Neiva, Mayo 2, 2024

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

John Alexander Gutierrez Gaviria, con C.C. No. 1075293047

Autor(es) de la tesis y/o trabajo de grado o _____

titulado SISTEMAS DE CLASIFICACIÓN AUTOMÁTICA DE GRADOS DE RETINOPATÍA DIABÉTICA MEDIANTE DEEP LEARNING

_____

presentado y aprobado en el año 2024 como requisito para optar al título de

Ingeniero Electrónico;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: John Alexander Gutierrez Gaviria

TÍTULO COMPLETO DEL TRABAJO: SISTEMAS DE CLASIFICACIÓN AUTOMÁTICA DE GRADOS DE RETINOPATÍA DIABÉTICA MEDIANTE DEEP LEARNING

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
John Alexander	Gutierrez Gavira

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Vladimir	Mosquera Cerquera

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva, Huila **AÑO DE PRESENTACIÓN:** 2024 **NÚMERO DE PÁGINAS:** 65

TIPO DE ILUSTRACIONES (Marcar con una **X**):

Diagramas X Fotografías X Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros X

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	Vision por computador	Computer vision	6.	Pixel	Pixel
2.	Inteligencia artificial	artificial intelligence	7.	Capa convolucional	Convolutional layer
3.	aprendizaje profundo	deep learning	8.	Keras	Keras
4.	redes neuronales convolucionales	convolutional neural network	9.	Tensorflow	Tensorflow
5.	retinoparía diabética	diabetic rethinopathy	10.	Imagenes de fondo de ojo	fundus pictures

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El fin de este proyecto de grado fue desarrollar una aplicación de software en lenguaje de programación de código abierto Python, que mediante el uso de redes neuronales convolucionales (CNN) posibiliten la detección y clasificación de la Retinopatía Diabética mediante imágenes de fondo de ojo.

La red neuronal convolucional adquiere imágenes de la base de datos Kaggle, cada imagen es pre-procesada adecuadamente con la finalidad de resaltar las características más relevantes y redimensionar la imagen a un tamaño de 512x512x3 permitiendo a la CNN un entrenamiento mas eficiente y rápido. La CNN realiza el proceso de extracción de características de las imágenes de fondo de ojo y clasificación de los cinco diferentes grados de Retinopatía Diabética. Se utilizó la métrica de evaluación Kappa Cuadrático Ponderado para medir el desempeño del sistema de aprendizaje. Usando el método de ensayo y error se implementó y entrenó la CNN.
Durante la experimentación se agregaron Custom Data Generators para facilitar el entrenamiento de la red e impedir el sobre-entrenamiento de esta. Se obtuvo un nivel de acuerdo con respecto a las etiquetas asignadas por un experto de 78%. De acuerdo con este resultado la red tiene un buen desempeño, indicando que fue capaz de extraer las características fundamentales para la clasificación de la enfermedad crónica retinopatía diabética.

ABSTRACT: (Máximo 250 palabras)

The purpose of this project is to develop a software application in the open-source programming language Python, which, using convolutional neural networks (CNN), enables the detection and classification of Diabetic Retinopathy through eye fundus images.

The convolutional neural network acquires images from the Kaggle data base, each image is pre-processed to highlight the most relevant features and resize the image to a size of 512x512x3, allowing the CNN to train more efficiently and faster. CNN performs the process of extracting features from the fundus images and classifying the five different grades of Diabetic Retinopathy. The Weighted Quadratic Kappa evaluation metric was extracted to measure the performance of the learning system. Using the trial and error method, the CNN was implemented and trained. During experimentation, custom data generators were added to facilitate network training and prevent network overfitting. A level of agreement was obtained according to the labels exposed by an expert of 78%. According to this result, the network the network has a “good” performance, indicating that it was able to extract the fundamental characteristics for the classification of chronic diabetic retinopathy.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Vladimir Mosquera C.

Firma:

Nombre Jurado: Martin Diomedes Bravo obando

Firma: Jose Fernando Barrera Compo

Nombre Jurado: José Fernando Barrera Compo.

Firma:

SISTEMAS DE CLASIFICACIÓN AUTOMÁTICA DE GRADOS DE RETINOPATÍA DIABÉTICA MEDIANTE DEEP LEARNING

JOHN ALEXANDER GUTIÉRREZ GAVIRIA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2024

SISTEMAS DE CLASIFICACIÓN AUTOMÁTICA DE GRADOS DE RETINOPATÍA DIABÉTICA MEDIANTE DEEP LEARNING

JOHN ALEXANDER GUTIÉRREZ GAVIRIA

Trabajo de tesis

Director

VLADIMIR MOSQUERA CERQUERA
M.Sc. Unicauca, Univalle.

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2024

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, Marzo 15 de 2024

Agradezco primero a Dios por darme la salud, las fuerzas y acompañarme en cada paso durante el tiempo de mi carrera universitaria y este proyecto de grado. A mi madre Sandra Rocío Gaviria por todas sus oraciones, enseñanzas sobre el sacrificio, por su respaldo, insistencia y amor. A mi padre John Alexander Gutierrez por todas sus enseñanzas sobre la vida y la familia. A mi hermana María Gutierrez por ser mi ejemplo a seguir y ayudarme en mi vida profesional. A mi hermana Catalina Gutierrez Gaviria por su orgullo y admiración, porque cada uno de mis logros los celebra sinceramente. A mi hermana María Gutierrez Gaviria por siempre cuidar de mí y alentarme a cada día ser una mejor persona. A mi novia Tatiana Marquez por ser una inspiración para alcanzar nuestros sueños y metas. A todas las personas que de una u otra manera ayudaron en este proceso de formación profesional. ¡Gracias!

John Alexander Gutierrez Gaviria

AGRADECIMIENTOS

Expreso mis más sinceros agradecimientos a mi director de tesis Vladimir Mosquera por todo su conocimiento compartido, por siempre estar disponible y atento ante mis inquietudes y guiarme de la manera correcta durante la realización de este proyecto de grado. Agradezco a nuestros jurados de tesis José de Jesús Salgado y Julián Molina por brindarme de manera excelente todos sus conocimientos académicos a lo largo de toda la carrera universitaria los cuales han sido indispensables para la realización de este proyecto de grado.

TABLA DE CONTENIDOS

Pág.

INTRODUCCIÓN. 13
PLANTEAMIENTO DEL PROBELMA.....15
JUSTIFICACIÓN. 17
1. OBJETIVOS 18
1.1 OBJETIVOS GENERAL……..18
1.2 OBJETIVOS ESPECÍFICOS……..…18
2. REDES NEURONALES CONVOLUCIONALES……19
3. RETINOPATÍA DIABÉTICA……..….25
4. METODOLOGÍA……..……27
4.1 SOFTWARE Y EQUIPO……27
4.2 BASE DE DATOS……27
4.3 MÉTRICA DE EVALUACIÓN: KAPPA CUADRÁTICO PONDERADO……...30
4.4 PRE-PROCESAMIENTO Y AUMENTO DE DATOS……32
4.4.1 Reducción de fondo……..……32
4.4.2 Escalamiento de radio……..……...32
4.4.3 Mapeo al 50% de gris……..……33
4.4.3.1 Eliminación del Efecto Aurora……..……33
4.4.4 Redimensionamiento de la imagen a 512x512x3……34
4.5 RED NEURONAL CONVOLUCIONAL……36
5. RESULTADOS……39
5.1 PRE-PROCESAMIENTO DE IMÁGENES……...39
5.2 BULK PRE-PROCESSING……45
5.3 RED NEURONAL CONVOLUCIONAL……49
5.4 ARQUITECTURA FINAL……53
5.4.1 Filtros y salidas de la red neuronal convolucional……...58
6. CONCLUSIONES……61
BIBLIOGRAFÍA……63

LISTA DE FIGURAS

Pág.

Figura 1. Ejemplo de primer bloque de una CNN. 20

Figura 2. Ejemplo de segundo bloque de una CNN. 21

Figura 3. Ejemplo de arquitectura de una red neuronal convolucional. 22

Figura 4. Signos de Retinopatía Diabética en una imagen de fondo de ojo. 26

Figura 5. Muestras de imágenes de retina del ojo izquierdo y derecho para dos pacientes de la base de datos Kaggle. 29

Figura 6. Imagen original a pre-procesar. 39

Figura 7. Imagen original a escala de grises e imagen con fondo reducido. 40

Figura 8. Imagen con radio escalado. 40

Figura 9. Imagen mapeada al 50% de gris. 41

Figura 10. Imagen sin efecto aurora. 42

Figura 11. Imagen final pre-procesada. 43

Figura 12. Histograma de una imagen pre-procesada. 44

Figura 13. Resultados del pre-procesamiento para una imagen de la base de Kaggle. 44

Figura 14. Muestra de arreglos creados con los nombres de las imágenes y el nivel de retinopatía diabética correspondiente. 45

Figura 15. Imágenes descartadas en el pre-procesamiento. 46

Figura 16. Muestra de los resultados del bulk pre-processing. 46

Figura 17. Imágenes antes y después de pasar por la etapa de pre-procesamiento. 47

Figura 18. Diagrama de flujo del pre-procesamiento. 48

Figura 19. Graficas de la exactitud y la pérdida del modelo inicial entrenado durante 50 épocas 50

Figura 20. Distribución de clases por batch. 52

Figura 21. Batch de Imágenes antes y después de aplicar data augmantation. 52

Figura 22. Arquitectura final de la CNN. 56

Figura 23. Grafica de la exactitud del modelo Final entrenado durante 150 épocas. 57

Figura 24. Filtros usados por la CNN en la primera capa convolucional, Conv2D(16, (3,3), padding='same', activation='relu'). 58

Figura 25. Imágenes de salida de la primera capa convolucional implementada en la arquitectura final de la CNN. 59

Figura 26. Matriz de confusión. 60

LISTA DE TABLAS

Pág.

Tabla 1. Relación entre las categorías de las etiquetas y el nivel de DR que representan. 28
Tabla 2. Interpretación del Kappa. 30
Tabla 3. Arquitectura de la CNN propuesta por Mathias Anthony y Stephan Brüggerma.....38
Tabla 4. Distribución de clases del set de entrenamiento de la base de datos de Kaggle. 51
Tabla 5. Distribución de clases del set de validación de la base de datos de Kaggle. 51
Tabla 6. Métricas de la matriz de confusión por clases. 60

GLOSARIO

Retinopatía Diabética: es una complicación de la diabetes que afecta los ojos. Es causada por el daño a los vasos sanguíneos del tejido sensible a la luz que se encuentran en el fondo del ojo (retina).

Imagen del fondo de ojo: imagen utilizada principalmente para diagnosticar enfermedades oculares.

Aprendizaje profundo: (*Deep Learning*) es un tipo de aprendizaje automático (*machine learning*, ML) e inteligencia artificial que imita la forma en que los humanos obtienen ciertos tipos de conocimiento.

Redes Neuronales: son un modelo para encontrar la combinación de parámetros que mejor se ajusta a un determinado problema.

CNN: Es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al cortex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver".

Capa de Convolución: procesa la salida de neuronas que están conectadas en "regiones locales" de entrada (es decir pixeles cercanos), calculando el producto escalar entre sus pesos (valor de píxel) y una pequeña región a la que están conectados en el volumen de entrada.

Keras: Es una biblioteca que funciona a nivel de modelo, proporciona bloques modulares sobre los que se pueden desarrollar modelos complejos de aprendizaje profundo.

Matplotlib: es una biblioteca de código abierto para la generación de gráficos a partir de vectores y arrays.

Píxel: es la parte homogénea en color más pequeña de una imagen digital generalmente compuesta por 8 bits.

RESUMEN

El fin de este proyecto de grado fue desarrollar una aplicación de software en lenguaje de programación de código abierto Python, que mediante el uso de redes neuronales convolucionales (CNN) posibiliten la detección y clasificación de la Retinopatía Diabética mediante imágenes de fondo de ojo.

La red neuronal convolucional adquiere imágenes de la base de datos Kaggle, cada imagen es pre-procesada adecuadamente con la finalidad de resaltar las características más relevantes y redimensionar la imagen a un tamaño de $512x512x3$ permitiendo a la CNN un entrenamiento mas eficiente y rápido. La CNN realiza el proceso de extracción de características de las imágenes de fondo de ojo y clasificación de los cinco diferentes grados de Retinopatía Diabética. Se utilizó la métrica de evaluación Kappa Cuadrático Ponderado para medir el desempeño del sistema de aprendizaje. Usando el método de ensayo y error se implementó y entrenó la CNN. Durante la experimentación se agregaron *Custom Data Generators* para facilitar el entrenamiento de la red e impedir el sobre-entrenamiento de esta. Se obtuvo un nivel de acuerdo con respecto a las etiquetas asignadas por un experto de 78%. De acuerdo con este resultado la red tiene un buen desempeño, indicando que fue capaz de extraer las características fundamentales para la clasificación de la enfermedad crónica retinopatía diabética.

PALABRAS CLAVES: Visión por computador, Inteligencia artificial, aprendizaje profundo, redes neuronales convolucionales, retinopatía diabética.

ABSTRACT

The purpose of this project is to develop a software application in the open-source programming language Python, which, using convolutional neural networks (CNN), enables the detection and classification of Diabetic Retinopathy through eye fundus images.

The convolutional neural network acquires images from the Kaggle data base, each image is pre-processed to highlight the most relevant features and resize the image to a size of $512x512x3$, allowing the CNN to train more efficiently and faster. CNN performs the process of extracting features from the fundus images and classifying the five different grades of Diabetic Retinopathy. The Weighted Quadratic Kappa evaluation metric was extracted to measure the performance of the learning system. Using the trial and error method, the CNN was implemented and trained. During experimentation, custom data generators were added to facilitate network training and prevent network overfitting. A level of agreement was obtained according to the labels exposed by an expert of 87%. According to this result, the network the network has a “good” performance, indicating that it was able to extract the fundamental characteristics for the classification of chronic diabetic retinopathy.

KEY WORDS: computer vision, artificial intelligence, deep learning, convolutional neural networks, diabetic retinopathy.

INTRODUCCIÓN

El Centro para el Control y la Prevención de Enfermedades de Estados Unidos estima que 29.1 millones de personas en los Estados Unidos tienen diabetes y la Organización Mundial de la Salud estima que 347 millones de personas padecen la enfermedad en todo el mundo. La Retinopatía Diabética (DR) del acrónimo en inglés Diabetic Retinopathy, es una enfermedad ocular asociada con diabetes de larga duración. La progresión hacia el deterioro de la visión puede ralentizarse o evitarse si se detecta DR a tiempo, sin embargo, esto puede ser difícil ya que la enfermedad a menudo muestra pocos síntomas hasta que es demasiado tarde para proporcionar tratamiento efectivo [1] siendo la causa más común de ceguera del ojo dependiendo de la diabetes. Por esta razón, la detección temprana de la Retinopatía Diabética es de importancia crítica.[2]

Actualmente, la detección de DR es un proceso manual que requiere mucho tiempo y así mismo, un médico capacitado para examinar y evaluar fotografías digitales en color del fondo de la retina. Para cuando los lectores humanos envían sus revisiones, a menudo uno o dos días después, los resultados retrasados conducen a un seguimiento perdido, falta de comunicación y tratamiento retrasado.

La causa principal de la Retinopatía Diabética se debe a un trastorno metabólico que aumenta los niveles de glucosa en la sangre de una persona, lo que hace que la persona presente altos niveles de presión arterial, que a su vez afectan el sistema circulatorio de la retina y el revestimiento sensible a la luz en la parte posterior del ojo [3]. Los médicos pueden identificar DR por la presencia de lesiones asociadas con las anomalías vasculares causadas por la enfermedad. Si bien este enfoque es efectivo, sus demandas de recursos son altas. La experiencia y el equipo requeridos a menudo faltan en áreas donde la tasa de diabetes en las poblaciones locales es alta y la detección de DR es más necesaria. A medida que el número de personas con diabetes continúa creciendo, la infraestructura necesaria para prevenir la ceguera debido a la DR se volverá aún más insuficiente.

---

[1] Disponible en: https://www.kaggle.com/c/diabetic-retinopathy-detection
[2] Nursel Yalçin, Seyfullah Alver, Necla Uluhatun. Classification of retinal images with Deep Learning for early detection of diabetic retinopathy disease.
[3] Marıa A. Bravo, Pablo A. Arbelaez. Automatic Diabetic Retinopathy Classification. Universidad de los Andes, Bogota, Colombia

La necesidad de un método integral y automatizado de detección de DR ha sido reconocida desde hace mucho tiempo, y los esfuerzos anteriores han hecho un buen progreso utilizando la clasificación de imágenes, el reconocimiento de patrones y el aprendizaje automático. Con la fotografía del fondo de retina como entrada, el objetivo de este trabajo de grado cobra gran importancia al desarrollar una herramienta basada en redes neuronales convolucionales que son capaces de automáticamente identificar y extraer de las imágenes sus características más relevantes, para la clasificación automática de las etapas de Retinopatía Diabética que agilicen el proceso y hagan más objetivos los diagnósticos del experto. Su objetivo será identificar y clasificar la Retinopatía Diabética en sus cinco etapas: Retinopatía Diabética no Proliferativa (No DR), Leve, Moderado, Severo y DR proliferativa.

PLANTEAMIENTO DEL PROBLEMA

La presencia de diferentes complicaciones oculares derivadas de la Diabetes, han llevado al ser humano a investigar métodos más sencillos, eficaces y confiables para la detección, clasificación y posterior tratamiento de ellas. Existen diferentes procedimientos para que el especialista pueda realizar el diagnostico de Retinopatía Diabética, como lo son: examen de agudeza visual, examen del iris y el ángulo iridocorneano, examen de fondo de ojo con dilatación pupilar, fotografía de fondo de ojo, angiografía con fluoresceína, Tomografía de Coherencia Óptica (O.C.T.).

El diagnóstico digital por imagen para la retina se vale de sistemas de procesamiento de imágenes de alta resolución para tomar fotografías del interior del ojo, esto ayuda a los especialistas a determinar la salud de la retina a la vez que les permite detectar y controlar enfermedades y complicaciones oculares. Es fundamental descubrir lo antes posible irregularidades en la retina para prevenir el avance de enfermedades potencialmente graves e incluso la pérdida de la visión. Además de ayudar a detectar enfermedades en forma precoz, las imágenes de la retina brindan un historial permanente de los cambios producidos en los ojos. Con las imágenes se pueden hacer comparaciones paralelas y anuales a fin de descubrir incluso los cambios más sutiles y ayudar al control de la salud.

Este problema se agrava principalmente cuando el crecimiento en el número de oftalmólogos es mucho menor que el crecimiento de diabéticos, al ser la Retinopatía Diabética una complicación en donde su detección a tiempo es fundamental para el correcto control, tratamiento y evitar la pérdida de visión.

En este orden de ideas, el análisis de imágenes de la retina requiere de un aprendizaje a partir del conocimiento previo de datos de diferentes pacientes, con el fin de facilitar y afirmar la detección de anomalías en estas.

El proyecto en cuestión aparece, entonces, como una alternativa confiable, eficiente, ágil y económica, pues tiene como objetivo otorgar una herramienta destinada a la detección y clasificación de las cinco etapas de la Retinopatía Diabética (No DR, Leve, Moderado, Severo, DR proliferativa) empleando el deterioro en los vasos sanguíneos a partir de imágenes de fondo de ojo, mediante procesamiento digital de imágenes el uso de las Redes neuronales convolucionales, siendo herramientas fundamentales de apoyo para el diagnóstico de Retinopatía Diabética. ¿Es posible desarrollar una aplicación de software que haciendo uso de procesamiento digital de imágenes y técnicas de inteligencia computacional tenga

la capacidad de detectar y clasificar la Retinopatía Diabética mediante imágenes de fondo de retina?

JUSTIFICACIÓN

La Retinopatía Diabética (DR) es una enfermedad en la cual la retina se daña debido al aumento de la presión arterial de los vasos sanguíneos pequeños del ojo humano. DR es la principal causa de ceguera para los diabéticos. Se ha demostrado que el diagnóstico temprano puede jugar un papel importante en la prevención de la pérdida visual y la ceguera. La gran población de pacientes diabéticos y sus requisitos de detección masiva han generado interés en un diagnóstico asistido por computadora y completamente automático de DR. Este trabajo propone un enfoque basado en computadora para la detección de DR en imágenes de fondo de ojo basadas en el uso de redes neuronales convolucionales (CNN) del acrónimo en inglés Convolutional Neural Network. La CNN a implementar utiliza aprendizaje profundo para clasificar las fotografías de la retina en el fondo del ojo en 5 etapas de DR.

De acuerdo con lo mencionado anteriormente, la importancia de este trabajo radica precisamente en la combinación de procesamiento digital de imágenes y técnicas de inteligencia computacional, para determinar características y patrones y usarlas en la clasificación de las diferentes etapas de Retinopatía Diabética.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Desarrollar una aplicación de software en lenguaje de programación Python, que mediante el uso de redes neuronales convolucionales posibiliten la detección y clasificación de la Retinopatía Diabética mediante imágenes de fondo de retina.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar preprocesamiento a las imágenes de fondo de retina para la detección y clasificación de Retinopatía Diabética.

- Seleccionar las características de las imágenes de fondo de retina más adecuadas para el proceso de detección y clasificación de Retinopatía Diabética.

- Implementar la arquitectura de red neuronal convolucional basada en Deep Learning que realice la detección y clasificación propuesta.

- Validar la robustez de la Red neuronal para la detección de la Retinopatía Diabética.

2. REDES NEURONALES CONVOLUCIONALES

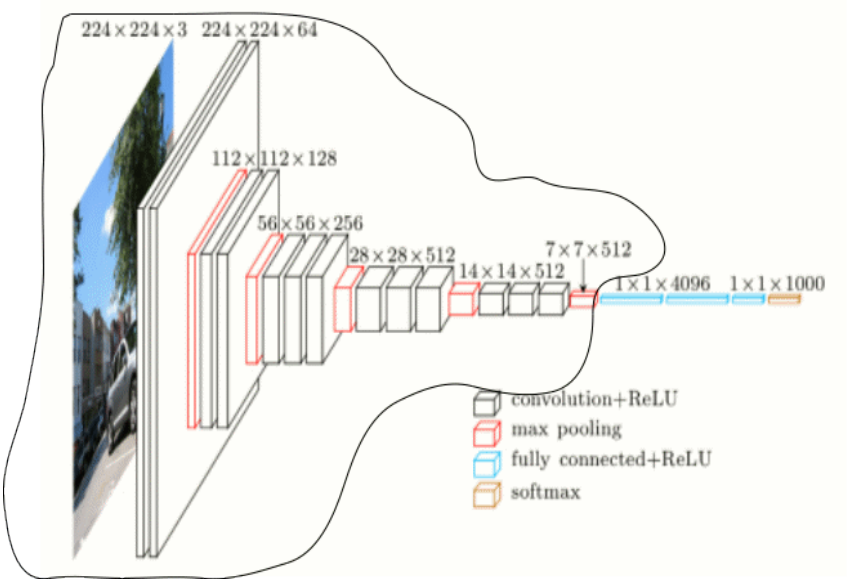
La Red Neuronal Convolucional (CNN) es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver". Para ello, la CNN contiene varias capas ocultas especializadas y con una jerarquía: esto quiere decir que las primeras capas pueden detectar líneas, curvas y se van especializando hasta llegar a capas más profundas que reconocen formas complejas como un rostro o la silueta de un animal.

El descubrimiento de que se podría usar una CNN para extraer progresivamente representaciones de alto y más alto nivel del contenido de la imagen trajo un gran avance en la construcción de modelos para la clasificación de imágenes; La CNN toma como entrada los datos de píxeles sin procesar de la imagen y aprende cómo extraer estas características y, en última instancia deduce qué objeto constituyen. La CNN está diseñada específicamente para procesar imágenes de entrada. Su arquitectura entonces más específicamente se compone de dos bloques principales.

El primer bloque de esta arquitectura tiene como cualidad para este tipo de redes neuronales su funcionamiento como un extractor de características, este procedimiento lo realiza mediante la comparación de plantillas a las cuales se les aplica un filtrado de convolución. Tenemos la primera capa cuya función es tomar una imagen y filtrarla con varios filtros de convolución, devolviendo mapas de características los cuales son normalizados con una función de activación y/o se redimensionan. Al tener la facilidad de repetir varias veces este proceso se filtran los mapas de características obtenidos con los nuevos filtros, lo que nos proporciona nuevos mapas de características para su respectiva normalización y redimensionamiento. Finalmente, los valores de los últimos mapas de características se concatenan en un vector. Este vector define la salida del primer bloque y la entrada del segundo.

Fuente:https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

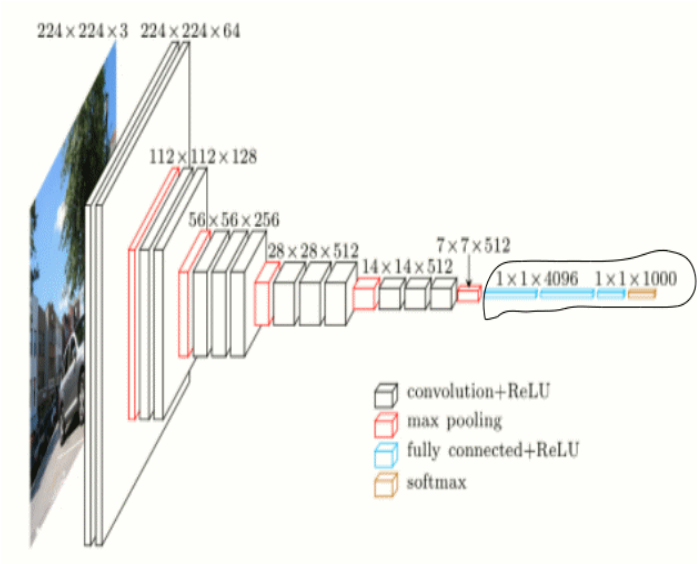
Figura 1. Ejemplo de primer bloque de una CNN.



Fuente:https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

El segundo bloque no es característico de una CNN. De hecho, está al final de todas las redes neuronales empleadas para la clasificación. Se necesita que los valores del vector de entrada se modifiquen mediante varias combinaciones lineales y funciones de activación, devolviendo un nuevo vector que será de salida. Dicho vector de salida tendrá tantos elementos como clases hay. El elemento i representa la probabilidad de que la imagen pertenezca a la clase i. Por tanto, cada elemento está entre 0 y 1, y la suma de todos vale 1. La última capa de este bloque se encarga de realizar un cálculo de estas probabilidades, mediante una función logística que realiza una clasificación binaria o a través de una función softmax que desarrolla una clasificación multiclase como una función de activación. De la misma forma que con las redes neuronales ordinarias, los parámetros de las capas están determinados por la retro propagación de gradiente. La entropía cruzada se minimiza durante la fase de entrenamiento. Sin embargo, en el caso de las CNN, estos parámetros se refieren en particular a las características de la imagen.

Figura 2. Ejemplo de segundo bloque de una CNN.



Fuente:https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Para comenzar, la CNN recibe un mapa de características de entrada la cuál es una matriz tridimensional donde el tamaño de las dos primeras dimensiones corresponde a la altura y el ancho de las imágenes en píxeles. El tamaño de la tercera dimensión es 3 (correspondiente a los 3 canales de una imagen en color RGB: rojo, verde y azul).[4] La CNN comprende una pila de módulos, cada uno de los cuales realiza tres operaciones.

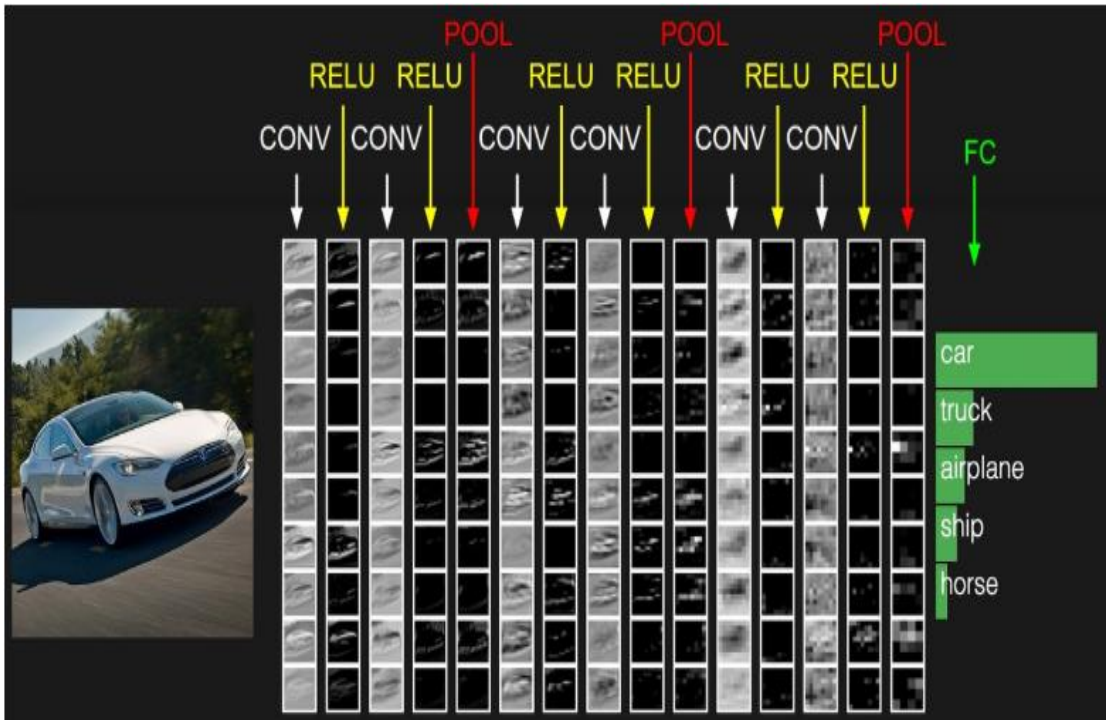
Una ConvNet del inglés Convolutional Network, simple es una secuencia de capas, y cada capa de una ConvNet transforma un volumen de activaciones a otra a través de una función diferenciable. Comunmente se utiliza tres tipos principales de capas para construir arquitecturas ConvNet: Capa convolucional, Capa de POOL y Capa totalmente conectada (FC).[5]

[4] Disponible en: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

[5] Disponible en: http://cs231n.github.io/convolutional-networks/

Con el fin de explicar la arquitectura básica de una CNN, se usará como ejemplo una red clasificadora, aplicada a la base de datos CIFAR-10 (imágenes a color de 32x32 de 10 clases). Esta arquitectura se presenta en la Figura 3. La ConvNet para la clasificación CIFAR-10 podría tener la arquitectura [INPUT – CONV – RELU – POOL – FC].

Figura 3. Ejemplo de arquitectura de una red neuronal convolucional.



Fuente: KARPATHY, ANDREJ. CS231n Convolutional Neural Networks for Visual Recognition, Convolutional Neural Networs (CNNs / ConvNets) [en línea]. Stanford University [Consultado 1 de abril de 2020]. Disponible en internet: http://cs231n.github.io/convolutional-networks/

En la Figura 3 en la parte izquierda se observa la imagen de un automóvil como entrada de la CNN [32x32x3] la cuál mantendrá los valores de píxeles sin procesar de la imagen, en este caso una imagen de ancho 32, altura 32 y con tres canales de color R, G, B. Se observa también que la capa CONV extrae mapas de características de la entrada y les aplica filtros para calcular nuevas características, produciendo un mapa de características de salida (que puede tener un tamaño y profundidad diferentes a los del mapa de entrada). Las convoluciones se definen por dos parámetros: Tamaño de los mosaicos que se extraen (normalmente 3x3 o

5x5 píxeles). La profundidad del mapa de características de salida, que corresponde al número de filtros que se aplican. Esto puede resultar en un volumen como [32x32x12] si se decide usar 12 filtros. Posteriormente la capa RELU – Después de cada operación de convolución, la CNN aplica una transformación de unidad lineal rectificada (ReLU del inglés Rectified Linear Unit) a la característica convolucionada, con el fin de introducir la no linealidad en el modelo. La función ReLU, $f(\mathrm{x}) = \max(0, \mathrm{x})$, devuelve x para todos los valores de $x > 0$, y devuelve 0 para todos los valores de $x \leq 0$. Esto deja el tamaño del volumen sin cambios ([32x32x12]). Consecutivamente la capa POOL - Después de ReLU viene un paso de agrupación, en el que la CNN cambia la configuración de la función convolucionada (para ahorrar en el tiempo de procesamiento), reduciendo el número de dimensiones del mapa de características, al tiempo que conserva la información de la característica más crítica. Un algoritmo común utilizado para este proceso se denomina max-pooling. Max-pooling funciona de manera similar a la convolución. Se desliza sobre el mapa de características y se extraen mosaicos de un tamaño específico. Para cada mosaico, el valor máximo se envía a un nuevo mapa de características y todos los demás valores se descartan. Las operaciones de agrupación máxima toman los siguientes parámetros. El primer parámetro es el tamaño del filtro de agrupación máxima (normalmente 2x2 píxeles) y el segundo parámetro es el Paso, que representa la distancia en píxeles que separa cada mosaico extraído. Finalmente, la capa FC (es decir, totalmente conectada) calculará los puntajes de la clase, lo que resultará en un volumen de tamaño [1x1x10], donde cada uno de los 10 números corresponde a un puntaje de clase, como por ejemplo entre las 10 categorías de CIFAR-10. Al igual que con las redes neuronales ordinarias y como su nombre lo indica, cada neurona en esta capa se conectará a todos los números en el volumen anterior.

Al entrenar redes neuronales profundas, se requiere de una gran cantidad de imágenes que contengan ejemplos de las diferentes categorías a clasificar para alcanzar desempeños significativos. En caso de que la base de datos contenga una cantidad limitada de imágenes, se recomienda realizar *data augmentation* para mejorar el desempeño de la red. Esta técnica busca principalmente construir datos sintéticos mediante la realización de transformaciones a datos etiquetados existentes, de esta manera ayudar al modelo de Deep Learning a aprender en un rango más amplio de variaciones dentro de las categorías. Algunas formas de aplicar *data augmentation* es, por ejemplo, operaciones como: girar la imagen vertical/horizontalmente, rotarla, realizar cortes aleatorios, variaciones de color y el

nivel de zoom. Normalmente en la práctica, estas muestras sintéticas son agregadas a la base de datos de entrenamiento para enriquecerlo.

3. RETINOPATÍA DIABÉTICA

La Retinopatía Diabética (DR) es la principal causa de pérdida visual no recuperable en los países industrializados en pacientes entre los 20 y 64 años, siendo responsable de un 10% de nuevos casos de ceguera cada año. El riesgo de ceguera en pacientes diabéticos seria aproximadamente 25 veces mayor al resto de la población. La Retinopatía Diabética (DR), ocurre cuando niveles altos de azúcar en la sangre generan daño en los vasos sanguíneos de la retina. Este daño, puede generar que los vasos sanguíneos se obstruyan, evitando el flujo de sangre a través de ellos. También es posible que se hinchen y goteen fluidos. Normalmente la DR se diagnostica con controles oculares anuales. Actualmente la retinopatía diabética comprende 5 etapas de clasificación de la enfermedad.

Fuente: D. ALISEDA, L. BERÁSTEGUI. Diabetic retinopathy Servicio de Oftalmología. Hospital de Navarra. Disponible en : https://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1137-66272008000600003

- **No DR:** En esta etapa, el paciente con la enfermedad de diabetes se considera sin Retinopatía Diabética.

- **Retinopatía no proliferativa ligera:** Está etapa es considerada la más temprana de la enfermedad en la que se originan los micro aneurismas, que aparecen como dilataciones de la pared de los capilares, estas pequeñas áreas de inflamación tienen una similitud con ampollas en los pequeños vasos sanguíneos de la retina.

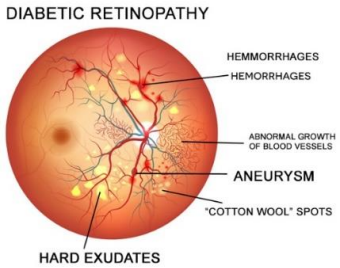
- **Retinopatía no proliferativa moderada:** Ya superada la etapa temprana de la enfermedad, aparecen hemorragias en una moderada cantidad y cómo consecuencia algunos vasos sanguíneos que alimentan la retina se obstruyen.

- **NPDR (Retinopatía diabética no proliferativa):** Es la etapa media de la enfermedad ocular diabética. En el mundo muchas personas padecen esta etapa de la enfermedad, la cual produce que la retina se inflame debido a la pérdida de muchos vasos sanguíneos pequeños. Lo que sucede es que se genera un edema macular cuándo se inflama la mácula del ojo. Esta es

comúnmente la razón por la que las personas con diabetes pierden la visión. Durante esta etapa de la enfermedad aparece una isquemia macular que es cuándo los vasos sanguíneos en la retina pueden cerrarse, debido a esto a la sangre le es imposible llegar a la mácula. En algunos casos, se forman pequeñas partículas en la retina que son llamadas exudados, y como consecuencia estas partículas pueden afectar la visión. La retinopatía diabética no proliferativa causa que la visión sea borrosa.

- **PDR (retinopatía diabética proliferativa)** Es considerada la etapa más avanzada de la enfermedad ocular diabética. Se genera por la denominada neovascularización, este proceso ocurre cuándo la retina empieza a desarrollar nuevos vasos sanguíneos. Estos vasos nuevos frágiles a menudo sangran hacia *el vítreo* y se puede visualizar en la Figura 4 como *hemorrhage*. Si sólo sangran un poco, quizá el paciente experimentará ver unas cuantas moscas volantes oscuras. Si sangran mucho, puede que bloqueen toda la visión. Estos vasos sanguíneos nuevos pueden desarrollar cicatrices. El tejido cicatrizante puede causar problemas con la mácula o derivar en un desprendimiento de retina. La retinopatía diabética proliferativa es muy grave, causará dolor y eventualmente daño al nervio del ojo y ceguera.[6]

Figura 4. Signos de Retinopatía Diabética (DR).



Fuente: ORCHARD, ELIZABETH. Diabetic Retinopathy Treatment [en línea] Eagle eye Centre [Consultado 1 de Abril de 2020]. Disponible en internet: http://www.eagleeyecentre.com.sg/service/diabetic-retinopathy/

[6] Disponible en: http://www.eagleeyecentre.com.sg/service/diabetic-retinopathy/

4. METODOLOGÍA

4.1 SOFTWARE Y EQUIPO

Se empleó un computador personal de la marca *MSI* modelo *GF65 Thin* y el sistema operativo Windows 11. Como librería para Deep Learning se utilizó Keras, configurando como backend Tensorflow[7]. El lenguaje de programación que se empleó fue Python y el código se desarrolló en la aplicación web Jupyter Notebook.

Las unidades de procesamiento central "CPU" son optimizadas para el procesamiento secuencial mientras que las unidades de procesamiento gráfico "GPU" son utilizadas para incrementar la velocidad de los cómputos vectoriales y matriciales requeridos en aplicaciones de Deep Learning. Por consiguiente, el entrenamiento del sistema de Deep Learning fue procesado por una GPU cuyas características técnicas son las siguientes[8]:

GPU: GeForce GTX 1660 Ti, memoria de video de 6GB GDDR6, velocidad de memoria de 12 Gbps

CPU: Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz, memoria RAM de 24GB y disco duro SSD de 512 GB.

4.2 BASE DE DATOS

Para desarrollar el sistema Deep Learning se utilizó la base de datos de Kaggle la cual consiste en un conjunto de gran tamaño de imágenes de fondo de ojo (Aproximadamente 88.000) de alta resolución entre 4 y 14 Megapíxeles. Las imágenes del conjunto de datos provienen de diferentes modelos y tipos de cámaras, lo que puede afectar la apariencia visual de la izquierda frente a la derecha. Algunas imágenes de la base de datos ilustran la anatomía la retina (mácula a la izquierda, nervio óptico a la derecha para el ojo derecho). Otros muestran cómo se vería a través de una lente de condensación de microscopio (es decir, invertida, como se ve en un examen ocular típico). Como cualquier conjunto

[7] CHOLLET, Francois. Keras: Deep Learning library for Theano and TensorFlow [en línea]. KERAS, 2017. Fuente: https://keras.io/

[8] NVIDIA Coporation [en línea] Fuente: https://www.nvidia.com/en-us/about-nvidia/

de datos del mundo real, se encontrará ruido tanto en las imágenes como en las etiquetas. Las imágenes pueden contener artefactos, estar desenfocadas, subexpuestas o sobreexpuestas. [9]

En esta base de datos se encuentran imágenes del ojo izquierdo y derecho para cada paciente, cada imagen ha sido etiquetada de acuerdo con el número de identificación del paciente y el ojo correspondiente (por ejemplo, la etiqueta 1_left.jpeg corresponde al ojo izquierdo del paciente con identificación 1). Un Oftalmólogo ha clasificado las imágenes diagnosticándolas con un nivel de Retinopatía Diabética en una escala de 0-4 (según la Tabla 1). De esta manera, el sistema automático que se desarrollará en este proyecto tiene como salida un valor DR basado en esta escala.

Tabla 1. Relación entre las categorías de las etiquetas y el nivel de DR que representan.

Categoría	Nivel de Retinopatía Diabética
0	No DR
1	Leve
2	Moderado
3	Severo
4	DR proliferativa

Fuente: Kaggle INC. Diabetic Retinopathy Detection – Data [en línea]. Kaggle Inc, 2017 [Consultado 4 de abril de 2020]. Disponible en Internet: https://www.kaggle.com/c/diabetic-retinopathy-detection/data

En la Figura 5, se puede observar muestras de la base de datos utilizada (Kaggle). Se muestra la etiqueta de cada imagen en la parte superior la cual representa la categoría de DR que ha sido diagnosticada para esta imagen de fondo de retina. Se detalla que las imágenes han sido tomadas bajo diferentes condiciones, se puede apreciar también diferentes tamaños y diferentes niveles de acercamiento a la retina.

[9] KAGGLE INC. Diabetic Retinopathy Detection [en línea]. Kaggle Inc, 2017. Fuente: https://www.kaggle.com/c/diabetic-retinopathy-detection

Figura 5. Muestras de imágenes de retina del ojo izquierdo y derecho para dos pacientes de la base de datos Kaggle.

Ojo Izquierdo	Ojo Derecho
Nivel: 0 	Nivel: 0 
Nivel: 2 	Nivel: 2 

Fuente: Kaggle INC. Diabetic Retinopathy Detection – Data [en línea]. Kaggle Inc, 2017 [Consultado 4 de abril de 2020]. Disponible en Internet: https://www.kaggle.com/c/diabetic-retinopathy-detection/data

4.3 MÉTRICA DE EVALUACIÓN: KAPPA CUADRÁTICO PONDERADO

Para el desempeño de los sistemas de aprendizaje se usará el Kappa Cuadrático Ponderado, llámese K_w (del inglés: weighted kappa). De acuerdo con Cohen[10], el K_w es un indicador del nivel de concordancia entre dos grupos de elementos cualitativos (variables categóricas), que tiene en cuanta el efecto del azar y permite medir los desacuerdos de manera eficiente. Al usar esta métrica, se producirán valores negativos cuando haya menos concordancia de la pronosticada por el azar, cero cuando la concordancia sea exactamente la pronosticada por el azar y uno al ocurrir una completa concordancia. En la Tabla 2, se muestra una interpretación del valor Kappa, de acuerdo con la fuerza del nivel de concordancia entre los evaluadores.

Tabla 2. Interpretación del Kappa.

Valor de *K*	Concordancia
< 0.20	Poor
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Good
0.81 - 1.00	Very Good

Fuente: ALTAM, Douglas G. Practical statistics for medical research. London: Chapman and Hall, 1991, p. 404

Se utilizará el K_w para medir el desempeño de la aplicación de software a desarrollar en este proyecto. La concordancia es medida de acuerdo con las etiquetas obtenidas según un evaluador humano(X), y las obtenidas por el sistema (P). Las categorías de clasificación posibles de las imagines son: 0, 1, 2, 3, 4. El K_w será calculado como se describe a continuación.

Primero, es construida una matriz de histograma *O* de dimensiones NxN (donde N es la cantidad de categorías), tal que O_{ij} corresponde con el número de imágenes que reciben una evaluación i por A y una evaluación j por B (donde A es la CNN y

[10] COHEN, Jacob. Weighted kappa: Nominal Scale agreement with provision for scaled disagreement or partial credit. En: Psychological Bulletin. October 1968 vol. 70 no. 4, p. 213-220.

B es el experto evaluador). Una matriz N-por-N de pesos W, es calculada, basada en la diferencia entre las clasificaciones indicadas por los evaluadores $W_{ij} = \frac{(i-j)^2}{(N-1)^2}$

Una Matriz de histograma (NxN) de clasificaciones esperadas, E, es calculada, asumiendo que no hay correlación entre las clasificaciones de los evaluadores. Esta, es calculada como el producto externo entre el vector de histograma de clasificaciones de cada evaluador, normalizado tal que E y O tengan la misma suma. A partir de estas tres marices, el Kappa cuadrático ponderado es calculado según:

Fórmula 1. Kappa cuadrático ponderado.

$$K_w = 1 - \frac{\sum_{ij} W_{ij} O_{ij}}{\sum_{ij} W_{ij} E_{ij}}$$ [11]

[11] KAGGLE INC, Op. cit. Disponible en Internet: https://www.kaggle.com/c/diabetic-retinopathy-detection#evaluation

4.4 PRE-PROCESAMIENTO Y AUMENTO DE DATOS

Un problema que se enfrenta con la base de datos de Kaggle es la variedad de imágenes de fondo de ojo, ya que fueron tomadas bajo diferentes condiciones lo que genera imágenes muy diversas. Para asegurarse de que el sistema de Deep Learning aprenderá las verdaderas características de DR, se necesita pre-procesar las imágenes de fondo de ojo y lograr un formato uniforme entre todas ellas a través de los siguientes pasos**.**

4.4.1 Reducción de Fondo

En las imágenes se encontrará un fondo de color negro que ocupa gran parte de la imagen, por lo cual, se decide como primer paso del pre-procesamiento la reducción del fondo de las imágenes de tal manera que se reduzca el número de pixeles que son ocupados por el fondo de la imagen y estandarizar la posición de la retina en cada imagen en el centro de esta. Se inicia este procedimiento con la conversión de la imagen a escala de grises, para posteriormente recorrer la imagen desde el centro hacia los cuatro extremos evaluando si la media de cada fila o columna de pixeles permanece por debajo de cierto limite, si la media supera este límite se considera que desde allí inicia el fondo negro de la imagen. Al detectar este cambio se obtienen los cuatro puntos indicando el inicio y final de los pixeles de altura y ancho, luego se corta la imagen original (RGB). Como resultado se obtiene una imagen con fondo reducido, por lo tanto, el tamaño de la imagen es menor.

4.4.2 Escalamiento de radio

Al ser las imágenes de fondo de ojo de forma circular, este proceso tiene como finalidad obtener imágenes con radios similares.

Primero, se halla el valor medio a lo largo del eje y haciendo el recorrido a través del eje x, obteniendo un arreglo en donde se tendrán los valores de los pixeles de cada canal RGB. luego se obtienen los pixeles que están por encima del 10% de la media, con esto se excluyen los pixeles que conforman el fondo de la imagen (color negro), así se adquiere el diámetro de la imagen de fondo de ojo y finalmente se divide este valor en 2 para obtener como resultado el radio de cada imagen de fondo de retina a lo largo de y.

De igual manera se halla el valor medio a lo largo del eje x y se recorre a través del eje y, obteniendo un arreglo en donde se tendrán los valores de los pixeles de cada canal RGB. luego se obtienen los pixeles que están por encima del 10% de la media,

con esto se descartan los pixeles que conforman el fondo de la imagen (color negro), así se obtiene el diámetro de la imagen de fondo de retina y finalmente se divide este valor en 2 para conseguir como resultado el radio de cada imagen de fondo de retina a lo largo de x.

El siguiente paso en este procedimiento es obtener el radio más grande, ya que algunas imágenes no son círculos completos, sino que están cortadas en x o en y.

Una vez obtenido el valor del radio, se calcula el factor de la escala correspondiente para que las imágenes procesadas queden con radios similares, se divide un valor constante igual al valor medio de las dimensiones en las que se desea la imagen final, este valor es igual a 256 (ya que las dimensiones deseadas de la imagen luego de su pre-procesamiento son de 512x512x3), por el valor del radio obtenido. El factor de escala se utiliza en el método *resize* [12] de la librería OpenCV y se redimensiona la imagen al tamaño deseado.

4.4.3 Mapeo al 50% de gris

El principal motivo del mapeo al 50% de gris es resaltar los detalles dentro de la imagen de fondo de retina para lo cual las tonalidades tanto del fondo de la imagen como el fondo de la retina deben ser similares.

Para realizar este procedimiento primero se debe obtener el color medio local de cada píxel en la imagen, para ello se pasa la imagen por un filtro Gaussiano haciendo uso del método *GaussianBlur* [13] de la librería OpenCV, ya que este filtro arroja como resultado una imagen en donde el valor de cada píxel ha sido obtenido como el promedio ponderado de sus pixeles vecinos.

Luego de obtener el color medio local de cada píxel en la imagen, se resta a cada píxel de la imagen original el color medio local para posteriormente sumar un valor constante que corresponde al 50% de la escala de gris (0-255) igual a 128.

4.4.3.1 Eliminación del Efecto Aurora

Una vez se realiza el procedimiento del mapeo de la imagen al 50% de gris, en la imagen resultante llámese (*image_mapped_50*) se genera un efecto de aurora el cual se debe remover de la imagen antes de seguir con el siguiente paso para

[12] Open Source Computer Vision.[en línea] OpenCV-Python [Consultado 3 de Noviembre de 2021] Fuente: https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html
[13] Open Source Computer Vision.[en línea] OpenCV-Python [Consultado 4 de Diciembre de 2021] Fuente: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

prevenir la extracción de características no deseadas y que puedan ocasionar que la red neuronal aprenda de forma errónea las características de las imágenes. para lograr una imagen sin el efecto aurora primero se crea una imagen llámese (*image_circle*) de las mismas dimensiones de la imagen resultante, con fondo negro y un círculo blanco cuyo radio es el 90% del radio de la retina de la imagen resultante, para ello se utiliza el método *circle* [14] de la librería OpenCV. Finalmente se realiza una multiplicación entre *image_mapped_50* y *image_circle*, al mismo tiempo se mapean los pixeles del fondo de la imagen al 50% de la escala de gris haciendo uso de la fórmula 2. Se obtiene una imagen igual a *image_mapped_50* pero con el radio de la retina en la imagen reducido al 90%.

Fórmula 2. Mapeo de los pixeles del fondo de la imagen al 50% de la escala de gris.

$$(\text{image\_mapped\_50} * \text{image\_circle} + 128 * (1 - \text{image\_circle}))$$

4.4.4 Redimensionamiento de la Imagen a 512x512x3

El último paso del pre-procesamiento es el redimensionamiento de las imágenes manteniendo la relación de aspecto. Las CNN reciben como entrada imágenes del mismo tamaño, por tal motivo este paso es indispensable para el correcto funcionamiento de la red neuronal a desarrollar en este proyecto. Normalmente se ingresan a las CNN imágenes con tamaños reducidos y no de alta calidad con el fin de agilizar tanto el pre-procesamiento de todas las imágenes como el entrenamiento de la red neuronal. Al trabajar con imágenes de fondo de retina se desea detectar detalles que son muy pequeños por lo cual el tamaño de la imagen debe ser considerable, para entrenar la red neuronal se utilizarán imágenes con dimensiones 512x512.

Al ser la forma de la imagen deseada un cuadrado y la mayoría de las imágenes en la base de datos son rectangulares se procede a realizar un relleno con fondo de color que corresponde al 50% de la escala de gris (128). En este procedimiento se determina en qué sentido se debe rellenar la imagen con fondo ya sea (izquierda/derecha o arriba/abajo) al evaluar la diferencia entre las dimensiones

[14] Open Source Computer Vision.[en línea] OpenCV-Python [Consultado 1 de enero de 2022] Fuente: https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html

(height, width). Posteriormente se emplea el método *copyMakeBorder*[15] de la librería OpenCV para agregar el relleno de tal manera que las dimensiones de la imagen resultante son de 512x512 pixeles.

Otro problema que se enfrentará es la cantidad de imágenes disponibles para entrenar el sistema Deep Learning. Tener suficientes datos de entrenamiento es la clave para entrenar una red neuronal con éxito; desafortunadamente, este requisito extraña vez se cumple en la mayoría de las aplicaciones de redes neuronales. Para aplicaciones de imágenes médicas, la falta de datos es más significativa debido al costo de las anotaciones y al desequilibrio en la aparición de enfermedades. Para mitigar la escasez de datos y utilizar plenamente los datos disponibles, se deben realizar ciertas técnicas de aumento de datos en el proyecto como: voltear la imagen horizontalmente, voltear la imagen verticalmente, girar aleatoriamente la imagen en un rango de grados, acercar o alejar al azar en un rango especifico, distorsionar la imagen al azar.

[15] Open Source Computer Vision.[en línea] OpenCV-Python [Consultado 4 de enero de 2022] Fuente: https://docs.opencv.org/3.4/dc/da3/tutorial_copyMakeBorder.html

4.5 RED NEURONAL CONVOLUCIONAL

Uno de los aspectos fundamentales para escoger las CNN es la capacidad de automáticamente identificar y extraer las características más relevantes en una imagen. Sus filtros son modificados mediante aprendizaje supervisado, de esta manera logran adaptarse al problema que se desea resolver. Esta particularidad de las CNN permite al ser humano saltar el proceso de identificación y extracción de características que es necesario para entrenar una red no convolucional. Por lo tanto, para desarrollar el proceso de identificación de signos de DR en la retina puede ser demasiado complejo, por este motivo se utilizará una CNN como modelo de solución.

Aunque las redes neuronales convolucionales han sido popularmente usadas para analizar imágenes, también son usadas para otro tipo de análisis de datos o problemas de clasificación. Las CNN son redes neuronales artificiales capaces de detectar y extraer patrones de las imágenes y dar sentido a estos, de esta manera aprende directamente de los datos, sin necesidad de extraer características manualmente[16], lo cual, para un humano la identificación y clasificación de signos de Retinopatía Diabética en la retina es un proceso complejo e incierto. Esta particularidad es lo que hace que las CNN sean tan útiles en el análisis de las imágenes. Las CNN tienen capas ocultas llamadas capas convolucionales, de igual manera pueden tener otro tipo de capas en su arquitectura, sin embargo, la base de las CNN son las capas convolucionales.

Normalmente, se utiliza una red que ha sido previamente entrenada haciendo uso de la metodología transferencia del aprendizaje lo cual es mucho más rápido y fácil que entrenar una red desde cero, ya que permite entrenar modelos con mucho menos datos con sus correspondientes etiquetas, por lo tanto, requiere menos poder computacional para su entrenamiento. La arquitectura de la red neuronal no tiene una metodología estandarizada, ya que el número de parámetros que la red debe tener, dependen de la complejidad de las imágenes a clasificar, es por esto por lo que la cantidad de capas, filtros, tipos de funciones de activaciones etc. se realiza de forma empírica.

Para este proyecto se decide entrenar una CNN desde cero, teniendo como referencia una arquitectura exitosa en la clasificación de DR. Para esto, se toma como arquitectura inicial la arquitectura propuesta por los participantes que ocuparon el segundo lugar en la competencia *Diabetic Retinopathy Detection*[17] de Kaggle, los cuales utilizaron una Deep CNN, obteniendo un coeficiente de

[16] https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html
[17] https://www.kaggle.com/c/diabetic-retinopathy-detection/overview

$K_w = 0.845$. Por esta razón, se toma como referencia inicial la arquitectura propuesta por Mathias Anthony y Stephan Brüggeman, la cual se muestra en la Tabla 3.

Para el entrenamiento de la CNN se utiliza una GPU Nvidia GeForce GTX 1660Ti y el desarrollo se realiza haciendo uso de la librería de Deep learning Keras con backend tensorflow. La metodología utilizada para encontrar el mejor modelo de arquitectura fue el ensayo y error, observando el comportamiento tanto de las gráficas de entrenamiento como de las gráficas de validación, se solucionan problemas como el *overfitting,* de igual manera se varían tanto el tipo como la cantidad de capas, además de utilizar diferentes tipos de funciones de activación para observar el comportamiento del modelo, así mismo se utiliza *Data augmentation* para incrementar los datos de las clases en que se encontraban menor número de muestras.

Tabla 3. Arquitectura de la CNN propuesta por Mathias Anthony y Stephan Brüggeman.

		units	filter	stride	size
1	Input				448
2	Conv	32	5	2	224
3	Conv	32	3		224
4	MaxPool		3	2	111
5	Conv	64	5	2	56
6	Conv	64	3		56
7	Conv	64	3		56
8	MaxPool		3	2	27
9	Conv	128	3		27
10	Conv	128	3		27
11	Conv	128	3		27
12	MaxPool		3	2	13
13	Conv	256	3		13
14	Conv	256	3		13
15	Conv	256	3		13
16	MaxPool		3	2	6
17	Conv	512	3		6
18	Conv	512	3		6
19	RMSPool		3	3	2
20	Dropout				
21	Dense	1024			
22	Maxout	512			
23	Dropout				
24	Dense	1024			
25	Maxout	512			

Donde, el término *units* hace referencia al número de filtros en cada capa de convolución de la CNN. El término *filter* hace referencia al tamaño del filtro usado en cada capa de convolución. Finalmente, el término *stride* hace referencia al número de pixeles que se desplaza el filtro en cada paso.

Fuente: Mathis Anthony, Stephan Brüggeman. Kaggle Diabetic Retinopathy Detection Team o_O solution [en línea]. [consultado 04 de Febrero de 2022]. Disponible en Internet: https://github.com/sveitser/kaggle_diabetic/blob/master/doc/report.pdf

5. RESULTADOS

5.1 PRE-PROCESAMIENTO DE IMÁGENES

En las siguientes páginas se presentan los resultados obtenidos durante el pre-procesamiento de las imágenes que dispone la base de datos Kaggle con las cuales se entrenará la red neuronal.

Para lograr el pre-procesamiento de miles de imágenes se inicia desarrollando el algoritmo para procesar una imagen a la vez, en donde se siguieron los pasos descritos en la sección 4.4. En la Figura 6 se muestra una imagen de fondo de retina original etiquetada como *1982_right* con nivel de retinopatía diabética *0.*

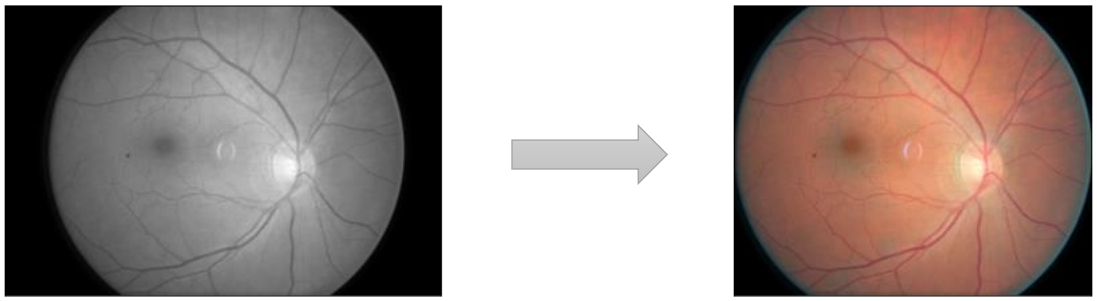
Figura 6. Imagen Original a pre-procesar.

$$Dimensiones = (2595, 3888, 3)$$

El primer paso en el pre-procesamiento realizado es la reducción del fondo de color negro de la imagen que está presente en todas las imágenes de la base de datos, esto con el fin de obtener una imagen en donde la mayoría de los pixeles pertenecen a la imagen de la retina y sus características, para facilitar este proceso ya que la imagen original es RGB con dimensiones $HxWx3$ en donde, H y W son el alto y ancho de la imagen en pixeles y el último término hace referencia al número de canales de la imagen. Se convierte la imagen a escala de grises lo cual da como resultados una imagen con dimensiones $HxWx1$, sin embargo, la reducción del fondo de color negro se aplica a la imagen original como se describe en la sección 4.4.1.

Figura 7. Imagen original a escala de grises e imagen con fondo reducido.

$Dimensiones = (2595, 3888)$ $Dimensiones = (2592, 3171, 3)$



Se puede observar en las imágenes anteriores que el fondo de color negro presente en la imagen a la izquierda de la Figura 7 se ha reducido drásticamente y ahora la mayoría de los pixeles conforman la retina en donde se encuentran las características que se requieren como entrada de la red neuronal como se muestra en la imagen a la derecha de la Figura 7. Es de resaltar que el tamaño de la imagen se ha reducido lo cual implica un mejoramiento en el rendimiento del sistema al entrenar la CNN con imágenes de tamaños menores.

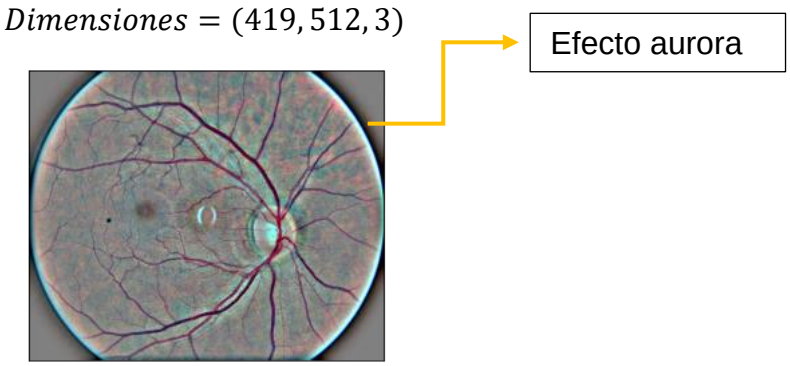
Una vez la reducción del fondo de la imagen a finalizado se procede a escalar el radio, como se sabe, las imágenes de la base de datos kaggle tienen diferentes tamaños por lo tanto el radio de la retina es diferente en todas las imágenes, con el objetivo de que todas las imágenes tengan radios similares se continua con el procedimiento denominado escalamiento del radio (sección 4.4.2). El resultado de este procedimiento se puede detallar en la Figura 8. Nótese que las dimensiones de todas las imágenes una vez terminado este procedimiento serán similares, por consiguiente, el radio de estas se asemejará.

Figura 8. Imagen con radio escalado.

$Dimensiones = (419, 512, 3)$

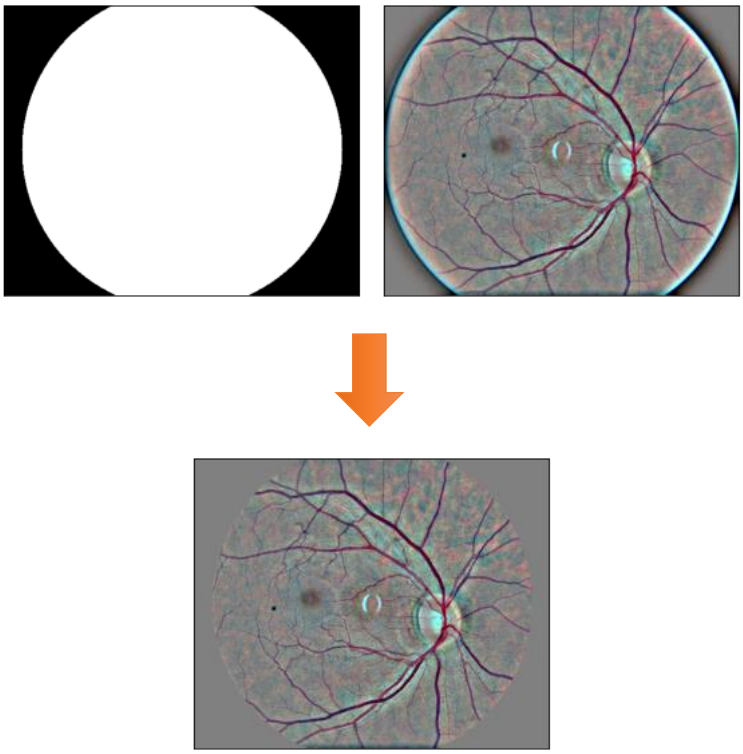
Con la finalidad de que le sea más fácil a la red neuronal aprender las características únicas de cada imagen se ha mapeado la imagen al 50% de gris, este proceso da como resultado una imagen con el fondo de la imagen y el fondo de la retina en coloraciones similares (sección 4.4.3). Sin embargo, se genera un efecto aurora alrededor de la retina como se muestra en la Figura 9.

Figura 9. Imagen mapeada al 50% de gris.



La red neuronal aprenderá las características más relevantes de las imágenes, por tal motivo es indispensable remover el efecto aurora producido después de realizar el procedimiento del mapeo al 50% de gris, de lo contrario la red neuronal aprenderá una característica errónea de las imágenes pre-procesadas. Se utiliza una imagen de fondo negro con un círculo blanco de igual diámetro al de la retina y se multiplica por la imagen de la Figura 9 siguiendo los pasos de la sección 4.4.3.1. Obteniendo como resultado una imagen sin características no deseadas como se observa en la Figura 10.

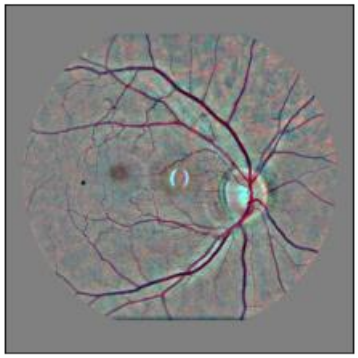
Figura 10. Imagen sin efecto aurora.



En este paso se tiene una imagen pre-procesada en la cual las características de la retina resaltan notablemente, sin embargo, el tamaño de la imagen es de (419, 512, 3) por lo que es necesario redimensionar la imagen para obtener el tamaño deseado que es (512, 512, 3). Para esto, se adiciona fondo a la imagen del color que corresponde al 50% de la escala de gris (128) proceso que se explica detalladamente en la sección 4.4.4. La imagen final luego de completar el pre-procesamiento es una imagen RGB de dimensiones 512x512 la cual se muestra en la Figura 11.

Figura 11. Imagen final pre-procesada.

$$Dimensiones = (512, 512, 3)$$



A continuación, se muestra en la Figura 12 la distribución de los distintos tonos de una imagen al completar el pre-procesamiento. Como se puede observar la mayoría de los pixeles tienen tonalidades correspondientes al 50% de la escala de gris (128) y los demás pixeles están homogéneamente distribuidos hacia la izquierda y derecha de este pico.

El histograma se obtiene convirtiendo la imagen a un arreglo aplanado contiguo, haciendo uso de la función *ravel* [18] la cual toma como entrada la imagen que es un arreglo 3D y devuelve un arreglo de 1D con todos los elementos del arreglo de entrada con el mismo tipo. Finalmente, en la Figura 13 se detalla cada una de las etapas del pre-procesamiento aplicado a una imagen de fondo de retina.

[18] NumPy [en línea], [Consultado 4 de enero de 2022] Fuente: https://numpy.org/doc/stable/reference/generated/numpy.ravel.html

Figura 12. Histograma de una Imagen Preprocesada.

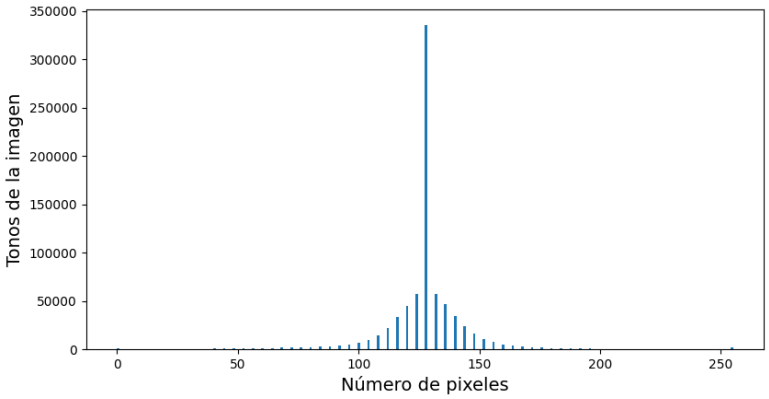
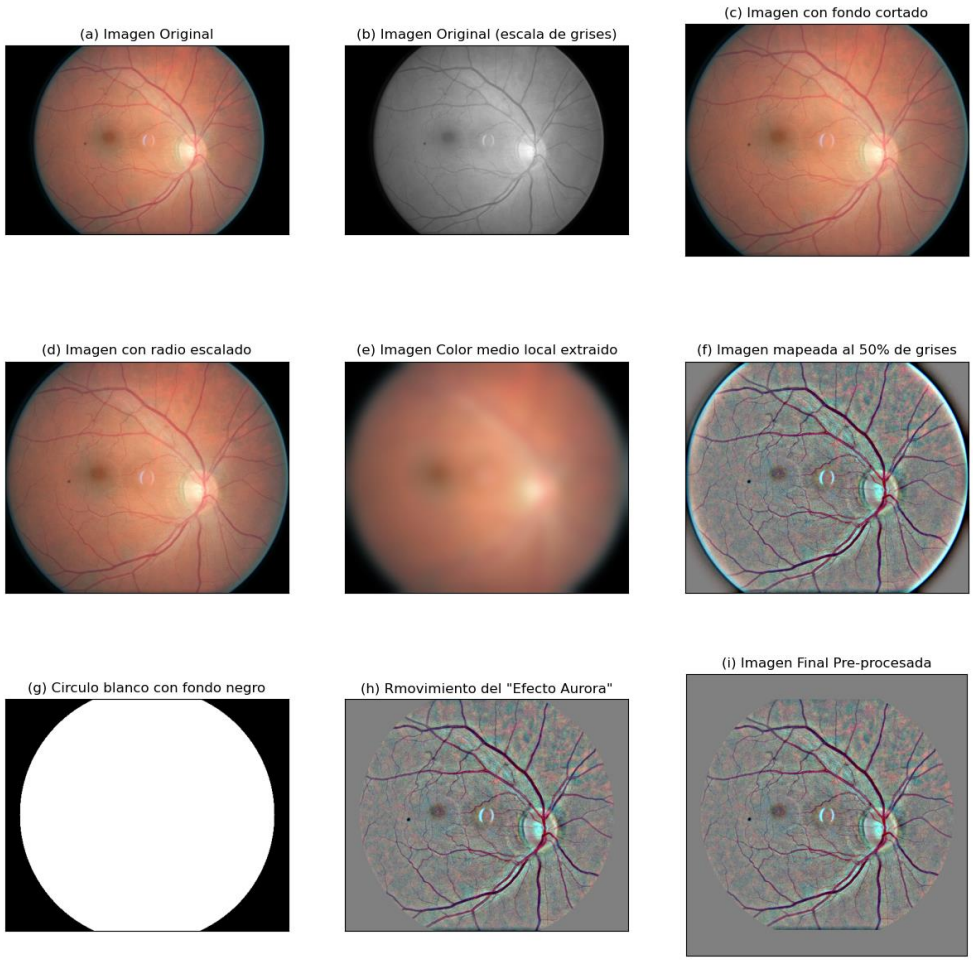


Figura 13. Resultados del pre-procesamiento para una imagen de la base de datos Kaggle.



5.2 BULK PRE-PROCESSING

Una vez el pre-procesamiento de una imagen ha sido exitoso se realiza el algoritmo para realizar el *bullk pre-processing* de todas las imágenes presentes en la base de datos de kaggle. Para ello, se crea un arreglo el cual contiene todos los nombres de las imágenes con su extensión *.jpeg*. Estos datos se obtienen de la carpeta en donde se encuentran almacenadas las imágenes originales, además se crea un arreglo que contiene todos los niveles de retinopatía diabética, estos datos se obtienen de la tabla en donde se encontraban el nombre de la imagen con su respectiva clasificación como se muestra en la Figura 14.

Figura 14. Muestra de arreglos creados con los nombres de las imágenes y el nivel de retinopatía correspondiente.

Niveles de Retinopatía Diabética: [0 0 2 0 0 0 1 1 0 0 0 0 0 0 0 0]

Nombre de las imágenes: ['9979_left.jpeg', '9979_right.jpeg', '9980_left.jpeg', '9980_right.jpeg', '9984_left.jpeg', '9984_right.jpeg', '9992_left.jpeg', '9992_right.jpeg', '9993_left.jpeg', '9993_right.jpeg', '9996_left.jpeg', '9996_right.jpeg', '9998_left.jpeg', '9998_right.jpeg', '9999_left.jpeg', '9999_right.jpeg']

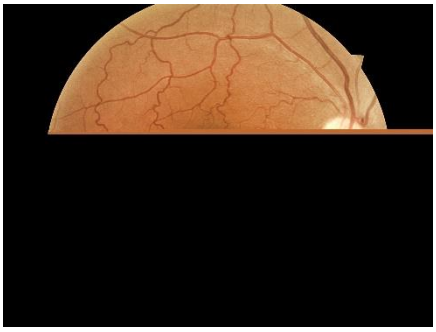
	A	B
1	Nobre de la Imagen	Nivel de Retinopatía Diabética
2	9979_left	0
3	9979_right	0
4	9980_left	2
5	9980_right	0
6	9984_left	0
7	9984_right	0
8	9992_left	1
9	9992_right	1
10	9993_left	0
11	9993_right	0
12	9996_left	0
13	9996_right	0
14	9998_left	0
15	9998_right	0
16	9999_left	0
17	9999_right	0

Luego de obtener estos arreglos, se recorre el arreglo denominado “Nombre de las imágenes” para así lograr realizar el pre-procesamiento explicado en la sección 4.4 a cada imagen. Las imágenes no fueron procesadas todas a la vez con el fin de evitar sobrepasar la capacidad máxima de la memoria RAM (24GB), se procesaron 500 imágenes a la vez hasta completar las 35126 imágenes de entrenamiento

presentes en la base de datos Kaggle. Durante este procedimiento no fue posible completar el pre-procesamiento de todas las imágenes ya que algunas de ellas eran completa o parcialmente negras como se muestra en la Figura 15.

Figura 15. Imágenes descartadas en el pre-procesamiento.

492_right.jpeg



766_left.jpeg

Fue posible identificar estás imágenes gracias a que dentro del desarrollado del algoritmo para el *bullk pre-processing* se implementó un bloque para el tratamiento de errores haciendo uso de *Try Except*. Una vez finalizado el *bullk pre-processing* se identificó todas las imágenes a las que no fue posible realizarles el pre-procesamiento y se retiraron de la base datos con sus respectivas parejas, por ejemplo, 492_left.jpeg y 492_right.jpeg. De la misma manera se removieron del archivo que contiene el nombre de las imágenes con su correspondiente nivel de Retimopatía Diabética.

Figura 16. Muestra de los resultados del *bullk pre-processing.*

```
Image Not Processed: 492_right.jpeg
Numero de Imagenes contenidas en el arreglo: 499
Dimensiones de las Imagenes: (512, 512, 3)
Image Not Processed: 766_left.jpeg
Numero de Imagenes contenidas en el arreglo: 499
Dimensiones de las Imagenes: (512, 512, 3)
Numero de Imagenes contenidas en el arreglo: 500
Dimensiones de las Imagenes: (512, 512, 3)
Image Not Processed: 1986_left.jpeg
Numero de Imagenes contenidas en el arreglo: 499
Dimensiones de las Imagenes: (512, 512, 3)
Numero de Imagenes contenidas en el arreglo: 500
Dimensiones de las Imagenes: (512, 512, 3)
Numero de Imagenes contenidas en el arreglo: 500
Dimensiones de las Imagenes: (512, 512, 3)
Image Not Processed: 3829_left.jpeg
Numero de Imagenes contenidas en el arreglo: 499
Dimensiones de las Imagenes: (512, 512, 3)
Numero de Imagenes contenidas en el arreglo: 500
Dimensiones de las Imagenes: (512, 512, 3)
Numero de Imagenes contenidas en el arreglo: 500
```

El proceso denominado *bullk pre-processing* se realizó nuevamente una vez las imágenes que no se pre-procesasaron anteriormente fueron retiradas de la base de datos.

Figura 17. Imágenes antes y después de pasar por la etapa de pre-procesamiento.

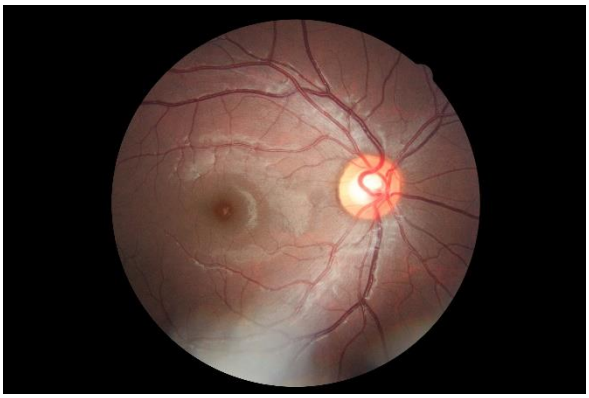
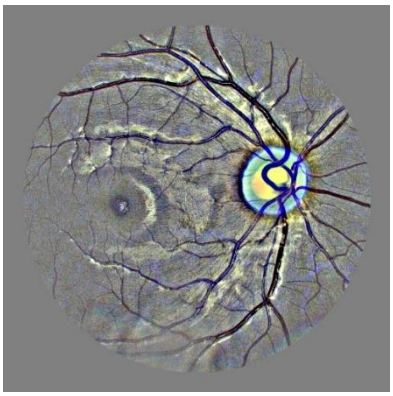
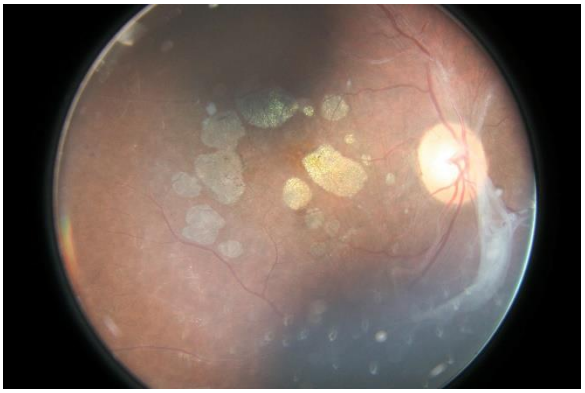
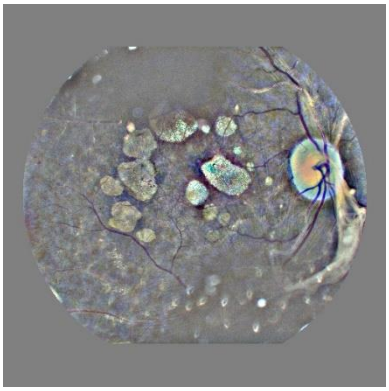
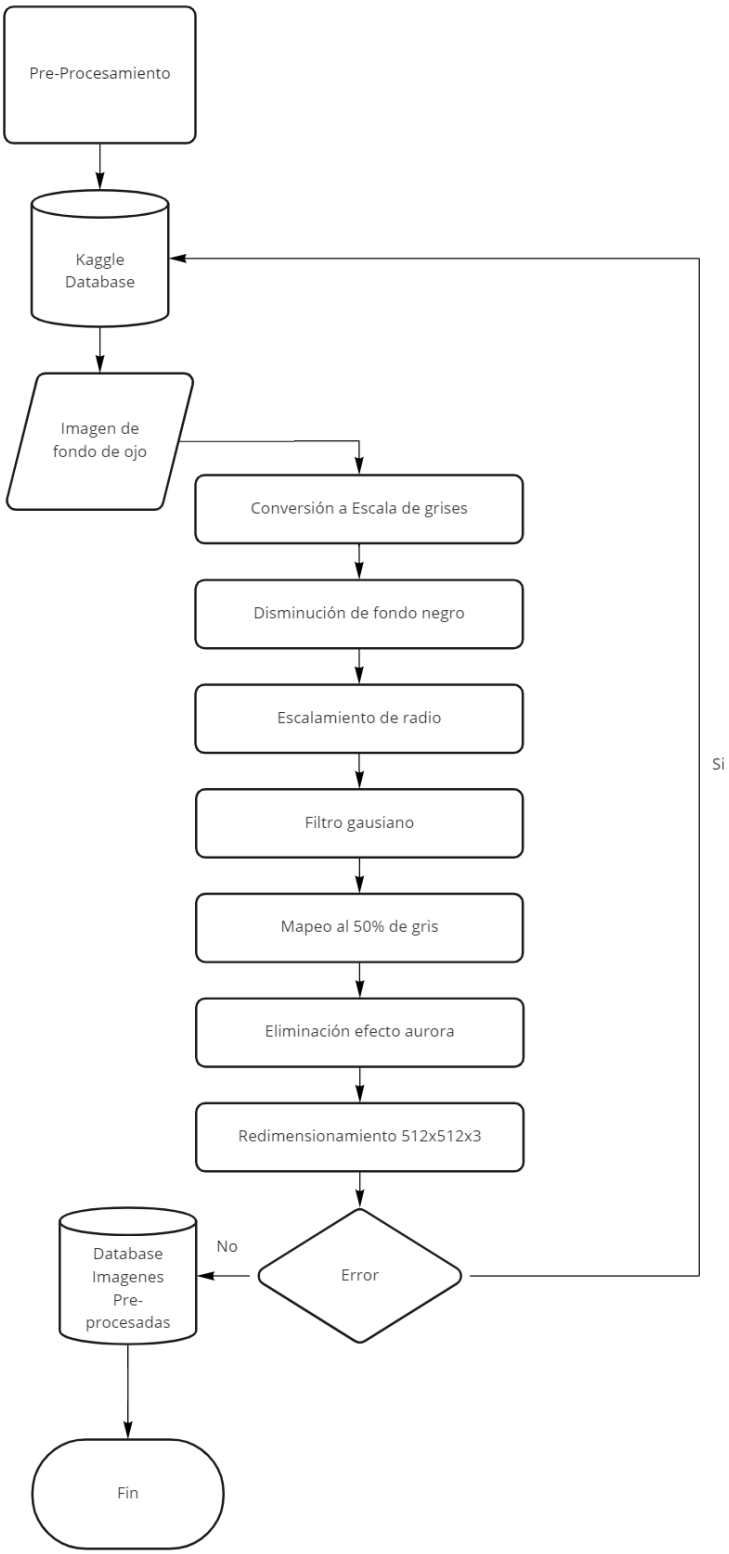
Imagen Original	**Imagen Pre-procesada**
910_right.jpeg Nivel de Retinopatía Diabética: 0	
	
1084_left.jpeg Nivel de Retinopatía Diabética: 4	
	

Figura 18. Diagrama de flujo del pre-procesamiento.



5.3 RED NEURONAL CONVOLUCIONAL

La primera prueba se realizó con un modelo inicial el cual se entrenó por 50 épocas, se implementó sin *data augmentation* y sin tener en cuenta la métrica de evaluación Kappa cuadrático ponderado, al revisar los resultados se observó un sobre entrenamiento (*overfitting*) considerable de la red neuronal. Se hizo uso de TensorBoard[19] que es el kit de herramientas de visualización de TensorFlow para poder visualizar y entender el comportamiento de la red neuronal durante el entrenamiento como se muestra en la Figura 19. En la gráfica de la exactitud vs épocas (a), se logra observar que la curva tanto de entrenamiento como de validación crecen rápidamente alcanzando un valor de exactitud aproximadamente de 99% y 67% respectivamente, lo cual indica que la red está memorizando en lugar de aprender las verdaderas características de las imágenes de fondo de retina suministradas a la red neuronal para su entrenamiento.

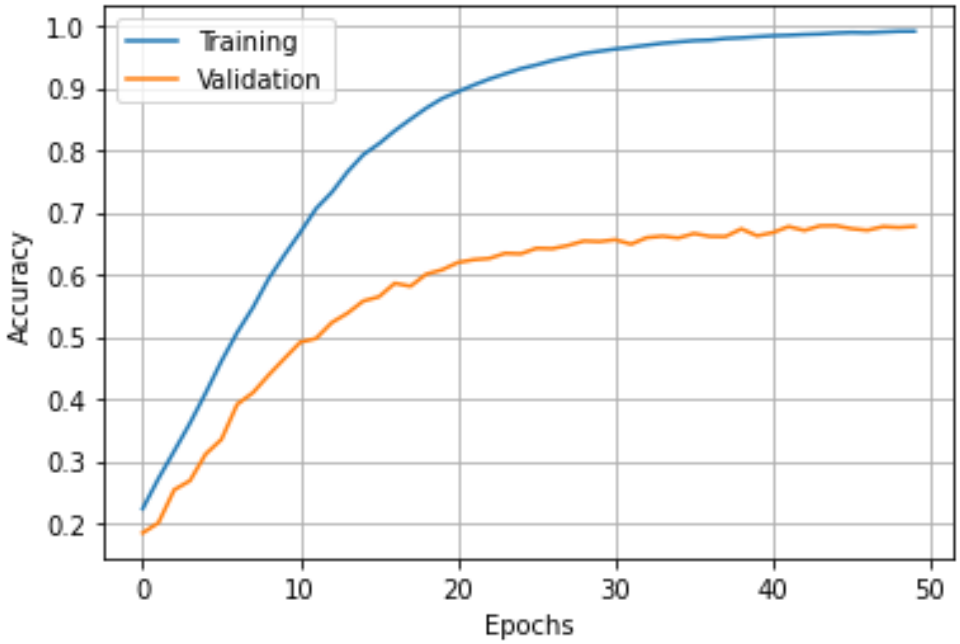
Al observar estás graficas se determinó que las posibles causas del *overfitting* de la red neuronal se debía a dos posibles razones, ya que la distribución de las clases (niveles de Retinopatía Diabética) de la base de datos de kaggle tanto para los datos de entrenamiento como para los datos de validación no es uniforme como se presenta en la Tabla 4 y en la Tabla 5 respectivamente. En estás tablas se puede observar que la clase 0 (No DR) contiene aproximadamente el 74% de las imágenes mientras que las clases 3 (Severo) y 4 (DR Proliferativa) contienen aproximadamente el 2% de las imágenes cada una. Se decidió añadir un método de muestreo de acuerdo con la probabilidad y así alimentar a la red neuronal convolucional con conjuntos de datos de tal manera que las clases sean uniformes. Este método de muestreo se implementó cargando los arreglos que contenían las imágenes (un arreglo contenía 500 imágenes), una vez cargado los arreglos en la memoria RAM se obtenían de manera aleatoria 16 imágenes (tamaño del batch) de tal manera que la imagen se seleccionaba con una probabilidad inversa al porcentaje del total de imágenes pertenecientes a dicha clase para los datos de entrenamiento. Este método generaba un arreglo en donde las imágenes pertenecientes a una clase con menor cantidad de imágenes en el set de entrenamiento eran escogidas con mayor probabilidad y viceversa. Este método también se utilizó para los datos de validación. En la Figura 20 se presenta la distribución de clases por batch antes y después de usar el método de muestreo.

[19] Tensorflow [en línea] [Consultado 20 de Enero de 2022] https://www.tensorflow.org/tensorboard?hl=es-419

Una segunda solución ante el problema de *overfitting* es la implementación de *data augmentation*. El aumento de datos permite crear nuevos datos a partir de datos existentes mediante la aplicación de algunas transformaciones. Es importante ya que incrementa el tamaño del conjunto de datos de entrenamiento y validación. Entrenar la red neuronal con más datos conduce a lograr una mayor precisión. Por esta razón se realizó la rotación de todas las imágenes de entrenamiento y de validación utilizando el método *random_rotation*[20] de la librería de keras con un ángulo aleatorio entre 0º y 360º, además se implementaron operaciones de flip (voltear las imágenes) izquierda/derecha, arriba/abajo o en ambas direcciones como se muestra en la Figura 21. Para lograr alimentar la red neuronal se implementó un *Custom Data Generator* el cual contenía el método de muestreo y el *data augmentation*.

Figura 19. Graficas de la exactitud y la pérdida del modelo inicial entrenado durante 50 épocas.

(a) Exactitud vs. Épocas



[20] Tensorflow [en línea] [Consultado 21 de Enero de 2022] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/random_rotation

(b) Perdida vs. Épocas

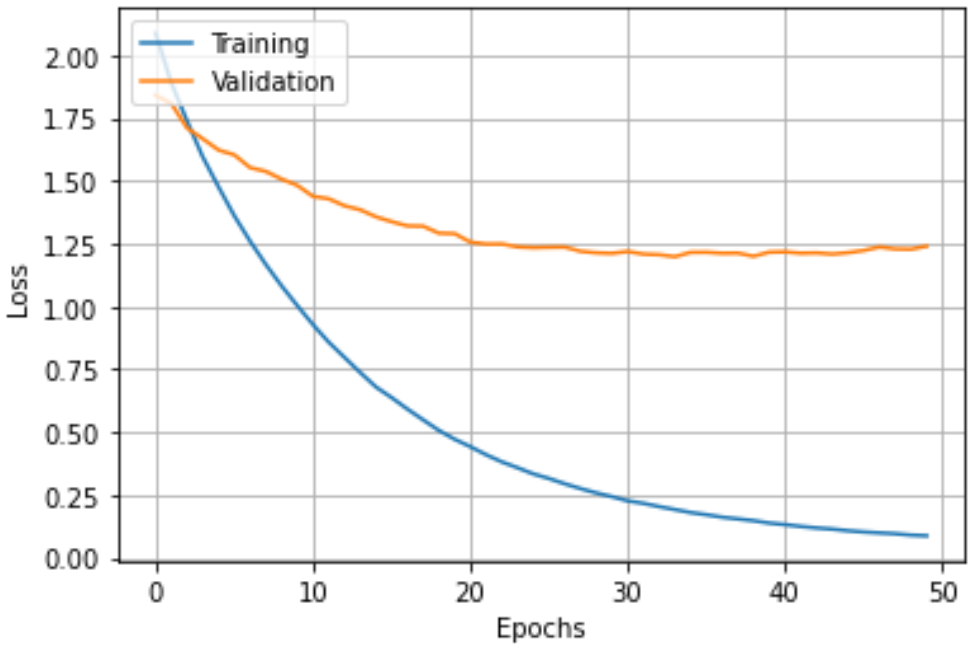


Tabla 4. Distribución de clases del set de entrenamiento de la base de datos de Kaggle.

Clases	Distribución de los datos	Cantidad de datos
0	73.5%	25803
1	6.9%	2432
2	15.1%	5290
3	2.5%	872
4	2%	708

Tabla 5. Distribución de clases del set de validación de la base de datos de Kaggle.

Clases	Distribución de los datos	Cantidad de datos
0	73.8%	39516
1	7%	3761
2	14.7%	7853
3	2.3%	1214
4	2.2%	1206

Figura 20. Distribución de clases por batch.

(a) Sin muestreo

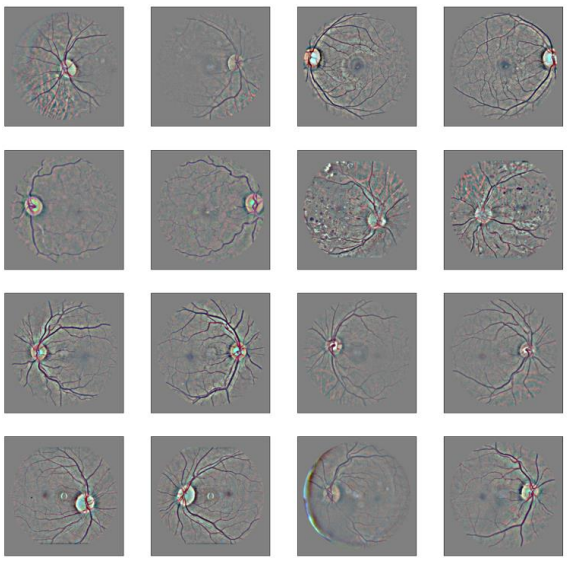
```
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.]]
```

(b) Con muestreo

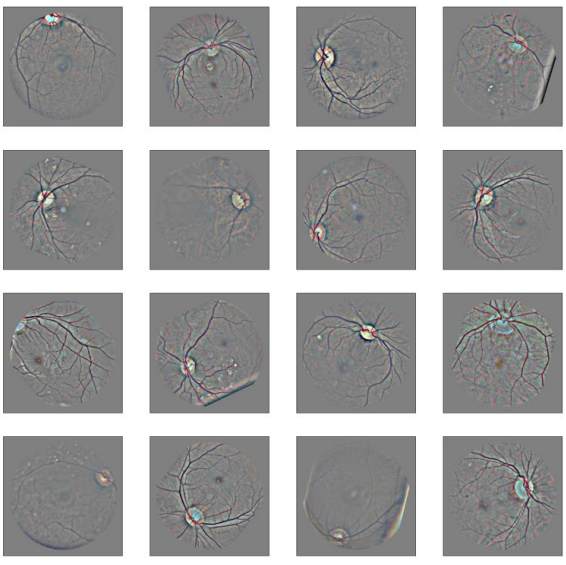
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0.]]
```

Figura 21. Batch de Imágenes antes y después de aplicar data augmentation.

(a) Imágenes sin data augmentation



(b) Imágenes con data augmentation



5.4 ARQUITECTURA FINAL

La Arquitectura de red neuronal convolucional final utilizada para la clasificación de Retinopatía Diabética se muestra en la Figura 22. En la Figura 22. (a) Etapa de aprendizaje y extracción de características, se puede observar que la red neuronal tiene 13 capas convolucionales y 6 capas de Pooling. Las capas convolucionales utilizan el método *Conv2D*[21], el tamaño de los filtros también llamados kernels en estas capas es de $3x3$, además se utilizó *padding* = 'same' lo cual da como resultado un relleno con zeros uniformemente a la izquierda/derecha o arriba/abajo de la entrada. Al configurar *padding* = 'same' y *strides* = 1, la salida tiene el mismo tamaño que la entrada. Como función de activación de las capas convolucionales se utilizó *ReLU*[22] del inglés (Rectified Linear Unit) es la función de activación más utilizada en los modelos de Deep Learning hoy en día, dicha función devuelve 0 si recibe una entrada negativa, pero para cualquier valor positivo devuelve ese valor, se puede representar con la siguiente expresión $y = \max(0, x)$. Cada una de las capas convolucionales es seguida por capas *BatchNormalization*[23], estas capas cumplen una función importante al hacer que la red neuronal se entrené más rápido, es decir, menos épocas necesarias para alcanzar una exactitud más alta y que sea más estable. Adicionalmente, en cada una de las capas de pooling, se hizo uso del método *MaxPooling*[24], es común añadir periódicamente una capa Pooling entre capas convolucionales en una arquitectura de red neuronal convolucional. Su principal objetivo es reducir progresivamente el tamaño espacial de la imagen para reducir la cantidad de parámetros y el poder computacional requerido en el entrenamiento, por lo tanto, controlar también el *overfitting.* Se utilizó un tamaño de kernel de $2x2$ aplicado con un *stride* de $2x2$, lo cual reduce tanto el ancho como el alto de la imagen a la mitad al pasar por cada capa de Pooling.

En la Figura 22 (b) Etapa de Clasificación, se adicionó una capa Flatten, la cual convierte los mapas de características de la capa anterior a esta en un vector plano. En la arquitectura implementada los mapas de características antes de ingresar a la capa Flatten tienen como dimensiones $7x7x128$ dando como resultado un vector de 6272 elementos. Continuamente se implementaron dos capas completamente

[21] Tensorflow [en línea] [Consultado 23 de Enero de 2022] https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
[22] Keras: the Python deep learning API [en línea] [Consultado 23 de Enero de 2022] https://keras.io/api/layers/activation_layers/relu/
[23] Tensorflow [en línea] [Consultado 24 de Enero de 2022] https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization
[24] Tensorflow [en línea] [Consultado 25 de Enero de 2022] https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D

conectadas (FC) del inglés Fully Connected layers, para la primera FC1 se utilizó la función de activación ReLU, seguida por una capa de *BatchNormalization* y para la última FC2 se utilizó la función de activación *softmax*, la cual da como resultado un vector en donde los elementos están en el rango de 0 a 1 y suman 1, este resultado puede interpretarse como una distribución de probabilidad. También se agregaron capas de Dropout, con un porcentaje de 25%, las cuales establecen aleatoriamente las unidades de entrada como 0 en cada paso durante el entrenamiento lo cual ayuda a evitar el *overfitting.*

Para la compilación del modelo se empleó el método *compile*[25] en el cual se utilizó como optimizador durante el entrenamiento de la red neuronal *adadelta* (Adaptive Learning Rate Method), como función de perdida se utilizó *WeightedKappaLoss*[26], al entrenar una CNN en donde existen desequilibrios de clases el uso de esta función de perdida es ideal para penalizar las clasificaciones erróneas con el fin de obtener un mejor aprendizaje de la red neuronal. Como métrica de evaluación se utilizó *CohenKappa*[27]*,* la cual es una estadística que mide la confiabilidad o el acuerdo entre dos evaluadores o métodos al evaluar elementos categóricos, esta métrica recibe como argumento *weightage,* este término hace referencia a la importancia asignada a diferentes elementos, factores o componentes dentro de un sistema, al cual se le dio el valor de 'quadratic', ponderación a considerar para calcular las estadísticas Kappa. Una vez la compilación se realizó correctamente, se procedió a entrenar la red neuronal convolucional utilizando el método *fit*[28]*.* Este método acepta generadores tanto para el set de entrenamiento como para el set de validación, en cada uno de los C*ustom Data Generators* desarrollados, se implementó *Data augmentation* y el método de muestreo como se explica en la sección **Red Neuronal Convolucional**. El tamaño tanto del set de entrenamiento como de validación era demasiado grande por lo cual no fue posible cargar todos los datos al mismo tiempo en memoria, por esta razón, los datos se cargaron mediante *batches* haciendo uso de los C*ustom Data Generators* descritos anteriormente*.* El entrenamiento de la red neuronal fue realizado por la GPU, mientras que en paralelo la CPU se encargaba de llamar los *Custom Data Generators* necesarios para alimentar a la red continuamente. El tamaño del batch

[25] Keras: the Python deep learning API [en línea] [Consultado 15 de Febrero de 2022] https://keras.io/api/models/model_training_apis/#compile-method
[26] Tensorflow [en línea] [Consultado 16 de Febrero de 2022] https://www.tensorflow.org/addons/api_docs/python/tfa/losses/WeightedKappaLoss
[27] Tensorflow [en línea] [Consultado 16 de Febrero de 2022] https://www.tensorflow.org/addons/api_docs/python/tfa/metrics/CohenKappa
[28] Keras: the Python deep learning API [en línea] [Consultado 17 de Febrero de 2022] https://keras.io/api/models/model_training_apis/#fit-method

escogido fue de 16, ya que se ha observado que para grandes tamaños de batches hay una degradación significativa en la calidad del modelo. La red neuronal fue entrenada con 53.550 imágenes para el entrenamiento de la CNN y se utilizaron aproximadamente 10% del total de imágenes (3.500) como validación. El total de parámetros que tenía la red neuronal era de 1’155.061 y se entrenó durante 150 épocas, su entrenamiento total tardó 38 horas.

Figura 22. Arquitectura final de la CNN.

(a) Etapa de aprendizaje y extracción de características

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 512, 512, 16)      448
_________________________________________________________________
batch_normalization (BatchNo (None, 512, 512, 16)      64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 512, 512, 16)      2320
_________________________________________________________________
batch_normalization_1 (Batch (None, 512, 512, 16)      64
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 255, 255, 32)      4640
_________________________________________________________________
batch_normalization_2 (Batch (None, 255, 255, 32)      128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 255, 255, 32)      9248
_________________________________________________________________
batch_normalization_3 (Batch (None, 255, 255, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 127, 127, 48)      13872
_________________________________________________________________
batch_normalization_4 (Batch (None, 127, 127, 48)      192
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 127, 127, 48)      20784
_________________________________________________________________
batch_normalization_5 (Batch (None, 127, 127, 48)      192
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 63, 63, 48)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 63, 63, 64)        27712
_________________________________________________________________
batch_normalization_6 (Batch (None, 63, 63, 64)        256
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 63, 63, 64)        36928
_________________________________________________________________
batch_normalization_7 (Batch (None, 63, 63, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 31, 31, 96)        55392
_________________________________________________________________
batch_normalization_8 (Batch (None, 31, 31, 96)        384
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 31, 31, 96)        83040
_________________________________________________________________
batch_normalization_9 (Batch (None, 31, 31, 96)        384
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 96)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 15, 15, 128)       110720
_________________________________________________________________
batch_normalization_10 (Batc (None, 15, 15, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
```

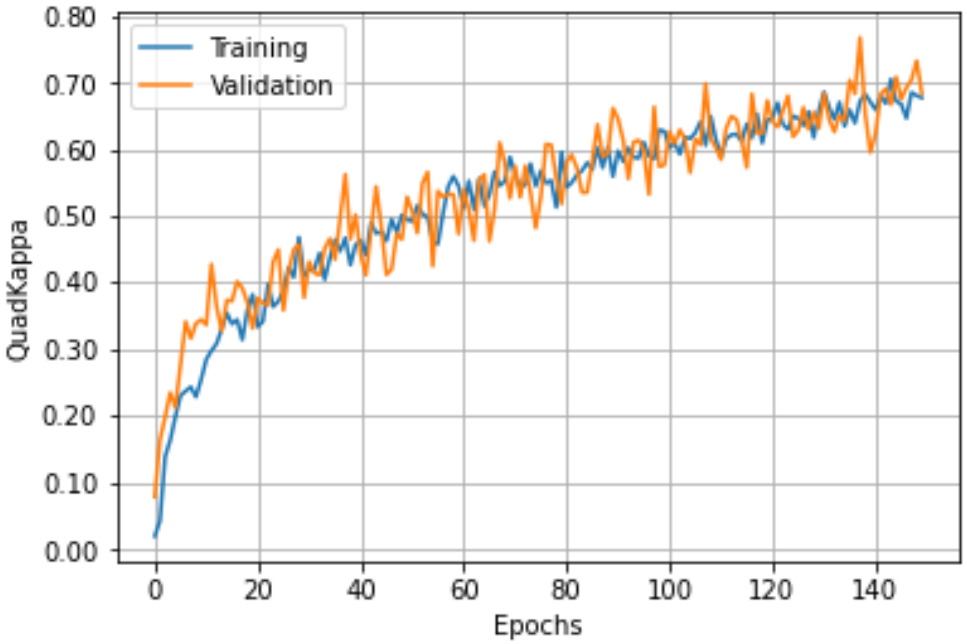
Figura 22. Arquitectura final de la CNN.

(b) Etapa de Clasificación.

dropout (Dropout)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 96)	602208
dropout_1 (Dropout)	(None, 96)	0
batch_normalization_11 (Batc	(None, 96)	384
dense_1 (Dense)	(None, 5)	485

Total params: 970,741
Trainable params: 969,269
Non-trainable params: 1,472

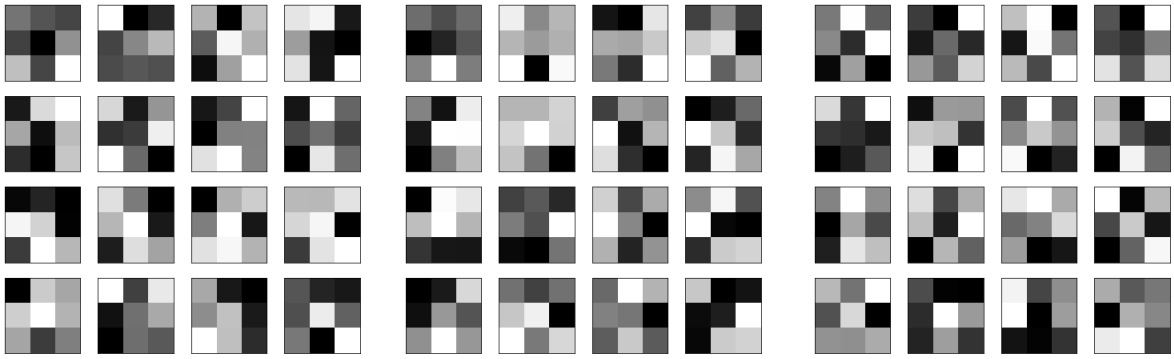
Figura 23. Grafica de la exactitud del modelo Final entrenado durante 150 épocas.



5.4.1 Filtros y salidas de la red neuronal Convolucional

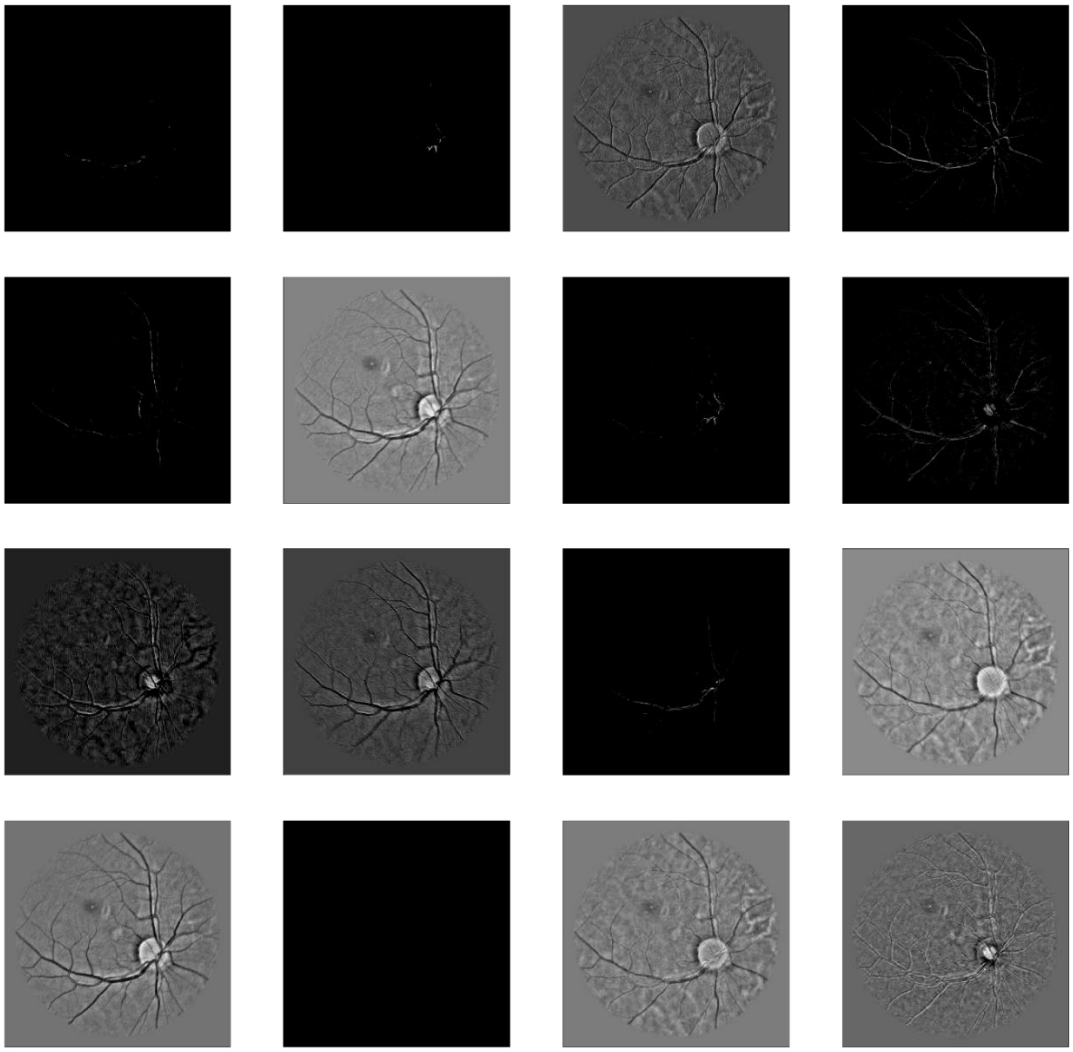
Se logró visualizar tanto los kernels de las capas convolucionales que fueron utilizados por la red neuronal convolucional, así como las imágenes de salida de cada una de ellas. Algunos de los filtros utilizados por la red neuronal en su primera capa convolucional se muestran en la Figura 24. La elección y la aplicación de los filtros la hace de forma automática el modelo, además las características aprendidas en cada capa convolucional varían significativamente. Se puede observar que cada uno de los 16 filtros es de tamaño $[3x3]$, además cada uno de los filtros es diferente logrando la extracción de diferentes características propias de las imágenes, para así aprender y conseguir clasificar en este caso los 5 diferentes niveles de Retinopatía Diabética en imágenes de fondo de retina.

Figura 24. Filtros usados por la CNN en la primera capa convolucional, Conv2D(16, (3,3), padding= 'same ', activation= 'relu ').



Se sabe con certeza que las capas iniciales capturan principalmente características de bajo nivel como lo son la dirección de la imagen (vertical/horizontal o diagonal), el color, etc. A medida que se añaden nuevas capas convolucionales, las CNN capturan características de más alto nivel que ayudan a diferenciar entre varias clases de imágenes. En la Figura 25 se muestran las imágenes de salida al pasar por la primera capa convolucional del modelo usado para el entrenamiento de la CNN, el cual se describe en la figura 22. En esta figura se puede observar que la primera capa convolucional utiliza 16 filtros en una imagen de entrada que ha sido pre-procesada con dimensiones $(512, 512, 3)$. Las imágenes resultantes muestran las características extraídas y aprendidas por la CNN en su primera capa convolucional, Intuitivamente, la red aprenderá las características que se activan cuando ven algún tipo de característica visual, como un borde de alguna orientación o una mancha de algún color en la primera capa.

Figura 25. Imágenes de salida de la primera capa convolucional implementada en la arquitectura final.



Se realiza la matriz de confusión con una batch de prueba de 500 imágenes. Se puede observar como se menciona anteriormente que las clases de las imágenes no están balanceadas por lo cuál existe una gran diferencia en el número de imágenes de una clase a otra.

En la Figura 26. Se puede observar que de 385 imágenes clasificadas como No DR el sistema predijo que 43 pertenecen al nivel de retinopatia diabetica Leve y 29 pertenecen al nivel de retinopatía diabetica Moderado. 28 imágenes se clasificaron como Leve, 4 se predijeron como No DR y 7 como Moderado. Para el nivel de retinopatia diabetica Moderado se predijeron 47 imágenes correctamente, 8 se predijeron como Leve y 11 como Severo. Durante la predicción del nivel Severo 7

predicciones fueron correctar, 2 se clasificaron como Moderado y 1 como DR proliferativa. Finalmente al clasificar las imágenes pertenecientes a DR Proliferativa 4 predicciones fueron correctas y 1 se clasificó como Severo.

En la Tabla 6. Se aprecian las metricas obtenidas de la matriz de confusión. La exactitud refleja el porcentaje de imágenes que el modelo ha acertado. La precisión evalua el acierto del modelo en la clasificación de cada clase y finalmente la exhaustividad informa sobre la cantidad que el modelo es capaz de clasificar.

Figura 26. Matriz de confusión

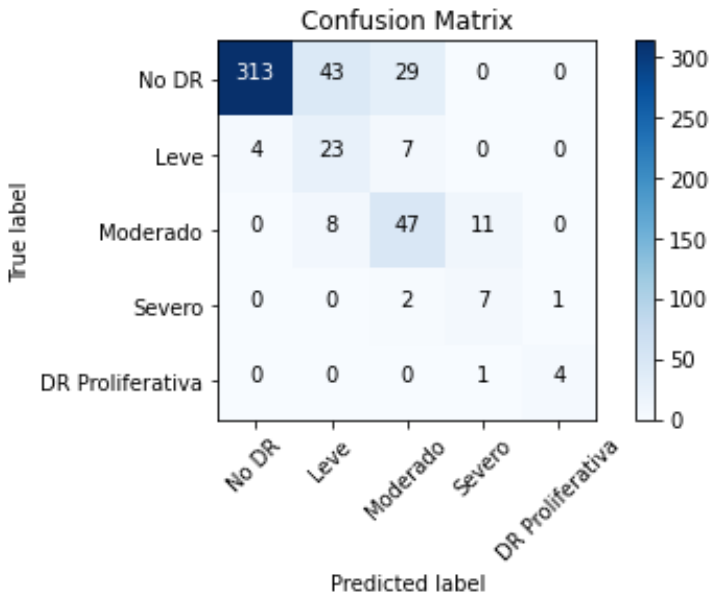


Tabla 6. Metricas de la matriz de confunsión por clases.

Metricas / Clases	Exactitud	Precisión	Exhaustividad
No DR	84.8 %	98.73 %	81.29 %
Leve	87.6 %	31.08 %	67.64 %
Moderado	88.6 %	55.29 %	71.21 %
Severo	97 %	36.84 %	70 %
DR Proliferativa	99.6 %	80 %	80 %

6. CONCLUSIONES

- Se propuso y se validó una aplicación desarrollada en el lenguaje de programación Python que fuese capaz de detectar y clasificar los cinco grados de Retinopatía Diabética en imágenes de fondo de ojo mediante el entrenamiento de una CNN, cumpliendo así el objetivo principal de esta tesis.

- Las redes neuronales convolucionales realizan la extracción de características necesarias para aprender diferentes patrones relevantes de las imágenes automáticamente, sin embargo, a través de este proyecto, se logró evidenciar que la etapa de pre-procesamiento es fundamental para suministrar las imágenes a la CNN, ya que, las imágenes de la base de datos de Kaggle tenían diferentes dimensiones a lo largo y ancho de la imagen, y la CNN acepta imágenes de entrada de iguales dimensiones, además, la red logra identificar y aprender patrones más fácilmente al suministrar imágenes estandarizadas, logrando un desempeño significativamente mejor.

- Es importante tener en cuenta los factores que pueden ocasionar que una red sufra de *overfitting* al momento del entrenamiento. En este proyecto se utilizaron diferentes técnicas para dar solución a este problema, tales como el muestreo de clases balanceadas, la adición de capas *Dropout* y *regularización.* A pesar de que se tenía una base de datos relativamente grande con alrededor de 53550 imágenes para entrenamiento, se utilizó *data augmentation,* rotando las imágenes en un rango de 360º, obteniendo una mejora significativa en el desempeño de la red neuronal convolucional, demostrando que el número de imágenes es de suma importancia cuando se requiere entrenar una red con imágenes de gran tamaño y características complejas.

- El modelo final con la arquitectura presentada en la Figura 23 tuvo un valor de kappa con los datos de prueba de 7.8 el cual es menor al obtenido por la CNN propuesta por Mathias Anthony y Stephan Brüggeman. se debe considerar que existen diferentes factores que influyen en el valor final obtenido de kappa, como lo es el ruido existente en la base de datos de Kaggle, en sus imágenes y en la clasificación de estas, a pesar es esto, el valor obtenido en la exactitud del modelo final evaluado por el Kappa

Cuadrático Ponderado de acuerdo con la fuerza del nivel de concordancia mostrado en la Tabla 2 es muy bueno.

- La red neuronal desarrollada en este proyecto obtuvo un Kappa de 7.8 como se expresa anteriormente este puntaje es considerado como bueno, por lo cual, la red neuronal convolucional aprendió a extraer los patrones más relevantes que presentan algún tipo de anomalía en las imágenes de fondo de ojo para clasificar retinopatía diabética, por esta razón la red neuronal convolucional podría llegar a detectar diferentes tipos de retinopatías, como lo es la retinopatía hipertensiva. Se debe tener en cuenta que cada tipo de retinopatía tiene una clasificación diferente de acuerdo con sus fases, por lo tanto, es necesario entrenar de nuevo la etapa de clasificación de la red neuronal convolucional y así permitir que lograse la clasificación de otras anomalías.

- Las técnicas de pre-procesamiento, data augmentation, y de entrenamiento utilizadas en este proyecto de grado y las usadas por Mathias Anthony y Stephan Brüggeman en la competencia Kaggle Diabetic Retinopathy Detection son diferentes. Mathias Anthony y Stephan Brüggeman únicamente removieron el fondo de la imagen y las cortaron con el fin de que fueran cuadrados de 128, 256 y 512 pixeles como técnicas de pre-procesamiento. Ellos utilizaron traslación, estiramiento, rotación y flipping (voltear) como técnicas de data augmentation. Una gran diferencia se encuentra en el proceso de entrenamiento de la CNN, Mathias Anthony y Stephan Brüggeman inicializaron y “pre-entrenaron” pequeñas arquitecturas con imágenes de 128 pixeles, luego utilizaron los pesos entrenados para (parcialmente) inicializar redes de tamaño intermedio las cuales fueron entrenadas con imágenes de 256 pixeles. Finalmente, repitieron este proceso para entrenar la CNN final con imágenes de 512 imágenes. Se logra evidenciar la relevancia en el proceso de entrenamiento de la CNN ante el pre-procesamiento de las imágenes. Un gran proyecto a futuro es combinar las técnicas de pre-procesamiento utilizadas en este trabajo de grado con las técnicas de entrenamiento utilizadas por Mathias Anthony y Stephan Brüggeman y complementar las técnicas de data augmentation con la finalidad de generar un mayor número de imágenes con las cuales se puede entrenar la CNN y buscar obtener una mayor exactitud en la clasificación de Retinopatía Diabética.

BIBLIOGRAFÍA

Anzola, N. (2016). *Máquinas de soporte vectorial y redes neuronales artificiales en la predicciòn del movimiento USD/COP spot intradiario. ODEON, 113-172.*

Aurélien, G. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc.*

Barrera, Jamie Areli-Toral. (s.f.). *Redes Neuronales*.

Pedro Ponce, Cruz. (2010). *Inteligencia artificial con aplicaciones a la ingeniería. Alfaomega.*

Toral Barrera, J. (s.f.). *Redes Neuronales.*

Giraldo Calderón, V. (2018). *Diseño de un sistema de seguimiento del ojo (pupila), usando técnicas de visión artificial. Neiva.*

García, A. (2012). *Inteligencia Artificial. Fundamentos, práctica y aplicaciones. Rc Libros.*

Nursel Yalçin, Seyfullah Alver, Necla Uluhatun. *Classification of retinal images with Deep Learning for early detection of diabetic retinopathy disease*. Disponible en: https://www.kaggle.com/c/diabetic-retinopathy-detection

ML Practicum: *Image Classification*. Disponible en: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Convolutional Neural Networks (CNNs/ConvNets). Disponible en: http://cs231n.github.io/convolutional-networks/

KARPATHY, ANDREJ. (Consultado 1 de abril de 2020). *Convolutional Neural Networks for Visual Recognition, Convolutional Neural Networs (CNNs / ConvNets)*. Stanford University. Disponible en internet: http://cs231n.github.io/convolutional-networks/

ORCHARD, ELIZABETH. (1 de Abril de 2020). *Diabetic Retinopathy Treatment Eagle Eye Centre.* Disponible en: http://www.eagleeyecentre.com.sg/service/diabetic-retinopathy/

CHOLLET, Francois. Keras: *Deep Learning library for Theano and TensorFlow*. KERAS, 2017. Disponible en: https://keras.io/

NVIDIA Coporation. Disponible en: https://www.nvidia.com/en-us/about-nvidia/

Kaggle INC. (Consultado 4 de abril de 2020). *Diabetic Retinopathy Detection – Data.* Kaggle Inc, 2017. Disponible en: https://www.kaggle.com/c/diabetic-retinopathy-detection/data

KAGGLE INC. *Diabetic Retinopathy Detection* . Kaggle Inc, 2017. Disponible en: https://www.kaggle.com/c/diabetic-retinopathy-detection

COHEN, Jacob. Weighted kappa: *Nominal Scale agreement with provision for scaled disagreement or partial credit*. En: Psychological Bulletin. October 1968 vol. 70 no. 4, p. 213-220. Disponible en: https://doi.apa.org/doiLanding?doi=10.1037%2Fh0026256

KAGGLE INC, Op. cit. Disponible en:https://www.kaggle.com/c/diabetic-retinopathy-detection#evaluation

OpenCV-Python. (Consultado 3 de Noviembre de 2021*). Open Source Computer Vision.* Disponible en: https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html

OpenCV-Python. (Consultado 4 de Diciembre de 2021). *Open Source Computer Vision.* Disponible en: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

OpenCV-Python. (Consultado 1 de enero de 2022). *Open Source Computer Vision.* Disponible en: https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html

OpenCV-Python. (Consultado 4 de enero de 2022). *Open Source Computer Vision.* Disponible en: https://docs.opencv.org/3.4/dc/da3/tutorial_copyMakeBorder.html

Convolutional Neural Networks. Disponible en: https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html

Diabetic Retinopathy Detection. Disponible en: https://www.kaggle.com/c/diabetic-retinopathy-detection/overview

Mathis Anthony, Stephan Brüggeman. (consultado 04 de Febrero de 2022]). *Kaggle Diabetic Retinopathy Detection Team o_O solution*. Disponible en Internet: https://github.com/sveitser/kaggle_diabetic/blob/master/doc/report.pdf

NumPy. (Consultado 4 de enero de 2022). Disponible en: https://numpy.org/doc/stable/reference/generated/numpy.ravel.html

Tensorflow. (Consultado 20 de Enero de 2022). Disponible en: https://www.tensorflow.org/tensorboard?hl=es-419

Tensorflow. (Consultado 21 de Enero de 2022). Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/random_rotation

Tensorflow. (Consultado 23 de Enero de 2022). Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

Keras: the Python deep learning. (Consultado 23 de Enero de 2022). Disponible en: https://keras.io/api/layers/activation_layers/relu/

Tensorflow. (Consultado 24 de Enero de 2022). Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization

Tensorflow. (Consultado 25 de Enero de 2022). Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/layers/MaxPool2D

Keras: the Python deep learning. (Consultado 15 de Febrero de 2022). Disponible en: https://keras.io/api/models/model_training_apis/#compile-method

Tensorflow. (Consultado 16 de Febrero de 2022). Disponible en: https://www.tensorflow.org/addons/api_docs/python/tfa/losses/WeightedKappaLoss

Tensorflow. (Consultado 16 de Febrero de 2022). Disponible en: https://www.tensorflow.org/addons/api_docs/python/tfa/metrics/CohenKappa

Keras: the Python deep learning. (Consultado 17 de Febrero de 2022). Disponible en: https://keras.io/api/models/model_training_apis/#fit-method