**Eye Disease Recognition Using CNN**

**Group Name:**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **First name** | **Last Name** | **Student number** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Submission date:**

Table of Contents

[**Abstract** 4](#_Toc174455666)

[**Introduction** 4](#_Toc174455667)

[Literature Review 5](#_Toc174455668)

[Methods and Results 5](#_Toc174455669)

[Conclusion 6](#_Toc174455670)

[**Methods** 6](#_Toc174455671)

[**Data Collection** 7](#_Toc174455672)

[**Image Preprocessing** 7](#_Toc174455673)

[**Model Building** 7](#_Toc174455674)

[CNN Architecture 7](#_Toc174455675)

[ResNet 10](#_Toc174455676)

[**Experiments** 10](#_Toc174455677)

[**Experiment 1: Simple CNN with Original Image Size** 11](#_Toc174455678)

[**Experiment 2:** **Simple CNN with Different Image Sizes** 11](#_Toc174455679)

[**Experiment 3: CNN with Hyperparameter Tuning** 12](#_Toc174455680)

[**Experiment 4: ResNet50 (pre-trained model)** 12](#_Toc174455681)

[**Deployment and GUI** 13](#_Toc174455682)

[**Results** 14](#_Toc174455683)

[**Experiment 1: Simple CNN with Original Image Size** 16](#_Toc174455684)

[**Experiment 2: Simple CNN with Different Image Sizes** 18](#_Toc174455685)

[Image size (128 x 128) 18](#_Toc174455686)

[Image size (224 x 224) 20](#_Toc174455687)

[Image size (448 x 448) 22](#_Toc174455688)

[**Experiment 3: CNN model with Hyperparameter Tuning** 24](#_Toc174455689)

[**Experiment 4: ResNet50 (pre-trained) Model** 28](#_Toc174455690)

[Image size (128 x 128) 28](#_Toc174455691)

[Image size (224 x 224) 29](#_Toc174455692)

[Image size (448 x 448) 31](#_Toc174455693)

[**Conclusions** 34](#_Toc174455694)

[**Future Work** 34](#_Toc174455695)

[**References** 35](#_Toc174455696)

[Figure 1: CNN Architecture 8](#_Toc174455583)

[Figure 2: Max Pooling Layer 9](#_Toc174455584)

[Figure 3: The Initial Phase of GUI 14](#_Toc174455585)

[Figure 4: Result 14](#_Toc174455586)

[Figure 5: After uploading image 14](#_Toc174455587)

[Figure 6: Graph of Loss and Accuracy Curve for Simple CNN Model 16](#_Toc174455588)

[Figure 7: Classification report for the best model in Experiment 1 17](#_Toc174455589)

[Figure 8: Confusion Matrix for the best model in Experiment 1 18](#_Toc174455590)

[Figure 9: Graph of Loss and Accuracy Curve for Simple CNN model for Image size (128 x128) 19](#_Toc174455591)

[Figure 10: Confusion matrix for CNN (128 x 128) 19](#_Toc174455592)

[Figure 11: Classification report for CNN (128 X 128) 19](#_Toc174455593)

[Figure 12: Graph of Loss and Accuracy Curve for Simple CNN model for Image size (224 x224) 21](#_Toc174455594)

[Figure 13: Confusion matrix for CNN (224 X 224) 21](#_Toc174455595)

[Figure 14: Classification report for CNN (224 X 224) 21](#_Toc174455596)

[Figure 15: Graph of Loss and Accuracy Curve for Simple CNN model for Image size (448 x 448) 23](#_Toc174455597)

[Figure 16: Confusion matrix for CNN (448 X 448) 23](#_Toc174455598)

[Figure 17: Classification report for CNN (448 X 448) 23](#_Toc174455599)

[Figure 18: Graph of Loss and Accuracy Curve for hyperparameter CNN tuned model 25](#_Toc174455600)

[Figure 19: Confusion matrix of model’s classification ability 25](#_Toc174455601)

[Figure 20: Graph of Loss and Accuracy Curve for hyperparameter tuned model 26](#_Toc174455602)

[Figure 21: Confusion matrix for hyperparameter tuned CNN (224 X 224) 27](#_Toc174455603)

[Figure 22: Classification report for hyperparameter tuned CNN (224 X 224) 27](#_Toc174455604)

[Figure 23: Graph of Loss and Accuracy Curve for ResNet50 model (124 x 124) 28](#_Toc174455605)

[Figure 24: Confusion matrix for ResNet model (124 x 124) 29](#_Toc174455606)

[Figure 25: Classification report for ResNet model (124 x 124) 29](#_Toc174455607)

[Figure 26: Graph of Loss and Accuracy Curve for ResNet50 model (224 x 224) 30](#_Toc174455608)

[Figure 27: Confusion matrix for ResNet model (224 x 224) 30](#_Toc174455609)

[Figure 28: Classification report for ResNet model (224 x 224) 30](#_Toc174455610)

[Figure 29: Graph of Loss and Accuracy Curve for ResNet50 model (448 x 448) 31](#_Toc174455611)

[Figure 30: Confusion matrix for ResNet model (448 x 448) 32](#_Toc174455612)

[Figure 31: Classification report for ResNet model (448 x 448) 32](#_Toc174455613)

# **Abstract**

Vision is vital as it allows individuals to interpret and interact with their environment. However, taking proactive measures to maintain healthy eyes is often neglected, resulting in avoidable visual impairments that may go unnoticed until great harm occurs.

In this project, we tackle the dilemma of detecting vision-threatening illnesses by creating a Convolutional Neural Network (CNN) model that can accurately diagnose ocular ailments using fundus images. The focus is on the high prevalence and significant impact of vision ailments, including Diabetic Retinopathy, Cataracts, and Glaucoma. The project has used ResNet-50 architecture for image classification problems, such as diagnosing ocular diseases in similar eye images to classify visually alike but diseased eyes into different categories.

The goal is to classify eye images for diagnosing these diseases using an application built with Streamlit for image uploading and classification. The Agile methodology enabled iterative development across tasks like problem definition, data collection, preprocessing, model building, evaluation, and deployment.

Considering the increasing occurrence rates of eye diseases as well as the urgent need for early detection, it is evident that developing an AI-based diagnosis model is no longer only an avenue for technological advancement but also an obligatory call to action. It can detect disease at an early stage, significantly reduce the incidence of preventable vision loss, improve the quality of life for millions of individuals worldwide, and reduce the global burden of preventable blindness.

# **Introduction**

The ability to see is one of man’s most essential abilities; through sight, one can gather valuable insight into one’s surroundings and make data-driven decisions. Though the ability to see is not often seen in this light, it is the fundamental basis for developing the camera or any other sensor device that captures or streams images. However, maintenance and care for the eyes are often ignored or delayed until there is a loss of sight wholly or partially.

According to (World Health Organisation, 2019), “at least 2.2 billion people have a vision impairment, and of these, at least 1 billion people have a vision impairment that could have been prevented or is yet to be addressed.” Though not the sole reason, it can be inferred that some preventable vision loss goes undiagnosed due to economic constraints. A recent report by (UNDP (United et al.), 2023) indicated that of the 6.1 billion people in the world the study covered, 1.1 billion are poor. The report further describes the dimensions of poverty faced by individuals globally and even those who fall beneath the poverty line.

That being said, there is a need to provide an accessible means by which marginalized individuals can have their eyes assessed for some of the common diseases associated with loss of vision and potentially receive help early to address them. (World Health Organisation, 2019) reported that most individuals who receive early diagnosis and treatment do not suffer vision loss.

This project's scope will seek to build a model using a Convolution Neural Network (CNN) to correctly diagnose (via classification) which eye disease is present in the patient's fundus images. Diseases of focus are the most popular and are known for their association with vision loss in the eyes. These diseases are *Diabetic Retinopathy, Cataract, and Glaucoma.*

The dataset used in training the model was retrieved from Kaggle and consisted of 14,400 fundus images, a CSV file that contained 19 features, and 6323 records.

### Literature Review

Currently, the diagnosis of the different types of ocular disease is conducted by a medical professional who physically examines the eye or examines special photographic scans of the eye(s). It is well understood that this method of diagnosis is not the most accurate or precise, with human error, either by executing poor medical diagnostic techniques or imagery unable to adequately capture the nuances of the eye, leading to incorrect diagnosis.

Over the last decade, academia has been interested in using deep learning models such as CNN as a means of providing accessible and accurate methods for diagnosis. This also would alleviate the resource constraints of medical systems in both developed and developing countries. The development and advancement of pre-trained models have seen these techniques used to address the problem, with each research focusing on a different range of diseases.

One such research (Deepak & Bhat, 2024) recognized a gap in previous works and decided to tackle the problem from two perspectives. According to (Deepak and Bhat, 2024), there is a need for further studies into the multi-classification of eye ocular disease, in particular glaucoma and cataracts. Additionally (Deepak and Bhat, 2024) have recognized from their research that further research is required into the effects of batch size and optimizer type on the accuracy of the classification deep learning models; theoretically, these two parameters could potentially be critical to improved predictive power.

### Methods and Results

The methodology of the research conducted by (Deepak & Bhat, 2024) used three pre-trained CNN models (*SqueezeNet, Darknet-53, and EfficientNet-b0) trained on ocular eye disease images while experimenting with parameters batch size and optimizer* type for enhanced model performance. The researchers ensured validation frequency, maximum epochs, and learning rate were kept constant through the experiments and found that *Darknet-53,* with a batch size of 6 and optimizer type Stochastic Gradient Descent with Momentum, yielded the highest accuracy, at 99.4%.

(Glaret Subin & Muthukannan, 2022) Also, it is agreed that hyperparameter tuning will significantly impact the model's predictive power during training and learning. This research tackled the classification of multiple eye diseases by preprocessing the fundus images using maximum entropy transformation, optimizing CNN through implementing flower pollination optimization algorithm (FPOA) for classification, and the use of multiclass support vector machine (SVM) for classification of multiple eye disease. The best results were accomplished with training executed with a learning rate 0.01 and an epoch set at 26.

The success of CNNs in single-classification models has spurred the academic world to explore the potential of multi-class classification further. (Yang & Yi, 2022) shared similar

sentiments, citing that many existing systems are designed for single disease classification, possibly due to the limitation of available fundus image data and the complexity of the fundus images. “One fundus image is likely to contain multiple fundus diseases, which greatly increases the complexity and difficulty of judging eye diseases,” according to (Yang & Yi, 2022).

The methodological approach to this multi-classification problem by (Yang & Yi, 2022) included preprocessing the images for improved quality using data screening, black border cropping, data amplification, and normalization. The processed images are then fed to the DSRA-CNN network for feature extraction and passed to a classifier multiclass classification. (Yang & Yi, 2022) reported that when DSRA-CNN was implemented before images were sent to a CNN classifier, the accuracy increased by 10.83% (accuracy yield 87.90 %) compared to the lowest-performing classifier (accuracy yield 77.07%). This demonstrated the network's improved ability to distinguish complex ocular diseases.

### Conclusion

The literature review has revealed that a critical challenge with the current methods of diagnosis through medical professionals is the use of visual assessment of the physical eye, which can be erroneous due to the complexity of how the disease manifests in the eye. Additionally, the eye may be impacted by multiple morbidities, which further increases the difficulty in diagnosis and the error in assessments. Though it is agreed that the use of deep learning techniques can and will combat this, there is a need for more focus on preprocessing techniques and feature extraction techniques to improve the model's ability to accurately diagnose a multiclass ocular disease.

This project seeks to contribute to the academic community by further exploring using CNN models to address the multi-classification problem by experimenting with image resolution and hyperparameter tuning for improved predictive power.

# **Methods**

The project aim is to classify the eye images to identify which eye is suffering from an ocular disease. The project provides a graphical user interface where images can be uploaded to the model, which then classifies the images according to the disease present. The technologies utilized in constructing the model were CNN for model development and Streamlit framework for model deployment.

This project employed the Agile methodology for the process of implementation and deployment. The flow of the project follows the software development lifecycle. The Basic framework of the requirement the project tries to accomplish is divided into multiple tasks. Tasks performed during the development lifecycle are Defining Problem Statement, Literary Review, Data collection, Data preprocessing, Model Building, Model evaluation and testing, and Deployment.

The rationale for selecting the Agile methodology is due to its critical concepts like quick fallback points and reiterating steps until the desirable goal is achieved, which is required to bring about the project. Since the classification of the images is not a straightforward method where a model is implemented, the requirement is achieved. It is the concept of utilizing the multiple layers of the Convolution Neural Network and experimenting with the parameters of the layers. This process must be iterative to do the experimentation, evaluate the experiment results, and re-iterate for further optimization until the desired objective is attained.

## **Data Collection**

The dataset used for this project included 14,400 fundus images retrieved from Kaggle’s web page titled *Ocular Disease Recognition.* The dataset has been cited in two research papers mentioned in the above literature review (Deepak & Bhat, 2024) and (Yang & Yi, 2022) respectively. The dataset simulates a medical database with 5000 patients at the Shanggong Medical Technology Co., Ltd. in China. This was potentially done on ethical grounds to protect the actual patient's personally identifiable data.

The fundus images were captured using standard market cameras from brands such as Canon, Seiss, and Kowa; this undoubtedly would have resulted in various image quality. The dataset provides 8 main classes for classifying the disease in the left and right eye, which are Diabetes(D), Glaucoma(G), Cataract(C), Age-Related Macular Degeneration (A), Hypertension(H), Pathological Myopia (M) and other diseases/abnormalities (O). (Maranhão, 2020) reported that the image for each eye was annotated with the corresponding class labels by trained human readers.

## **Image Preprocessing**

While doing a medical image classification analysis, preprocessing plates play a vital role in making the models produce accurate and impartial outcomes. The project comprises classifying the eye images into different categories based on the disease they may be diagnosed with.

To begin with, the eye images in the dataset were classified into three groups. This decision was influenced by the fact that eye images, especially the left and right eye images in the dataset, are pretty similar. If these images are to be classified into more categories, this leads to undesirable biases in the model's functioning.

The images have default labels; these labels are matched with the three disease categories and assigned to those categories correspondingly if a match is found. Some images that do not fit these categories were initially labeled “other categories.” However, this bale has caused the class imbalance, so it was dropped to put more emphasis on the three categories selected.

## **Model Building**

### CNN Architecture

The project involves working on images. So, making images compatible and easily understandable to the model is the baseline before going further. So, the initial goal is to decide on the best algorithm to analyze the images. Traditional algorithms like SVM and random forest are not up to the mark when dealing with image data as they have severe drawbacks like manual feature extraction, scalability issues when given with data-heavy objects like images, and Disability to handle muti-dimensional data, which is what vectors from the image upon extraction appertain to(Lin, 2024).

A convolutional Neural Network is a deep learning technique developed for specialized tasks like image classification, Object recognition, Image detection, and segmentation. To overcome the drawback of the Traditional machine learning models while working with image data, CNN employs several techniques like Automated feature extraction, utilizing multiple convolutional layers to identify and extract patterns from the data irrespective of the position, orientation, scale, and translation of the image (Kelta, 2023).

To better understand the workings and architecture of CNN, research done by (Alzubaidi, L 2021) on Deep learning concepts is referred to, regarding the structure of CNN (Alzubaidi, L 2021) stated that “the structure of CNNs was inspired by neurons in human and animal brains, similar to a conventional neural network.” CNN employs weights and local connections to fully use 2D input data structures like images (Alzubaidi, L 2021).

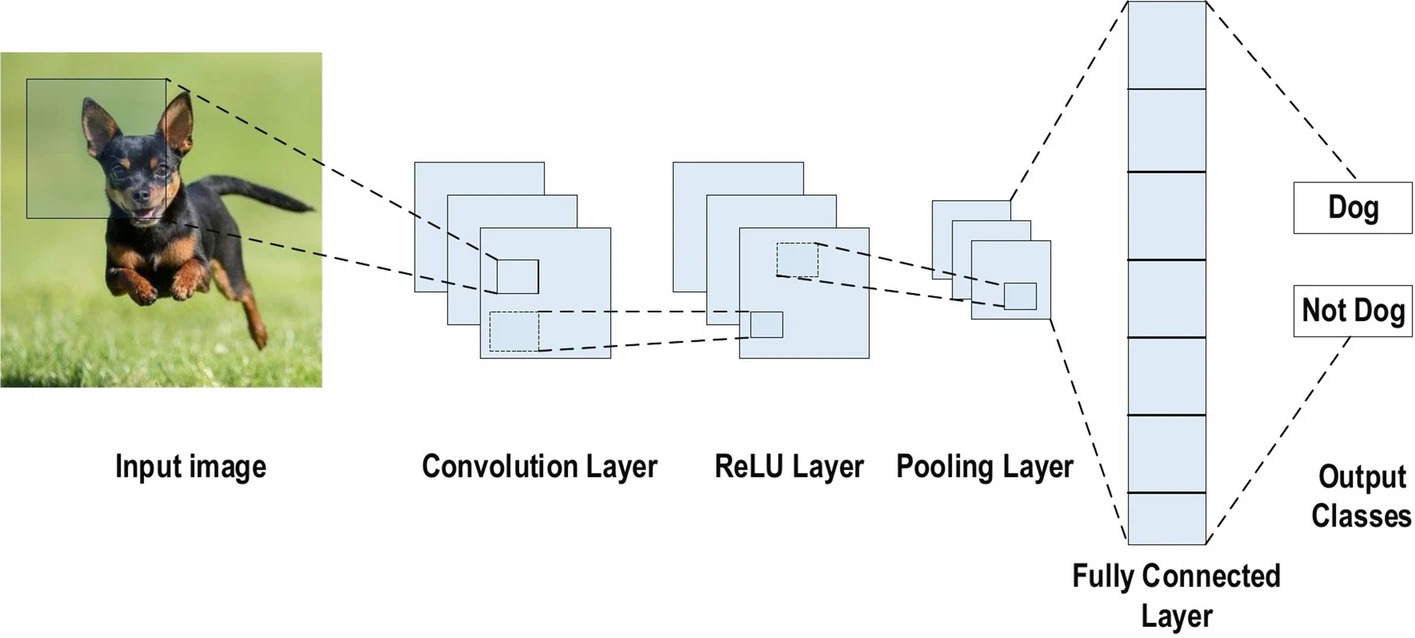


Figure 1: CNN Architecture

**Convolutional Layers**

In the first layer, images are fed to the convolution layers. It contains a sliding window function to an image matrix called Kernel or filter; multiple filters of equal size will applied to the image; these filters act as grids and flow through the image, looking for the patterns where each filter based on the layer kernel parameters look for different information (Kelta, 2023). for our custom simple CNN, three convolutional layers were designed and implemented, all having the kernel size of 3\*3, as all the filters must be of equal size. The first convolutional layer, ‘conv1’, takes the inputs the size of 3 to capture the RGB channels of the image. The padding is set to 1 for all three convolutional layers, which ensures that the output dimensions of the layers are preserved so that they will be given to the next layer (Alzubaidi, L 2021).

**Activation function**

The output of the convolutional layers will be linear; due to this, the model will not be able to learn complex patterns and relations. So, an activation function layer will be introduced after the convolutional layer, which will convert linear output to non-linear. Relu activation is adapted for the model as it reduces the computational load and optimizes efficiency better than other activation functions like step and sigmoid (Kizrak, 2019). The Relu function is applied to all three convolutional layers.

**Pooling layer**

After the Relu activation layer is applied to each convolutional layer, it is given to the pooling layer. Each convolutional layer has its respective pooling layer. The main aim of the pooling layer is to retrieve the most significant features from the input. This is achieved with the help of aggregation functions, which reduce the dimensions of the feature matrix and, in turn, help reduce the computational load and memory the network uses while training (Kelta, 2023). The max pooling aggregation function is used for all three pooling layers in the model, and stride is selected as 2. This will select a 2\*2 matrix from the input and take only max values from the stride, thereby reducing the feature map of the image.



Figure 2: Max Pooling Layer

**Fully connected layer**

According to (Alzubaidi L 2021), the fully connected layers are “Located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so-called Fully Connected (FC) approach.” The first fully connected layer in the network takes inputs from the convolutional layers. The activation function, which is 3D (128 \* 16 \*16), flattens the input into 1D (512) for the output to be simple, and the model can be classified easily. The second FC layer maps the output of the first layer (512 features) to the classes of the output, corresponding to the number of categories in the classification task.

### ResNet

A diagram of a weight layer

Description automatically generatedCompared to prior networks, ResNet's goal was to create an ultra-deep network that mitigates the vanishing gradient problem. As CNN already trains the model, ResNet was used to delve into deeper networks using residual or skip connections. The various types of ResNets can be categorized according to their layer counts. The most common form is ResNet50, consisting of 49 convolutional layers and one fully connected layer. Its weight is 25.5 million, while the MACs are 3.9 million (2015). Highway Nets proposed the basic idea behind residual connections, which gave birth to an innovation called ResNet. It comprises a classical feed-forward network and a skip or a shortcut connection between them within a single block (a group of units). Thus, one may call the output from this tame as the input layer’s previous outputs. This layer may also be a convolution using filters with various sizes or normalization, such as batch normalization, and activation functions, such as ReLU afterward, producing this output. Finally, mathematically, it could be described by all these premises contained in the equation:

x1 = F(x1-1) + x1 - 1

Here, The ResNet-50 model’s final fully connected layer (FC) is adjusted to match the number of classes in the dataset. By default, ResNet-50 outputs predictions for 1,000 ImageNet classes. This modification ensures that the model can output predictions for the specific number of classes in the dataset.

# **Experiments**

As part of the project for classifying the eye image data for diseases, reiterating steps have been followed to employ different techniques and improve the model and methods.

Initial Trail: A Support Vector Machine (SVM), a basic Machine Learning model, was first employed to evaluate performance. While SVMs are recognized for their accuracy in binary classification challenges, this experiment showed severe inadequacies regarding image data. Moreover, it performed poorly due to a lack of GPU optimization, resulting in prolonged processing to the point where the model could not complete at least one epoch within 20 minutes, which exemplified the issue with image data processing. This meant SVM would have significant hurdles when processing image data computations. This is most likely because it relies on the CPU rather than the GPU, which performs better in parallel processing tasks such as deep learning. Even though insights gained from the SVM trial helped understand the data structure and establish a comparison benchmark, they also showed why moving toward advanced models such as convolutional neural networks (CNNs) is necessary. This makes them much more suitable for this project since CNNs are designed to manage large-scale image data through efficient parallel computation.

## **Experiment 1: Simple CNN with Original Image Size**

A SimpleCNN model was created using original image sizes to establish a baseline for comparison purposes. The SimpleCNN architecture consisted of convolutional layers for feature extraction, a max pooling layer to reduce feature map size, and fully connected layers to map features into output categories. Initially, this model had problems with images' varying dimensions (height and width). Every image should be of particular pixels to be fed into the model; otherwise, because of the dimension mismatch, the code will crash and will not be able to run the code further.

These were solved by flattening outputs from convolutional layers and changing input size dynamically at the fully connected layer. i.e., Flattening the output of the convolutional layers to get a fixed-size feature vector and dynamically adjusting the input size of the first fully connected layer to accommodate the varying size of this feature vector. This resolved the issue of dimension mismatch and allowed the model to be run.

The model’s capacity was measured using k-fold cross-validation, thereby showing increased training and validation accuracy, though some overfitting was noticed within one of the folds. Nonetheless, despite these hurdles encountered, simple CNN showed promising learning abilities.

## **Experiment 2:** **Simple CNN with Different Image Sizes**

Concerning the input image shape in classification problems, the structure of the input or, more precisely, the input image size affects the CNNs significantly. The source of images in the project was the ‘eye image dataset’ that contained images of random sizes; these may have been inconvenient in the CNN and ResNet architectures necessary for the project. As such, one of the things that was done was to rescale the images to other sizes relevant to the models and hyperparameter-tune them.

Three resized versions were considered to train the models upon which are 128x128, 224x224, and 448x448; the main reasons behind these size considerations were as follows: the 128 X 128 size was selected to reduce computational load on the model as the image data taken for the project is very large, and the original data size of the images is enormous. This resized version is smaller than the original; this will help the model recognize the pertinent features and patterns rather than focus on the delectable features, leading to overfitting (Howard et al., 2017).

The 224 X 224 is a reliable standard for the many pre-trained models of the CNN, such as ResNet, because of its computation efficiency while also retaining enough details from the image; the 448 x 448 size is large enough to grasp the granular and finer details in the image which is very important while dealing with medical data (He et al., 2016). The K-fold cross-validation is employed in this experiment to validate the model's performance. All three resized versions were employed on the Simple CNN model, which was implemented for the project.

## **Experiment 3: CNN with Hyperparameter Tuning**

After developing the Simple CNN model, it is very important to hyperparameter-tune it to improve its performance. The Grid search method is implemented to identify the better set of parameters to tune the model. This was done using a GridSearchCV procedure that aimed to identify the best hyperparameters for the SimpleCNN model, which sequentially selects the parameters manually from the given set of parameters grid. In doing so, this systematic search will help determine which parameters' settings make them work better.

When training the deep learning models for image classification, regularization is Crucial to mitigate overfitting, especially when working with medical images that are high dimensional. One such is dropout, one of the most commonly used regularization techniques in which arbitrary units are removed from the neural network during training to build a generalized model(Srivastava et al., 2014). Dropout rates of 0.3 and 0.5 were taken as one of the grid parameters in the training phase, where 30 percent and half of the neurons in the network are masked to eliminate contribution, respectively. Another parameter that is considered is the learning rate. As stated (Brownlee. ,2020), “The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

Choosing the learning rate is challenging as a value too small may result in a lengthy training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process”. Two learning rates were considered parameters, 0.001 and 0.01, representing the optimal and more aggressive learning steps. The third parameter utilized for parameter tuning is Batch size. It is an important hyperparameter tuning technique while training the models, as it arbitrates the number of training samples processed in each iteration before the model’s parameters are updated (Masters & Luschi, 2018). Two batch sizes were considered for the parameter grid, which are 16 and 32, respectively. The primary rationale behind choosing 16 and 32 is that batch size 16 is small enough to introduce noise and can prevent the model from becoming overfit, and 32 speeds up the training process efficiency and reduces the computational load on the model (Masters & Luschi, 2018).

## **Experiment 4: ResNet50 (pre-trained model)**

As with the Simple CNN model, which was trained on the three different images in experiment 1, experiment 4 involves using the Resnet50 pre-trained model to train on the same resized images. ResNet50 is considered for this experiment as it is a deep convolutional neural network of 50 layers and is quite complex compared to simple CNN networks, which usually contain fewer layers. The residual blocks of ResNet50 enable it to take into a deeper hierarchy of the image and enhance the features it learns from the image. This capability is instrumental in medical diagnosis through imaging, where slight differentiation can mean between right and wrong diagnosis (He et al., 2016).

# **Deployment and GUI**

The deployment and hosting of “Eye Image Classification,” a specialized ResNet-50 model fine-tuned to classify eye images, are streamlined by Streamlit, ensuring accessibility and efficiency. Streamlit - a framework that simplifies the design and implementation of interactive web applications. It is ideal for hosting the model while providing a comparable user interface, making the app scalable and easy to maintain. It allows deployment on cloud platforms and takes care of backend operations like model inference and image processing while providing an intuitive frontend for users. Streamlit allows machine learning models to integrate with user-friendly interfaces easily.

The “Eye Image Classification” app is accessible through the link <https://eye-disease.streamlit.app>, where users can interact with the model and classify eye images.

The environment is prepared by installing all dependencies, including the pre-trained model file. After thorough local testing to verify functionality, the app is deployed to Streamlit Cloud. Streamlit Cloud makes this more accessible since it automatically handles the installation of dependencies and app hosting. Users interact with the app through its interface by uploading eye pictures to get immediate and accurate classifications. Thus, streamlit’s seamless deployment and hosting capabilities provide users a smooth experience.

The Eye Image Classification app's Graphical User Interface (GUI) is thoughtfully designed to provide a seamless and intuitive user experience. It is a simple interface. A clear and prominent title on the top communicates the app’s purpose: “Eye Image Classification.” A significant aspect is its file uploader, which users can find extremely easy to navigate; here, different formats are accepted (jpg, jpeg, png), thus making it simple to upload images that require going through analysis.

When one uploads an image, it appears on the screen, allowing them to check if they chose correctly before moving on with other tasks. The next thing they do is to press the “Classify Image” button that is adequately marked. The app processes quickly and provides the result, i.e., a statement as to whether or not there are indications of cataract, diabetes, or glaucoma in the eye. It attains responsiveness by automatically re-sizing according to any given screen dimensions, making it possible for laptops and tablets, besides mobile devices, to have similar interface and user-friendliness at all times. Combining straightforward navigation with fundamental features makes an app suitable for every technical user level.



Figure : The Initial Phase of GUI

A black screen with white text

Description automatically generatedA screenshot of a cellphone

Description automatically generated

Figure : Result

Figure : After uploading image

# **Results**

This project's scope will seek to build a model using Convolution Neural Network (CNN) to correctly diagnose (via classification) which eye disease is present in the patients' fundus images. This project seeks to contribute to the academic community by further exploring using CNN models to address the multi-classification problem by experimenting with image resolution and hyperparameter tuning for improved predictive power.

The investigation into the ideal hyperparameters and dimension resolution for developing a stable and accurate classification model revealed some exciting results, which can be used to enhance the model further. The adjustment of image size was included in the experiment as this significantly impacts the resolution of an image. A large image may lose some information or detail, which could aid in the classification process, while for a smaller image, increasing the size could help enlarge some details at the cost of lower quality.

In product development, the optimal point is usually a decision made around trade-offs; this must be balanced in the context of the problem trying to be resolved. This project addresses the medical challenge of accessible and accurate diagnosis of multiple ocular diseases. When it comes to medical diagnosis, high accuracy is always a key metric that is assessed for; however, equally important is a diagnostic tool's precision and the accuracy of classifying class labels correctly.

The latter speaks to the results generated from a confusion matrix, commonly known as true positive, true negative, false positive, and false negative results. For this project, the ideal metrics would be for the model to have high accuracy and high precision as well as a high number of correctly classifying, for example, glaucoma as glaucoma and a low number of incorrectly classifying glaucoma as usual or some other type of of of eye disease.

(Frost, 2021) accuracy indicates that a series of measurements are correct on average; it does not say how close measurements are to the target value, while precision indicates how close values are together; in other words, it indicates the reproducibility or reliability of the results. Thus, a diagnostic system needs to be accurate (in statistical terms, low bias) and precise in diagnosing whether a patient has or does not have a particular disease. For further confidence in the confusion matrix results, the key focuses in this project are true positive and false negative. A true positive result reflects a sample correctly identified as a positive sample, and a false negative result reflects a sample incorrectly identified as a negative but positive sample.

It is apparent that for this model and for the context of the problem to be solved, the ability to correctly and reliably diagnose the correct disease is critical. Throughout this section, a more detailed examination of the results generated from each experiment will be provided. The model's performance, whether overfitting or underfitting, will be assessed using the model loss curve.

(IBM, 2019) states that “overfitting occurs when an algorithm fits too closely or even exactly to its training data, resulting in a model that cannot make accurate predictions or conclusions from any data other than the training data.” Another way to understand this concept is to evaluate the accuracy from the training and test phase; if the training accuracy is higher than the test, it indicates overfitting. Conversely, underfitting indicates a model’s inability to capture the relationship between the input and output values, which results in a high error for the training and test phase (IBM, 2024).

Thus, using the model loss curve is another critical visualization in assessing how the various adjustments impact the model’s performance regarding overfitting and underfitting. Whether the model overfits or underfits, this suggests an inability to learn during training and make good generalizations on unseen data. According to (IBM, 2019), an underfit model has high bias and low variance while an overfit model has low bias but high variance; the aim is to identify the optimal point for bias/variance tradeoff. This optimal point suggests that the model can adequately apply good generalizations on unseen data.

## **Experiment 1: Simple CNN with Original Image Size**

**Simple CNN Model**

This code implements a CNN training and evaluation process using K-Fold Cross-Validation to ensure the best model performance. The dataset is split into 5 folds, each serving as a validation set once while the remaining folds are used for training. For each fold, a new CNN model is initialized and trained over 10 epochs, tracking training and validation accuracy and loss. The best-performing epoch within each fold is identified, and the model's state and classification report are saved.

Loss and accuracy curves are plotted for each fold, focusing on analyzing these graphs for the best-performing fold. After evaluating all folds, the model with the highest validation accuracy across folds is selected as the final model. This model is saved, and its classification report and the confusion matrix are presented, ensuring the best possible generalization to unseen data.

Table 1: Matrix showing performance across the K-Fold cross-validation for Exp 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/9 | 73 | 71 | Overfit |
| 2 | 2/10 | 77.68 | 78.34 | Good |
| 3 | 3/10 | 77 | 70 | Overfit |
| 4 | 4/7 | 69 | 71 | Underfit |
| 5 | 5/10 | 73 | 74 | Good |

The best model comes out to be Fold 2/Epoch 10, and this model is saved.

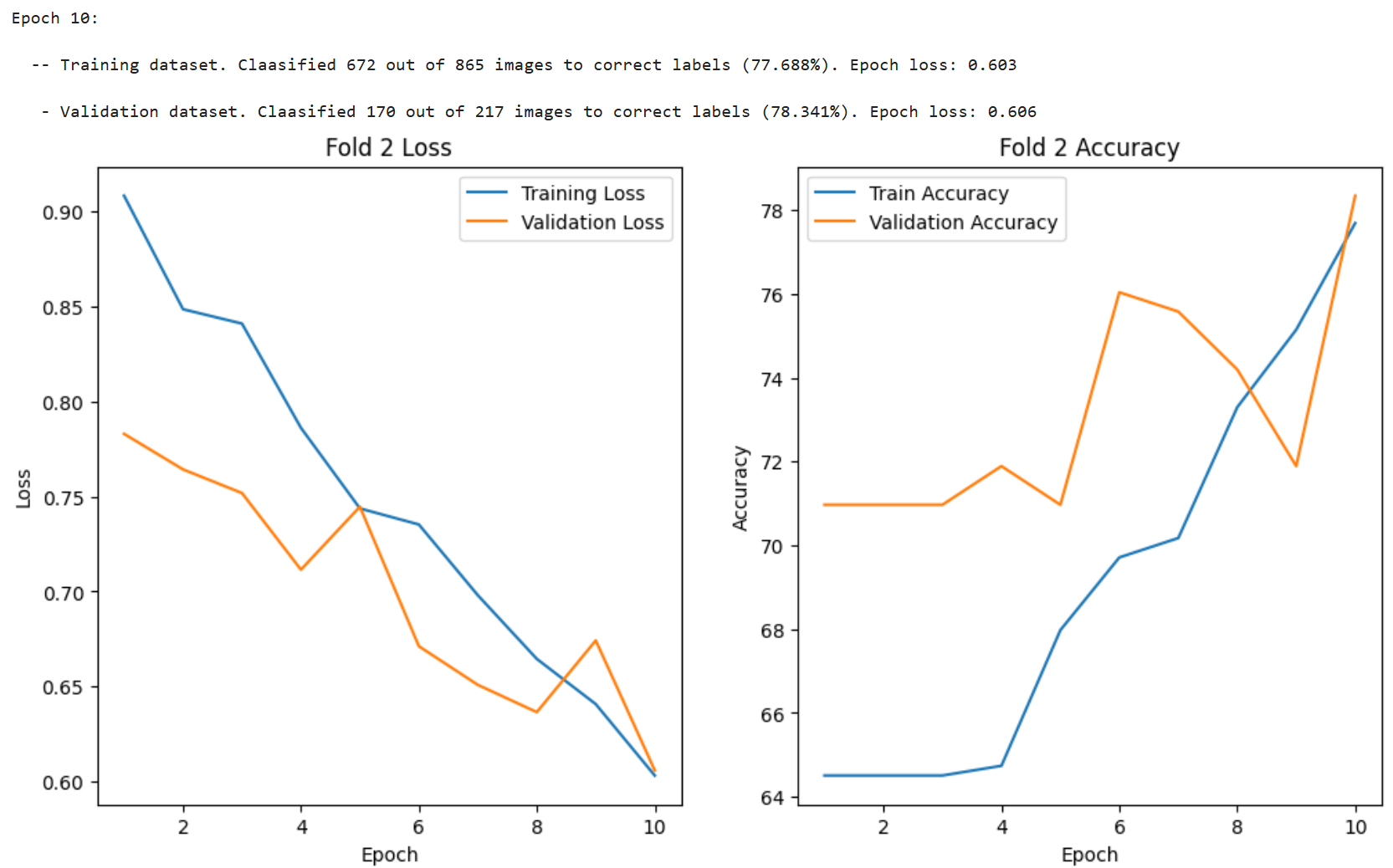


Figure : Graph of Loss and Accuracy Curve for Simple CNN Model

In Fold 2, the loss and accuracy graphs reveal the model's progression over the 10 epochs. Initially, the training accuracy started at 64.51%, with a corresponding validation accuracy of 70.97%, showing that the model was generalizing better on the validation set than on the training set. The model consistently improved as training progressed, with training and validation accuracies increasing steadily. By Epoch 10, the model achieved its best performance, with a training accuracy of 77.69% and a validation accuracy of 78.34%. The loss curves mirrored this improvement, with both training and validation losses gradually decreasing, indicating effective learning and reduced prediction errors. The convergence of the loss curves in the final epochs suggests that the model was well-tuned, with minimal overfitting. Overall, the graphs for Fold 2 highlight the model's ability to learn effectively, with a balance between training and validation performance, leading to the selection of this fold's model as the best overall.

A screenshot of a graph

Description automatically generated

Figure : Classification report for the best model in Experiment 1

The CNN model from Fold 2 achieves an overall accuracy of 0.78 on the validation data. The performance metrics for each class are as follows:

* **Cataract:** Precision is 0.65, recall is 0.49, and f1-score is 0.56, indicating moderate effectiveness in identifying cataract cases, with lower recall suggesting that the model misses many actual cataract instances.
* **Diabetes:** Precision is 0.83, recall is 0.93, and f1-score is 0.87, demonstrating strong performance in identifying diabetes cases, with high recall reflecting the model's ability to identify most diabetes instances correctly.
* **Glaucoma:** The model achieves a precision of 0.56, recall of 0.36, and an f1-score of 0.43, showing that the model has difficulty in accurately identifying glaucoma cases, with particularly low recall.

The macro average across all classes is 0.68 for precision, 0.59 for recall, and 0.62 for f1-score, indicating balanced but moderate performance across different conditions. The weighted averages, accounting for the support of each class, align closely with the overall accuracy, reflecting that the model is generally good performance, especially in recognizing diabetes cases.

A blue square with white text

Description automatically generatedThe inference on the precision, recall, and f1-score results is supported/ corroborated by the breakdown of class predictions shown in the confusion matrix. The model performs strongly in classifying diabetes but is challenged in correctly classifying both Cataracts and Glaucoma.

Figure 8: Confusion Matrix for the best model in Experiment 1

## 

## **Experiment 2: Simple CNN with Different Image Sizes**

Initial attempts to train the model on fundas images in the original size/resolution were challenging and often resulted in the model crashing. However, a solution to this was to understand how the model receives input, which required that each image was of one particular dimension. In order to process each image in its original dimension, the view(x.size(0), -1) operation was used to reshape the image tensor into a 2d tensor; in other words, the tensors are flattened into a single vector.

### Image size (128 x 128)

The image size is reduced to 128 x 128 pixels and evaluated with the Simple CNN model.

Table 2: Matrix showing performance across the K-Fold cross-validation for Exp 1 (128 x 128)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/6 | 70 | 72 | Good |
| 2 | 2/10 | 73.41 | 76.95 | Good |
| 3 | 3/8 | 72 | 69 | Overfitting |
| 4 | 4/10 | 74 | 73 | Good |
| 5 | 5/10 | 75 | 76 | Good |

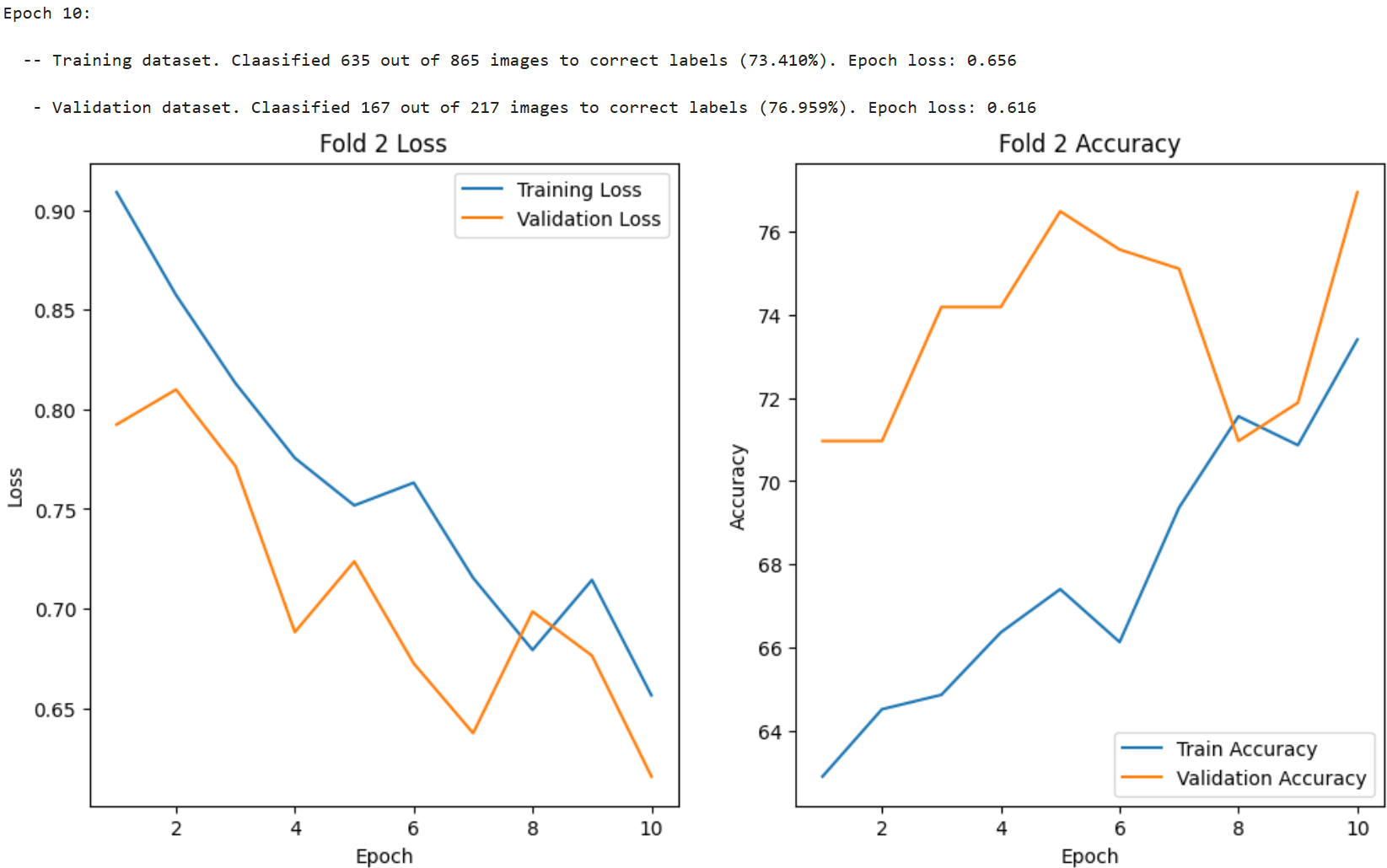


Figure : Graph of Loss and Accuracy Curve for Simple CNN model for Image size (128 x128)

In Fold 2, the graphs for loss and accuracy over 10 epochs show a clear improvement in the model's performance. Initially, the training accuracy was 62.89%, with a validation accuracy of 70.97% and corresponding losses of 0.909 and 0.792, indicating better generalization than fitting. As the epochs progressed, training and validation accuracies steadily increased, reaching 73.41% and 76.96%, respectively. The loss curves reflected this improvement, with the training loss decreasing to 0.656 and the validation loss to 0.616 by the final epoch. The alignment of accuracy improvements with the consistent drop in loss suggests that the model effectively learned with minimal overfitting, achieving its best performance in the last epoch.

Figure : Confusion matrix for CNN (128 x 128)

A blue squares with white text

Description automatically generatedA screenshot of a white sheet with black text

Description automatically generated

Figure : Classification report for CNN (128 X 128)

The classification report for the best model in Fold 2 shows an overall accuracy of 0.77 on the validation set. The performance metrics for each class are as follows:

* **Cataract:** Precision is 0.69, recall is 0.26, and the f1-score is 0.37, indicating that while the model can correctly identify cataract cases relatively well, it struggles to recall actual cataract instances. This is seen in the confusion matrix, with Cataract's true label being 9 out of possible 33.
* **Diabetes:** The model achieves a high precision of 0.79, an impressive recall of 0.97, and an f1-score of 0.87, showing strong performance in identifying and recalling diabetes cases.
* **Glaucoma:** Precision is 0.60, recall is 0.32, and the f1-score is 0.42, reflecting moderate ability in predicting glaucoma, with challenges in accurately recalling glaucoma instances. Similar to Cataract, the true label for Glaucoma is relatively, only 9 correctly classified out of possible 28.

The macro average f1-score is 0.55, and the weighted average f1-score is 0.73, demonstrating that the model performs well overall, with its strongest performance in classifying diabetes.

However, though the best model selected for this experiment was solely based on the highest test accuracy, it is not the most stable model. The performance of fold 5 epoch 10 yielded a train and test accuracy, with a difference of 1%, 75%, and 76%, respectively. Ideally, a model’s train accuracy should not exceed its testing, but the difference between them should not be significant. This speaks to how stable the model is in classifying labels correctly, suggesting possibly greater overall precision.

### Image size (224 x 224)

The image size is reduced to 224 x 224 pixels and evaluated with the Simple CNN model.

Table 3: Matrix showing performance across the K-Fold cross-validation for Exp 1 (224 x 224)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/5 | 73 | 73 | Good (Consistent) |
| 2 | 2/7 | 71 | 77 | Good |
| 3 | 3/8 | 76 | 74 | Overfit |
| 4 | 4/8 | 76 | 70 | Overfit |
| 5 | 5/7 | 75.05 | 81.48 | Good |

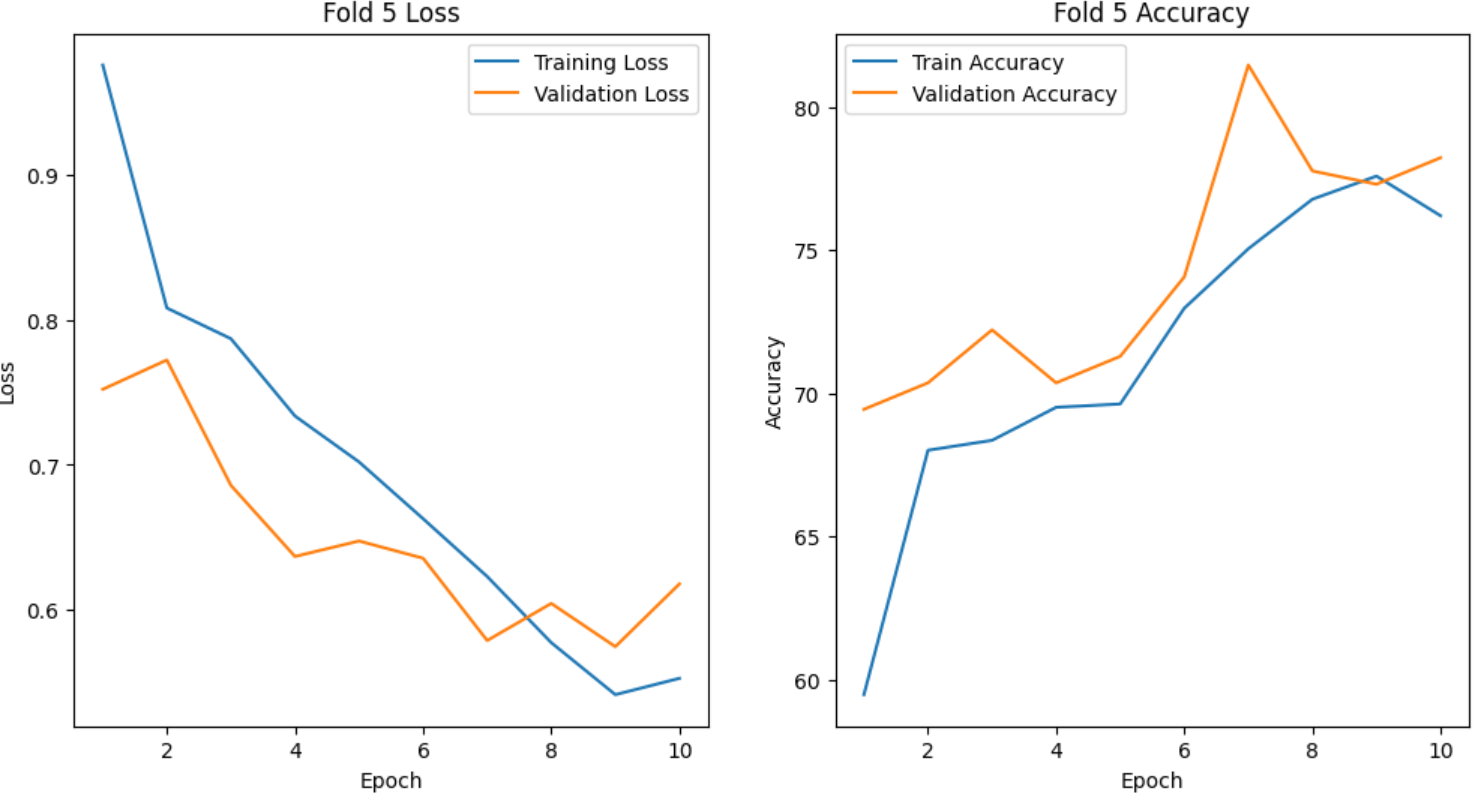


Figure : Graph of Loss and Accuracy Curve for Simple CNN model for Image size (224 x224)

A screenshot of a graph

Description automatically generatedIn Fold 5, the model training process consistently improved accuracy and loss throughout 10 epochs. Initially, the training accuracy started at 59.47% with a relatively high loss of 0.976, while the validation accuracy began at 69.44% with a loss of 0.752. As training progressed, the model's performance steadily improved. By Epoch 7, the training accuracy reached 75.06%, and validation accuracy peaked at 81.48%, with corresponding losses of 0.623 and 0.579, respectively. This suggests that the model effectively learned to generalize, as seen by the decreasing loss and increasing accuracy on both the training and validation datasets. However, after Epoch 7, while the validation accuracy slightly fluctuated, the validation loss began to increase, indicating potential overfitting beyond this point. The balance between training and validation metrics in Epoch 7 suggests it was the optimal point, offering the best trade-off between model complexity and generalization performance.

A screenshot of a graph

Description automatically generated

Figure : Classification report for CNN (224 X 224)

Figure : Confusion matrix for CNN (224 X 224)

The classification report for this model indicates an overall accuracy of 0.81 on the test data. The performance metrics for each class are as follows:

* **Cataract:** The model achieves a precision of 0.67, a recall of 0.72, and an f1-score of 0.69, reflecting a balanced performance with slightly higher recall than precision.
* **Diabetes:** With a precision of 0.84, recall of 0.93, and f1-score of 0.88, the model demonstrates strong effectiveness in identifying and correctly classifying cases of diabetes.
* **Glaucoma:** The model shows a high precision of 0.88 but a lower recall of 0.42, resulting in an f1-score of 0.57, indicating that while it is good at predicting positive instances, it misses a significant portion of actual glaucoma cases.

The macro average scores—precision of 0.80, recall of 0.69, and f1-score of 0.72—suggest that the model performs well across the classes, although there is some variability. The weighted averages indicate that the model maintains a good balance between precision and recall across all classes, with an overall weighted f1-score of 0.80.

It should be noted, however, that though the criteria for this experiment was to select the model with the best validation (test) accuracy, it was not the most stable. This was demonstrated by a difference of 6% between the train and validation accuracy, which was higher than the train, suggesting underfitting. However, model 1 (see Table 3), based on both the train and test accuracy being 73%, indicates a more stable model despite the significantly lower test accuracy. This model is quite likely not to exhibit properties of an overfit or underfit model.

This model shows consistent performance across training and testing, indicating a good balance between fitting the training data and generalizing to unseen data, which is crucial for reliable real-world performance.

### Image size (448 x 448)

The image size is reduced to 448 x 448 pixels and evaluated with the Simple CNN model.

Table 4: Matrix showing performance across the K-Fold cross-validation for Exp 1 (448 x 448)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/8 | 75 | 73 | Good |
| 2 | 2/8 | 72 | 73 | Underfit |
| 3 | 3/8 | 76 | 72 | Overfit |
| 4 | 4/6 | 71 | 65 | Underfit |
| 5 | 5/9 | 74.92 | 75.92 | Good |

The fifth model is selected primarily because its test accuracy is the highest. However, the training and test accuracies are also closely aligned, indicating a well-generalized model without signs of overfitting or underfitting. This balance suggests that the model has effectively learned from the training data and performs consistently on unseen data. Additionally, the slightly higher test accuracy (75%) reflects its strong performance in real-world scenarios.

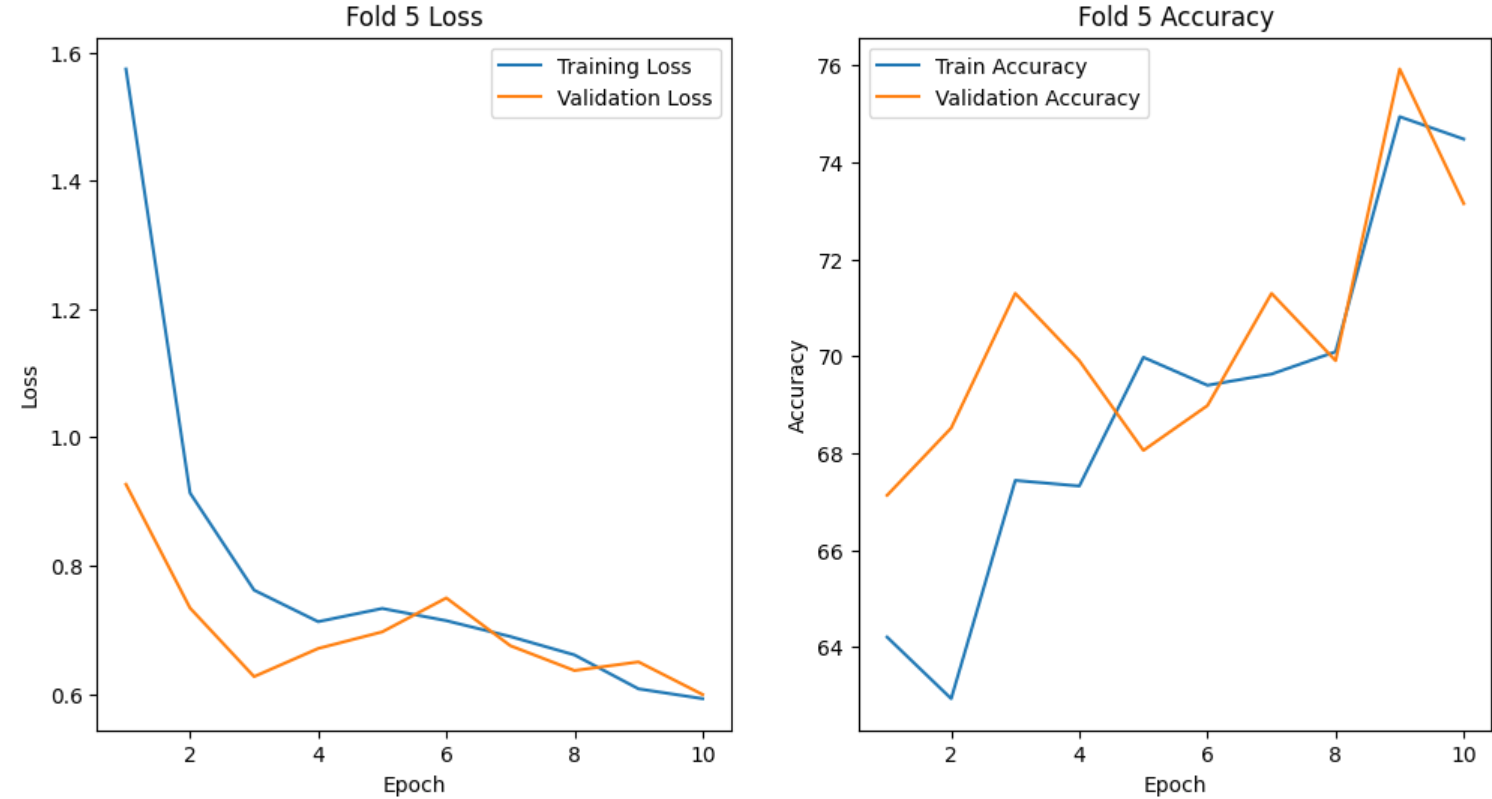


Figure : Graph of Loss and Accuracy Curve for Simple CNN model for Image size (448 x 448)

In this fold 5, the model displayed a clear learning trajectory, improving both accuracy and loss over the epochs. Initially, training and validation accuracies were closely aligned, suggesting that the model was learning effectively without overfitting. By Epoch 10, the model achieved a training accuracy of 74.48% and a validation accuracy of 73.15%. The gradual decrease in loss across both datasets indicates that the model effectively minimized errors.

A blue square with white text

Description automatically generatedHowever, there is a slight divergence between training and validation accuracies in the later epochs, which could be a sign of the model beginning to overfit. Despite this, the gap is minimal, and the validation accuracy remains high, suggesting that the model maintains a good balance between fitting the training data and generalizing to new data. This makes the model from this fold a strong candidate, showing effective learning while performing well on the validation set.

Figure : Confusion matrix for CNN (448 X 448)

A screenshot of a white paper with black text

Description automatically generated

Figure : Classification report for CNN (448 X 448)

The classification report for this model indicates an overall accuracy of 0.76 on the test data. The performance metrics for each class are as follows:

* **Cataract:** The model achieves a precision of 0.61, a recall of 0.61, and an f1-score of 0.61. This indicates a balanced but modest performance in identifying cataract cases, with neither high precision nor recall.
* **Diabetes:** The model performs well with a precision of 0.79 and a recall of 0.93, resulting in an f1-score of 0.85. This suggests that the model is highly effective in detecting diabetes, with reasonable accuracy and coverage.
* **Glaucoma:** The model's precision is 0.83, but the recall is low at 0.15, leading to an f1-score of 0.26. This reflects that while the model correctly identifies a few glaucoma cases, it struggles with recall, missing many actual instances.

Overall, the model demonstrates good performance with an accuracy of 0.76, suggesting it performs well across all classes, but there is room for improvement in the detection of glaucoma. The macro average f1-score of 0.57 indicates that the model has variable performance across different classes, with better performance for some classes than others. This is also observed in the confusion matrix below, which depicts the model's ability to classify diabetes accurately. However, it struggles to classify Glaucoma correctly, only correctly classifying 5 out of 33. The model does a better job with Cataract as previously indicated by the results in the classification report, correctly classifying 22 out of 36.

## **Experiment 3: CNN model with Hyperparameter Tuning**

In this experiment, three parameters were selected for hyperparameter tuning: learning rate, batch size, and dropout. Two values were provided for each: learning rate, 0.001 and 0.01; batch size, 16 and 32; and dropout\_rate, 0.3 and 0.5. After applying the user-defined function for executing hyperparameter tuning, during the training and validation phase, the best combination of the parameters were batch\_size, 32; dropout\_rate, 0.5; and learning\_rate, 0.001. The model was trained for 10 epochs and achieved a peak accuracy of 71.43 % (see Figure 18).

A closer examination of the loss curve shows that the model was learning and increasing its ability to classify during the training phase, as demonstrated by the steady decrease in loss over the epoch. However, the validation curve suggests the model struggled to generalize well on unseen data and is tending towards overfitting, demonstrated by the gap between the two curves towards the end of the epochs. A higher training accuracy also corroborates this compared to the test for most training and validation sessions (10 epochs).

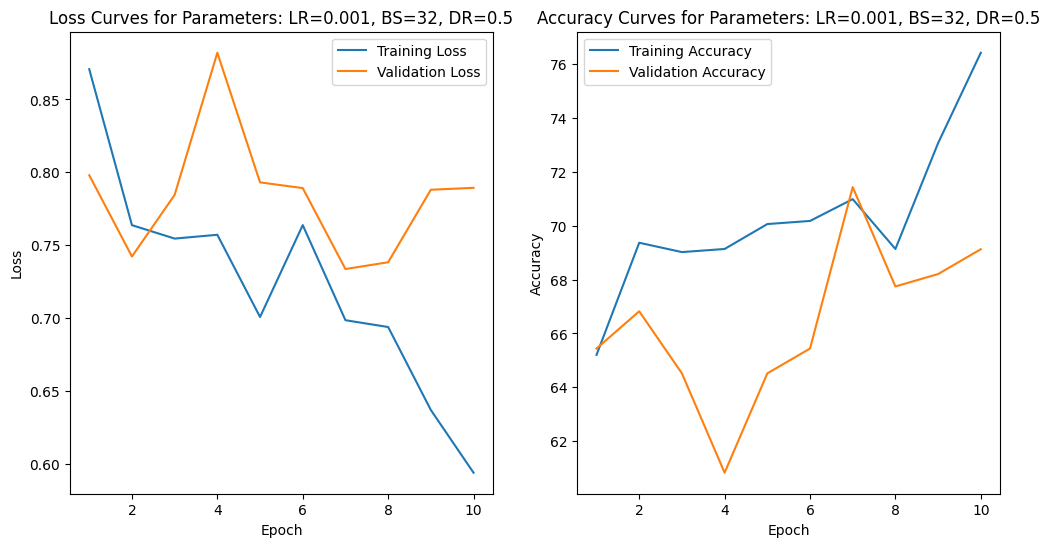
****

Figure : Graph of Loss and Accuracy Curve for hyperparameter CNN tuned model

A blue squares with white text

Description automatically generatedSimilarly, a close inspection of the confusion matrix for the best combination of parameters for the CNN model also showed features of an overfitting model. The model could predict the Diabetes class of the three with the highest accuracy, correctly predicting diabetes approximately 90% of the time. However the other two diseases showed less impressive results with Cataract being correctly classified at a rate of 52% and Glaucoma, 7%.

Figure : Confusion matrix of model’s classification ability

This shows the impact of the class imbalance on the model’s ability to learn and classify correctly Cataract and Glaucoma, with both classes contributing significantly fewer images to the overall image count.

**Tuned CNN model with K-Fold cross-validation**

After deriving the best combination of parameters from hyperparameter tuning, K-fold cross-validation was employed better to understand the model performance across the limited dataset. Considering the computational resources for executing the experiment, though a high K-fold value could lead to a better estimate of the model performance, 5 was selected as the value. This was a trade-off between the demand for computational resources and the accuracy of the estimate, with 5 being the best compromise.

Table : Matrix showing performance across the K-Fold cross-validation for hyperparameter tuned CNN model (224 x 224)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/3 | 67 | 74 | Underfit |
| 2 | 2/9 | 74 | 77 | Overfit |
| 3 | 3/10 | 73 | 70 | Overfit |
| 4 | 4/9 | 71 | 64 | Overfit |
| 5 | 5/5 | 68 | 74 | Underfit |

According to the results from the K-fold cross-validation, Fold 2 yielded the best validation(test) accuracy of 77%. However, the model is still overfitting (see Table 5) and did not achieve a best-fit line on the loss or accuracy curves.

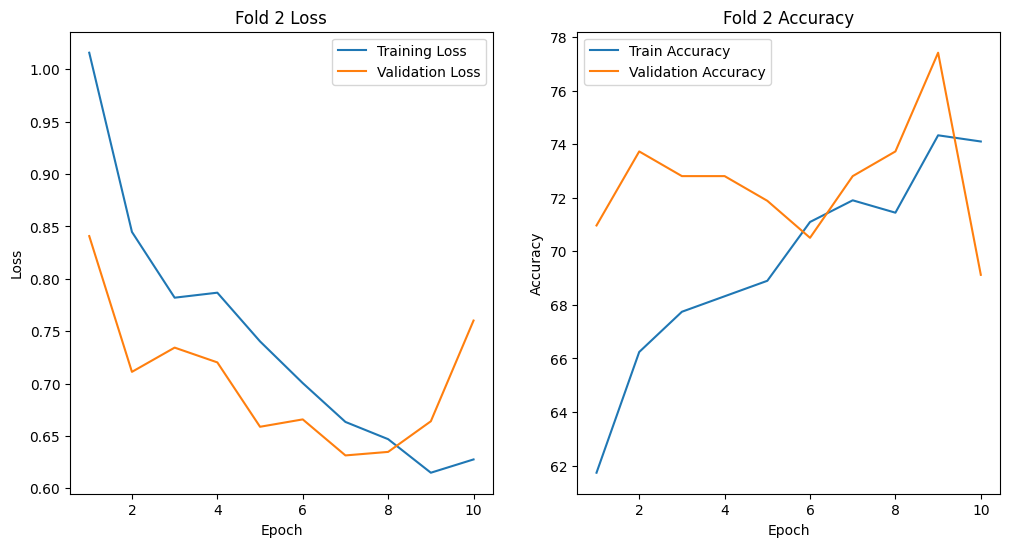


Figure : Graph of Loss and Accuracy Curve for hyperparameter tuned model

A close examination of the loss curve for Fold 2 shows that training and validation loss generally decreased steadily. At Epoch 8, there appears to be a convergence of booth curves, though momentarily, afterward, the validation curve generates a sharp increase in loss. The sharp increase suggests that the model starts to overfit just after the eighth epoch and can no longer generalize well on unseen data. This is also reflected in the accuracy curve as it peaks at about 77% accuracy for validation and then declines sharply after that, suggesting that the model is now memorizing the training data and the noise and cannot generalize on further unseen data.

A blue squares with white text

Description automatically generatedHowever, the model shows strong learning during the first 8 epochs and could benefit from experimenting with regularization techniques and early stopping.

Figure : Confusion matrix for hyperparameter tuned CNN (224 X 224)

A white paper with black text

Description automatically generated

Figure : Classification report for hyperparameter tuned CNN (224 X 224)

The overall accuracy of the model is 0.77; its performance varies across the different classes as follows:

* **Cataract:** The model performs moderately in detecting cataract cases, with a precision of 0.62 and a recall of 0.43, resulting in an f1-score of 0.51. This suggests that while the model is fairly precise, it misses many true cataract cases, reflecting lower recall.
* **Diabetes:** The model performs very well in identifying diabetes cases, with a high precision of 0.80 and an even higher recall of 0.95, leading to a strong f1-score of 0.87. This indicates that the model is precise and effective in identifying most diabetes cases.
* **Glaucoma:** For glaucoma, the model shows lower performance compared to the other conditions, with a precision of 0.60 and a recall of 0.21, resulting in a lower f1-score of 0.32. This indicates that the model struggles to correctly identify glaucoma cases, with a high rate of missed true positives.

The macro average f1-score of 0.57 and weighted average f1-score of 0.74 indicate that the model's overall performance is moderate, with a more vital ability to classify diabetes cases but with noticeable weaknesses in detecting cataract and glaucoma cases.

## **Experiment 4: ResNet50 (pre-trained) Model**

This code uses a pre-trained Resnet model, which performs well with smaller images. This model is trained and tested with three different image sizes (128, 224, 448).

### Image size (128 x 128)

Table : Matrix showing performance across the K-Fold cross-validation for ResNet50 (128 x 128)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/9 | 94.22 | 93.62 | Good |
| 2 | 2/7 | 93 | 90 | Overfit |
| 3 | 3/7 | 91 | 89 | Overfit |
| 4 | 4/5 | 88 | 85 | Overfit |
| 5 | 5/10 | 95 | 90 | Overfit |

The model from Fold 1/ Epoch 9 comes out to be the best model with an accuracy of 93% for image size (128 x128).

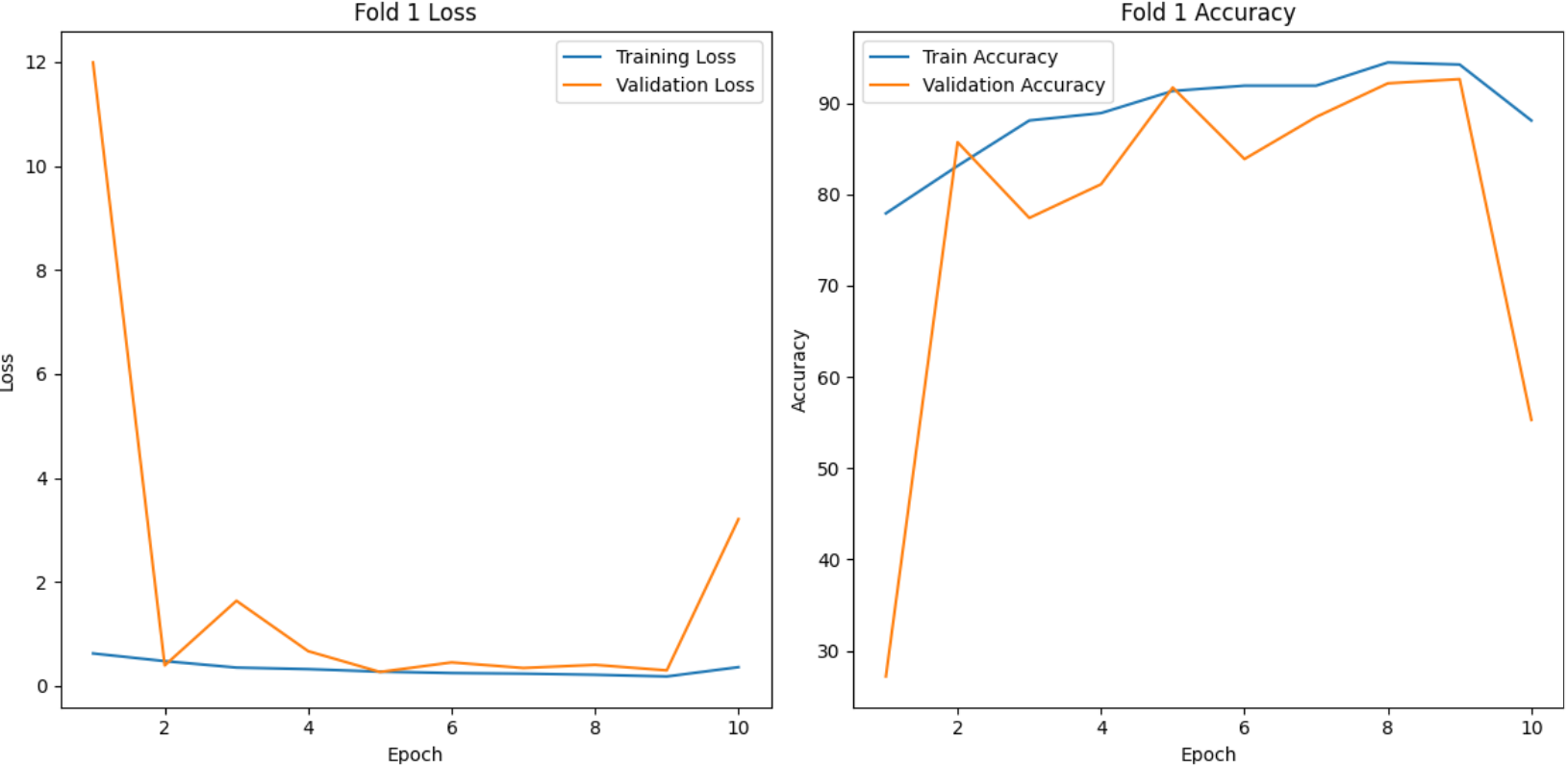


Figure : Graph of Loss and Accuracy Curve for ResNet50 model (124 x 124)

In fold 1, the training loss starts just below 2, peaks at around 12 by epoch 2, and then decreases sharply to near zero by epoch 10. Training accuracy starts around 70%, rises to 90% by epoch 2, and remains close to 90% throughout. The validation loss begins around 7, drops to just over zero by epoch 3, spikes to nearly 12 at epoch 7, and then falls back near zero by epoch 10. The validation accuracy starts at around 75% and increases to over 90% by epoch 2 but shows significant dips at epochs 3 and 7, ending slightly above 85%. This suggests effective learning on training data but inconsistent performance on validation data, indicating potential overfitting.

A blue square with white text

Description automatically generated

Figure : Confusion matrix for ResNet model (124 x 124)

A screenshot of a graph

Description automatically generated

Figure : Classification report for ResNet model (124 x 124)

The classification report for this model indicates an overall accuracy of 0.93 on the test data. The performance metrics for each class are as follows:

* **Cataract:** The model achieves an impressive precision, recall, and f1-score of 0.93 for cataract detection, indicating that it correctly identifies nearly all cataract cases with very few false positives or negatives.
* **Diabetes:** With a precision of 0.93 and a recall of 0.97, the model performs excellently in identifying diabetes, reflected in a high f1-score of 0.95. This suggests both high accuracy and robustness in detecting diabetes cases.
* **Glaucoma:** The model's performance in detecting glaucoma is slightly lower, with a precision of 0.87, a recall of 0.71, and an f1-score of 0.78. While still strong, it indicates some challenges in recalling all glaucoma cases, though most predicted cases are correct.

The macro average f1-score of 0.89 and weighted average f1-score of 0.92 further confirm that the model is consistently effective across different classes, with a slight variance in performance for glaucoma.

### Image size (224 x 224)

Table : Matrix showing performance across the K-Fold cross-validation for ResNet50 (224 x 224)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/9 | 92.02 | 91.24 | Good |
| 2 | 2/7 | 94 | 89 | Overfit |
| 3 | 3/5 | 87 | 90 | Underfit |
| 4 | 4/5 | 89 | 84 | Overfit |
| 5 | 5/7 | 90 | 84 | Overfit |

The model from Fold 1/ Epoch 9 comes out to be the best model, with an accuracy of 91%.

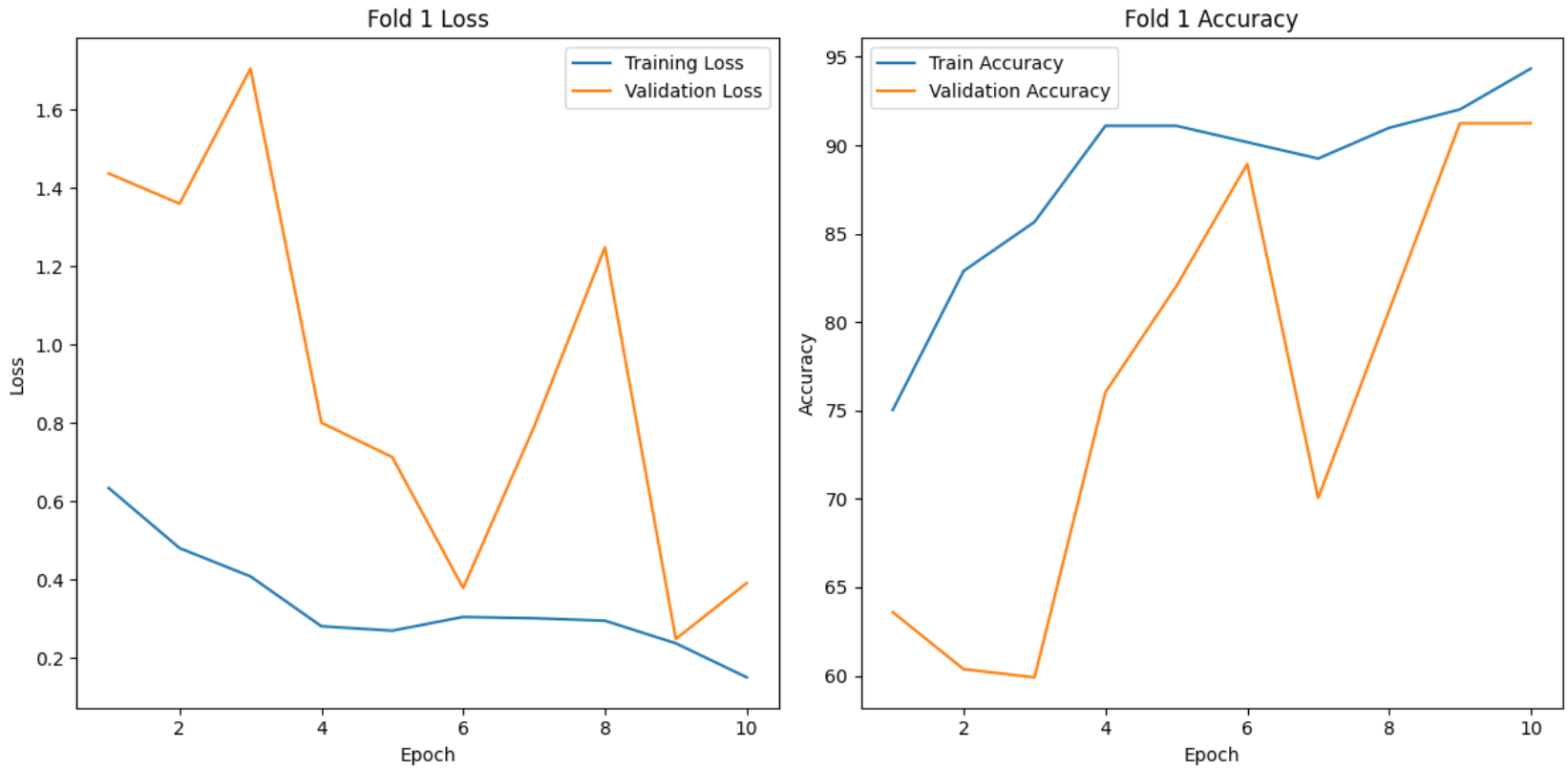


Figure : Graph of Loss and Accuracy Curve for ResNet50 model (224 x 224)

A screenshot of a graph

Description automatically generatedThe “Fold 1 Loss” graph, the training loss starts just below 1.6, fluctuates, and ends slightly above 0.4. The validation loss begins around 1.4, with dramatic fluctuations, reaching its lowest point below 0.8 between epochs 5 and 6. In the “Fold 1 Accuracy” graph, the training accuracy starts near 75%, increases steadily with minor dips, and ends near 95%. The validation accuracy also starts around 75% and rises similarly but shows more variability, ending below 90%. These trends suggest effective learning on training data but indicate some variability in performance on validation data.

Figure : Confusion matrix for ResNet model (224 x 224)

A screenshot of a graph

Description automatically generated

Figure : Classification report for ResNet model (224 x 224)

The model's overall accuracy is 0.91, reflecting its strong performance across the board.

* **Cataract:** The model performs excellently in detecting cataract cases, with a high precision of 0.89 and an even higher recall of 0.95, leading to an f1-score of 0.92. This indicates that the model is precise and effective in identifying most cataract cases.
* **Diabetes:** The model performs very well in identifying diabetes, with a precision of 0.93 and a recall of 0.94, resulting in a strong f1-score of 0.94. This suggests that the model reliably detects diabetes cases with minimal errors.
* **Glaucoma:** For glaucoma, the model shows a slightly lower performance compared to the other conditions, with a precision of 0.83 and a recall of 0.71, leading to an f1-score of 0.77. This indicates that while the model is fairly accurate in its predictions, it misses some true glaucoma cases.

The macro average f1-score of 0.87 and weighted average f1-score of 0.91 further support the model's truthfulness in classifying the different conditions.

### Image size (448 x 448)

Table : Matrix showing performance across the K-Fold cross-validation for ResNet50 (448 x 448)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fold/Epoch | Train Accuracy | Test Accuracy | Notes |
| 1 | 1/6 | 80 | 88 | Underfit |
| 2 | 2/8 | 88 | 86 | Overfit |
| 3 | 3/4 | 87 | 88 | Good |
| 4 | 4/9 | 87 | 81 | Overfit |
| 5 | 5/9 | 90.41 | 89.81 | Good |

The model from Fold 5/ Epoch 9 is the best model, with an accuracy of 89%.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Figure : Graph of Loss and Accuracy Curve for ResNet50 model (448 x 448)

A blue square with white text

Description automatically generatedThe training and validation process in Fold 5 highlights a model that shows promise but also instability. The training accuracy steadily improves, reaching over 90% by Epoch 9, with a corresponding decrease in loss, indicating effective learning. However, validation accuracy fluctuates significantly, with solid performance in Epochs 4 and 9, up to 89.%. Despite this, the model also experiences severe drops in validation accuracy in Epochs 1, 5, and 10, where accuracy plummets to as low as 15%. These fluctuations and varying validation losses suggest that the model struggles with generalization, possibly due to overfitting or sensitivity to the validation data.

Figure : Confusion matrix for ResNet model (448 x 448)

A screenshot of a graph

Description automatically generated

Figure : Classification report for ResNet model (448 x 448)

The model's overall accuracy is 0.90, indicating strong performance across the board.

* **Cataract:** The model demonstrates excellent performance in detecting cataract cases, with a high precision of 0.95 and an even higher recall of 0.97, leading to an F1-score of 0.96. This indicates that the model is highly accurate and effective in identifying most cataract cases with few false negatives.
* **Diabetes:** The model also performs very well in identifying diabetes, achieving a precision of 0.92 and a recall of 0.94, resulting in a strong F1-score of 0.93. This suggests that the model reliably detects diabetes cases with minimal errors, making it a robust predictor for this condition.
* **Glaucoma:** The model's performance is slightly lower for glaucoma, with a precision of 0.72 and a recall of 0.64, leading to an F1-score of 0.68. This indicates that the model has difficulty accurately identifying all true glaucoma cases, resulting in a higher rate of missed diagnoses.

The macro average F1-score of 0.86 and weighted average F1-score of 0.90 further confirm the model's reliability and effectiveness in classifying the different conditions, though there is room for improvement in glaucoma detection.

Table 9: Summary of Experiment Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiment | Image Size in pixels | Train Accuracy | Test Accuracy | Notes |
| **Simple CNN** | Original | 77.68 | 78.34 | The training and test accuracy are close, suggesting the model is well-fitted without significant overfitting or underfitting. |
| **Simple CNN** | 128 x 128 | 73.41 | 76.95 | The lower training accuracy indicates potential underfitting. The model is not complex enough to capture the patterns in the data, leading to lower performance. |
| 224 x 224 | 75.05 | 81.48 | The increase in test accuracy compared to training accuracy suggests a well-generalized model. This configuration seems to strike a balance between model complexity and data representation. |
| 448 x 448 | 74.92 | 75.92 | The minimal difference between training and test accuracy indicates a lack of overfitting. However, the lower accuracy could suggest that the larger image size does not benefit the simple CNN architecture, possibly due to overfitting complex features. |
| **Simple CNN with Hyperparameter**  **(batch\_size = 32, dropout\_rate = 0.5, learning\_rate = 0.001)** | 224 x 224 | 74.33 | 77.41 | Despite hyperparameter tuning, this model underperforms compared to the simple CNN without hyperparameter tuning on 224x224 images. This suggests that the chosen hyperparameters may not be optimal for this dataset. |
| **ResNet 50**  **(pre-trained model)** | 128 x 128 | 94.22 | 93.62 | With minimal difference, the high training and test accuracy suggests that ResNet 50 is neither overfitting nor underfitting. It is a well-performing model for this image size. |
| 224 x 224 | 92.02 | 91.24 | Similar to the above image size experiment, this model shows strong performance with slightly lower accuracy due to the increased image size, indicating that it handles the data well. |
| 448 x 448 | 90.41 | 89.81 | The slight drop in accuracy with the most prominent image size may suggest some overfitting, as the model might capture noise rather than meaningful features. |

# **Conclusions**

As expected, the pre-trained ResNet 50 model outperformed the custom-built CNN models across all image sizes. Pre-trained models like ResNet 50 have the advantage of being fine-tuned on large datasets, allowing them to capture complex patterns and features that custom models, notably simpler architectures, might miss.

Among the various configurations tested, ResNet 50, with an image size of 128x128, emerged as the best-performing model. It achieved the highest accuracy while maintaining a balance between training and test performance, indicating strong generalization without overfitting.

In scenarios where computational resources or memory constraints are a concern, the Simple CNN with an image size of 224x224 could be considered an alternative. It showed decent performance with good generalization, making it a viable option when a simpler, more lightweight model is needed. However, for the most robust and reliable results, the pre-trained ResNet 50 with a 128x128 image size is recommended as the best choice for deployment.

# **Future Work**

For future work, mainly with the CNN model, increased training time (number of epochs) and increased number of images for training, especially for cataract and glaucoma, should be explored. This could help improve classification performance between those classes. Another consideration is experimenting with regularization techniques, including L2 regularization, early stopping, and data augmentation, to combat overfitting observed during experiments.

For the next phase, an important area of improvement is model adaptability to real-world scenarios where mobile cameras mainly capture images. While the current model only focuses on high-quality fundus images captured from professional equipment, the scope can be expanded if a model is trained with real-time images. This could also be deployed to mobile applications for health care, and eye disease detection could be easy. Moreover, this approach is achievable if the model is trained on a combination of fundus and authentic eye images combined with examinations like Boris Baneko mentioned in his study Detecting Signs of Disease from External Images of the Eye. He mentioned how they could train a model to classify diseases based on authentic eye images captured by mobile devices. (*Detecting Signs of Disease from External Images of the Eye*, 2022).

Further, GPU limits resources and data labeling accuracy. Labeled datasets must improve quality and accuracy to train an accurate model. By using unlabeled data better, approaches like semi-supervised learning or supervised learning could improve the quality of datasets. Improving the structure of models and using methods like model pruning or quantization could lower computational requirements to overcome GPU restrictions. Additionally, exploring edge computing options may make it possible to process fundus photos directly on devices in real-time, improving the system's usability and availability for more users. These developments are intended to improve the model's performance, provide scalability, and successfully handle real-world problems.

# **References**

World report on vision. Geneva: World Health Organization; 2019. License: CC BY-NC-SA 3.0 IGO UNDP (United et al.), .. (2023). 2023 Global Multidimensional Poverty Index (MPI). *UNDP (United et al.)*, 31, <https://hdr.undp.org/system/files/documents/hdp-document/2023mpireporten.pdf>

Maranhão, A. (n.d.). *Ocular Disease Recognition*. Retrieved July 21, 2024, from Kaggle, <https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k>

Deepak, G. D., & Bhat, S. K. (2024). Deep learning-based CNN for multiclassification of ocular diseases using transfer learning. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization, 12(1). <https://doi.org/10.1080/21681163.2024.2335959>

Glaret subin, P., & Muthukannan, P. (2022). Optimized convolution neural network-based multiple eye disease detection. *Computers in Biology and Medicine*, *146*. <https://www.sciencedirect.com/science/article/pii/S0010482522004401>

Yang, X. L., & Yi, S. L. (2022). Multi-classification of fundus diseases based on DSRA-CNN. *Biomedical Signal Processing and Control*, *77*. <https://www.sciencedirect.com/science/article/pii/S1746809422002853>

Lin, P. (2024, February 24). *ML vs Traditional Programming: Which is Better for Your Career?* Retrieved August 10, 2024, from <https://www.codemotion.com/magazine/ai-ml/machine-learning/ml-vs-traditional-programming-which-is-better/>

Frost, J. (2021). *Accuracy vs Precision: Differences & Examples*. Retrieved August 11, 2024, from <https://statisticsbyjim.com/basics/accuracy-vs-precision/>

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications <https://arxiv.org/abs/1704.04861>

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 770–778. <https://doi.org/10.1109/CVPR.2016.90>

IBM, (2024). *What is overfitting?* Retrieved August 11, 2024, from <https://www.ibm.com/topics/underfitting> Masters, D., & Luschi, C. (2018). Revisiting small batch training for deep neural networks. [*https://arxiv.org/abs/1804.07612*](https://arxiv.org/abs/1804.07612)

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition <https://ieeexplore.ieee.org/document/7780459>

*Detecting Signs of Disease from External Images of the Eye*. (2022). Research. Google. <https://research.google/blog/detecting-signs-of-disease-from-external-images-of-the-eye/>