the models.
- JSON 2.0.9: Used for handling configuration files and saving model metadata, ensuring that model parameters and results were efficiently stored and retrieved.

Glossary

Term	Definition
Diabetic Retinopathy (DR)	A diabetes-related eye condition that affects the blood vessels in the retina, leading to vision impairment and potentially blindness if untreated.
Artificial Intelligence (AI)	The simulation of human intelligence in machines, enabling them to perform tasks such as learning, reasoning, problem-solving, and decision-making.
Non-Proliferative Diabetic Retinopathy (NPDR)	Early stage of DR where the blood vessels in the retina are weakened.
Proliferative Diabetic Retinopathy (PDR)	An advanced stage of DR characterized by the growth of new, abnormal blood vessels in the retina.
International Centre for Dispute Resolution (ICDR)	Established in 1996 as the global component of the American Arbitration Association, the ICDR provides conflict-management services in more than 80 countries with a staff fluent in 12 languages.
Early treatment diabetic retinopathy study (ETDRS)	The ETDRS trial was designed to assess: the efficacy of argon laser photocoagulation (both macular and scatter laser) and aspirin therapy in deterring the progression of early DR into more advanced DR. the best time to initiate photocoagulation treatment where necessary
Seamless Electronic Health Record (EHR)	A unified digital platform that integrates patient health data, ensuring continuous access and exchange of information across healthcare providers.
Optical Coherence Tomography (OCT)	A non-invasive imaging technique that uses light waves to capture detailed cross-sectional images of the retina, aiding in eye disease diagnosis.
Low- and Middle-Income Countries (LMICs)	LMICs are nations classified by the World Bank based on their gross national income (GNI) per capita. These countries often face challenges such as limited healthcare infrastructure, making it harder to address public health issues like diabetic retinopathy.
General Data Protection Regulation (GDPR)	A European Union regulation designed to protect personal data privacy, setting guidelines for data collection, processing, and storage.

Term	Definition
Asia Pacific Tele-Ophthalmology Society (APTOS)	An organization promoting telemedicine and digital solutions to improve eye care across the Asia-Pacific region.
ReLu	An activation function used in neural networks, converting negative input values to zero while maintaining positive input values.
Quadratic Weighted Kappa (QWK)	A statistical measure that evaluates the agreement between two raters or models, considering the extent of disagreement, often used in classification tasks.
Stochastic Gradient Descent (SGD)	An optimization algorithm used in machine learning that updates model parameters iteratively by sampling a small batch of data points at each step.
ImageNet Large Scale Visual Recognition Challenge (ILSVRC)	A prestigious annual competition that tests algorithms on large-scale image classification and object detection using the ImageNet dataset.

Members and Roles

Members	A.M.	Notes	Roles
Merlou Anna	**f2822307**	▪ BSc in Management, Science and Technology, AUEB ▪ Current working in Satori Analytics, as AI Engineer	1. Execution: ▪ Preproccessing ▪ Augmentation ▪ Inception V3 2. Report 3. Github Repository 4. Frontend 5. Presentation
Ziaragkalis Stavros	**f2822304**	▪ BSc in Statistics, AUEB ▪ Current working in Satori Analytics, as AI Engineer	1. Strategy 2. EDA 3. Execution: ▪ Preproccessing ▪ Augmentation ▪ Densenet121 4. Report
Koutsogianni Agapi	**f2822316**	▪ BSc in Mathematics, NUOA ▪ Current working …	1. Data Collection 2. Conceptualize the methodology 3. Setting up the environment 4. Execution: ▪ CNN ▪ Inception V3 ▪ Densenet121 5. Report
Xanalatou Athina	**f2822406**	▪ BSc in Mathematics, NUOA ▪ Current working in BGM OMD, as Adopts Strategist	1. Data Collection 2. Conceptualize the methodology 3. Execution: ▪ Preproccessing ▪ Augmentation ▪ CNN 4. Report

Time Plan and Deliverables

The current project followed a structured timeline to ensure efficient progress and thorough evaluation of different models. The plan was divided into distinct phases, each focusing on specific tasks as shown below:

- **Weeks 1-3: Data Collection & Preprocessing**
  The dataset was collected, and any issues related to missing labels and dataset size were addressed, while we explored potential business problems to solve. Subsequently, initial checks were performed to ensure the dataset was ready for further analysis.
- **Week 4: Exploratory Data Analysis**
  An initial data analysis was conducted to understand the dataset's structure, label distribution, and potential issues like class imbalance. Visualization techniques were employed to gain insights into the images, which informed key preprocessing steps later for the image preprocessing.
- **Weeks 5: Image Preprocessing**
  Images were cropped, resized, normalized, and sharpened to prepare them for training. This process was done to enhance contrast among the vessels for better classification.
- **Weeks 6-8: Model Design & Training (CNN, InceptionV3, DenseNet121)**
  The dataset was split into 70% for training, 15% for validation, and 15% for testing to evaluate model performance. Three models—CNN, InceptionV3, and DenseNet121—were implemented and trained on the training set. The validation set was used to assess the models' performance and to tune hyperparameters such as batch size, learning rate, and epochs.
- **Week 9-10: Model Evaluation & Optimization**
  The models were evaluated using metrics such as accuracy, the Quadratic Weighted Kappa (QWK) score, and a confusion matrix. Based on these results, further optimization was performed to improve the models' performance. The best model was selected based on the QWK score.
- **Weeks 11-13: Testing & Reporting**
  The final models were applied to the test dataset, consisting of unseen data, and the results were compared to evaluate the overall performance of CNN, InceptionV3, and DenseNet121. A report was drafted to document the methodology, results, and technical details of the project.
- **Week 14: Presentation Preparation**
  A project presentation was prepared. The final review of the project ensured that the key insights and results were clearly communicated.

References

(na svistoun ta hyperlink)

[1] Bastawrous, A., Giardini, M. E., Livingstone, I. A., Jordan, S., Bolster, N. M., Peto, T., & MacCormick, I. J. (2016). Results of automated retinal image analysis for detection of diabetic retinopathy from the Nakuru Study, Kenya. *PLOS ONE*, 11(10), e0164314. https://doi.org/10.1371/journal.pone.0164314

[2] Acharya, U. R., Lim, C. M., Ng, E. Y. K., Chee, C., & Min, L. C. (2009). Diagnosis of diabetic retinopathy based on holistic texture and local retinal features. *Expert Systems with Applications*, 36(4), 9634–9641. https://doi.org/10.1016/j.eswa.2008.10.012

[3] Li, Y., Shen, L., Hu, J., & Liu, Y. (2019). Referable diabetic retinopathy identification from eye fundus images with weighted path for convolutional neural network. *IEEE Access*, 7, 151601–151611.

[4] https://doi.org/10.1109/ACCESS.2019.2947163

[5] Yan, Z., Yang, F., Wen, J., & Wu, Q. (2020). Multi-stream deep neural network for diabetic retinopathy severity classification under a boosting framework. *IEEE Transactions on Artificial Intelligence*, 1(2), 1-12 https://www.researchgate.net/publication/342933701_Multi-Stream_Deep_Neural_Network_for_Diabetic_Retinopathy_Severity_Classification_Under_a_Boosting_Framework

[6] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22), 2402–2410. https://doi.org/10.1038/s41467-019-09587-9

[7] Rahman, M. F., Hossain, F. K., & Billah, A. S. M. (2020). Severity classification of diabetic retinopathy using an ensemble learning algorithm through analyzing retinal images. *Applied Sciences*, 10(24), 1-12. https://doi.org/10.3390/app10249761

[8] Cleland, C. R., Rwiza, J., Evans, J. R., Gordon, I., MacLeod, D., Burton, M. J., & Bascaran, C. (2023). Artificial intelligence for diabetic retinopathy in low-income and middle-income countries: A scoping review. *BMJ Open Diabetes Research & Care*, 11(4), e003424. Artificial intelligence for diabetic retinopathy in low-income and middle-income countries: a scoping review | BMJ Open Diabetes Research & Care

[9] Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv*, 1608.06993v3. https://doi.org/10.48550/arXiv.1608.06993

[10] Ismail, F., Lee, M. H., Isa, D., & Rajamony, S. (2018). Medical image segmentation using the fast-marching algorithm: A study of digital retinal images. *Information Sciences*, 454-455, 1-13. https://doi.org/10.1016/j.ins.2018.02.060

[11] Abdolmanafi, A., Zhang, Y., Rashidi, T., & Taylor, S. (2020). Macular disease classification based on optical coherence tomography images using deep convolutional neural networks. *Biomedical Signal Processing and Control*, 57, 101675. https://doi.org/10.1016/j.bspc.2020.101675

[12] American Academy of Ophthalmology. (2022, April 1). What is diabetic retinopathy? *American Academy of Ophthalmology*. https://www.aao.org/eye-health/diseases/what-is-diabetic-retinopathy

[13] Rahman, M. S., & Mukhopadhyay, S. C. (2018). Medical image segmentation using the fast marching algorithm: A study of digital retinal images. *Information Sciences*, 454-455, 1-13. https://doi.org/10.1016/j.ins.2018.02.060

[14] Ahmad, S., Liu, W., Iqbal, W., Hanif, M. S., & Zaman, F. (2023). Identification of diabetic retinopathy from retinal fundus images using deep convolutional neural networks. *Applied Sciences*, 13(5), 3108. https://doi.org/10.3390/app13053108

[15] Zhao, L., & Wu, S. (2023). Deep learning models for computer vision. *arXiv*, 2306.01289v4. https://doi.org/10.48550/arXiv.2306.01289

[16] Li, X., & Zhang, Y. (2023). Automated macular disease classification using attention-based deep learning. *arXiv*, 2304.07461v2. https://doi.org/10.48550/arXiv.2304.07461

[17] Islam, F., Lee, M. H., Isa, D., & Rajamony, S. (2018). Medical image segmentation using the fast marching algorithm: A study of digital retinal images. *Information Sciences*, 454-455, 1-13. https://doi.org/10.1016/j.ins.2018.02.060

[18] Srinivasan, P. P., Kim, L. A., & Witkin, A. J. (2019). Optimized deep convolutional neural networks for identification of macular diseases from optical coherence tomography images. *ResearchGate*. https://doi.org/10.1016/j.compmedimag.2019.02.004

[19] Langer, R., & Tirrell, D. A. (2017). Design and application of synthetic materials in biology and medicine. *Annual Review of Biomedical Engineering*, 19, 1-28. https://doi.org/10.1146/annurev-bioeng-071516-044442

[20] Yim, J., Zhao, C., & Brais, S. J. (2021). Convolutional neural networks for macular disease diagnosis using OCT images. *Nature Communications*, 12(1), 3471. https://doi.org/10.1038/s41467-021-23458-5

[21] Virdee, J. K., Thomas, D., Chong, M., & Dasgupta, K. (2017). Evaluating the efficacy of convolutional neural networks for diabetic retinopathy screening: A systematic review and meta-analysis. *BMJ Open*, 7(9), e016280. https://doi.org/10.1136/bmjopen-2017-016280

[22] TensorFlow. (n.d.). Adam optimizer. *TensorFlow API Documentation*. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

Appendices