Eye Diseases Classification

Systems & Biomedical Engineering Department

Special Functions & Partial Differential Equations (MTH2245)

ماشي بنور الله

Dr. Samah El-Tantawy

Faculty of Engineering - Cairo University

Table of contents

List of figures

List of Tables

Abstract

In view of the increasing incidence of eye diseases worldwide, there's an urgent need for reliable diagnostic tools to enable fast intervention and tailored treatment plans. This research investigates the use of deep learning techniques for classifying eye diseases using medical image analysis. Employing convolutional neural network (CNN) architectures, incorporating partial differential equations (PDEs) in their foundational construction, we explore the automated identification and categorization of various pathological conditions, such as glaucoma, cataract and diabetic retinopathy. The study involves extensive training on a diverse large-scale dataset, incorporating data augmentation strategies to improve model robustness. Remarkably, the utilization of the pre-trained VGG19 model exhibits superior performance, achieving a minimum validation loss of 0.3127 and an accuracy of 91.46% at its optimal epoch. This research significantly contributes to the evolving field of medical image analysis, paving the way for the integration of deep learning models into ophthalmic diagnostic workflows. The proposed approach holds promise for early disease detection, highlighting the transformative potential of deep learning in eye disease classification and improving patient outcomes.

1. Introduction

World Health Organization estimates that at least 2.2 billion people have near or distant vision impairment. 1 billion of those cases could have been prevented or still have to be addressed.

The leading causes of blindness or distance vision impairment among those mentioned 1 billion are cataracts (94 million), refractive error (88.4 million), age-related macular degeneration (8 million), glaucoma (7.7 million) and diabetic retinopathy (3.9 million) [1].

Among the above eye conditions, we will study cataracts, glaucoma, and diabetic retinopathy.

Cataract is an eye disease causing the eyes to be cloudy, vision to be frosty or fogged-up and the person suffering from it to face difficulties reading, driving, and even recognizing another person's face. It is caused by proteins that build up on the lens of the eyes preventing light from passing through it.

Cataracts account for more than 51 percent of blindness throughout the world. [2]

Glaucoma is caused by fluid building up in the front part of the eyes that increases the pressure inside the eye causing the optic nerve that connects the eye to the brain to be damaged.

According to the Centers for Disease Control and Prevention (CDC), it is the second leading cause of blindness worldwide, 50% of people with glaucoma don't know they have it as there are usually no early symptoms. [3]

It must be detected and diagnosed early to stop the vision from getting worse and causing vision loss as it is impossible to reverse any damages that happened before the diagnosis.

Diabetic retinopathy is a common complication of diabetes where high blood sugar levels cause damage to the blood vessels of the retina. Globally, 27% of diabetic patients have it. [4]

It can't usually be noticed in the early stages due to the lack of obvious symptoms until it's more advanced however it can be picked up during diabetic eye screening.

The primary objective of this project is to leverage the power of deep learning and computer vision to build a system that enable early detection of eye diseases like glaucoma, cataract and diabetic retinopathy as early intervention and treatment can significantly improve patient outcomes and prevent vision loss.

The project reduces human diagnostic errors and subjectivity by providing a consistent and objective assessment of medical images. It also makes eye disease diagnosis more accessible, especially in underserved or remote areas where specialist medical facilities may be limited.

2. <u>Literature Review</u>

Early detection of eye diseases like cataract, glaucoma and diabetic retinopathy is crucial in the treatment especially that they are the main cause of blindness around the world and some of them have irreversible effects.

At the present time, a trained ophthalmologist performs manual analysis of the fundus images to detect the presence of these diseases. This method requires the availability of trained individuals and is subjected to human errors. In addition, the early diagnosis is very difficult as the early symptoms are not obvious. For these reasons, automated methods that can detect the eye diseases in fast and accurate manner are needed.

In the past few years, there were various studies conducted using machine learning and deep learning algorithms to analyze the fundus images for the classification of retinal diseases.

Maheshwari et al. [5] conducted an automated glaucoma diagnosis by optimizing kernel parameters and applying cross validation to achieve accuracy of 81.32% while Acharya et al. [6] used adaptive histogram equalization convolved with several filter banks processed to create local configuration patterns that feed a k-nearest neighbor (KNN) classifier.

A study [7] proposed automated glaucoma diagnosis system using higher order spectra (HOS), discrete wavelet transform (DWT) features then feeding the extracted features to a support vector machine classifier (SVM) reaching an accuracy of 91.67%.

Yang et al. [8] proposed that artificial neural network (ANN) could be used to automatically classify the severity of cataracts in patients' eyes, using a high-quality carpet conversion and a three-dimensional filter to improve the image quality. the ANN was able to classify cataracts as normal, mild, moderate, or severe with an accuracy of 84.2%.

In 2019, Agarwal et al. [9] carried out a study focusing on cataract detection using Android technology. They evaluated the performance of different classification methods, including KNN, SVM, and Naïve Bayes, to distinguish between normal and cataract cases. The KNN method proved to be the most accurate, achieving an 83.07% accuracy rate, while SVM reached 75.2%, and Naïve Bayes attained a 76.64% accuracy rate in the classification task.

The study [10] classifies diabetic retinopathy (DR) stages depending on detecting blood vessels and hemorrhages in retinal images. Random Forests are employed to classify into normal, moderate non-proliferative diabetic retinopathy (NPDR), and severe NPDR using area and perimeter features of blood vessels and hemorrhages, achieving 90% accuracy for normal cases and 87.5% for NPDR cases.

the paper [11] had the main goal of classifying the grade of NDPR at any retinal Image using Support vector mechanism (SVM) classification. It was tested on 400 retinal images that resulted in 93.8 % predictive capacity value.

The previously mentioned studies have significant drawbacks that hinder the usability in large scale as some of these algorithms have been derived from small data sets of fundus images obtained in singular clinical environments meaning that different types of fundus cameras, eye dilation methods are not put into consideration. In addition, many of these algorithms depend on manual feature extraction and selection for the diseases' characterization such as optic disc or blood vessels. [12][13]

Thus, the presence of deep learning algorithm was a must to avoid the problems of both present way of detection and the already mentioned studies by developing a way of detection that is accurate, efficient, fast, uses large data sets that are similar to the real clinical world and to be independent on experts that their existence is rare especially in rural and underdeveloped areas.

H. Jiang [14] presents an automatic image-level diabetic retinopathy (DR) detection using three deep learning models which were Inception V3, ResNet151, and Inception-ResNet-V2. They individually performed with an accuracy of 87.91%, 87.20% and 86.18% respectively. When all these models were integrated using the AdaBoost algorithm, it performed with a better accuracy of 88.21%.

"AD2Net" model is a deep learning model proposed by Qian et la. [15] having the main purpose of speeding up DR diagnosis and using attention mechanism to encourage the network to focus on learning useful information in images which improves classification effect that yielded to accuracy of 83.2%.

The study [16] explained the usage of texture features extracted from different binary pattern (LBP) variants, with artificial neural network classifier (ANN) in the detection of exudates which is the primary sign of DR and reported 96.73% classification performance.

Shankar et al. [17], proposed a Synergic Deep Learning (SDL) model for fundus image classification, which could classify images as either healthy or one of three stages of DR with maximum detection rate.

In 2021, Weni et al. [18] conducted research on cataract detection based on image features having the main goal of improving diagnostic precision while minimizing loss. Cataract detection was performed using Convolutional Neural Networks. This study utilizes the Google Net architecture and divides it into two classes, namely normal and cataract. The accuracy rate was 95% however when tested on ten real time images, the accuracy decreased to 88%.

Some researchers used digital camera instead of fundus camera for cataract detection like Ju Lai et la. [19] where this study used the digital camera to train a convolutional neural network classifier for the detection, but it had its disadvantages like it was affected by reflection of light, the pictures contained noise and the person taking the pictures needed a lot of training.

Hasan et al. [20] examined the performance of four pretrained CNN models, including DenseNet121, InceptionV3, Xception, and InceptionResNetV2, to diagnose cataracts from retinal

images. Moreover, the authors discovered that InceptionResNetV2 outperforms all other models, with a two-class classification with
accuracy of 98.17%.

Chen et al. [21] developed a deep learning (DL) framework for glaucoma detection based on deep CNN, which is able to capture the discriminative features that better characterize the hidden patterns related to glaucoma. In order to further boost the performance, dropout and data augmentation strategies are utilized in the proposed deep CNN, effectively improving the model's robustness and generalization capabilities.

The study by Gomez et la. [22] used the application of different CNN scheme for glaucoma detection to show the effect of data set size, the architecture and the training strategy. They used three different datasets and five architectures which are standard CNN, VGG 19, ResNet50, DENet and GoogleNet. The best option for the data set used was VGG 19 with transfer learning and fine tuning which showed similar result to human evaluation.

The primary objective of the study by Sudhan et la. [23] is to identify if the person has glaucoma or not using U-Net architecture and a pertained DenseNet-201 architecture for feature extraction and DCNN for classification reaching testing accuracy of 96.9%.

Nath adapted the same objective as Sudhan [24] but this study used 18- layer CNN model training five different database. The accuracies varied among them having the highest accuracy of 96.64% with ACRIMA database.

Previous researches aimed to automate the diagnosis of conditions like cataract, glaucoma, and diabetic retinopathy, improving accuracy and efficiency compared to manual analysis. Studies, including Maheshwari et al.'s glaucoma diagnosis and Yang et al.'s neural network-based cataract classification, demonstrated the potential of these approaches. However, challenges such as limited datasets, manual feature extraction, and concerns about Explainability persist. These gaps highlight the need for deep learning algorithms to develop precise, scalable detection methods. Addressing these issues is crucial for advancing practical implementations of deep learning in eye disease classification.

Table 1 shows a comparison between the different mentioned studies.

Table 1 comparison and limitations of previous studies

Study	Disease	Methodology	Accuracy	Limitations
Weni et al. [18]	Cataract	Convolutional Neural Networks (Google Net)	88%	Limited to cataract classification
Yang et al. [8]	Cataract	Artificial Neural Network	84.2%	Limited to cataract classification, Low accuracy
Nath [24]	Glaucoma	18-layer CNN model	96.64%	Limited to classification
Maheshwari et al. [5]	Glaucoma	Optimized kernel parameters, cross-validation	81.32%	Small dataset, single clinical environment, Low accuracy
"AD2Net" [15]	Diabetic Retinopathy	Deep learning with attention mechanism	83.2%	Single-class classification, Low accuracy
Study [11]	Diabetic Retinopathy	Support Vector Mechanism (SVM)	93.8%	Limited to a single database

3. <u>Mathematical Modelling</u>

3.1. CNN

A convolutional neural network (CNN) is a class of artificial neural network most commonly applied to analyze visual imagery. It is influenced by the working mechanism of the visual cortex and the connectivity pattern of neurons in the human brain. Its main advantage is shrinking input images to a simplified form while retaining their features for prediction. It consists of three layers: the convolutional layer, the pooling layer, and the fully connected layer. [25]
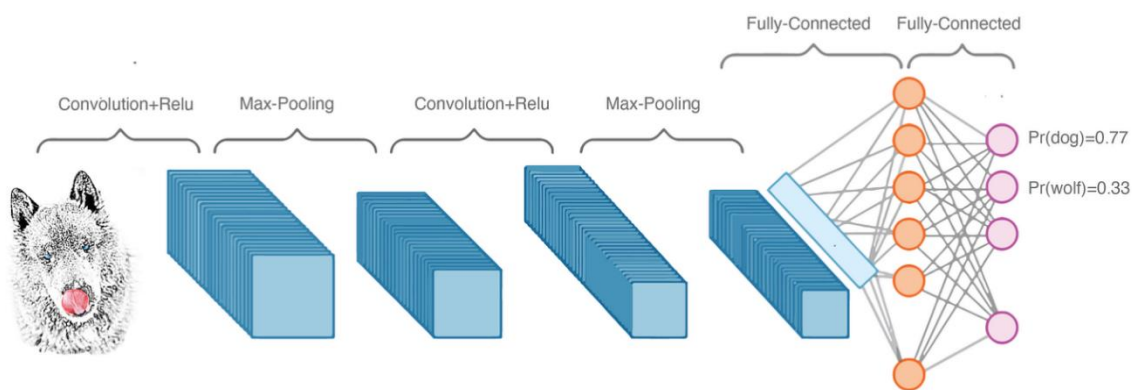

Figure 1 Sequence representation of a basic CNN architecture

I.   Kernel

The convolutional layer is the core building block of a CNN, used to extract features from input images. It requires input data, a filter, and a feature map. The input data is color images with 3D pixels. The filter, or kernel, is a 3x3 matrix with numerical coefficients. The dot product is calculated between input pixels and the filter and fed into an output matrix. [25]
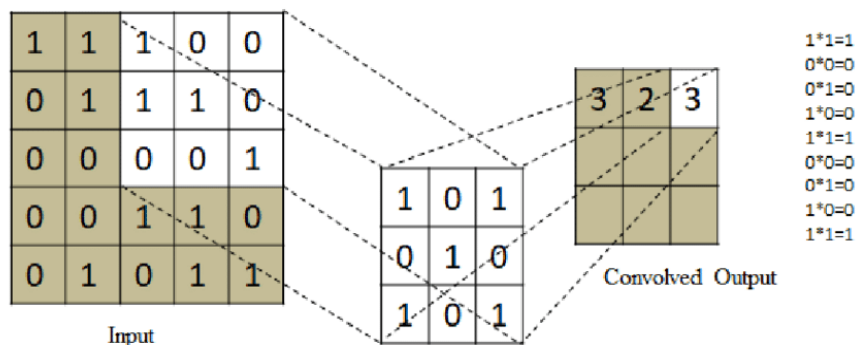

Figure 2 Kernel with size of 3x3

II.   Stride

Stride is a setting that controls how the filter moves over the input image. A larger stride means that the filter moves more quickly, which can reduce the size of the output feature map but also cause some loss of information. Stride is often set to a whole number, such as 1 or 2, to ensure that the output feature map is a consistent size. [26]
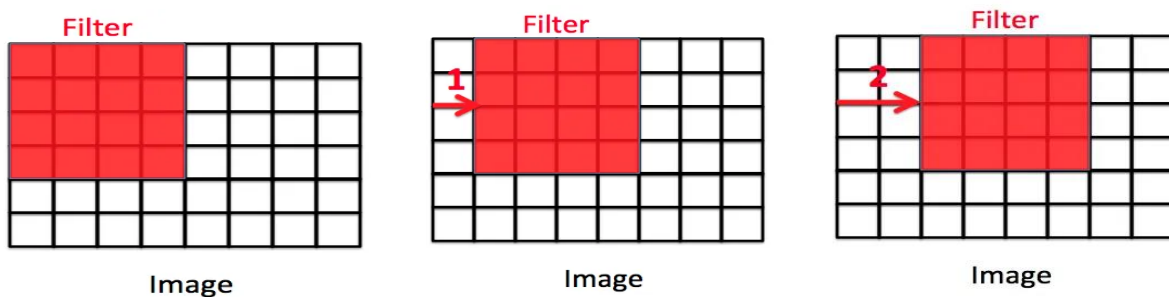

Figure 3 Difference between stride value of 1 and that of 2

III.   Padding

The filter sweeps the input image, causing the loss of corners' features and shrinking the output size. To preserve the original size and extract low-level features, padding is set before convolution. The output shape can be calculated using the formula:

$$\text{Output Shape} = (n - k + 2p + 1) * (n - k + 2p + 1)$$

Where **P** represents the number of padding layers added.

It has two types: valid and same padding. Valid padding is the absence of any additional pixels around the input data, while same padding adds zero pixels, aiming to maintain the output size the same as the input size. [27]
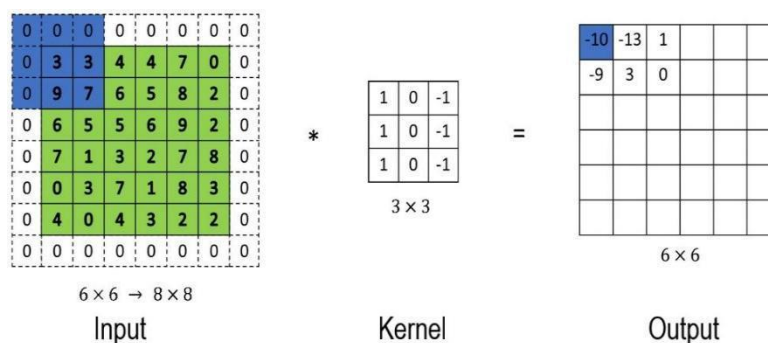

Figure 4 Same padding technique

IV. Pooling

The Pooling Layer in a CNN reduces feature map size while retaining key information. It's a 2D kernel applied to individual maps, summarizing features within the filter's coverage zone. Max pooling is more common.
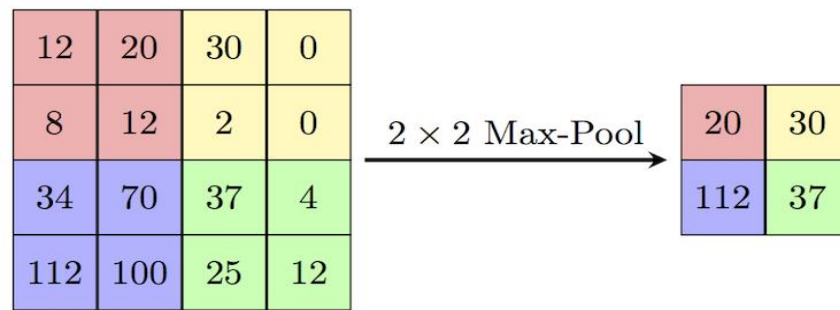

Figure 5 2x2 max pooling method

The CNN initial stages involve repeated convolutional and pooling layers to extract hierarchical features, then the pooled feature maps are flattened to a single-dimension input and passed to the fully connected layer. [28]
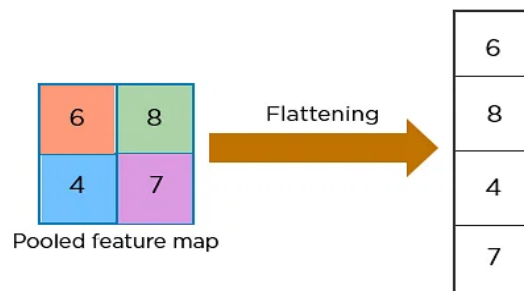
Figure 6 Flattening of a pooled feature map

3.2.   Forward propagation and Backpropagation

The fully connected layer consists of weights, biases, and neurons. It connects the neurons of the hidden layers with each other and with the neurons of the output layer. It takes the previous flattened vectors and performs many mathematical operations on them. In this stage, the classification process takes place to correctly classify the inputs, giving them probabilities from 0 to 1.

To calculate the weights and the biases in all the CNN layers and to train the neural networks, forward propagation and backward propagation are used.
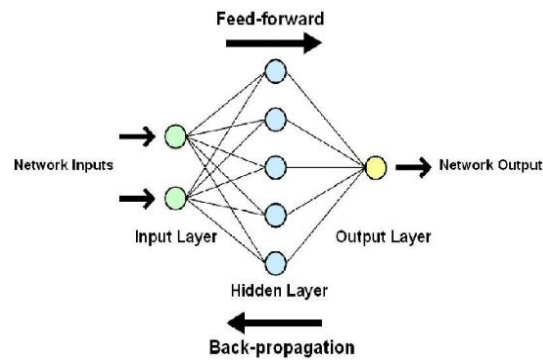
Figure 7 A simple neural network with one hidden layer

I.   Forward propagation

Forward propagation is the process of feeding an input into a neural network, multiplying it by a set of weights then passing it through an activation function -explained soon- then producing the output. It is the first step in training a neural network and is also used to make predictions on new data. [29]

An activation function is a function that generates inputs and discovers relationships from outputs, adding non-linearity to the network and helping it to learn complex relationships between the input and output data. Common activation functions include: the Sigmoid Function, Tanh Function, Rectified Linear Unit (ReLU), and Softmax Function. [30]

The ReLU function is defined as the following:

$$g(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

Figure 8 Relation between input and output values in ReLU activation function

The ReLU function takes an input value x and returns either zero (if x is negative) or the input value itself (if x is non-negative).
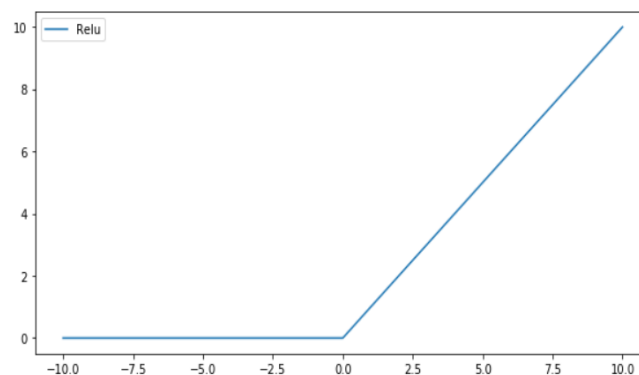
Softmax activation function, which is used in the output layer, is used for multi-classification problems. The mathematical formula involved in the Softmax activation function is given as:

$$\frac{e^{zi}}{\sum_{j=1}^{k} e^{zj}} \qquad for\ i = 1,2,\dots,k\ and\ z = z_1, z_2, \dots, z_k$$

Where z is the input vector, z_i is the elements of the input vector and k is the number of output classes.
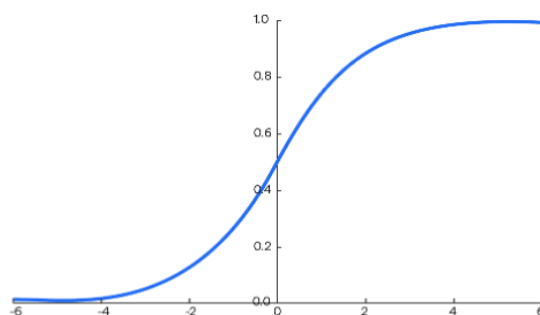
Figure 9 Relation between input and output values in softmax activation function

The Softmax function is a mathematical tool that calculates probabilities by dividing the exponential of each input value by the sum of all exponentials, resulting in a vector of probabilities. [31]
Now proceeding with forward propagation:

Let's say the input of a neural network is a vector $x^{(i)}$, and the output is ŷ. Given a feature vector x, the softmax function returns a vector of probabilities. Assuming the network to be like the following figure:
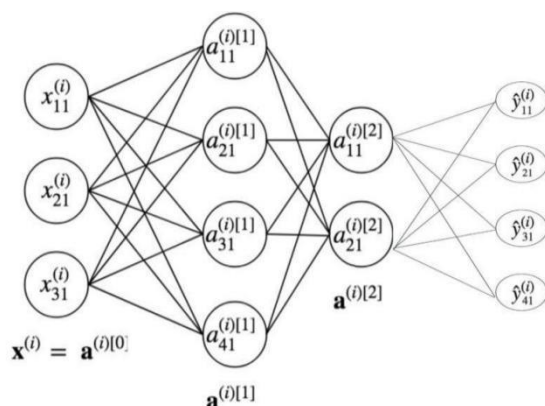
Figure 10 A neural network with 2 hidden layers

14

In this diagram, we have two hidden layers $a^{(i)[1]}$ and $a^{(i)[2]}$ and output layer $a^{(i)[3]}$ we calculate $a^{(i)[j]}$ which is the first entry for the first hidden layer for the training i^{th} example. In order to calculate $a^{(i)[j]}$ we take each entry from $x^{(i)}$ and multiply it by a weight.

$$\mathbf{W}^{[1]} = \begin{bmatrix} W_{11}^{[1]} & W_{12}^{[1]} & W_{13}^{[1]} \\ W_{21}^{[1]} & W_{22}^{[1]} & W_{23}^{[1]} \\ W_{31}^{[1]} & W_{32}^{[1]} & W_{33}^{[1]} \\ W_{41}^{[1]} & W_{42}^{[1]} & W_{43}^{[1]} \end{bmatrix}$$

The weight corresponding to $a_1^{(i)[j]}$ be $W^{[1]} \cdot W^{[1]}$ is a (4, 3) matrix:

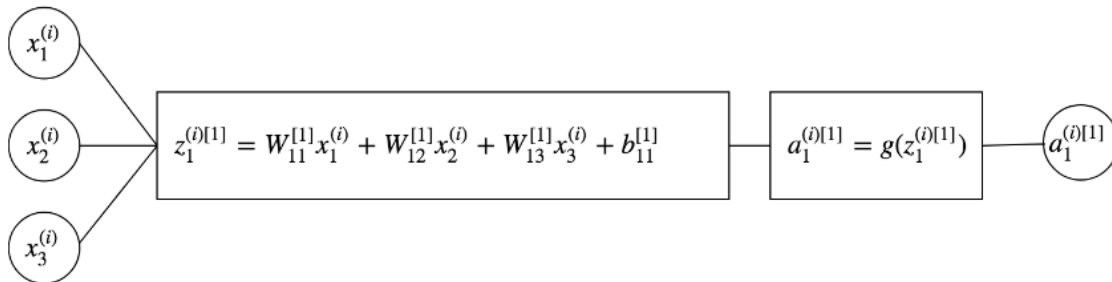The weights that are used to calculate $a_1^{(i)[j]}$ are the weights in the first row.

Then this vector is multiplied by $x^{(i)}$ to get the following representation:

A bias $b_1^{[1]}$ is added to $W_1^{[1]}\_x^{(i)}$. The bias are other parameters besides the weights that the model learns. Bias terms improve generalization and enable the model to make non-zero predictions even with zero inputs by learning optimal values during training.

In summary, forward propagation looks like this:

$$x^{(i)} \rightarrow a^{(i)[1]} \rightarrow a^{(i)[2]} \rightarrow \hat{y}^{(i)}$$

Digging deeper in the inner-workings of each of these transitions:

- **Input → 1st Hidden Layer**

When we transition from our vector of features for the first training example I to the activations from our first hidden layer. We start by multiplying $x^{(i)}$ by the weights and bias of the first hidden layer, $w^{(1)}$ and $b^{(1)}$ to get $z^{(i)[1]}$

$$z^{(i)[1]} = w^{[1]}x^{(i)} + b^{[1]}$$

Once we get the activity matrix $z^{(i)[1]}$, we apply the activation function to each element in $z^{(i)[1]}$. Recall that the activation function that we chose is ReLU. So, we get:

$$a^{(i)[1]} = g\big(z^{(i)[1]}\big)$$

- **1st Hidden Layer → 2nd Hidden Layer ($a^{(i)[1]} \rightarrow a^{(i)[2]}$)**

Just like the previous section, we start by multiplying $a^{(i)[1]}$ by the weights and bias of the second hidden layer, $w^{[2]}$ and $b^{[2]}$ to get $z^{(i)[2]}$ and we get:

$$a^{(i)[2]} = g\big(z^{(i)[2]}\big)$$

- **2nd Hidden Layer → Output ($a^{(i)[2]} \rightarrow \hat{y}^{(i)}$)**

There is $a^{(i)[2]}$ again so $a^{(i)[2]}$ is multiplied by $w^{[3]}$ then add bias $b^{[3]}$.

Since this is the final layer of the neural network, the Softmax activation function will be applied to each of these output neurons. It calculates the relative probabilities, that means it uses the value of $\hat{y}^{(i)}_{11}$, $\hat{y}^{(i)}_{21}$, $\hat{y}^{(i)}_{31}$ and $\hat{y}^{(i)}_{41}$ to determine the final probability value yielding to:

$$\hat{y}^{(i)} = soft\big(z^{(i)[3]}\big)$$

II.   Backpropagation

The input data is propagated through the network, computing activation and producing predicted output. Backpropagation is used to compute the loss by comparing the predicted output to the desired output. This algorithm calculates gradients of the loss function using the chain rule of calculus, adjusting the network's weights and biases to minimize loss. [32]

Let the dataset denoted by X consists of the pair (x_i, y_i), where x_i is the input and y_i is the desired output of the neural network. Let the predicted output be denoted by $\hat{y}_i$ then the error function E (X, θ), which is used to quantify how well the neural network is performing, can be calculated as follows: [33]

$$E\,(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \qquad\qquad (1)$$

Where θ is the parameter of the network and N is the set of natural numbers

The bias b_i^k for node *I* in layer *k* will be incorporated into the weights as w_{0i}^k with a fixed output of o_0^{k-1} = 1 for node 0 in layer *k*−1. Thus, the activation function will be:

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ji}^k o_j^{k-1} = \sum_{j=0}^{r_{k-1}} w_{ji}^k o_j^{k-1} \tag{2}$$

Calculating the value of $\frac{\partial E}{\partial w_{ij}^k}$ applied to equation (1) to each weight w_{ij}^k results in:

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^N \frac{\partial\left(\frac{1}{2}(\hat{y}_d - y_d)^2\right)}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^N \frac{\partial E_d}{\partial w_{ij}^k}$$

For the sake of simplicity, only one input-output pair will be considered. So, the error function, equation (1), will be:

$$E = \frac{1}{2}(\hat{y} - y)^2 \tag{3}$$

By applying the chain rule to the error function partial derivative.

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k} \cdot \frac{\partial a_j^k}{\partial w_{ij}^k} \tag{4}$$

The first term is usually called the error. It is denoted by:

$$\delta_j^k \equiv \frac{\partial E}{\partial a_j^k} \tag{5}$$

The second term can be calculated from the equation for a_j^k, equation (2):

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial\left(\sum_{l=0}^{r_{k-1}} w_{lj}^k o_l^{k-1}\right)}{\partial w_{ij}^k} = o_i^{k-1} \tag{6}$$

By substituting equations (5) and (6) into (4), we get:

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} \tag{7}$$

Applying the previous steps to the output layer as backpropagation starts from that layer moving backwards, the error function, equation (3), will be:

$$E = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}\left(g_o(a_j^m) - y\right)^2$$

Where $g_o(x)$ is the activation function for the output layer and m is the final layer.

Applying the chain rule to the error function partial derivative will yield to:

$$\frac{\partial E}{\partial w_{i1}^m} = \delta_j^m o_i^{m-1} \tag{8}$$

Where: $\delta_j^m = \left(g_o(a_j^m) - y\right) \cdot g_o'(a_j^m) = (\hat{y} - y) \cdot g_o'(a_j^m)$

So, equation (8) can be written as follows:

$$\frac{\partial E}{\partial w_{i1}^m} = (\hat{y} - y) \cdot g_o'(a_j^m) o_i^{m-1}$$

To calculate the error function of layers other than the output layer. The error term δ_j^k in layer 1 $\leq k < m$ can be calculated from the following equation:

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial E}{\partial a_l^{k+1}} \cdot \frac{\partial a_l^{k+1}}{\partial a_j^k} \tag{9}$$

Where a_l^{k+1} is equal to:

$$a_l^{k+1} = \sum_{j=1}^{r^k} w_{jl}^{k+1} \cdot g(a_j^k)$$

Where $g(x)$ is the activation function for the hidden layers, so that expression $\frac{\partial a_l^{k+1}}{\partial a_j^k}$ can be calculated as follows:

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1} \cdot g'(a_j^k) \tag{10}$$

Applying equation (10) into equation (9) yields to what is called as propagation formula:

$$\delta_j^k = g'(a_j^k) \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1} \tag{11}$$

Therefore, applying equation (11) into error function, equation (7), will result to:

$$\frac{\partial E}{\partial w_{ij}^k} = g'(a_j^k) o_i^{k-1} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1}$$

Finally, Network's weights can be updated, reducing the loss, using equation (12):

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X,\theta)}{\partial w_{ij}^k} \tag{12}$$

Where α represents the learning rate, that determines the step size of the update

4. <u>Proposed Methodology</u>

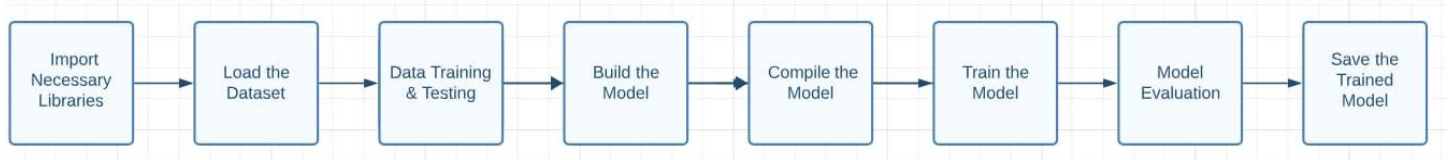Before delving into the methodology, presented here is a block diagram illustrating the key components:

Figure 11 Key components of methodology

4.1   Dataset

The dataset utilized in our model, titled "Eye Disease Classification," was obtained from Kaggle. This dataset comprises a diverse collection of retinal images categorized into four distinct disease types: Normal, Diabetic Retinopathy, Cataract, and Glaucoma. Each class is well-represented, with approximately 1000 images, contributing to a balanced dataset.

The distribution of images across disease types is as follows:

- Diabetic Retinopathy: 1098 images
- Normal: 1074 images
- Cataract: 1038 images
- Glaucoma: 1007 images

These images are sourced from various reputable repositories, including IDRiD (Indian Diabetic Retinopathy Image Dataset), Ocular Recognition, HRF (Human Retina Fundus), and others. The incorporation of images from diverse sources ensures the dataset's richness and relevance in capturing the wide spectrum of eye diseases.

The distribution of diseases within the dataset is visually depicted in the following chart:
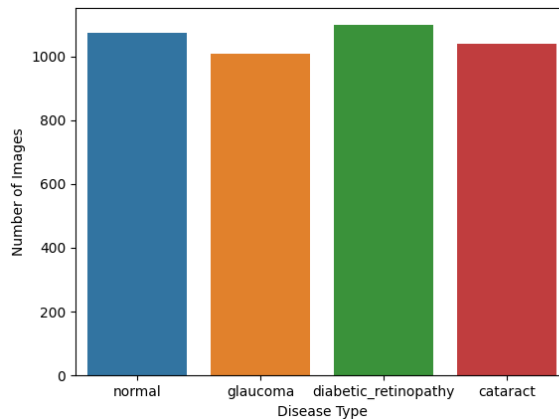

Figure 12 Distribution of the dataset among the diseases

Sample training images from distinct disease categories, including Diabetic Retinopathy, Normal, Cataract, and Glaucoma, are presented for visual exploration. These images, carefully chosen from the training set, offer a visual understanding of the unique characteristics associated with each disease type. The visualization provides insights into the diversity of visual cues present in the training dataset, contributing to a precised comprehension of the data. [34]
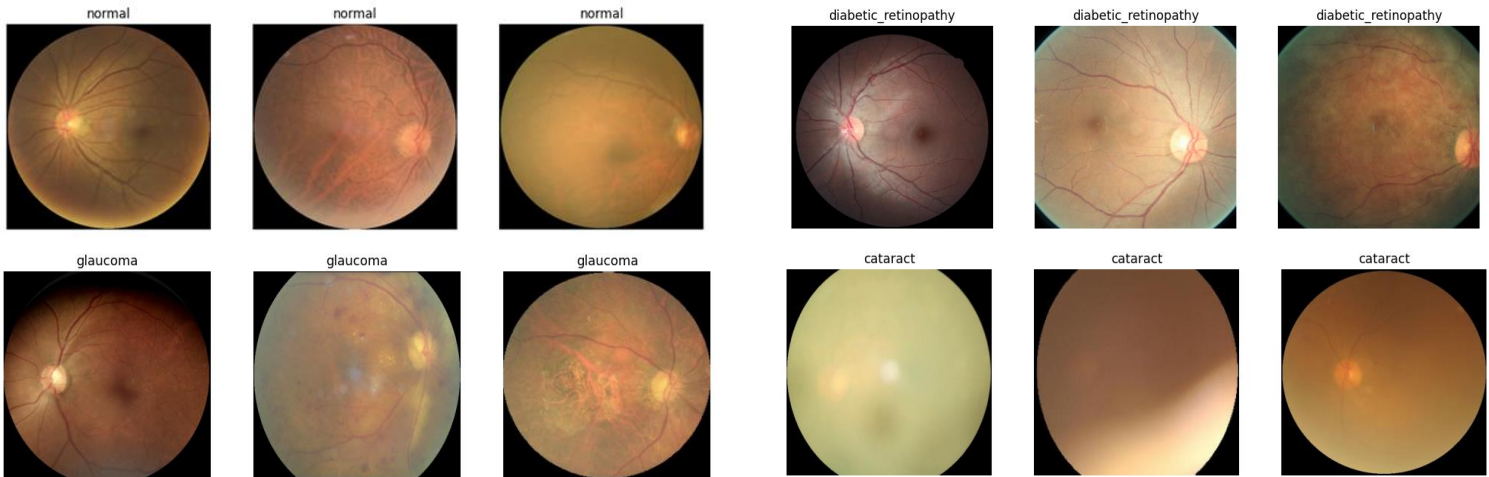

Figure 13 Sample training images for the different diseases

4.2   Data Training Phase

The data training phase involved meticulous steps to prepare the dataset for model development. The dataset was meticulously divided into training and validation sets, with 20% allocated to the latter, ensuring a thorough evaluation of the model's performance. The training set comprised 3374 images, while the validation set contained 843 images.

Central to enhancing the model's generalization capabilities was the use of an Image Data Generator. This tool served a dual purpose by seamlessly integrating data augmentation and preprocessing during the model training process. Data augmentation played a pivotal role in mitigating overfitting and bolstering the model's capacity to generalize. Random transformations, including rotation, zooming, and horizontal flipping, enriched the training dataset, exposing the model to a diverse array of examples.

The preprocessing function within the generator ensured that the input images were appropriately formatted for the neural network. This involved the normalization of pixel values to a standardized scale, aligning with the architectural requirements of the model.

Data generators were instantiated using the 'flow_from_dataframe' method, seamlessly bridging the gap between the DataFrame and the model. The training data generator was meticulously configured with specific parameters: a target size of (224, 224), a batch size of 32, and shuffling enabled to inject randomness during training. This precision resulted in the training data generator yielding 3,374 validated image filenames, distributed across four distinct classes.

Simultaneously, a separate data generator for the validation set adhered to comparable specifications. Notably, shuffling was disabled to uphold order during the validation phase. Consequently, this generator generated 843 validated image filenames, encompassing the same four classes. [35]

4.3   Building the model

Before delving into the model development, it's essential to understand the concept of transfer learning and its relevance to our eye disease classification project.

Transfer learning involves leveraging knowledge acquired from solving one problem and applying it to a different but related problem. By harnessing the learned features, patterns, or representations of a pre-trained model, we can enhance the model's performance or generalization on a new, similar task. For this project, we opted to use the pre-trained VGG19 model due to its ability to understand intricate image features. [36]

VGG19 is a Convolutional Neural Network (CNN) designed for image classification. Here are the key architectural aspects:

- Input Size: A fixed size of (224 * 224) RGB image was used, creating a matrix shape of (224, 224, 3).

- Preprocessing: Mean RGB value subtraction from each pixel, calculated across the entire training set, was the only preprocessing step applied.

- Convolutional Layers: Utilized (3 * 3) sized kernels with a stride size of 1 pixel, enabling comprehensive coverage of image features.

- Spatial Padding: Employed to retain the spatial resolution of the image.

- Pooling: Max pooling over 2*2-pixel windows with a stride of 2 was used for down sampling.

- Activation Function: Rectified Linear Unit (ReLU) was utilized for introducing non-linearity, improving classification and computational efficiency over earlier models that used tanh or sigmoid functions.

- Fully Connected Layers: Included three fully connected layers, with the first two having a size of 4096, followed by a layer with 1000 channels for 1000-way ILSVRC classification. The final layer employed a softmax function for classification.

4.4 Model Structure and Integration

To enhance our model, we've integrated the pre-trained VGG19 architecture and augmented it with additional layers to further refine its capabilities. [37]

I. VGG19 Integration:

This step enables our model to comprehend complex patterns and textures within eye images more effectively.

II. Flattening Layer:

After the VGG19 integration, a flattening layer is introduced. This layer serves to convert the multi-dimensional output from the VGG19 into a one-dimensional array, facilitating the seamless transition to subsequent dense layers.

III. Dense Layers:
   1. First Dense Layer (256 neurons, ReLU activation):

Following the flattened output, the model features a dense layer comprising 256 neurons activated by Rectified Linear Unit (ReLU). This layer is instrumental in learning complex patterns and representations derived from the preceding layers.

Batch normalization is introduced after the first dense layer helping in stabilizing and accelerating the training process. It aids in reducing internal covariate shift, ultimately improving the model's performance and convergence speed.

   2. Second Dense Layer (256 neurons, ReLU activation):

A subsequent dense layer consisting of 256 neurons, also activated by ReLU, further refines the model's capability to extract and comprehend intricate features, enhancing its ability to discern patterns within the data.

IV. Output Layer (4 neurons, Softmax activation):
The final layer comprises 4 neurons, each representing a specific eye disease class. Utilizing the Softmax activation function, this layer conducts the ultimate classification, assigning probabilities to each disease class. The class with the highest probability becomes the model's prediction for a given input image.
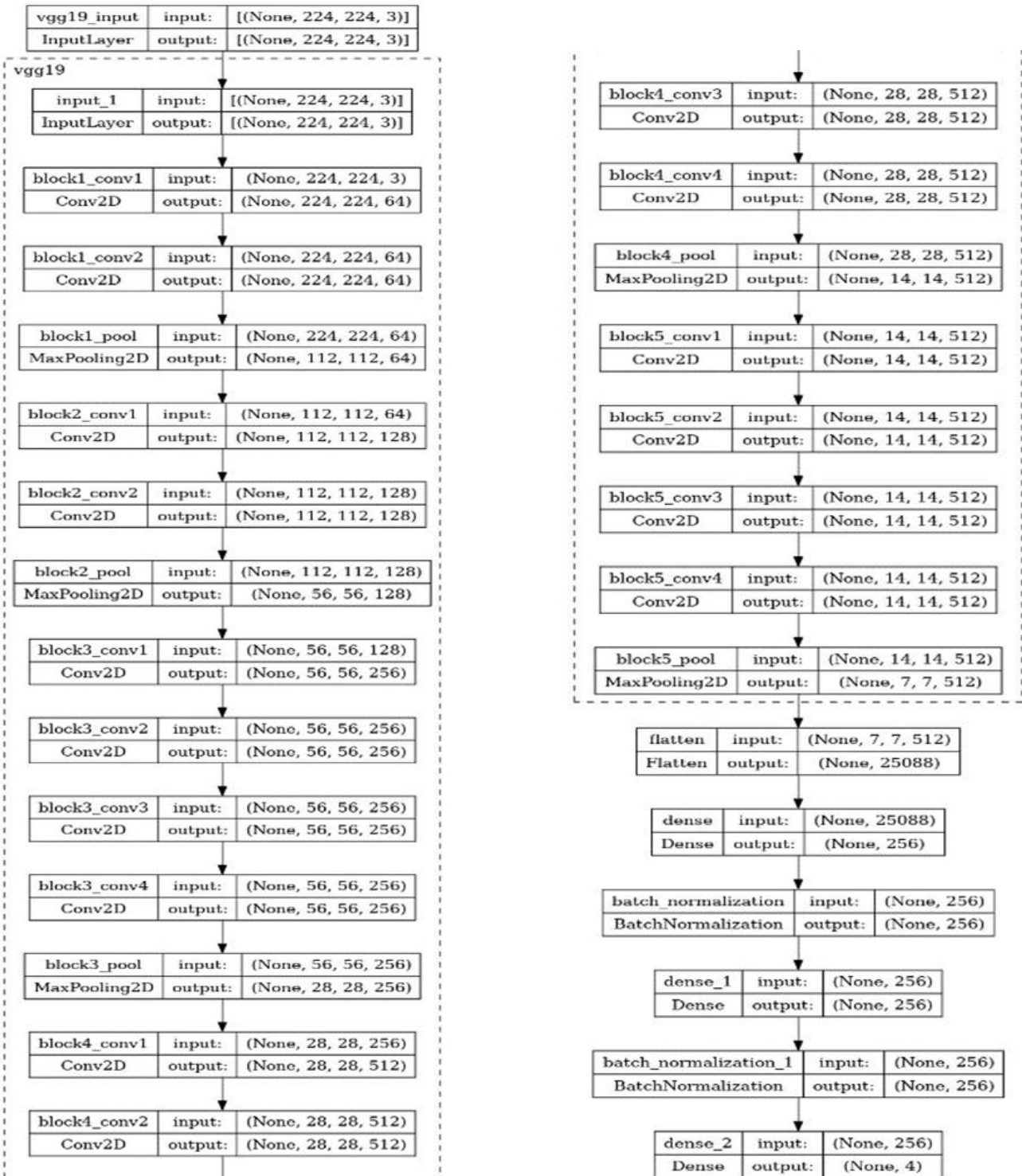
Figure 14 CNN Model

5. <u>Experimental Work</u>

5.1 Training the model

After constructing the deep learning model architecture, the next crucial step involved initiating the training process through the model compilation phase. This pivotal step utilizes a built-in function within the Keras framework, designed explicitly to configure the model for training. The function accommodates various essential arguments that define crucial aspects of the training procedure, such as the optimizer, loss function, and metrics.

The optimizer parameter within the compile function serves to select the optimization algorithm that dictates how the model's parameters are updated during training. In this context, 'Adam' was chosen, known for its adaptive learning rates and efficiency in adjusting model weights.

Loss function is the function used for backpropagation. It computes the quantity that a model should seek to minimize during training.

Categorical_crossentropy was employed, tailored for multi-class classification tasks. It quantifies the dissimilarity between predicted class probabilities and the true class labels.

Categorical_crossentropy function is described by the following equation:

$$J(w,b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} \sum_{j=1}^{C} y_j^{(i)} \cdot log\left(f\left(w, b\big(x^{(i)}\big)\right)_j \right)$$

where $J(w,b)$ is the loss function, w and b as weights and biases, m_{train} as the number of training examples, $y_j^{(i)}$ as the true label for class j in the ith example, $x^{(i)}$ as the input, and $f\left(w, b\big(x^{(i)}\big)\right)_j$ as the predicted probability for class j given the input and model's parameters.

The metrics argument outlines the metrics to be observed and tracked during the training process. This could encompass metrics such as accuracy, precision, or recall. In this instance, accuracy was selected to evaluate the model's performance.

This model summary provided an overview of the sequential architecture, detailing layer parameters and a total of 26,516,036 parameters, of which 6,490,628 were trainable.

```
Model: "sequential"
_____
 Layer (type)                Output Shape              Param #
=================================================================
 vgg19 (Functional)          (None, 7, 7, 512)         20024384

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 256)               6422784

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dense_1 (Dense)             (None, 256)               65792

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 dense_2 (Dense)             (None, 4)                 1028

=================================================================
Total params: 26,516,036
Trainable params: 6,490,628
Non-trainable params: 20,025,408
_____
```

Figure 15 Model Summary

Now, transitioning to the training phase, let's explore the **fit ()** function, an essential step that employs defined data, parameters, and callbacks to train the model effectively.

Here's an overview of its components:

- Training Data: It's the dataset used for model training.

- Batch Size: Specifies the number of samples used in each iteration during training.

- Epochs: Denotes the number of times the model iterates over the entire dataset.

- Validation Data: This set serves as a benchmark during training to evaluate model performance.

- Verbosity: Controls the level of details shown in the training logs.

- Callbacks: These are functions called during training at defined points. In this case, Model Checkpoint saves the best model based on validation accuracy, and Early Stopping halts training if there's no improvement in validation accuracy after a certain number of epochs (determined by the patience parameter).

5.2 Test scenarios

I.   Cataract Detection Model

The primary objective at the inception of the project was the development of an accurate cataract detection model. The focus was on implementing a Convolutional Neural Network (CNN) tailored for binary classification. This involved the usage of a dataset comprising images categorized into two classes: normal and cataract as shown below.
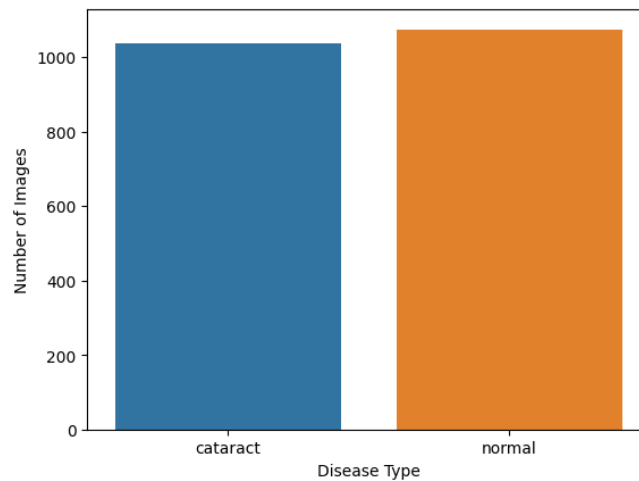

Figure 16 Distribution of the dataset

The CNN architecture excelled in binary classification, achieving promising results. Rigorous validation procedures ensured the model's reliability (Accuracy: 96.68%, Loss: 0.4733 at best epoch).
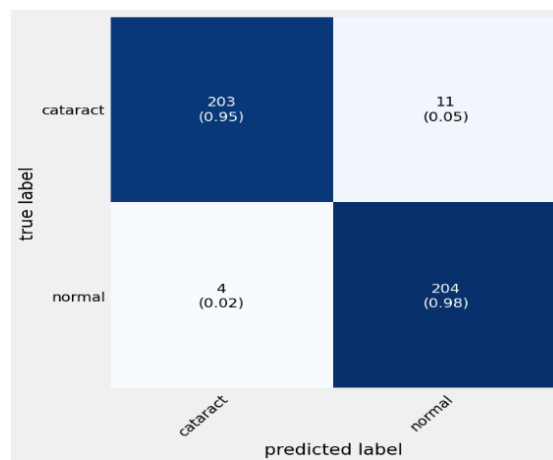

Figure 17 Predicted & True Label for cataract detection model

Figure 18 Training and validation accuracy & loss for cataract detection model

The success of this phase laid the foundation for the project's evolution into a more comprehensive eye disease classification system.

II.   Evolution to Eye Disease Classification

1.   Initial Attempt with a Simple CNN

The first step towards multi-class classification involved the implementation of a simple Convolutional Neural Network (CNN) with four convolutional layers. Despite the simplicity of the architecture, the model demonstrated limited success, as reflected in the validation metrics (Accuracy: 86.95%, Loss: 0.9638).

2.   Transition to Advanced Models: VGG19

Recognizing the need for a more advanced approach, the project shifted towards utilizing pre-trained models. VGG19, a deep convolutional neural network renowned for its performance in image classification tasks, as mentioned in the methodology section, was selected. The model exhibited highly satisfying results, reaching a minimum validation loss of 0.3127 and an accuracy of 91.46% at its best epoch.
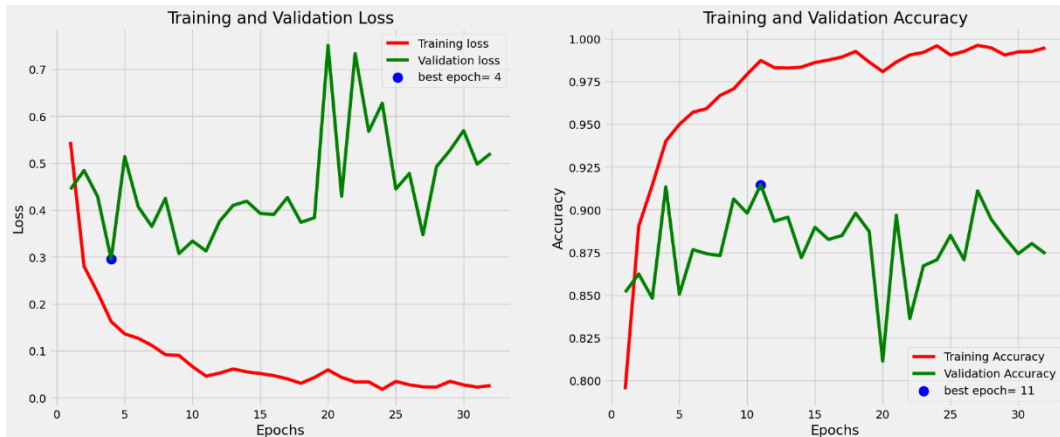
Figure 19 Training and validation accuracy & loss for eye diseases classification model

5.3 Model Deployment

I.   Web Application

The creation of a user-friendly web application using JavaScript, CSS, and HTML was done. This application is complemented by a Restful API implemented in ASP.NET for seamless integration with the model. Users can conveniently access the model through a singular endpoint, namely:
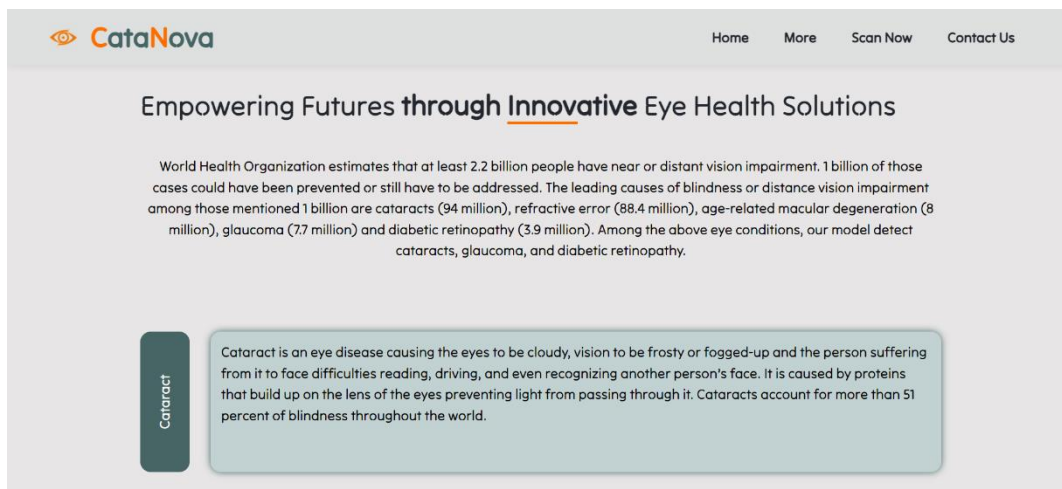
https://dlmodel-001-site1.btempurl.com/?

Figure 20 snapshot of the web application

II.   Mobile Application

To broaden the model's accessibility, a mobile application has been developed using the Flutter framework. The app serves to showcase the model's capabilities, deliver disease information, and enable users to upload retinal images for accurate predictions via the integrated API.
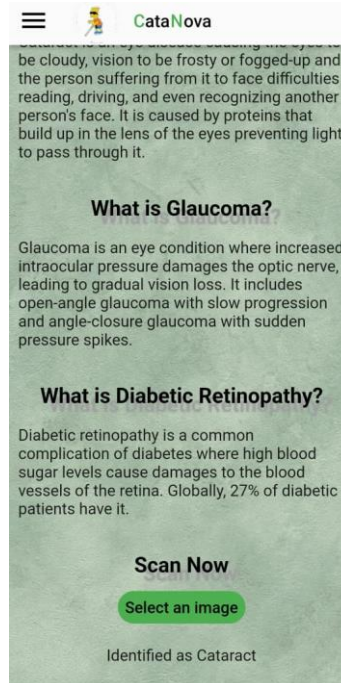
Figure 21 snapshot of the mobile application showing the result of a scanned image

III.   API Integration

The API allows users to input test images, which undergo preprocessing through a Python script integrated into the C# codebase. The processed images are then sent to the model for prediction, with results returned as text. This integration ensures consistent user experience across our website and mobile application.

Figure 21 The result of scanning an image in the web application

6. <u>Result and Analysis</u>
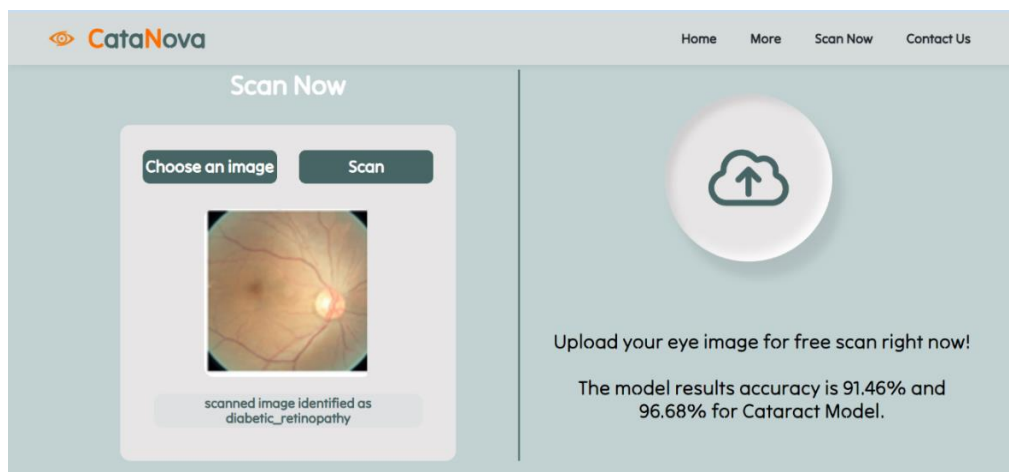
The Eye Disease Classification model demonstrated robust performance across multiple eye disease categories. The following metrics were calculated for each class:

```
                      precision   recall  f1-score   support

            cataract       0.90     0.94      0.92       203
 diabetic_retinopathy      1.00     1.00      1.00       223
            glaucoma       0.80     0.83      0.81       204
              normal       0.85     0.79      0.82       213

            accuracy                          0.89       843
           macro avg       0.89     0.89      0.89       843
        weighted avg       0.89     0.89      0.89       843
```

Figure 22 Classification Report

The model was evaluated on an independent test set, confirming its real-world applicability.

Figure 23 Predicted & True Label for eye diseases classification model.
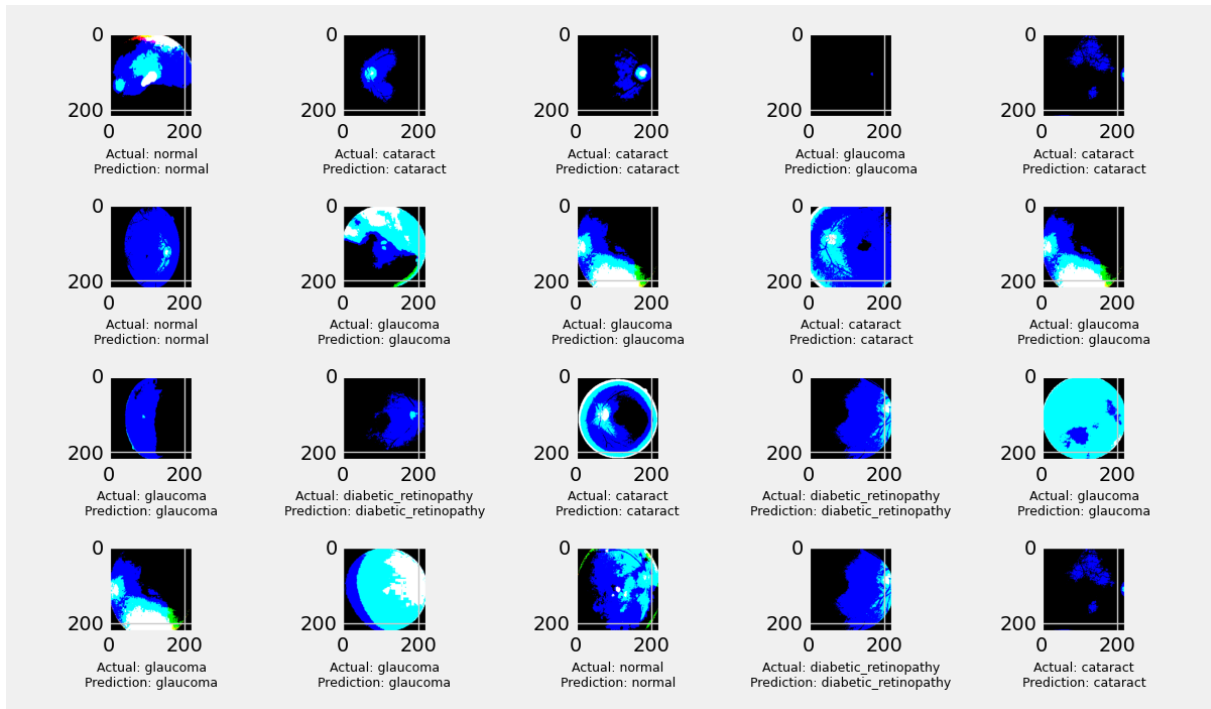
Figure 24 Model Evaluation

The results showcase the evolution from cataract detection to a comprehensive eye disease classification system, emphasizing the model's effectiveness and potential areas for enhancement.

7. Conclusion

This research tries to offer an accurate and instant solution for detection and categorization of various eye diseases by extracting and analyzing features from retinal fundus images for glaucoma, diabetic retinopathy, cataracts and normal conditions. On implementing a Convolutional Neural Network (CNN) for detection of only cataract eye condition. It reached accuracy of 96.68% and validation loss of 0.4733 at best epoch. Then improving the model to detect and classify the previously mentioned four eye conditions first by using CNN, it reached an accuracy of 86.95% and loss of 0.9638 then by using VGG19 model reaching an accuracy of 91.46% and minimum validation loss of 0.3127 at best epoch.

These findings highlight deep learning's potential to transform the categorization of eye diseases and hold promise for early disease detection, facilitating timely intervention and personalized treatment strategies that can prevent vision loss.

8. Future work

Future work can consider broadening the diversity of the dataset by having additional images that encompass a more extensive array of scenarios, including diverse lighting conditions and demographic representations that can improve the model's robustness.

Using advanced data augmentation techniques like CutMix, MixUp, or domain-specific augmentations can also be considered to generate a more diverse set of training samples, potentially boosting the model's ability to handle various input variations.

The project can be enhanced through understanding which parts of an image contribute most to predictions that will improve decision-making process, transparency and trust in the model. This can be done by implementing interpretability techniques like Grad-CAM, SHAP (SHapley Additive exPlanations), or LIME (Local Interpretable Model-agnostic Explanations).

9. <u>Reference</u>

1. World Health Organization (2023), *Blindness and vision impairment*
2. Ophthalmology, *A Brief Look at Cataract Statistics*
3. Centers for Disease Control and Prevention (2020), *Don't Let Glaucoma Steal Your Sight!*
4. National Library of Medicine (2021), *Prevalence and factors associated with Diabetes retinopathy among type 2 diabetic patients*
5. Maheshwari, S., Pachori, R. B., & Acharya, U. R. (2017*). Automated Diagnosis of Glaucoma Using Empirical Wavelet Transform and Correntropy Features Extracted From Fundus Images*. IEEE Journal of Biomedical and Health Informatics
6. Acharya, U. R., Bhat, S., Koh, J. E. W., Bhandary, S. V., & Adeli, H. (2017*). A novel algorithm to detect glaucoma risk using texton and local configuration pattern features extracted from fundus images* . Computers in Biology and Medicine
7. KRISHNAN, M. M. R., & FAUST, O. (2013). *AUTOMATED GLAUCOMA DETECTION USING HYBRID FEATURE EXTRACTION IN RETINAL FUNDUS IMAGES* . Journal of Mechanics in Medicine and Biology
8. Yang M, Yang J-J, Zhang Q, Niu Y, Li J (2013). *Classification of retinal image for automatic cataract detection*. IEEE international conference on e-health networking, applications & services
9. Agarwal, V., Gupta, V., Vashisht, V. M., Sharma, K., & Sharma, N. (2019).*Mobile Application Based Cataract Detection System*. *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*
10. Verma, K., Deep, P., & Ramakrishnan, A. G. (2011).*Detection and classification of diabetic retinopathy using retinal images* . *2011 Annual IEEE India Conference.*
11. 10 Carrera, E. V., Gonzalez, A., & Carrera, R. (2017).*Automated detection of diabetic retinopathy using SVM* . *2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON).*
12. Gargeya, R., & Leng, T. (2017). *Automated Identification of Diabetic Retinopathy Using Deep Learning. Ophthalmology*
13. Ouda, O., Abdelmaksoud, E., Abdelaziz, A. A., & Elmogy, M. (2022*).Multiple Ocular Disease Diagnosis Using Fundus Images Based on Multi-Label Deep Learning Classification*. 2022 Electronics.
14. Jiang, H., Yang, K., Gao, M., Zhang, D., Ma, H., & Qian, W. (2019*). An Interpretable Ensemble Deep Learning Model for Diabetic Retinopathy Disease Classification*. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).*
15. Qian, Z., Wu, C., Chen, H., & Chen, M. (2021).*Diabetic Retinopathy Grading Using Attention based Convolution Neural Network* . *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC).*
16. Omar, M., Khelifi, F., & Tahir, M. A. (2016). *Detection and classification of retinal fundus images exudates using region based multiscale LBP texture approach*. *2016 International Conference on Control, Decision and Information Technologies (CoDIT).*

17. Kathiresan, S., Sait, A. R. W., Gupta, D., Lakshmanaprabu, S. K., Khanna, A., & Pandey, H. M. (2020).*Automated detection and classification of fundus diabetic retinopathy images using synergic deep learning model* . *Pattern Recognition Letters.*

18. Weni, I., Utomo, P. E. P., Hutabarat, B. F., & Alfalah, M. (2021).*Detection of Cataract Based on Image Features Using Convolutional Neural Networks.* 2021 Indonesian Journal of Computing and Cybernetics Systems (IJCCS)

19. Ju Lai, C., Feng Pai, P., Marvin, M., Han Hung, H., Han Wang, S., & Nan Chen, D. (2022).*The Use of Convolutional Neural Networks and Digital Camera Images in Cataract Detection.* 2022 Electronics

20. Hasan, M. K., Tanha, T., Amin, M. R., Faruk, O., Khan, M. M., Aljahdali, S., & Masud, M. (2021).*Cataract Disease Detection by Using Transfer Learning-Based Intelligent Methods.* Knowledge-Based Intelligent Systems in E-Health and Medical Communication Services

21. Chen X, Xu Y, Kee Wong DW, Wong TY, Liu J (2015). *Glaucoma detection based on deep convolutional neural network.* 2015 37th annual international conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan.

22. Gómez-Valverde, J. J., Antón, A., Fatti, G., Liefers, B., Herranz, A., Santos, A., … Ledesma-Carbayo, M. J. (2019).*Automatic glaucoma classification using color fundus images based on convolutional neural networks and transfer learning* *Biomedical Optics Express*

23. Sudhan, M. B., Sinthuja, M., Pravinth Raja, S., Charlyn Pushpa Latha, G., Sheeba Rachel, S., Anitha, T., Rajendran, T., & Asrat Waji, Y. (2022) *Segmentation and Classification of Glaucoma Using U-Net with Deep Learning Model.* Explainable Artificial Intelligence for Medical Applications

24. Elangovan, P., & Nath, M. K. (2020). *Glaucoma assessment from color fundus images using convolutional neural network. International Journal of Imaging Systems and Technology*

25. Mairal, Julien & Koniusz, Piotr & Harchaoui, Zaid & Schmid, Cordelia. (2014). *Convolutional Kernel Networks. Advances in neural information processing systems.*

26. DeepAI. (2019, May 17). Stride (machine learning). *https://deepai.org/machine-learning-glossary-and-terms/stride*

27. CNN | Introduction to padding. (2023, May 23). GeeksforGeeks. *https://www.geeksforgeeks.org/cnn-introduction-to-padding/*

28. *What are Convolutional neural networks? (n.d.). IBM - United States. https://www.ibm.com/topics/convolutional-neural-networks*

29. Osajima, J. (n.d.). The math behind neural networks - Forward propagation. Jason {osa-jima}. *https://www.jasonosajima.com/forwardprop*

30. What is activation function? (n.d.). H2O.ai | The fastest, most accurate AI Cloud Platform. https://h2o.ai/wiki/activation-function/

31. Feedforward Neural Networks. Brilliant.org. Retrieved 21:47, November 24, 2023, from *https://brilliant.org/wiki/feedforward-neural-networks/*

32. Backpropagation with Softmax / Cross entropy. (n.d.). Cross Validated. *https://stats.stackexchange.com/questions/235528/backpropagation-with-softmax-cross-entropy*

33. Backpropagation. Brilliant.org. Retrieved 21:29, November 24, 2023, from *https://brilliant.org/wiki/backpropagation/*

34. Eye_diseases_classification. (n.d.). Kaggle: Your Machine Learning and Data Science Community. *https://www.kaggle.com/datasets/gunavenkatdoddi/eye-diseases-classification*

35. *Shah, T. (2020, July 10). About train, validation and test sets in machine learning. Medium. https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7*

36. *Guide to transfer learning. (2023, September 5). Data Engine for AI Model Development | Encord. https://encord.com/blog/transfer-learning/*

37. *Bilal, Muhammad & Maqsood, Muazzam & Yasmin, Sadaf & Ul Hasan, Najam & Rho, Seungmin. (2022). A transfer learning-based efficient spatiotemporal human action recognition framework for long and overlapping action classes. The Journal of Supercomputing. 78. 10.1007/s11227-021-03957-4.*

APPENDIX A

Eye Diseases Classification Repo on GitHub

Presented by:

NAME	ID
AHMED RAAFAT MOHAMED	9220038
AHMED AMGAD EID IBRAHIM	9220021
ABDULLAH MAHMOUD HANAFY	9220448
HASNAA HOSSAM HASSANEIN	9220267
SOMAIA AHMED ABDELRAHMAN	9220374
SALAH MOHAMED SALAH ISMAEL	9220413
MOHAMED AHMED ALI HASSAN	9220650
AYAT TAREK ELROUBY ALI	9210263