Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

DIABETIC RETINOPATHY IMAGE CLASSIFICATION WITH NEURAL NETWORKS

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN SISTEMAS COMPUTACIONALES

Presenta: **IVÁN SÁNCHEZ FERRUSCA**

Asesor **MTRO. VÍCTOR HUGO MARTÍNEZ SÁNCHEZ**

Tlaquepaque, Jalisco. noviembre de 2021.

ACKNOWLEDGMENTS

I would like to acknowledge:

To Mtro. Víctor Hugo Martínez, the person who guides and supports me with information material and his vast knowledge in programming and deep learning.

The program coordinator of the master's degree in Computer Systems at ITESO, Dr. Iván Villalón, for offering the opportunity to study this postgraduate.

To CONACYT, for the scholarship that was very helpful for me during all the master periods.

To Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO), for the agreement support with my current company employee, Continental Automotive.

DEDICATION

This work is dedicated to:

To my parents. The achieved goals are a reflection of the dedication, support, and love that parents invest in their children. Thanks to my parents I am a person wanting to improve, proudly and with my face held high I thank José Luis Sánchez and Leticia Ferrusca, my greatest inspiration.

To my brothers, José Luis, and Edgar. You have been a very important part of my life and because you trust in me and always give me your unconditional support.

ABSTRACT

The world is experiencing an increased life expectancy, which results in a natural increase in the chance of getting a disease. The main concern is that some of the methods to determine an affectation are not so fast and need expert people. Therefore, it is necessary to create new low-cost mechanisms of diagnosis that can give us fast and better results.

Recent studies have been implemented using known architectures getting high scores of accuracies. An experimental classification model was implemented in this work using Python libraries. This is an experimental model with custom neural network architecture. This work intends to contrast the results using a model based on the AlexNet against my experimental architecture.

The 2 main reasons to compare my work versus AlexNet is that during my investigation of the state of the art I did not find researches to solve the DR categorization using this architecture and also if I had chosen other architecture, I would need more powerful computing. In the end, AlexNet was not a good solution.

This solution will help the healthcare industry to have a less expensive and non-invasive way to determine if a person is being affected by diabetic retinopathy, depending on the damage shown on their retinas.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

AI		Artificial Intelligence
AUC		Area Under Curve
API		Application Programming Interface
CL		Convolutional Layer
CNN		Convolutional Neural Network
CPU		Central Processing Unit
CSV		Comma Separated Value
DL		Deep Learning
DR		Diabetic Retinopathy
DT		Decision Tree
EAD		Encoder, Attention, Decoder
FC		Fully Connected
FN		False Negative
FP		False Positive
GPU		Graphics Processing Unit
HDF5		Hierarchical Data Format version 5
IRMA		Intra Retinal Microvascular Anomalies
LR		Logistic Regression
MAE		Mean Absolute Error
ML		Machine Learning
MSE		Mean Squared Error
NLP		Natural Language Processing
NN		Neural Network
PL		Pooling Layer
ReLU		Rectified Linear Activation Unit
RFC		Random Forest Classifier
RGB		Red-Green-Blue
RNN		Recurrent Neural Network
ROC		Receiver Operating Characteristic
SQL		Structured Query Language
TP		True Positive
VGG		Visual Geometry Group
WHO		World Health Organization

1. INTRODUCTION

Diabetic retinopathy (DR) is the main cause of blindness. It's estimated that more than 93 million people are affected by this complication. The World Health Organization (WHO) estimates that 422 million people have diabetes around the world in 2014 [1]. In USA 40% to 45% of people with diabetes have DR that is classified according to the severity stage, it can be moderate, severe, and Proliferative.

Progression to vision disability can be slow and high cost or averted if DR is detected on time. Currently, the detection of this complication involves a manual process and it requires a long time and an expert physician to examine and evaluate digital color photographs of the fundus of the retina. [2]

That is why alternative methods are necessary, to be able to provide solutions that can be faster and more effective.

Artificial intelligence (AI), Machine Learning (ML), and Deep Learning (DL) have been some of the tools used recently in technology development for healthcare. ML is used in many approaches for the detection and classification of DR severity. DL is a method that can improve object detection performance and visual recognition. DL uses multiple layers with nonlinear processing units and it is used for object detection of features in images. [3]

Convolutional Neural Network (CNN) is a DL model that contains convolutional, pooling, and fully connected layers. Convolutional is the layer that extracts the features from the image, then the result of this goes through the pooling layer that calculates the number of feature extraction, and finally, the fully connected layer is classified as a deep learning algorithm. CNN has been used for image classification and it is useful for applications in medical images. [3]

In this study, I propose a CNN method to classify DR using a dataset available in Kaggle.

1.1. Background

A significant cause of blindness in the working-age population is the DR. Some people present late in the course of this disease and the treatments are more difficult. If this is caught early, laser treatment and vascular endothelial growth factor inhibitors decrease the risk of visual loss. Vitrectomy operations with modern surgical techniques provide improving results for patients with advanced retinopathy, however, it is better to detect this disease on time [4].

In most of the studies, the segmentation of blood vessels in the base for retinal image analysis. There is a wide variety of approaches to get this information from the image such as the grayscale decomposition instead of using the green channel proposed by Coye [5] and the image segmentation techniques applied by [6].

However, the analysis made by the approaches described is typically binary classification problems, where the classes were only a healthy eye or the detection of disease. Furthermore, the classification is made based on more than one disease type (retinopathy and glaucoma) due to the lack of data available in the medical area.

Edge detection has been proposed in several ways in the classic image analysis algorithms [6] and the techniques have been introduced to deep learning algorithms to improve the performance of the models. Deeper analysis has been performed in the image generation itself working with additional filters to reduce the noise and improve the quality of the retinal image from the source itself, typically the camera lens. This addition of this technique might improve the performance of the deep learning model, but this analysis is out of the scope of this document.

1.2. Justification

By the time you finish reading this project, 3,400 people in the world, with low resources, will have died because of some undiagnosed disease. I will focus on the investigation, application, and verification of algorithms that allow early detection of one of the main effects of diabetes.

The idea stems from looking for a way to improve and make faster the early detection of DR and it can be treated before the damage can be higher. As part of the process, I seek to study recent academic documents, through the review of the state of the art.

This allows a clear picture of the techniques used in image recognition and how it is applied. Then I will evaluate the current approaches, through the model implementation, using reference frameworks such as Tensor Flow. Finally, a dataset through supervised training, which will predict with high accuracy the category of affectation eyes by diabetes.

1.3. Problem

By the time human readers submit their reviews, often a day or more, delayed results may mean loss of follow-up and treatment will be delayed as well. The expertise and equipment required are often in areas where the diabetes rate is high and resources are sometimes insufficient.

Some investigations have been done to detect this affectation in an early way. Most of them are based on CNN with high performance.

It is important to have some tools that help ophthalmologists to detect DR grade more easily and cheaply so that complications can be detected on time

1.4. Hyphotesis

CNN can be used for image classification through DL models. In this research, it will be necessary to find a good predictive model to classify DR affection. In a predictive model, data cleaning favors retina image processing for the detection of eyes damage, for that reason it will be necessary to apply some strategies to improve the dataset that will be used.

1.5. Objectives

1.5.1. General Objective:

Obtain predictive models, contrasting the results using a model based on AlexNet and an experimental model, for the detection of eyes damage through retina images. This model can be used by ophthalmologists or common people can have access to the developed DR predictive model at a lower cost.

1.5.2. Specific Objectives:

This research looks to satisfy the next specific objectives:

- To define a strategy to manage the dataset. It will be necessary to use techniques of the unbalanced dataset.
- Make a qualitative analysis of the effectiveness of the model.
- Understand the causes for which the model can fail (if there is any wrong prediction).
- Categorize based on the characteristics of the different affectations due to DR.

1.6. Scientific or technologic share/innovation

This document has the intention to provide the results of the overall behavior of a model trained by a DR dataset that will be able to classify the DR affectation grades.

2. STATE OF THE ART OR THE TECHNIQUE

DR is a disease caused by diabetes and it affects the retina. It is the most common cause of blindness. Patients and the responsible for diabetes in each case, play an important role to avoid permanent blindness. So, early detection can prevent complications.

The current research used for the detection and classification of DR is based on CNNs. Some of them used different algorithms techniques to obtain better results than individual classifiers. For example, the ensemble model extracts the best features from each classifier. [7]

Some architectures as GoogleNet [3] and Visual Geometry Group (VGG) like VGG-16 [8] have been used to classify images getting high accuracies, up to 97% and 74% respectively.

Other techniques are also combined before introducing a classifier model, for example, an attention mechanism that focuses on pathological areas or segmentation meth comprised. [9]

Also, others compare the results of their systems versus the typical classification of ophthalmologists to know the efficacy of the systems built. [10] [11]

2.1. Ensemble classification technique

Health records sensitivity is one of the causes of the reduced accuracy of disease detection.

In the article named "An Ensemble Classification Techniques Based on ML Model for Automatic DR Detection" [7], it is proposed an ensemble based on machine learning model using algorithms as DT classifier, RFC, K-Nearest Neighbor, Ad boost classifier, J48graft classifier, and LR.

The individual algorithms did not give satisfactory results, so an ensemble approach with existing machine learning algorithms was proposed.

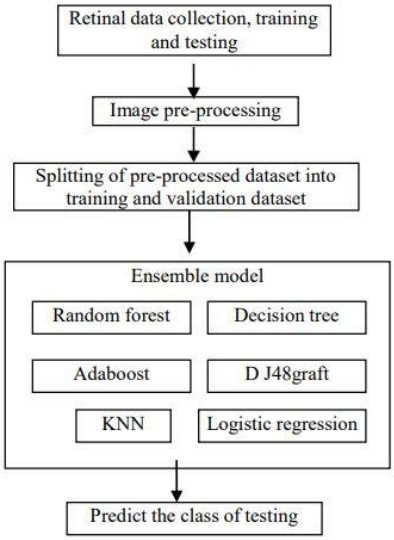


Fig 1 Ensemble classification diagram

This proposed work of DR uses Kaggle's collection of image data sets with 3,624 images found.

Table 1 Data set distribution

Level	Type of DR	No. of Records
0	NO DR	1,789
1	Mild	410
2	Moderate	950
3	Sever	170
4	Proliferative	305

The advantages of each model are combined into the ensemble model that gives better results than individual classifiers. The ensemble model has the objective of extracting the best features from the classifiers.

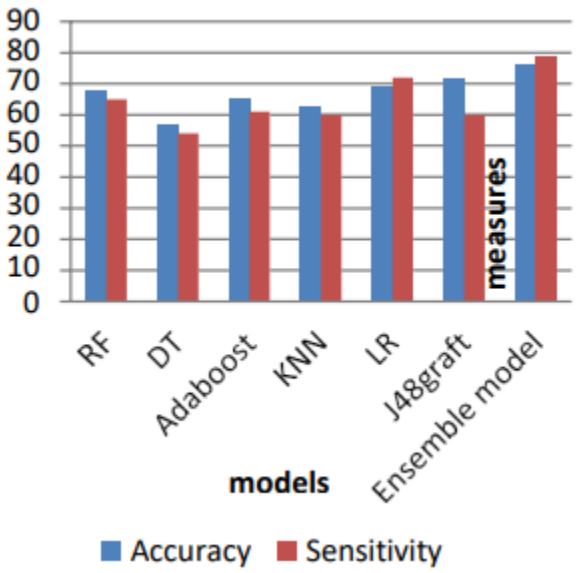


Fig 2 Accuracy and Sensitivity

2.2. DR Detection using GoogleNet and Attention Mechanism

Patients with Diabetes can suffer blindness caused by DR. Big Data has challenges concerning image processing in the medical area. In this research, Attention Mechanism and GoogleNet algorithms are used to detect and classify DR according to classifications such as normal, mild, moderate, severe, and proliferative [3]. The attention mechanism has the function of focusing on the pathological area in the fundus images and GoogleNet focuses on the classification of the fundus images at the DR levels.

The accuracy gotten was up to 97% with 250 images for training data. These datasets were obtained from Kaggle.

2.3. EAD-Net as Segmentation Method in DR

DR is a common disease of the fundus and it has four types of microvascular lesions: hard exudates, soft exudates, microaneurysms, and hemorrhages. the accurate detection and the count of them are significant work.

Hard work in the clinical analysis is the manual annotation of the lesions in the fundus. In this research, it is proposed a solution that uses the segmentation method for the different kinds of DR lesions. It is based on a CNN and it is divided into encoder, attention, and decoder modules. These 3 parts are named EAD Net.

The fundus images were passed to EAD Net but before that, a normalization and augmentation were used. EAD Net extracts the features and pixel-wise label prediction. The method achieved a specificity of 99.98%, a sensitivity of 92.77%, and an accuracy of 99.97% using the e_ophtha_EX dataset and comparable AUC Recall Curve scores on the IDRiD dataset. [9]

2.4. Development of mobile eye care services in integrated home-based medical care

In Taiwan, many older adults suffer from vision-impairing eye conditions. In this research, mobile eye care services were constructed in integrated home-based medical care and an efficacy evaluation of portable fundus camera is done to evaluate DR at home.

A new portable handheld nonmydriatic fundus camera using a telemedicine system was built. The quality of the images of fundus cameras used by ophthalmologists and non-ophthalmologists was compared by taking photographs of 112 eyes from 56 diabetic people. All the fundus images were graded with the three classifications.

Results were using portable fundus camera by non-ophthalmologists a 65.2% of quality, 29.9% were fair and 4.9% inadequate. On the other hand, using portable fundus cameras by ophthalmologists, 71% of quality was obtained, 25.4% were fair and 4.6% were inadequate. DR was easily and accurately identified even at home without a visit of an ophthalmologist. [10]

2.5. Automated detection of Diabetic Retinopathy using VGG-16 architecture

DR is caused by chronic diabetes and it is a disease that affects the retina. It is important to have early detection of this disease to have a significant benefit.

This research used the Asia-Pacific Tele-Ophthalmology Society 2019 Blindness Detection dataset for training, and it contains 3,668 retinal images. Fundus photography was used for these pictures. It was used a pre-trained VGG-16 model to detect the DR severity. This model was tested with a new dataset of over 1728 images that were not included in the dataset of training. The model process flow is shown in Fig 3.

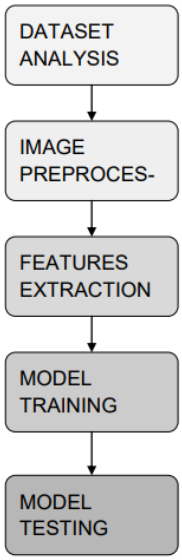


Fig 3 Process flow

This model was able to achieve an accuracy of 74.58% and a loss of 5.06% with 30 epochs and using Categorical Cross Entropy as loss function and ADAM optimizer. [8]

2.6. Screening DR

Automated screening represents an opportunity for patients at risk of developing DR to improve their mid-term and have a lower cost of common vision-threatening complications caused by diabetes.

The target of this study was to develop an automated DL system to classify images of the fundus of the retina of DR cases, of international and Mexican people. Evaluated the performance of automated image analysis (ARIA) under a scheme that is independent and two assistive schemes (i.e., hybrid ARIA and ophthalmologist screening) using a web-based platform for remote image analysis to compare sensitivity and specificity of the three schemes.

The objective of this study was to develop an automated DL system to classify retinal fundus images of DR cases, from international and Mexican people. It evaluated the performance of automated image analysis (ARIA) under an independent scheme and two assistive schemes (i.e., hybrid ARIA and ophthalmologist screening) using a web-based platform based for remote analysis of images to compare the sensibility and specificity of the three schemes.

A randomized experiment was carried out where 17 ophthalmologists had to classify a series of fundus images of the retina in 3 different conditions. Condition 1: screen the images by themselves. Condition 2: screen the images after the exposure to the retina imagen classification of the ARIA system. Condition 3: like condition 2 but now also its confidence level and attention map remarking the important areas of interest in the image.

The results from the ARIA system and ophthalmologists were compared and it was concluded that the ARIA system was able to classify with an area under the Receiver Operating Characteristic (ROC) curve of 98%, 95.1% of sensitivity, and 91.5% of sensitivity for international patient cases. For Mexican patients, it was obtained a 98.3% of the area under the ROC curve, with a sensitivity of 95.2% and 90% of specificity. ARIA system showed a better performance than the average performance of the 17 ophthalmologists that participated in the study. [11]

3. THEORIC/CONCEPTUAL FRAMEWORK

3.1. Artificial Intelligence

Artificial Intelligence is the engineering and science of making machines with intelligence, in most cases, it is related to intelligent computer programs. The intention is to use computing to understand human intelligence.

There is not yet a solid definition of intelligence that is not related to human intelligence. We cannot characterize what is intelligent in a computational procedure. Some of the mechanisms of intelligence we can understand but there are others that we cannot.

Intelligence involves different mechanisms and research of AI has discovered how to carry out some of them but not others.

Sometimes AI tries to simulate human intelligence but not always. We can learn something and make machines that can solve problems just by observing our methods or from other people. On the other hand, a big part of the work in AI is related to the studying of problems that the world presents to intelligence rather than studying animals or people. In other words, AI is free to use methods not observed in people. [12]

3.1.1. Machine Learning

Machine Learning is defined as computational methods that improve performance using experience or can make accurate predictions. Experience is related to the past information that is available to the learner. This information can be presented in labeled training datasets typically. The quality and size of the data are crucial to obtaining good predictions.

ML consists of choosing efficient and accurate prediction algorithms. Theoretical learning guarantees are based on the complexity of the concept classes and the size used for the training data. Learning techniques are data-based methods combined with concepts in computer science as statistics, probability, and optimization.

Some the applications of ML include the following:

- Text or document classification

- Natural language processing (NLP)
- Speech processing applications
- Computer vision applications
- Computational biology applications

Some of the standards learning tasks are the next:

- Classification: assignation of a class to each item
- Regression: prediction of a true value for each item
- Ranking: learn to sort items according to some criteria
- Clustering: this is often used to analyze large datasets. It is related to partitioning a set of items into homogenous subsets
- Reduction of dimensionality or multiple learning: the transformation of an initial representation into a representation of less dimension but preserving the properties

There are common terminology and definitions used in ML:

- Examples: it corresponds to the collection of data used for learning and evaluation
- Features: it is the set of attributes that can be represented by a vector
- Labels: categories assigned to the examples. It is used in classification methods
- Hyperparameters: free parameters and they are not determined by the algorithm, but rather specified as inputs
- Training sample: examples used by the learning algorithm to train
- Validation sample: it is the subset of examples to select appropriate parameters of a learning algorithm
- Test sample: it is the subset of examples to evaluate the performance of the learning algorithm
- Loss function: a function that calculates the variance between a predicted label and the true label

Scenarios differ in the types of training available to the learner and the training method used. The most common scenarios can be supervised and unsupervised learning. In supervised learning, there is a set of labeled examples as training data, and it makes predictions for invisible points. Unsupervised learning uses unlabeled training data and predictions are performed for invisible points. [13]

3.1.2. Deep Learning

Deep learning is a subset of ML, which is a NN with 3 or more layers. This kind of NN attempt to simulate the behavior of the human brain allowing it to learn from data in large amounts through a combination of data inputs, weights, and bias.

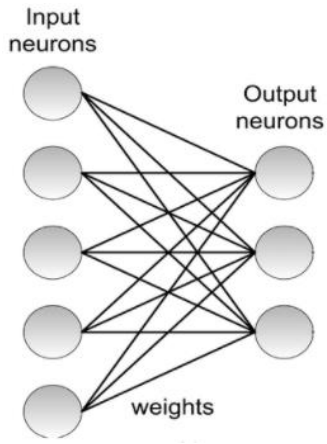


Fig 4 Model with single layer

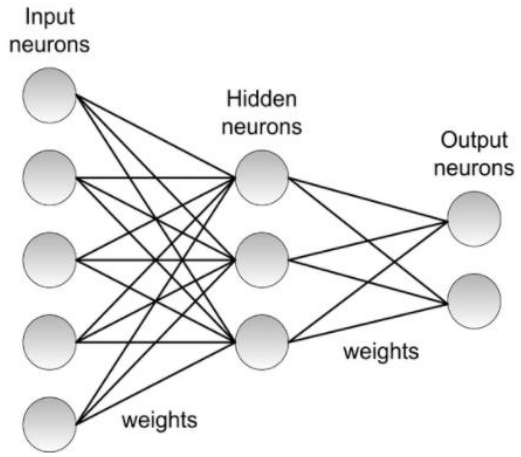


Fig 5 Model with multilayer

DL is used in many artificial intelligence applications that improve automation, performing physical and analytical tasks without intervention on humans.

These neural networks consist of multilayers of interconnected nodes, refinement, and optimization of the prediction are done on each of the buildings upon the previous layer. Forward propagation is the progression of computations through the neural network. The input and output layers are the visible layers in the NN. The deep learning model ingests the data through the input layer to be processed and the final prediction or classification is done by the output layer.

Gradient descent is an algorithm used by backpropagation, it is used to calculate errors in predictions and adjust the weights and biases by moving backward through the layers to train the model. Forward and backpropagation works together in a NN to make predictions and correct errors, so the algorithm becomes more accurate over time. [14]

There are different types of NN to direct specific problems or datasets. Some examples of that are:

- CNN: They are used primarily in computer vision and classification of images; they can detect patterns and features like object detection
- RNN: They are used in natural language and speech recognition

3.2. Convolutional Neural Network

Convolutional networks have an important role in the deep learning history. They were some of the first deep models with a good performance. These nets were also some of the first deep learning networks trained with backpropagation. [15]

CNN are distinguished because they have superior performance with image, speech, and audio signal inputs than other neural networks. They have 3 main types of layers: CL, PL, and FC layer.

CL is the first layer of a CNN. They can be followed by additional convolutional layers or pooling layers. The last layer is the FC. At each layer, the CNN increases in complexity. The first layers focus on simple features like colors and borders. As the image data progresses through CNN, it begins to recognize large shapes and elements and eventually identifies the desired object.

3.2.1. Convolutional layer

This layer is the core building of CNN and is the part where most of the computation occurs. It requires some components as input data, filter, and feature map. For example, we can assume that the input is a color image which is a matrix of pixels in 3D, so the input has 3 dimensions (weight, width, and depth), which means a Red-Green-Blue (RGB) image. Also, we have a kernel or filter that will be a feature detector, it will move across the receptive fields of the image checking for the feature. It is known as convolution. [16]

The feature detector is a 2D matrix of weights and depicts part of the image and can vary in size. The filter size is commonly a 3x3 matrix and defines the size of the receptive field. This filter is put onto an area of the image and then a dot product is performed between the input pixels and the filter. And the product is fed to an outlet array. The filter is repeated shifting by a stride across the entire image. The result of the final output from the series is known as a convolved feature, activation feature, or feature map.

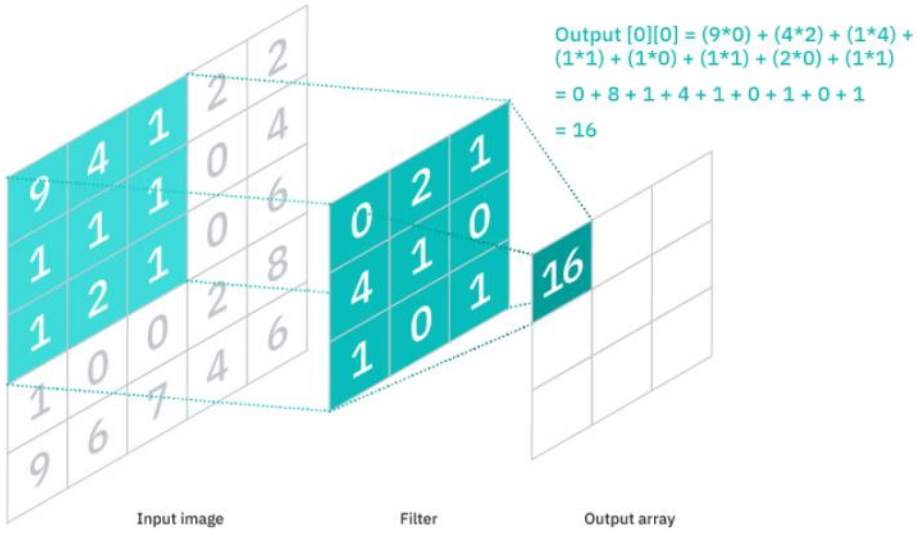


Fig 6 Example of convolution

As it can be observed in Fig 6 each output value in the feature map is not connected to each pixel value in the input image, it just needs to connect to the receptive field. Convolutional and pooling layers are referred to as partially connected layers because the output arrays are not mapped directly to each input value.

The weights remain fixed in the feature detector as it moves through the image. The volume of the output is affected by 3 hyperparameters:

1) Depth of the output is affected by the number of filters. If 3 different filters are applied, it will create 3 feature maps using a depth for each one.
2) Stride is the pixels number or the distance that the kernel shifts over the input matrix. It is rare for a stride of 2 or more, however, a larger stride will produce smaller output.
3) Zero padding is occupied when the input image is not fit by the filters. All the elements that are outside of the matrix will be zero, producing a larger or equal output size. There are 3 types of padding:
    - Valid padding: it is a no padding, if dimensions do not align the last convolution is dropped.
    - Same padding: the output layer will have the same size as the input layer.
    - Full padding: the output size is increased by aggregate zeros to the edge of the input.

3.2.2. Pooling layer

It is also known as downsampling. this layer reduces the number of parameters in the input. It is a kind of similar to CL due to PL uses a filter across the entire input but this filter does not have weights.

Instead, an aggregation function is applied by the kernel to the values within the receptive field, population the array of the output. Two principal types of pooling are used:

Max pooling: it selects the pixel with the maximum value to the output array as the filter moves across the input. Average pooling: like max-pooling but it calculates the average within the receptive field. Too much information is lost in the PL; however, it improves efficiency, reduces complexity, and limits the risk of overfitting.

3.2.3. Fully connected layer

As it was mentioned, the pixel values of the input image are not connected to the output layer directly in partially connected layers. In a fully connected layer, each node of the output is connected directly to a node in the previous layer.

FC performs classification based on the extracted features through the previous layers and the filters applied. CLs and PLs tend to use Rectified Linear Activation Unit (ReLU) functions and fully connected can use SoftMax to classify inputs.

3.3. Evaluation metrics

Sometimes the model can have satisfying results when it is evaluated using an accuracy score but also it can give poor results versus metrics as logarithmic loss or any other. Classification accuracy is used to measure the performance of the model most of the time but it is not sufficient. There are some other evaluation metrics. [17]

3.3.1. Classification accuracy

It is the relation between the number of correct predictions and the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made} \quad (1)$$

It works fine when the dataset is balanced on each class. If the dataset is not balanced, classification accuracy can be great, but it gives a false sense of high accuracy.

3.3.2. Logarithmic Loss

It works well in multiclass classification. It penalizes the false classifications. Log loss assign probability to each class for all samples. This metric is calculated as:

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log\,(p_{ij}) \quad (2)$$

Were:

- N represents the number of samples
- M represents the classes number
- y_{ij} tells if sample i belongs to class j
- p_{ij} is the probability that sample i belongs to class j

Higher accuracy is indicated when Log loss is nearer to 0

3.3.3. Confusion Matrix

It gives an output matrix that describes the performance of the model.

For example, if we assume that we have a binary classification problem, the classes are YES and NO and the model predicts 165 samples with the following results:

Table 2 Confusion Matrix

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

- True positives: number of values where the prediction is YES and the current output is YES
- True negatives: number of values where the prediction is NO and the current output is NO
- False positives: number of values where the prediction is YES and the current output is NO
- False negatives: number of values where the prediction is NO and the current output is YES

The accuracy can be calculated as:

$$Accuracy = \frac{TruePositive+TrueNegative}{TotalSample} \quad \textbf{(3)}$$

$$\therefore Accuracy = \frac{150 + 50}{165} = 0.91$$

3.3.4. Area Under Curve

It is used for classification and it is one of the most used metrics for evaluation. Area Under Curve (AUC) is the probability that the model ranks a random positive example higher than a random negative example.

Sensitivity or recall is related to True Positive Rate, it is the relation of the proportion of positive data points that are correctly considered positive concerning the all-positive data points.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative+TruePositive} \quad (4)$$

Specificity is related to True Negative Rate; is the ratio of negative data points that are correctly considered negative to totally negative data points.

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative+FalsePositive} \quad (5)$$

False Negative Rate is the ratio of negative data points that are not correctly considered positive to totally negative data points.

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative+FalsePositive} \quad (6)$$

FP Rate and TP Rate have values in the range [0,1] Both are varying thresholds. AUC is the area under the curve of the plot. AUC has a range of [0,1], a greater value means a better performance of the model.

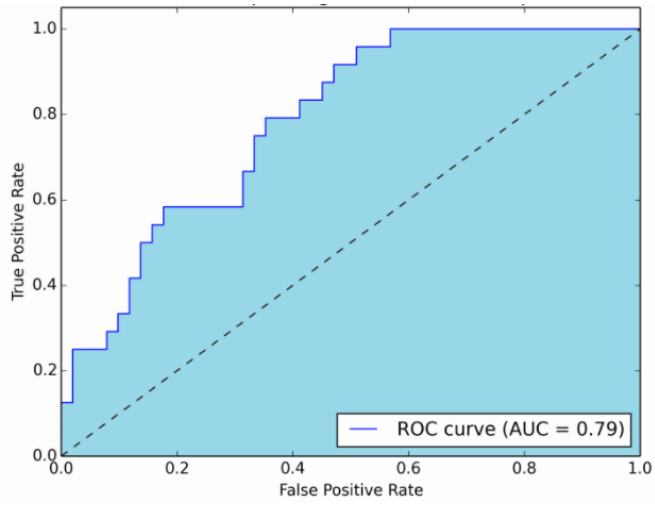


Fig 7 Receiver operating characteristic

3.3.5. F1 Score

It is the harmonic mean between precision and recall. The range of this metric is [0,1]. It says how precise is the classifier, the number of instances that are classified correctly, and also it determines how robust it is, a significant number of instances is not missed.

F1 Score tries to find the balance between precision and recall. It can be represented as:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \quad (7)$$

Were:

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

$$Recall = \frac{TP}{TP+FN} \quad (9)$$

3.3.6. Mean Absolute Error

It is the average of the difference between original values and prediction values. But it does not give information about the error direction, for example under predicting or over predicting data.

$$MeanAbsoluteError = \frac{1}{N}\sum_{j=1}^{N} |y_j - \hat{y}_j| \quad (10)$$

3.3.7. Mean Squared Error

The Mean Squared Error (MSE) is similar than Mean Absolute Error (MAE) takes the average of square of the difference between original values and the predicted values.

$$MeanSquadError = \frac{1}{N}\sum_{j=1}^{N} (y_j - \hat{y}_j)^2 \quad (11)$$

3.4. Libraries

3.4.1. NumPy

NumPy is a package used mainly for Python programmers. This library provides multidimensional array objects, masked arrays, and matrices, and a variety of routines to be used on routines like mathematical operations, logical operations, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, linear algebra, statical operations, and random simulation. [18]

Ndarray is an object at the core of the NumPy package. N-dimensional arrays of homogenous data can be encapsulated by ndarray object and some operations can be performed. The main differences between standard Python sequences and NumPy arrays are:

- The size of the NumPy array is fixed since the creation, unlike Python list can grow dynamically
- NumPy must contain elements with the same data type
- Operations with NumPy are executed more efficiently than Python sequences

3.4.2. NDarray Attributes

The attributes of an array reflect the intrinsic information of the array. It is possible to access to get and sometimes set intrinsic attributes without creating a new array [19]. Some of the attributes are the next:

Table 3 NDarray attributes [19]

ATTRIBUTE	DESCRIPTION
ndarray.flags	Information about the memory layout of the array.
ndarray.shape	Tuple of array dimensions.
ndarray.strides	Tuple of bytes to step in each dimension when traversing an array.
ndarray.ndim	Number of array dimensions.
ndarray.data	Python buffer object pointing to the start of the array's data.
ndarray.size	Number of elements in the array.
ndarray.itemsize	Length of one array element in bytes.
ndarray.nbytes	Total bytes consumed by the elements of the array.
ndarray.base	Base object if memory is from some other object.
ndarray.dtype	Data-type of the array's elements.
ndarray.T	The transposed array.
ndarray.real	The real part of the array.
ndarray.imag	The imaginary part of the array.
ndarray.flat	A 1-D iterator over the array.
ndarray.ctypes	An object to simplify the interaction of the array with the ctypes module.
array_interface__	Python-side of the array interface
array_struct__	C-side of the array interface
ndarray.ctypes	An object to simplify the interaction of the array with the ctypes module.

3.4.3. Pandas

Pandas is a high-level block to perform data analysis in Python [20]. Some of the characteristics are:

- Data frame object has a fast and efficient manipulation
- It is possible to hand varied formats like Comma Separated Value (CSV), text files, Microsoft Excel, databases of Structured Query Language (SQL), and Hierarchical Data Format version 5 (HDF5) format
- Reshaping is flexible
- Intelligent slicing based on labels
- Columns can be inserted or deleted from data structures
- A group by engine allows to split, apply, and combine operations in a data set
- Merging and joining are done with a high performance
- The hierarchical axis index provides an intuitive way to work with high-dimensional data in a lower-dimensional structure
- Functionality with time series
- Optimized for a high performance
- Academic and commercial domains make use of Python with pandas

3.4.4. Keras

Keras is an Application Programming Interface (API) and one of the leading high-level neural networks. Keras is developed in Python and supports multiple computation engines for the back end.

Standalone modules can be used to create a new model. These modules are neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization. Keras support multiple back-end engines but his main back end is TensorFlow.

Layers and models are the core structure of Keras. The sequential model is the simplest model type [21]. Below is an example of a sequential model:

```
from tensorflow.keras.models import Sequential

model = Sequential()
```

Fig 8 Sequential model example

It is easy to add layers

```
from tensorflow.keras.layers import Dense

model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

Fig 9 Layers example

compile() is used to configure the learning process

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

Fig 10 Compile example

It is possible to perform more configurations to the optimizer

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True))
```

Fig 11 Optimizer example

Batches can be used to iterate in the training data

```
# x_train and y_train are Numpy arrays
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Fig 12 Batches example

Evaluation of test loss and metrics

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

Fig 13 Loss and metrics example

Generation of predictions using new data

```
classes = model.predict(x_test, batch_size=128)
```

Fig 14 Predictions example

3.4.5. TensorFlow

TensorFlow is a library of open-source developed by Google Brain Team. This can run in Graphics Processing Unit (GPU) and Central Processing Unit (CPU). The distributed engine execution abstracts many devices supported and gives a high-performance core implemented in C++ for the platform of TensorFlow.

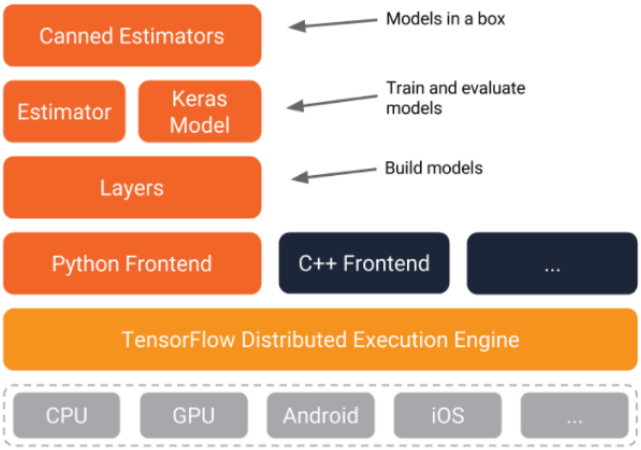


Fig 15 TensorFlow Engine

As it can be observed in Fig 8, Python and C++ are on the top of the frontends. API layers give a simpler interface for layers commonly used in DL models. At a high level, we can find APIs like Keras and Estimator, which makes the training and evaluating distributed models easier. [22]

4. DEVELOPMENT AND METODOLOGY

The first step towards data collection was to research scientific articles, better known as the state of the art. These articles aim to identify patterns in the human iris to detect diseases. A dataset of retinal images was obtained from Kaggle, which includes large sets of high-resolution images labeled with either right or left eye.

The next step was to separate images according to the grade of damage. Then a balance of the dataset is needed to have a similar size on each of the classes and do not have overfitting in the CNN. An experimental CNN architecture was used to train a model to classify the grade of damage of DR.

4.1. Dataset by EyePACS

The dataset contains images of the retina that were provided by EyePACS (a free platform for the detection of retinopathy). It includes a big dataset of high-resolution retina images. These pictures were captured in different conditions and can be downloaded from Kaggle by navigating to the following path:

https://www.kaggle.com/c/diabetic-retinopathy-detection/data

The Left and right fields are provided for each subject. Images are tagged with a subject ID as well as left or right (e.g., 1_left.jpeg is the left eye of patient id 1).

DR has been rated on the images on a scale of 0 to by a clinician. Table 4 shows the scales:

Table 4 DR damage

GRADE	FEATURE
R0	No disease
R1	**Mild background DR** It includes microaneurysms, flame exudates, >4 blot hemorrhages in one or both hemifields, and/or cotton wool spots
R2	**Moderate background DR** >4 blot hemorrhages in one hemifield
R3	**Severe non-proliferative or pre-proliferative DR:** >4 blot hemorrhages in both hemifields, IRMA, venous beading
R4	**Proliferative retinopathy** Neovascularization elsewhere, neovascularization of the disc, vitreous hemorrhage, retinal detachment

Next images in Fig 16, Fig 17, Fig 18, and Fig 19 shows the different grades and we can appreciate that as long as the degree is higher, the hemorrhages are more noticeable

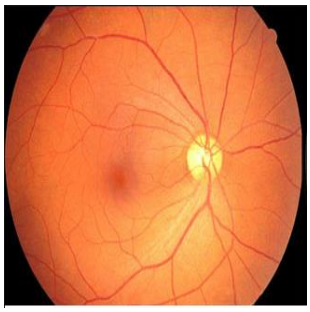


Fig 17 Grade R0

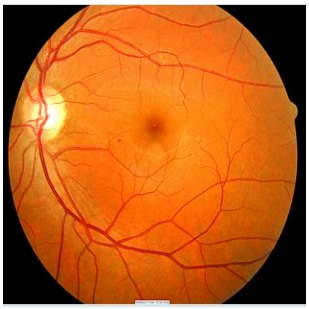


Fig 16 Grade R1

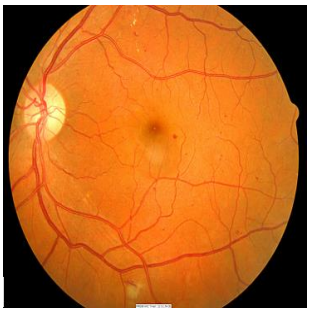


Fig 18 Grade R2

Fig 19 Grade R3

Fig 20 Grade R4

Due to the extremely large size of this dataset, they have separated in files into multi-part archives:

- train.zip.* - it contains 5 files and it is the training set
- test.zip.* - it contains 7 files and it is the test set
- sample.zip – it is a preview of the full data
- sampleSubmission.csv – it is a file with samples with the correct format
- trainLabels.csv – it contains the training set scores

4.1. Data preparation

The trainLabels.csv file contains two columns, the first for the image ID and the second for the scale. The value of each row in the first column called image concatenates the patient ID as well as either left or right. This is the unique prefix associated with the filename.

The first step is to import the required libraries. One of the main is Pandas because it contains all relations between the image files and the metadata. Other libraries like TensorFlow and Keras are for ML and DL proposes.

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import datetime

from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import MaxPool2D

from datetime import date
from sklearn.model_selection import train_test_split
from pathlib import Path
from shutil import copyfile
```

Fig 21 Import libraries

Read the trainLabels.csv dataset into a data frame. As you can observe below, there is a list with the first five elements where we describe if the image belongs to the left or right eye and the severity of damage on it.

```
# Read CSV from WHO dataset
# https://www.kaggle.com/c/diabetic-retinopathy-detection/data

data_dir = os.path.join('data')
severity = [
    "No DR",
    "Mild",
    "Moderate",
    "Severe",
    "Proliferative DR"
]
retina_df = pd.read_csv('data/trainLabels.csv')
retina_df.head()
```

	image	level
0	10_left	0
1	10_right	0
2	13_left	0
3	13_right	0
4	15_left	1

Fig 22 Data frame of the document

Append a new column into the data frame called “patiend_id”. Append a new column into the data frame to store the path to the image. Append a new column into the data frame that contains if an image exists. Append a new column into the data frame that contains the eye side. Append a new column into the data frame that contains the category level and convert it to a binary class matrix using the to_categorical method from Keras.utils object.

A resultant data frame is observed below. It contains the information of the number of the patient, if the image belongs to the left or right eye, the path of the image, if the image exists, a '0 'or '1' value if the eye is left or right and a subfolder category where the image will be deposited.

```
# Generate required columns

image_dir = os.path.join(data_dir, 'train')
retina_df['patient'] = retina_df['image'].map(lambda x: x.split('_')[0])
retina_df['path'] = retina_df['image'].map(lambda x: os.path.join(image_dir, '{}.jpeg'.format(x)))
retina_df['exists'] = retina_df['path'].map(os.path.exists)
retina_df['eye'] = retina_df['image'].map(lambda x: 1 if x.split('_')[-1] == 'left' else 0)
retina_df['subfolder'] = retina_df['level'].map(lambda x: severity[x])
retina_df.head()
```

	image	level	patient	path	exists	eye	subfolder
0	10_left	0	10	data\train\10_left.jpeg	True	1	No DR
1	10_right	0	10	data\train\10_right.jpeg	True	0	No DR
2	13_left	0	13	data\train\13_left.jpeg	True	1	No DR
3	13_right	0	13	data\train\13_right.jpeg	True	0	No DR
4	15_left	1	15	data\train\15_left.jpeg	True	1	Mild

Fig 23 Data frame adjustment

It is necessary to remove missing values using the "dropna" method provided by the Pandas library from the dataset object and to analyze the distribution on each eye according to their severity. As it can be observed in Fig 24, we have 17,500 for the left eye and 17,500 for the right eye but the severity on each one is not balanced.

```
# Remove empty

retina_df.dropna(inplace = True)
retina_df = retina_df[retina_df['exists']]
retina_df[['level', 'eye']].hist(figsize = (10, 5))
print(retina_df.shape)
```

```
(35126, 7)
```

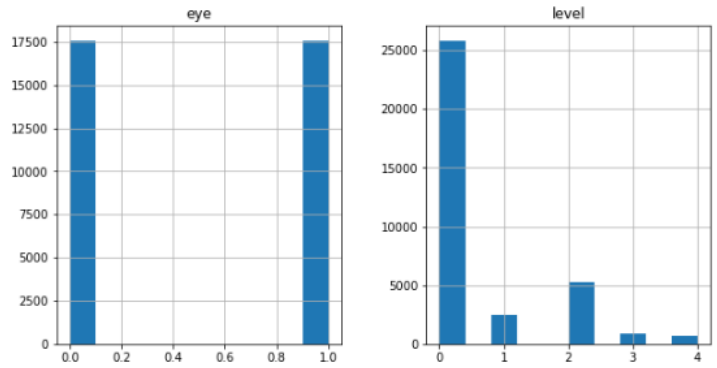


Fig 24 Remove missing values

Ensure to remove duplicates using the "drop_duplicates" method and then split data in training and validation. It results in a dataset of 32,018 images for training and 3,916 for tests.

```
# Split Data

retina_dup_df = retina_df[['patient', 'level']].drop_duplicates()
train_ids, valid_ids = train_test_split(
    retina_dup_df['patient'],
    test_size = 0.10,
    stratify = retina_dup_df['level']
)
train_df = retina_df[retina_df['patient'].isin(train_ids)]
valid_df = retina_df[retina_df['patient'].isin(valid_ids)]
print('train', train_df.shape[0], 'validation', valid_df.shape[0])
```

```
train 32018 validation 3916
```

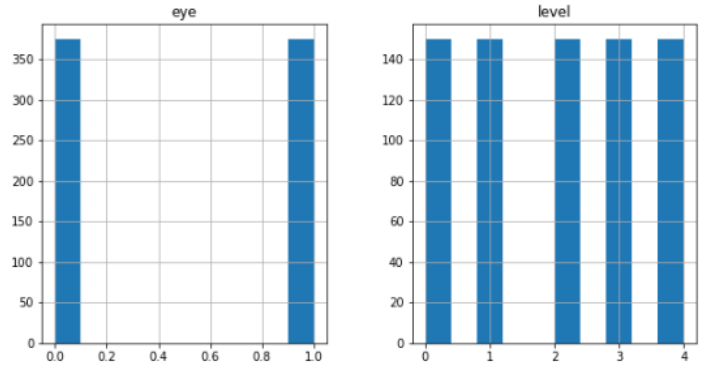
Fig 25 Remove duplicates

Balance training and validation datasets. Due to the original dataset being very big, I decided to reduce it and make a balance of all classes for training and validation. Each class contains 150 images, a total of 750 images for training and 750 for validation. Balance can be observed in Fig 26 and Fig 27.

```
# Balance the Distribution in training data

train_df = train_df.groupby(['level', 'eye']).apply(lambda x: x.sample(75, replace = True)).reset_index(drop = True)
train_df[['level', 'eye']].hist(figsize = (10, 5))
print('Balanced training size:', train_df.shape[0])
```

```
Balanced training size: 750
```



,

Fig 26 Training data balanced

```
# Balance the Distribution in validation data

valid_df = valid_df.groupby(['level', 'eye']).apply(lambda x: x.sample(75, replace = True)).reset_index(drop = True)
valid_df[['level', 'eye']].hist(figsize = (10, 5))
print('Balanced validation size:', valid_df.shape[0])
```

```
Balanced validation size: 750
```

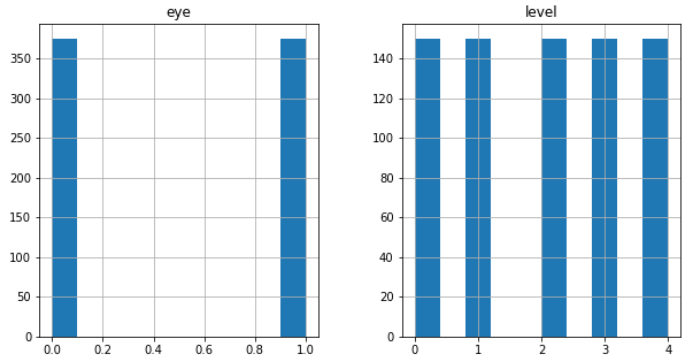


Fig 27 Validation data balanced

4.2. AlexNet architecture

AlexNet is a CNN architecture that consists of 5 CLs and 3 more FC layers. The first 2 CLs are followed by the overlapping max-pooling layers. Third, fourth and fifth CLs are connected directly with each other. An overlapping max-pooling layer follows the fifth convolutional layer, the output of which goes into a series of 2 fully CLs and each one contains 4096 neurons. The second FC layer feeds into a softmax classifier that can have 1000 class labels [23]. Fig 28 shows the AlexNet architecture.

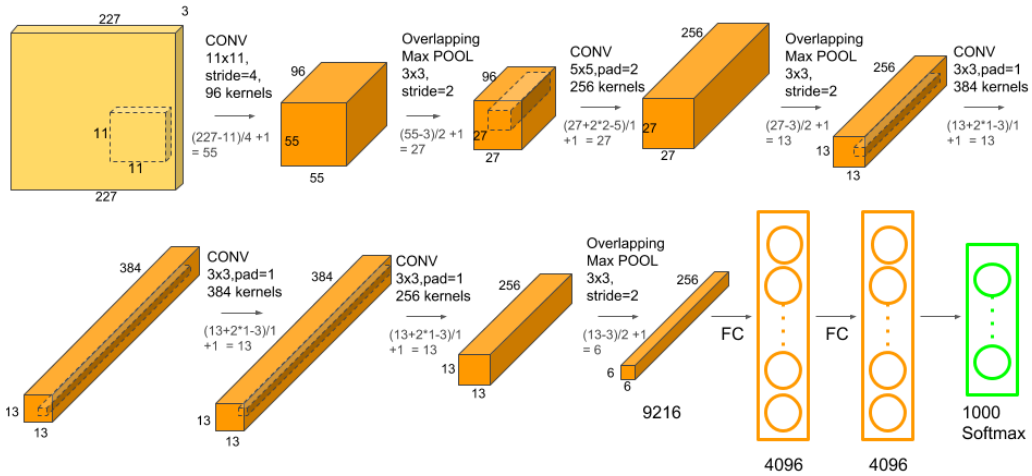


Fig 28 AlexNet Architecture

Due to AlexNet is a basic architecture, it does not require high computing power. For that reason, an architecture based on AlexNet was implemented as part of this work. The output layer was modified to 5

labels, the same number of classes that the dataset contains. This architecture based on AlexNet is shown in Fig 29.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 96)	34944
activation (Activation)	(None, 30, 30, 96)	0
batch_normalization (BatchNo	(None, 30, 30, 96)	384
max_pooling2d (MaxPooling2D)	(None, 14, 14, 96)	0
conv2d_1 (Conv2D)	(None, 14, 14, 256)	614656
activation_1 (Activation)	(None, 14, 14, 256)	0
batch_normalization_1 (Batch	(None, 14, 14, 256)	1024
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 256)	0
conv2d_2 (Conv2D)	(None, 6, 6, 384)	885120
activation_2 (Activation)	(None, 6, 6, 384)	0
batch_normalization_2 (Batch	(None, 6, 6, 384)	1536
conv2d_3 (Conv2D)	(None, 6, 6, 384)	1327488
activation_3 (Activation)	(None, 6, 6, 384)	0
batch_normalization_3 (Batch	(None, 6, 6, 384)	1536
conv2d_4 (Conv2D)	(None, 6, 6, 256)	884992
activation_4 (Activation)	(None, 6, 6, 256)	0
batch_normalization_4 (Batch	(None, 6, 6, 256)	1024
max_pooling2d_2 (MaxPooling2	(None, 2, 2, 256)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 4096)	4198400
dense_1 (Dense)	(None, 4096)	16781312
dense_2 (Dense)	(None, 5)	20485

Fig 29 Architecture based on AlexNet

The model was trained with a loss function of categorical cross-entropy and stochastic gradient descent as the optimizer. The results were not the best. As it can be observed in Fig 30 and Fig 31, the loss function and accuracy for validation shows that the model has overfitting.

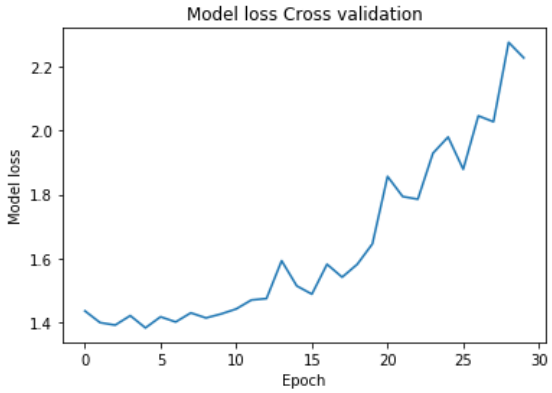


Fig 30 AlexNet model loss cross validation

The loss function observed in Fig 30 is ramping up instead of ramp down.

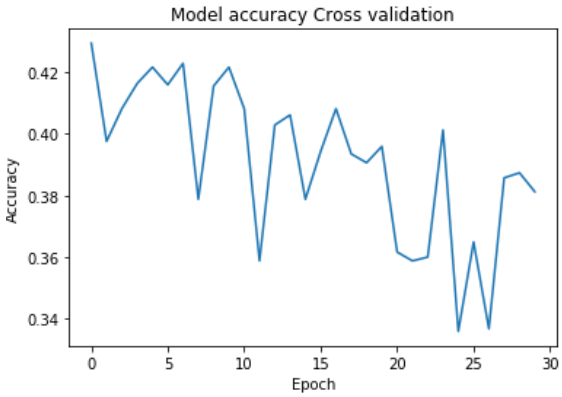


Fig 31 AlexNet model accuracy validation

Accuracy shown in Fig 31 has a not stable behavior. It is similar to a sawtooth wave; it is observed as a damped signal.

4.3. Proposed experimental model

Due to the AlexNet architecture did not show proper behavior during the training, I decided to work with an experimental model using the Tensor Flow API called Keras.

This experimental architecture contains four main blocks defined. The first one corresponds to a 32-filter convolutional layer with a 3x3 kernel and an input image resolution of 256x256. An activation function based on the ReLU one is used for this purpose. Subsequently, it is applied a 2x2 pooling that reduces the size of the resulting matrix in half.

It is repeated the same block three times and add it to the sequential model. Then it is added a flat layer that positions each neuron right next to the other in size nx1. To the previous layer, it is added a dense layer with a ReLU activation function that allows us to start classifying the images into 4 classes corresponding to the levels of DR. Additionally, a Dropout layer at 0.5 is used, which allows turning off 50% of the neurons during training. This avoids dependence between neighboring neurons in the training process. Finally, a SoftMax activation function at the end of the model is used.

```
# Creating the secuential model

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(600, 400, 3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(filters=32, kernel_size=(3, 3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3, 3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(units=64))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(units=5))
model.add(Activation('softmax'))
```

Fig 32 Experimental model

The next stage is the compilation stage. It was used "rmsprop" as an optimizer, an algorithm that is used for the optimization of complete batches. It tries to solve the problem that gradients can vary widely in magnitudes. Some gradients can be small and some can be huge, resulting in a very difficult problem.

Categorical cross-entropy was used as a loss function. A loss function is used for unique label categorization. This is when only one category applies for each data point. In other words, an example can belong to just one class.

```
# Compilation

model.compile(
    optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Fig 33 Compilation

Normalization is done 1/25 to the training a validation dataset. Later it was used the "flow_from_directory" method to point to the working directories with the corresponding images. It was used a batch size of 20 for the training set and also 20 for the test set. This is to pass all the images in batches due to the memory problem of the machines used in training.

```
# Image Processing

train_img_data = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)

test_img_data = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
)

train_set = train_img_data.flow_from_directory(
    directory=os.path.join('dataset', 'train'),
    target_size=(600, 400),
    batch_size=20
)

test_set = test_img_data.flow_from_directory(
    directory=os.path.join('dataset', 'test'),
    target_size=(600, 400),
    batch_size=20
)
```

Fig 34 Normalization and directories

Finally, the model is trained. In this step, it is defined a callback to the Tensor board which will collect the metrics in each training period.

```
# Trainning

log_dir = os.path.join('logs', datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

history = model.fit(
    x=train_set,
    steps_per_epoch=36,
    epochs=50,
    validation_data=test_set,
    validation_steps=36,
    callbacks=[tensorboard_callback]
)

print('classes', train_set.class_indices)
```

Fig 35 Model training

The use of this experimental model gave promising results, multilayer neural network was able to collect information to be trained, to finally probe with high accuracy that the model can properly categorize individuals, based on the level of damage of DR.

The behavior of training and validation were obtained with the help of the Tensor Board. This toolkit provides visualization and tooling needed for machine learning experimentation.

For accuracy, as it is shown in Fig 36, it is observed a value of 0.9496 for training and 0.9958 for validation.

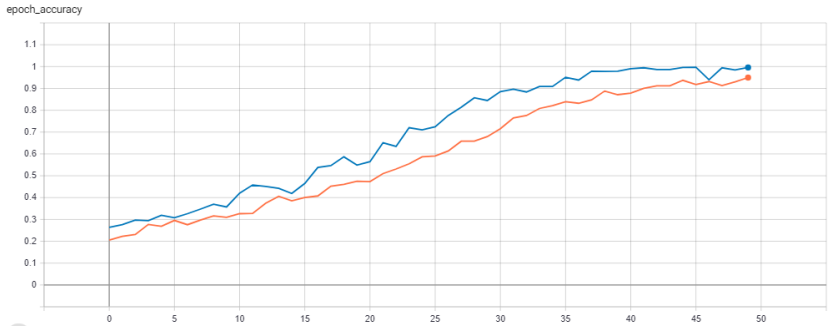


Fig 36 Accuracy results

In addition, for loss, it was gotten a value of 0.156 for training and 0.03418 for validation is shown in Fig 37.

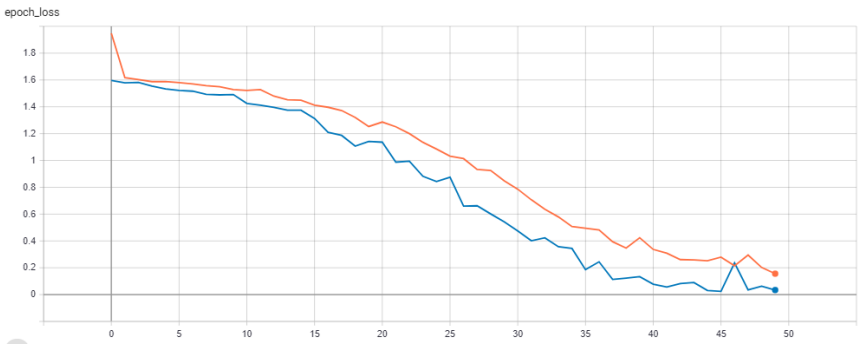


Fig 37 Loss results

The model representation by TensorBoard is shown in Fig 34.

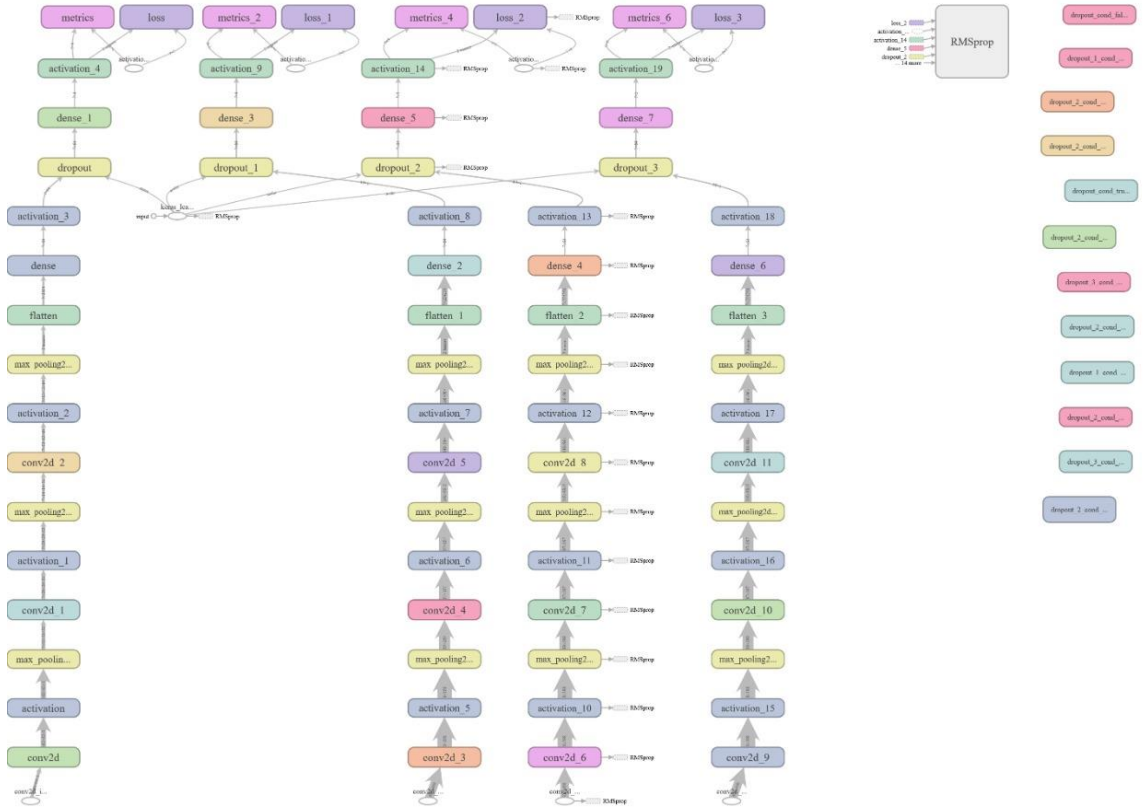


Fig 38 Model representation

The resultant confusion matrix can be observed in Table 5. We can observe very good results, just 3 images of 'No DR' were misclassified and 1 more for 'Mild' that was not classified correctly. The rest of the classes were classified correctly.

Table 5 Confusion matrix DR

n=750	Predicted as NO DR	Predicted as Mild	Predicted as Moderate	Predicted as Sever	Predicted as Proliferative DR
No DR	147	3	0	0	0
Mild	1	149	0	0	0
Moderate	0	0	150	0	0
Sever	0	0	0	150	0
Proliferative DR	0	0	0	0	150

5. RESULTS Y DISCUSSIONS

This project is an opportunity to cooperate with the healthcare industry, and especially with the most vulnerable sectors. This development work is capable to categorize iris images based on the level of deterioration.

This is an alternative and non-invasive methodology, that does not require complex and expensive laboratory tests to understand the condition of a person and potentially diagnose him/her with diabetes.

I decided to do not to work with other architectures as GoogleNet or newer due to these types CNNs demand a high computing power because they contain many layers. For that reason, I did a comparison of results between a CNN based on AlexNet, which is a basic CNN, and an experimental CNN.

As we observed, in this case, AlexNet was not a good architecture to solve the problem faced for classification. For this CNN I just got an accuracy of 0.38. Perhaps it is needed to adjust some of the parameters to have better results, but this work is intended to show the performance of an experimental model.

As it was shown in the results graphs, the trust level of this experimental model is high; it can classify the different levels of eyes damage and have the certainty that the model will be ~ 95% accurate.

The balance of the dataset is an important key to getting good results. Probably if the dataset was not balanced, I would have an overfitting model.

This experimental model was trained with a short dataset, similar to many of the authors mentioned in the state of the art. It would be interesting to have a powerful computer to work with the complete dataset and be able to process the original images non-resized to do not lose characteristics.

6. CONCLUSIONS

It is possible to use image-processing techniques, to be able to categorize patients depending on their level of DR damage grade.

Using Keras makes it very easy for us to write code to generate models without having to delve as deeply into the algorithms used. This used model belongs to Deep Learning since it was created using multilayer neural networks.

Multilayer neural network is a powerful technique that can help to predict behavior with a high level of accuracy; if we combine it with the proper analyzing data technique, the accuracy is high, getting to levels above 95% of accuracy.

When merging multiple techniques within one same model, the results are better than when using one single technique. If additionally, we apply a data cleaning using data analysis techniques, we are dramatically increasing the possibility of getting positive results on our analysis

Diabetes among other diseases in the world is a chronic degenerative condition that if not treated appropriately, can lead a patient to die. That is why early detection is crucial to elevate the survival possibilities.

Engineers and especially researchers should think on alternative ways for detection of these chronic diseases, making special emphasis on accessible solutions for sectors of the population that do not have access to the existing ones.

6.1. Future work

After getting successful results on this investigation, I got excited about the possibility to take this model and apply it to other problems in the same healthcare industry.

For example, what if our model is also capable to analyze x-rays images, to have an early diagnose of osteoporosis. As another example, we might be able to check blood close-up pictures and analyze components of the proteins to determine if a person is willing to develop kidney or liver issues.

The possibilities with this model are practically unlimited; this opens the door for applying this model to many other issues, and expands the capability of the healthcare industry to provide solutions to the majority of the population, and not only to the people that have access to the existing ones.

7. REFERENCES

[1] "WHO," [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes. [Accessed 07 September 2021].

[2] U. Ishtiaq, S. Abdul Kareem and E. Abdullah, Diabetic retinopathy detection through artificial intelligent techniques: a review and open issues, 2020.

[3] S. Amnia, B. Alhadi, Y. Anggun Rama, V. Andi Arus and M. Wibowo, "Diabetic Retinopathy Detection and Classification Using GoogleNet and Attention," 2021.

[4] P. H. Scanlon, «Complications of Diabetes,» 2019.

[5] "RGB Definition," [Online]. Available: https://techterms.com/definition/rgb. [Accessed 20 August 2021].

[6] Q. Wu and A. Cheddad, "Segmentation-based Deep Learning Fundus Image Analysis," in *Ninth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Istanbul, Turkey, 2019.

[7] J. Manjula, S. R. N. R. and Y. M. , An Ensemble Classification Techniques Based On "Ml" Model For Automatic Diabetic Retinopathy Detection An Ensemble Classification Techniques Based On "Ml" Model For Automatic Diabetic Retinopathy Detection, 2021.

[8] A. Deshpande and J. Pardhi, "Automated detection of Diabetic Retinopathy using VGG-16 architecture," *International Research Journal of Engineering and Technology*, vol. 08, no. 03, 2021.

[9] C. Wan, Y. Chen, H. Li, B. Zheng, N. Chen, W. Yang, C. Wang and Y. Li, "EAD-Net: A Novel Lesion Segmentation Method in Diabetic Retinopathy Using Neural Networks," vol. 2021, p. 13, 2021.

[10] L.-C. Woung, "Development of a mobile eye care services in integrated home-based medical care," in *ICIC20 Virtual Conference*, Taiwan, 2020.

[11] A. Noriega, D. Meizner, D. Camacho, J. Enciso, H. Quiroz-Mercado, V. Morales-Canton, A. Almaatouq and A. Pentland, "Screening Diabetic Retinopathy Using an Automated Retinal Image Analysis System in Independent and Assistive Use Cases in Mexico: Randomized Controlled Trial," *JMIR*, vol. 8, no. 5, 2021.

[12] J. McCarthy, "What is Artificial Intelligence," Stanford University, 2007.

[13] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of Machine Learning, London: Cambridge, 2018.

[14] "IBM," 01 May 2020. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning. [Accessed 03 October 2021].

[15] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[16] I. C. Education, "IBM," 20 October 2020. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Accessed 03 October 2021].

[17] A. Mishra, "Towards Data Science," 24 February 2018. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234. [Accessed 03 October 2021].

[18] "Numpy Org," 22 June 2021. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Accessed 04 October 2021].

[19] "Numpy," 22 June 2021. [Online]. Available: https://numpy.org/doc/stable/reference/arrays.ndarray.html#array-attributes. [Accessed 04 October 2021].

[20] "Pandas," [Online]. Available: https://pandas.pydata.org/about/. [Accessed 04 October 2021].

[21] "InfoWorld," 2019 January 2019. [Online]. Available: https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html. [Accessed 04 October 2021].

[22] "Opensource," 2017 November 2017. [Online]. Available: https://opensource.com/article/17/11/intro-tensorflow. [Accessed 04 October 2021].

[23] P. Nepal, "Analytics Vidhya," 30 July 2020. [Online]. Available: https://medium.com/analytics-vidhya/alexnet-architecture-explained-5d19e3dca2bb. [Accessed 09 November 2021].