

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

*Trabajo de titulación previo
a la obtención del título
de Ingeniero Electrónico*

PROYECTO TÉCNICO:

**“PROCESAMIENTO DE IMÁGENES DIGITALES DEL FONDO DE
OJO CON EL USO DE INTELIGENCIA ARTIFICIAL PARA
BRINDAR UNA HERRAMIENTA DE SOPORTE DE DIAGNÓSTICO
PRESUNTIVO DEL GLAUCOMA HUMANO”**

AUTOR:

EDISON ARTURO BURI ABAD

TUTOR:

ING. EDUARDO GUILLERMO PINOS VÉLEZ, Ph.D.

CUENCA - ECUADOR

2022

CESIÓN DE DERECHOS DE AUTOR

Yo, Edison Arturo Buri Abad con documento de identificación N° 0302044953, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación: **“PROCESAMIENTO DE IMÁGENES DIGITALES DEL FONDO DE OJO CON EL USO DE INTELIGENCIA ARTIFICIAL PARA BRINDAR UNA HERRAMIENTA DE SOPORTE DE DIAGNÓSTICO PRESUNTIVO DEL GLAUCOMA HUMANO”**, mismo que ha sido desarrollado para optar por el título de: *Ingeniero Electrónico*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, febrero de 2022.



Edison Arturo Buri Abad

C.I. 0302044953

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación:
**“PROCESAMIENTO DE IMÁGENES DIGITALES DEL FONDO DE OJO
CON EL USO DE INTELIGENCIA ARTIFICIAL PARA BRINDAR UNA
HERRAMIENTA DE SOPORTE DE DIAGNÓSTICO PRESUNTIVO DEL
GLAUCOMA HUMANO”**, realizado por Edison Arturo Buri Abad, obteniendo el
Proyecto Técnico que cumple con todos los requisitos estipulados por la Universidad
Politécnica Salesiana.

Cuenca, febrero de 2022.

A handwritten signature in blue ink, appearing to read 'Eduardo Pinos Vélez', with a horizontal line drawn through the middle of the signature.

Ing. Eduardo Guillermo Pinos Vélez, Ph.D.

C.I. 0102942190

DECLARATORIA DE RESPONSABILIDAD

Yo, Edison Arturo Buri Abad con documento de identificación N° 0302044953, autor del trabajo de titulación: **“PROCESAMIENTO DE IMÁGENES DIGITALES DEL FONDO DE OJO CON EL USO DE INTELIGENCIA ARTIFICIAL PARA BRINDAR UNA HERRAMIENTA DE SOPORTE DE DIAGNÓSTICO PRESUNTIVO DEL GLAUCOMA HUMANO”** certifico que el total contenido del *Proyecto Técnico*, es de mi exclusiva responsabilidad y autoría.

Cuenca, febrero de 2022.



Edison Arturo Buri Abad

C.I. 0302044953

DEDICATORIA

Me gustaría dedicarle este trabajo a mi Madre Zaida y mi Tío Diego, que Dios los cuide en su gloria; gracias al esfuerzo y sacrificio de mi Madre he logrado cumplir una meta más en el transcurso de mi vida, al apoyo de mi tío que desde lo alto me sigue alentando para continuar esforzándome, a mi padre Arturo que con su apoyo incondicional logre seguir adelante. A mis Abuelos y Tíos por estar siempre a mi lado y ser pilar fundamental para levantarme en momentos duros para convertirme en un profesional, a mis amigos y familiares por brindarme sus valiosos conocimientos con el fin de ayudar a la sociedad.

Edison Buri Abad.

AGRADECIMIENTOS

Agradezco primeramente a Dios por brindarme ciencia y sabiduría en cada día de mi vida para afrontar las pruebas y obstáculos que se han presentado en el desarrollo de esta tesis. A mis padres Zaida Abad y Arturo Buri, gracias por brindarme la oportunidad de vivir, darme un cariño inmenso y educarme para ser un hombre de bien, a mis Abuelos Isolina Yascaribay, José María Buri y Luz Sánchez que me han dado su apoyo incondicional le estoy inmensamente agradecido, a mis tíos Sara Buri, Diego Carangui y Mauricio Buri les doy gracias por alentarme día a día a cumplir mis metas y continuar creciendo como profesional y persona.

Le Agradezco al Doctor Eduardo Pinos por tenerme tanta paciencia y brindarme sus conocimientos para el buen desarrollo del presente trabajo de investigación llevando a concluir satisfactoriamente este proyecto.

Agradezco a mis Amigos Gabriel Sarmiento, Roció Urgilez, Estefanía Figueroa, Christian Figueroa, Christian Cedillo y Darío Carrión por estar durante tantos años compartiendo experiencias, penas y alegrías, aunque la pandemia nos separe, sé que siempre estarán en mi corazón, en verdad, gracias por tan buena amistad.

Edison Buri Abad.

ÍNDICE

DEDICATORIA.....	4
AGRADECIMIENTOS.....	5
ÍNDICE	I
GLOSARIO	III
RESUMEN.....	IV
INTRODUCCIÓN	V
ANTECEDENTES DEL PROBLEMA DE ESTUDIO	VII
JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES).....	IX
OBJETIVOS.....	X
OBJETIVO GENERAL	X
OBJETIVOS ESPECÍFICOS	X
1. Fundamentación Teórica.....	1
1.1. Fisiología del ojo	1
1.2. Glaucoma	7
1.2.1. Clasificación del Glaucoma	7
1.3. Métodos de Diagnóstico	10
1.4. Inteligencia Artificial (IA).	15
1.4.1. Redes Neuronales (NN).	16
1.4.2. Red Neuronal Convolucional (CNN).....	19
1.5. Especificidad y Sensibilidad.....	24
1.6. Transferencia de Conocimiento.	25
2. Procesamiento de imágenes y Software.....	27
2.1. Base de Datos	27
2.2. Software y Hardware	29
2.3. Pre-Procesamiento	29
3. Implementación y análisis de resultados.....	37

3.1.	CNN con Una Capa Convolutiva.....	38
3.2.	CNN con Dos Capas Convolutivas	49
3.3.	CNN con Tres Capas Convolutivas.....	60
3.4.	CNN con Cuatro Capas Convolutivas.....	71
3.5.	Transferencia de Conocimiento.	82
3.6.	Desarrollo de la Interfaz de la Herramienta de Soporte.	95
3.7.	Sensibilidad y Especificidad.....	96
3.8.	Análisis de Resultados	98
4.	Conclusiones y Recomendaciones.....	102
	Trabajos Futuros	105
	ANEXOS.....	106
	ANEXO A.....	106
	ANEXO B	110
	ANEXO C	111
	ANEXO D.....	118
	ANEXO E	119
	ANEXO F	121
	ANEXO G.....	123
	ANEXO H.....	133
	REFERENCIAS BIBLIOGRAFICAS.....	143

GLOSARIO

CNN	Red Neuronal Convolucional.
CONADIS	Consejo Nacional para la Igualdad de Discapacidades.
GDE	Gradiente Descendiente Estocástico
mm Hg	milímetros de mercurio, escala de la presión ocular.
NN	Red Neuronal
OMS	Organización Mundial de la Salud.
PIO	Presión Intraocular.
ReLU	Unidad Lineal Rectificada
TIC	Tecnología de la Información y la Comunicación.

RESUMEN

Cada año el Glaucoma causa que millones de personas pierdan la vista, de acuerdo con la Organización Mundial de la Salud (OMS), 2.2 billones de personas padecen de discapacidad visual, muchas de ellas, ya sea por falta de conocimiento o no disponer de suficientes recursos económicos conforman al menos 1 billón de personas que podrían haber controlado esta enfermedad con un diagnóstico temprano.

Por tal motivo, este proyecto busca proporcionar a los especialistas una herramienta que permita determinar un diagnóstico más acertado sobre si los pacientes presentan sospecha de Glaucoma. Para ello, se utilizan Imágenes de Fondo de Ojo tomadas a partir de bases de datos públicas, las imágenes son procedentes de un Tomógrafo de Coherencia Óptica (OCT), cada una de ellas anteriormente catalogadas por especialistas con varios años de experiencia en el diagnóstico de esta enfermedad.

Con el uso de Inteligencia Artificial, IA, se desarrolla 12 modelos basados en Redes Neuronales Convolucionales, de los cuales los primeros 4 modelos utilizan el optimizador de Adagrad, en los 4 siguientes se implementan el optimizador de Adam, y en los últimos 4 se aplica transferencia de conocimiento mediante el uso de modelos pre entrenados que es similar a compartir los conocimientos de un profesor con un estudiante, para posteriormente aplicar una prueba de Gold Estándar.

Esta prueba toma los datos de la matriz de confusión para calcular la sensibilidad y especificidad, este proyecto está centrado en lograr el valor de especificidad más alto posible, de esta manera se logra confirmar un diagnóstico presuntivo de la enfermedad. Dentro de la medicina, esta prueba indica la fiabilidad de los resultados obtenidos.

En el presente proyecto se obtuvo un modelo con 100% de especificidad utilizando un modelo creado desde cero y la misma probabilidad con el modelo pre entrenado Resnet 50. Por último, se desarrolla una herramienta de Soporte con una interfaz intuitiva para el usuario bajo software libre, en donde, el usuario primero selecciona la Imagen de Fondo de Ojo, entonces con ayuda del cursor recorta la región de interés, que, en este proyecto es el Disco Óptico y finalmente el programa clasifica la imagen en Sano o Sospecha de Glaucoma usando la Red Neuronal Convolutiva entrenada. De esta manera se brinda a los especialistas un apoyo adicional al momento de indicar el diagnóstico a sus pacientes.

INTRODUCCIÓN

La tecnología es una herramienta que indudablemente permite avanzar a la sociedad y desarrollarse en áreas como el trabajo, educación, cultura, y muchos otros más [1], [2].

Cada año se realizan estudios referentes a discapacidades visuales, dado que la visión es uno de los sentidos más importantes que el ser humano utiliza, y la privación o deterioro de éste, afecta directamente su estilo de vida; existen dos grupos de enfermedades oculares, las multifactoriales y las de gen único; dentro de las enfermedades multifactoriales se encuentra el Glaucoma, que es la degeneración macular relacionada con la edad, la catarata senil, la retinopatía diabética entre otras más, y en enfermedades de gen único o enfermedades monogénicas se encuentran las degeneraciones retinianas, la catarata congénita, el estrabismo congénito, el glaucoma infantil y juvenil. De acuerdo con la Organización Mundial de la Salud (OMS), indica que: “el glaucoma es la causa más frecuente de ceguera irreversible en el mundo y se calcula que 12.3% de los ciegos es atribuido al glaucoma, que por lo menos el 50% de los pacientes con glaucoma desconoce su padecimiento” [2].

En el Ecuador, la entidad encargada de llevar el registro de las personas con discapacidades es el Consejo Nacional para la Igualdad de Discapacidades (CONADIS), que para el año 2019 registró un total de 485.325 personas que sufren de discapacidad visual, de las cuales, más del 50% pueden padecer de glaucoma y no estar conscientes [3].

Las Instituciones Educativas dentro del país, trabajan conjuntamente para elaborar proyectos, tanto investigativos, como técnicos, que permitan el desarrollo y conocimiento acerca de esta enfermedad debido a que causa un deterioro muy serio en el globo ocular; la Universidad Politécnica Salesiana, junto con la Clínica de Especialidades “Santa Lucía” de la ciudad de Quito, lleva a cabo el desarrollo de proyectos con el Grupo de Investigación e Inteligencia Artificial y Tecnología de Asistencia (GI-IATa), desarrollando software que pueda ser utilizado como soporte por los médicos especialistas y así tener más certeza al indicar un diagnóstico a sus pacientes.

En este proyecto se plantea el desarrollo de una herramienta de software libre como herramienta de soporte para identificación presuntiva del glaucoma mediante el procesamiento de imágenes s digitales con el uso de inteligencia artificial.

ANTECEDENTES DEL PROBLEMA DE ESTUDIO

El informe publicado por la Organización Mundial de la Salud (OMS) en octubre de 2019, estima que para el año 2030 cerca de 95.4 millones de personas padecerán de glaucoma, a diferencia del año 2019 que se registraron 76 millones. Este pronóstico es alarmante, actualmente el glaucoma es catalogado como una enfermedad común, que no tiene cura [4].

El glaucoma se ha convertido en una enfermedad ocular crónica y progresiva, al ser silenciosa el paciente perderá su visión con el tiempo, lo que finalmente le causará ceguera [5], [6].

El termino glaucoma, se deriva del griego: *glaukos* que significa “azul húmedo o acuoso”, atribuido a las personas de edad avanzada., fue empleada por Hipócrates (469 a.C.) [7], [8].

En el campo de la oftalmología, el primer aporte para identificación del glaucoma fue hecho por Schiötz en 1905, con su tonómetro de indentación¹, para el año 1950, sería tomado como un estándar por la Academia Americana de Oftalmología. En el mismo año Hans Goldmann, propuso su perímetro, que cinco años más tarde, sería conocido como tonómetro de aplicación [9], [10].

Entre 1936 y 1940, Otto Barkan, impulsó la denominada gonioscopía, que se convirtió en parte vital contra el glaucoma, propuso el uso de goniolentes² como el goniolente directo de Koepe y el indirecto de Goldman, que se conformaba de tres espejos [9], [11].

En 1862, nace el primer fármaco para combatir el glaucoma, desarrollado por Thomas R. que permitía una reducción de la presión intraocular, y recibió el nombre de pilocarpina [8].

En la actualidad algunos de los métodos mencionados se siguen utilizando por los oftalmólogos, razón por la que, la inclusión de la tecnología de computación es una mejora significativa para identificación presuntiva del glaucoma.

¹ Tonómetro de Identación: Instrumento usado para medir la presión intraocular. <https://www.cun.es/diccionario-medico/terminos/tonometria-indentacion>

² Goniolente: Son lentes utilizados por los médicos para visualizar la estructura del globo ocular, los hay de varios tipos y dimensiones. <https://blogs.sld.cu/elierortiz/2013/03/30/goniolentes/>

En los últimos años se han realizado varios estudios como: “*Simulación y modelamiento del ojo humano como herramienta para la prevención del glaucoma a través la medición de la presión intraocular*”, indica que, a través de un modelo matemático y software, se simula como afecta la velocidad de producción de humor acuoso en la presión intraocular (PIO), llegando a la conclusión que el glaucoma tiene un 68% de probabilidad de afectar a personas que superen los 40 años, sean hombres o mujeres [12].

En: “*Implementación de Técnicas aplicadas para la identificación y prevención del glaucoma mediante el uso de imágenes médicas y software*”, se sugiere identificar la relación copa disco en el globo ocular, a través de un algoritmo y procesamiento de imágenes, para un posterior diagnóstico presuntivo con ayuda de la Regla ISNT [13].

De manera similar en: “*Desarrollo de un sistema de soporte a la detección del glaucoma a través de procesamiento digital de imágenes biomédicas del fondo de ojo y uso de software libre*”, se propone el desarrollo de una interfaz de software libre a través de QtDesigner y OpenCV, en donde, se realiza un procesamiento de imágenes con ayuda de métodos de filtrado, y así obtener una relación Copa-Disco, y la Regla ISNT; de esta manera brindar a los oftalmólogos una herramienta para el diagnóstico de glaucoma en sus pacientes [14].

En el trabajo “*Modelamiento y simulación del ojo humano para establecer una correlación entre el aumento de la presión intraocular y el grosor central de la córnea*”, se formula un modelo matemático que relaciona la presión intraocular con el grosor central de la córnea, mediante software libre, como conclusión el modelo fue examinado con una curva de ROC, que dio resultados positivos para la identificación del glaucoma [15].

De forma semejante: “*Desarrollo de una herramienta para la identificación y prevención del glaucoma a través de procesamiento digital de imágenes biomédicas mediante la relación de volumen de fibras nerviosas*”, se presenta la identificación de glaucoma mediante la relación del grosor de la cantidad de volumen de fibras nerviosas retinianas, mediante procesamiento de imágenes se aplica métodos de filtrado y se identifica los puntos entre la región superior e inferior para finalmente obtener una ecuación que indique el grosor de la fibra nerviosa; como conclusión se obtiene un error de 4.11%, lo que indica el correcto funcionamiento del algoritmo aplicado [16].

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

La Organización Mundial de la Salud (OMS), cada año publica un informe, con los detalles más importantes que afecta a los seres humanos alrededor del mundo, en 2019 se publicó un informe, indicando que 2.2 billones de personas sufren de discapacidad visual, de esta cantidad alrededor de 1 billón se pudo evitar, si se hubiese tomado las medidas necesarias. El Glaucoma a nivel global afectó a 76 millones de personas, de ellas 6.9 millones se podrían haber evitado, sin mencionar, que actualmente 11.9 millones padecen de discapacidad moderada o severa [4].

El Ecuador lleva un registro de todas las personas que sufren de alguna discapacidad, dicha institución recibe el nombre de Consejo Nacional para la Igualdad de Discapacidades (CONADIS), en cuyos registros indican que de un total de 17.475.205 de ciudadanos, alrededor de 485.325 sufren de alguna discapacidad, siendo 56.570 las que tienen discapacidad visual, con grados de pérdida de visión entre 30 - 49% un total de 19.033 personas, entre 50 - 74% un total de 15.116, de 75 - 84% un total de 16.986 y un 100% un total de 5.434; los más afectados tienen una edad de entre 36 – 64 años, con porcentaje de 42,79% [3].

OBJETIVOS

OBJETIVO GENERAL

- Identificar el fondo de ojo humano en imágenes digitales aplicando técnicas de procesamiento de imágenes e inteligencia artificial.

OBJETIVOS ESPECÍFICOS

- Revisar Estudios relacionados con la fisiología del ojo y el Glaucoma para adquirir los conocimientos que requiere el tema a tratar.
- Revisar trabajos relacionados con el Glaucoma propuestos por otros autores para definir las técnicas que se utilizan en su diagnóstico.
- Revisar estudios acerca del procesamiento de imágenes de fondo de Ojo Humano.
- Desarrollar un algoritmo con el uso de software especializado, que mejor se adecue al proyecto, así como las modificaciones necesarias.
- Validar los Resultados

CAPÍTULO 1

1. FUNDAMENTACIÓN TEÓRICA

Como primer paso se realiza una descripción de la fisiología del ojo, como está conformado, los diferentes tipos de Glaucomas, además de los métodos que utilizan los especialistas para determinar el Glaucoma en sus pacientes y, por último, los métodos y arquitecturas que maneja la Inteligencia Artificial.

1.1. Fisiología del ojo

El ojo es un órgano complejo (ver Figura 1), que se encarga de proveer el sentido de la visión, procesa gran cantidad de información para luego seleccionar y extraer la información útil que ira al cerebro y así determinar el color, la forma, el movimiento y la profundidad de los objetos, para finalmente enviar esta información al cerebro que se encarga de la toma de decisiones [17].

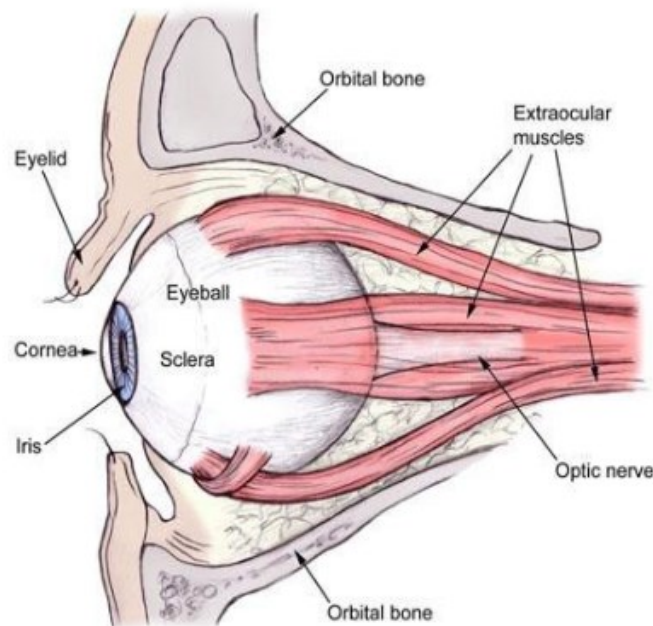


Figura 1. Órbita del Ojo [17].

El ojo está conformado de tres capas: capa fibrosa, capa vascular y capas nerviosas [17].

CAPA FIBROSA

Es la parte externa del ojo, conformada de dos partes, la córnea y la esclera.

Córnea: Es un órgano transparente que se encuentre delante del iris (ver Figura 1), tiene una estructura convexa de un diámetro horizontal de 12 mm y vertical de 10 - 11 mm, su función principal es repeler el agua, evita que pasen fluidos, también se encuentra la *Membrana de Bowman*³, la *Estroma*⁴ la *Membrana de Descemet*⁵ y el *Endotelio*⁶ [17], [18], [19].

La Esclera: El tejido es vascular y blanco, aproximadamente de 1mm, su firmeza, fuerza y propiedades elásticas mantienen la forma del tejido del ojo, la unión con la córnea y la conjuntiva se llama episclera. La **conjuntiva** es una membrana que cubre la esclera, formando el saco conjuntival (ver Figura 2), hay tres conjuntivas: la **conjuntiva bulbar**, recubre la esclera blanca, la **conjuntiva palpebral**, recubre la superficie trasera del párpado, y el **fórnix**, es el lugar en donde la conjuntiva bulbar y la palpebral se encuentran [17], [18], [19].

CAPA MEDIA

Es una superficie óptica y líquida dentro del ojo, permite el paso de la luz hacia la retina, compuesta de: la cámara anterior, los lentes cristalinos y la vítrea.

Cámara anterior y ángulos: Esta encerrada por la córnea y el iris, (ver Figura 3), está llena de un líquido de color claro e incoloro llamado *humor acuoso*, este viaja a través de la cámara posterior del iris, baña la córnea mediante la pupila, y sale mediante el ángulo, por un punto de drenaje en donde se encuentran la córnea y el iris. El ángulo se crea por la raíz del iris y la periferia de la córnea, tiene una apertura de 30 grados, si llegara a estrecharse impediría el paso del humor acuoso, y cuando este se cierra se genera el Glaucoma, la función del **humor acuoso** es la de aportar nutrientes a la córnea, al cristalino y mantener la presión intraocular [17], [18], [19].

³ Membrana de Bowman: Compuesta de fibras de colágeno acelular, mantiene la forma de la córnea, evita que fluidos y microorganismos entren en contacto con la estroma [17].

⁴ La Estroma: Representa el 90% del grosor de la córnea le da la transparencia a la córnea [17].

⁵ Membrana de Descemet: esta sobre la capa posterior a la estroma, compuesta de colágeno y glicoproteínas secretadas por el endotelio, resistente a los productos químicos y traumas y puede regenerarse si es dañada [17].

⁶ El Endotelio: Es la última de las membranas, su función es la de mantener la membrana permeable y dar oxígeno, funciona como barrera para las impurezas, y no se puede regenerar [17].

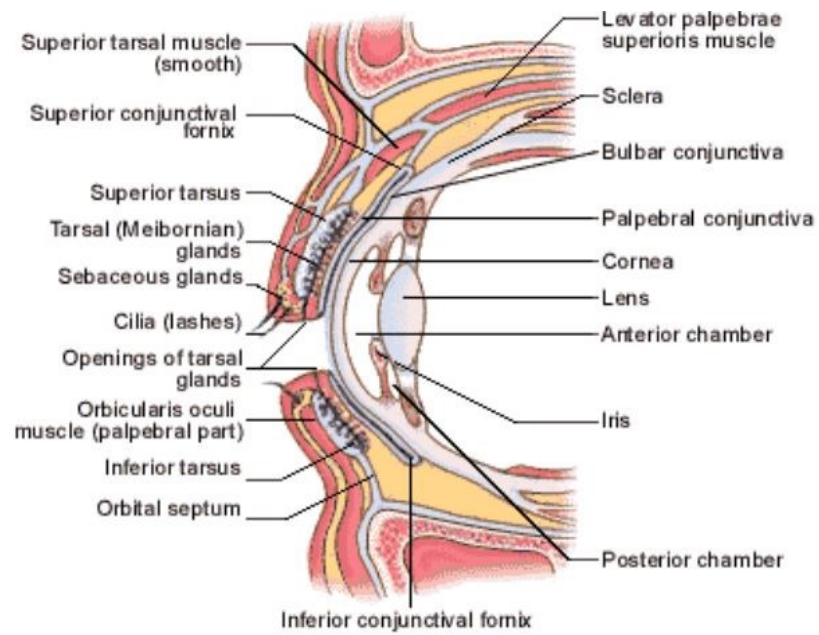


Figura 2. Anatomía del Ojo [17].

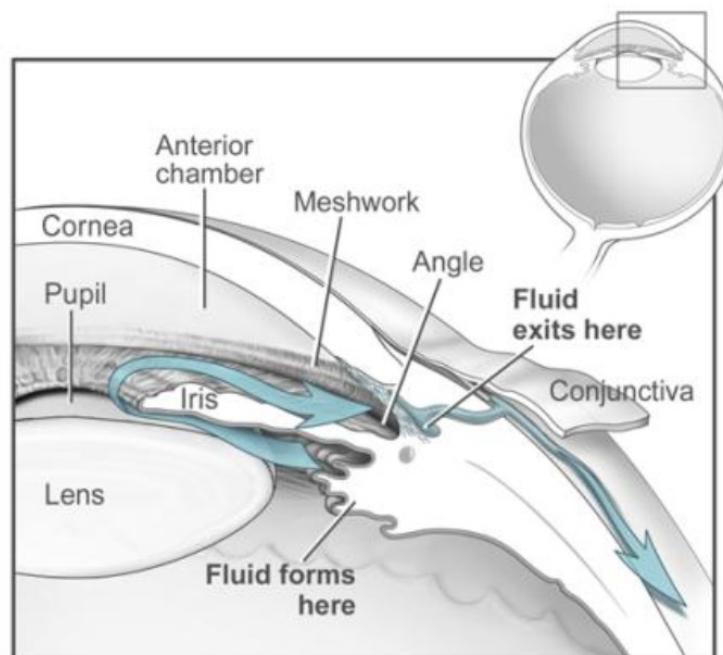


Figura 3. Flujo Acuoso [17].

Lentes Cristalinos: Estos son biconvexos, de estructura transparente ubicados detrás del iris y delante del vítreo. Es la segunda estructura más refractiva del ojo, sin nervios o tejidos conectivos, es flexible y se mantiene en su lugar por zónulas que los sujetan al cuerpo ciliar. Está compuesta por: **La Capsula**, esta membrana mantiene la forma de los lentes, además, actúa como una barrera contra el vítreo, la fluoresceína⁷ y las bacterias; **Células epiteliales**, estas tienen forma cubica, y están ubicadas debajo de la Capsula. **La Corteza**, está ubicada en el centro del lente cristalino. Al nacer mide cerca de 6.5mm y crece hasta 10mm en la edad adulta (ver Figura 4). El Lente cristalino cambia su forma para enfocar los objetos según este cerca o lejos [17], [18], [19].

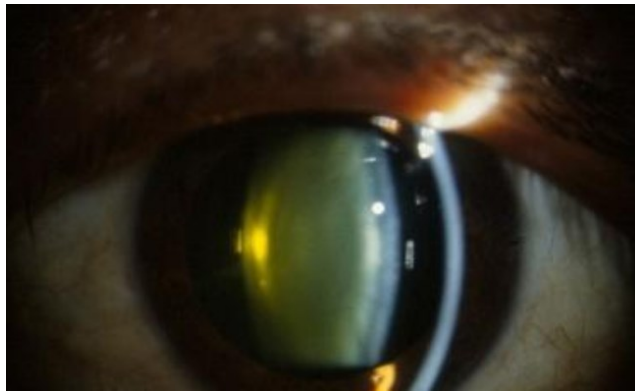


Figura 4. Lente Cristalino de un Adulto [17].

El Vitreo: constituye el 80% del ojo, es clara y gelatinosa. Está ubicada entre el Cristalino y la retina, Su función es la de mantener la transparencia y la presión interna del ojo [17], [18].

CAPA VASCULAR

Conocida como tracto uveal (ver Figura 5), compuesta de tres partes, el cuerpo ciliar, el iris y las coroides. Está situada entre la esclera y la retina, consiste mayormente de vasos sanguíneos y canales capilares. Sus funciones son: **Producir** humor acuoso en el proceso ciliar y, **Alterar** la forma de los lentes cristalinos para enfocar el ojo en los objetos cercanos o lejanos [17], [18].

Cuerpo Ciliar: Se extiende desde la base del iris (ver Figura 5), se divide en dos partes: la *pars plicata* que produce acuosa en el proceso ciliar, y la *pars plana* está en la parte más alejada del cuerpo ciliar y es un punto de referencia para los instrumentos de microcirugía [17], [18], [19], [20].

⁷ Fluoresceína: La fluoresceína es una sustancia colorante orgánica utilizada en el examen de los vasos sanguíneos del ojo y en ciertas técnicas odontológicas. <https://www.vademecum.es/principios-activos-fluoresceina-s01ja01>

El Iris: es la parte colorida del ojo, se encarga de transmitir la luz para cambiar el tamaño de la pupila (ver Figura 5), está compuesta de dos músculos para controlar la pupila: el **musculo dilatador**, como su nombre lo indica dilata la pupila, también llamado midrasis y, el **musculo esfínter**, encargado de reducir la cantidad de luz que entra en el ojo, llamado miosis. El iris tiene tres capas [17], [18], [19], [20]:

- **Anterior:** Contiene melanocitos y colágenos.
- **Estromal Medio:** Contiene Fibroblastos, melanocitos, y colágeno.
- **Posterior:** Consiste en un musculo dilatados y un pigmento de epitelio, el estroma, o capa media, tiene la mayor parte del iris y contiene vasos sanguíneos, nervios y melanocitos.

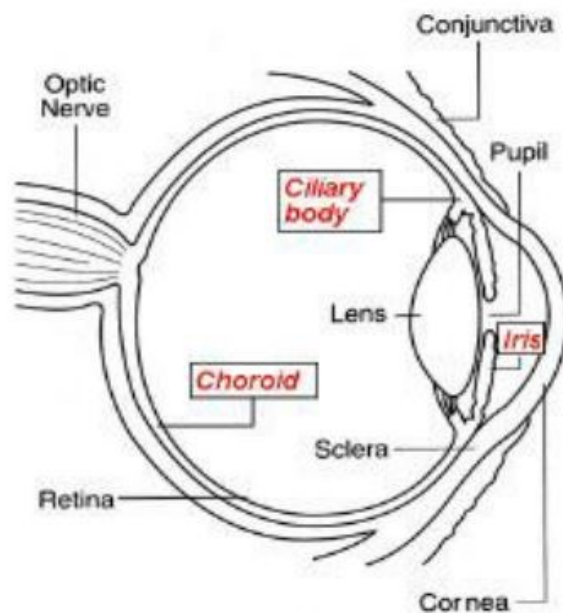


Figura 5. Tracto Uveal [17].

Coroides: encargado de nutrir y proveer de oxígeno a las capas exteriores de la retina (ver Figura 5), está compuesto de cuatro capas: Lamina fusca, estroma, la coriocaliparis y membran de Bruch, comparte una arteria oftálmica y, es la arteria de mayor suministro de sangre [17], [18], [19].

Pupila: Se encuentra en el centro del iris, es de color negro, debido a que el tejido tiene que absorber la mayor cantidad de luz posible, su función es similar a la de una cámara, el músculo dilatador abre la pupila, y el músculo esfínter la encoje [17], [18], [20].

CAPA NERVIOSA

Receptores Celulares: Es la capa más interna del ojo y, es una extensión directa del cerebro [17].

Retina: Es una capa fina de tejido transparente, compuesta de 10 capas, pero solo tres se encarga de convertir la luz en señales electroquímicas, los encargados de transmitir la información son los **fotorreceptores**, estos envían la señal a través de las fibras nerviosas, atravesando las diferentes capas hasta la retina, posteriormente quien recibe esta información es la **mácula**, llega la imagen nítida, para que luego los conos y las barras le den color e iluminación a la imagen (ver Figura 6), [17], [18], [19].

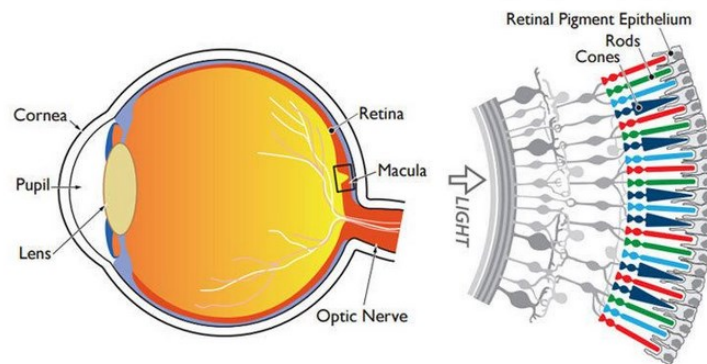


Figura 6. Macula seccionada [17].

Nervio óptico: Las fibras nerviosas llevan información de la retina al cerebro, se puede encontrar en esta área el disco óptico, aparece de color amarillo y se encuentra debajo de los vasos sanguíneos (Ver Figura 7), [17], [18].

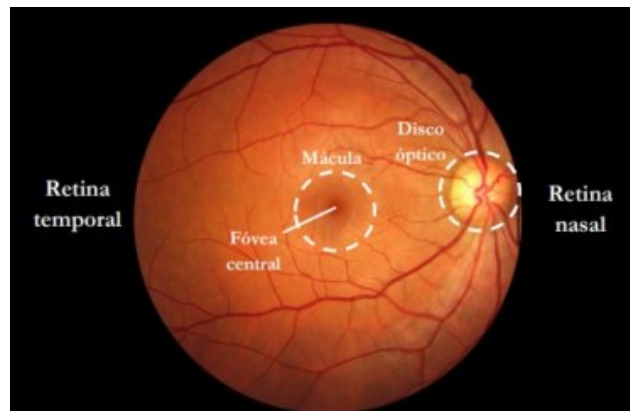


Figura 7. Retina [17].

1.2. Glaucoma

A. Saucedo et al de Glaucoma, lo define de esta manera: “Es un conjunto de neuropatías ópticas crónicas y progresivas que sufren degeneración de los axones de las células ganglionares y cambios morfológicos característicos de la pupila y la capa de fibras nerviosas, además de la pérdida del campo visual”[2], El término se refiere al conjunto de patologías con diferentes presentaciones clínicas, patogénesis y tratamientos. [21]

1.2.1. Clasificación del Glaucoma

Se dividen en *Glaucoma Primarios* y *Secundarios* [2], [12–16], [21] y a su vez se subdividen en:

GLAUCOMA PRIMARIO

Glaucoma Primario de Angulo Abierto (GPAA): Son alteraciones progresivas, y asimétricas que afectan las fibras nerviosas de la retina, los pacientes tienen PIO elevada, con un valor de 21mm Hg, que al incrementarse puede desarrollar glaucoma; algunas pruebas usadas para su diagnóstico son: la ingestión hídrica⁸ y la prueba de ibopamina⁹. Se encuentra clasificado de acuerdo con su nivel de riesgo en: **alto riesgo**, presentan Excavación aumentada del disco óptico, Presión intraocular elevada, Vejez, Razas afroamericanas e hispana, Grosor Central corneal Delgado, Baja presión de perfusión, **riesgo moderado**: presentan antecedentes

⁸ Ingestión Hídrica: este método de diagnóstico se dejó de usar debido a su sensibilidad, especificidad y bajo valor de diagnóstico. <https://oftalmologaldia.com/blog/2019/05/21/la-prueba-de-sobrecarga-hidrica-y-su-uso-en-el-manejo-del-glaucoma/>

⁹ Prueba de Ibopamina: es la mejor prueba para diagnóstico del glaucoma. <https://temas.sld.cu/glaucoma/2010/02/23/ibopamina-test-de-provocacion/>

familiares, y en **bajo riesgo**: presentan Diabetes mellitus, Migraña, Miopía Alta, e Hipotiroidismo. Por lo general aparece en personas que superan los 40 años, por lo que varía mucho con la edad y la raza. Sus síntomas al inicio son leves y no se nota hasta presentar ceguera. Para realizar un diagnóstico se identifica alteraciones en el campo visual, la cabeza del nervio óptico, y la presión intraocular; El daño avanza conforme se incrementa la **PIO**. Las alteraciones más comunes son: Adelgazamiento del anillo neuro-retiniano, excavación de más de seis décimas, asimetrías de la excavación de los dos ojos mayor a dos décimas, pérdida en la regla ISNT con presencia de excavación en el lado nasal. Entre los Equipos Utilizados para su diagnóstico se encuentran: El Tomógrafo de la retina de Heidelberg (**HRT**) realiza una valoración en tres dimensiones del disco óptico y produce una imagen que permite determinar el daño en el anillo (ver Figura 8.a), El Tomógrafo de Coherencia Óptica (**OCT**) da la posibilidad de cuantificar el grosor de la capa de fibras nerviosas de la retina mediante interferometría laser (ver Figura 8.b); Polarimetría de Barrido Laser (**Gdx**) que permite determinar el grosor de la capa de fibras nerviosas de la retina, por último, fotografías del fondo ocular a color y tridimensionales (ver Figura 8.c), [2].



a)



b)



c)

Figura 8. Equipos de Tomografía, a) HRT, b) OCT, c) Gdx [2].

Glaucoma Primario de Angulo Cerrado: Tiene una incidencia en la población bastante baja, los factores que pueden dar lugar a este son: Mayor de Edad, Raza Asiática, Sexo Femenino, Hipermetropía, diámetro corneal pequeño. Cámara anterior estrecha, también puede ser producto de: bloqueo del disco óptico inducidos por el cristalino, Iris plateau¹⁰, proyección del diafragma, aumento de Volumen de Iris y mayor engrosamiento de la coroides. Se caracteriza por presentar: dolor considerable, ojo rojo, baja agudeza visual, cefalea, náusea, sudación, dolor torácico o abdominal. Requiere atención inmediata en el lugar de bloqueo del disco óptico, para que pase el humor acuoso por detrás del iris, a través de una iridectomía. Aunque para el caso de iris plateau, es necesario un tratamiento con iridoplastia¹¹ [2].

GLAUCOMA SECUNDARIO

A diferencia del Glaucoma Primario de ángulo abierto, en este se puede identificar con claridad la razón de la obstrucción del humor acuoso.

Glaucoma Secundario de Angulo Abierto: Este tipo de Glaucoma se tiene una presión intraocular elevada debido a células inflamadas, sangre o pigmento. Se tiene los siguientes: **Glaucoma pigmentario**, se desarrolla en personas miopes masculinas, su causa se debe a un roce anormal de la cara posterior del iris con la capsula anterior del cristalino, lo que genera presión intraocular; **Glaucoma secundario a pseudoexfoliación o Glaucoma capsular**, se caracteriza por presentar depósitos de proteína en la parte anterior del ojo, aparece a edad tardía sin distinción de sexo, **Glaucoma facolítico**, es el resultado de una degradación, sucede en una catarata madura o hipermadura, **Glaucoma por partículas cristalinas**, efecto de la rotura de la cápsula del cristalino, **Glaucoma facoanafilático**, efecto de desarrollar sensibilización inmunológica después de un traumatismo u operación en el paciente, **Glaucoma secundario a esteroides**, producido por el uso de esteroides tópicos, **Glaucoma secundario a uveítis**, consecuencia de obstrucción por células inflamatorias y proteínas, **Glaucoma secundario a receso angular**, producto de haberse producido en el musculo ciliar una separación de las capas circulares y longitudinales, **Glaucoma neovascular**, generalmente se presenta en pacientes con retinopatía diabética proliferativa u obstrucción de vena central de la retina [2].

¹⁰ Iris plateau: es una condición que se da cuando el iris bloquea el paso del humor acuoso hacia el iris.
<https://blogs.sld.cu/elierortiz/iris-en-meseta-plateau/>

¹¹ Iridoplastia: Es un procedimiento con láser de argón que tiene como objetivo abrir el trabéculo en el caso que el iris tape la zona de filtrado de humor acuoso.
<https://www.imo.es/es/enfermedades-de-los-ojos/tratamientos/iridoplastia>

GLAUCOMA CONGÉNITO

Glaucoma Congénito Primario: Se da en niños, se diagnostica en el lapso del primer año de vida, es del tipo hereditario y se caracteriza por lagrimeo, blefaroespasmos¹² y fotobobia¹³. Ocasionalmente es posible ver la membrana de *Barkan*, suele aparecer delante de iris con un color grisáceo [2].

Glaucoma congénito relacionado con anomalías congénitas: Se relaciona con anomalías de otras estructuras, las tres principales alteraciones son: **Síndrome de Axenfeld-Rieger**, son trastornos producidos por alteraciones del ángulo, el iris y el trabéculo, **Anomalía de Axenfeld**, se encuentra en 10-15% de los ojos normales, caracterizada por una línea de Schwalbe prominente y, **Síndrome de Rieger**, comprende alteración del iris, como pupila descentrada, zonas atrofiadas y adelgazamiento [2].

1.3. Métodos de Diagnóstico

GONIOSCOPIA

Es un examen indoloro en el que se analiza el ángulo de drenaje, es decir, por donde se drena el humor acuoso y se encuentra al frente del ojo, entre el iris y la córnea (ver Figura 9). Para realizar este examen se utiliza un soporte, se coloca antes unas gotas sobre el globo ocular, de esta manera queda anestesiado, luego con una lampara de hendidura y una luz se resalta el ángulo de drenaje. Para terminar con ayuda de un lente de contacto portátil se determina si el ángulo está cerrado y bloqueado, o amplio y abierto [11], [22], [23].

Existen dos sistemas de clasificación: el de Shaffer y el de Scheie, sin embargo, el de Shaffer es utilizado internacionalmente, de acuerdo con [2] y [24], existen diferentes grados:

- **Grado 0:** Tiene un cierre inminente, sin posibilidad de valorar ninguna estructura (ver Figura 10.a).
- **Grado 1 (10°):** Angulo muy estrecho, alto Riesgo, se puede ver la parte frontal del trabéculo (ver Figura 10.b).
- **Grado 2 (20°):** Angulo moderadamente estrecho, solo se ve el trabéculo, pero el cierre es poco probable.

¹² blefaroespasmos: Es una anomalía en los párpados que causa contracciones involuntarias. <https://medlineplus.gov/spanish/ency/article/000756.htm>

¹³ Fotofobia: No es una enfermedad, más bien es un cierre espontaneo de los ojos como medida de protección ante el exceso de luz o iluminación. <https://www.clinicabaviera.com/blog/bye-bye-gafas/que-es-la-fotofobia/>

- **Grado 3 (25-35°):** Angulo abierto, se puede ver el espolón escleral¹⁴, cierre poco probable (ver Figura 10.c).
- **Grado 4 (35-45°):** Angulo más amplio, se puede ver el cuerpo ciliar, el cierre es imposible (ver Figura 10.d).

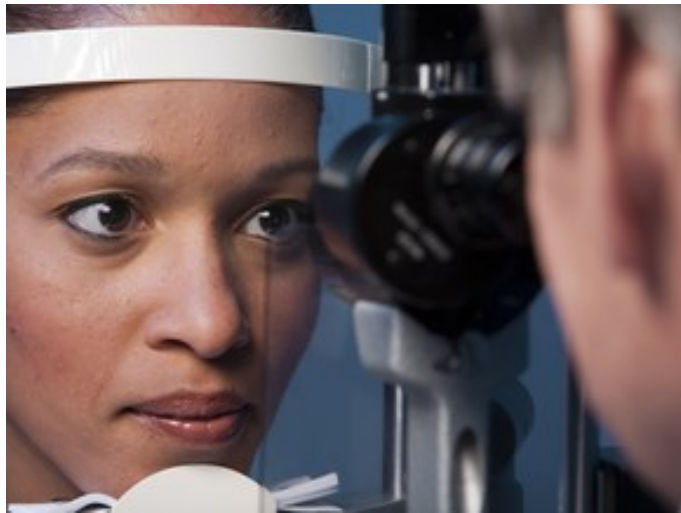


Figura 9. Imagen de Gonioscopia [23].

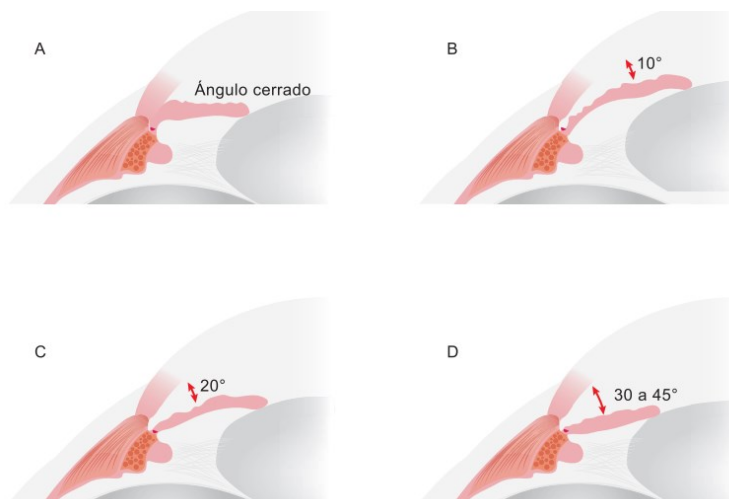


Figura 10. Sistema de Clasificación de Shaffer A) Grado 0, B) Grado 1, C) Grado 3, D) Grado 4 [2].

¹⁴ Espolon escleral: se encuentra detrás del trabéculo y tiene forma de banda estrecha, densa y blanquecina.
<https://blogs.sld.cu/elierortiz/tag/espilon-escleral/>

Durante el tratamiento se busca reducir la presión intraocular (PIO), para una presión leve se reduce por debajo de 18 mm Hg, moderado 16 mm Hg, y en los graves por debajo de 14 mm Hg, además incluye fármacos, finalmente se aplica tratamiento quirúrgico en el caso de que este persista [24].

Por otro lado, en **gonioscopia indirecta**, se puede mencionar dos tipos: **Lentes de Goldman**, de uno o dos lentes con diámetro de 12mm de superficie, se necesita de un agente que puede producir visión borrosa, permite observar la estructura del ángulo de forma exitosa y estable, este proceso se lleva en un cuarto oscuro; también están los **lentes de Zeiss**, estos lentes tienen una superficie de 9mm, dispone de una curvatura como los de Goldman para observar la córnea, necesita de una película lagrimal y, requiere poca rotación o ninguna [25].



Figura 11. Dos lentes usados en gonioscopia [25].

TONOMETRÍA

Se rige bajo el principio de que a mayor presión en la esfera, mayor es la fuerza requerida para hundirla [25], entre ellos tenemos:

Tonómetro de Aplanación de Goldman (GAT): Se rige bajo el principio de Imbert Fick, en donde, la presión de una esfera es igual a la fuerza dividida por el área de la superficie: $P = F/A$, este asume que el radio de curvatura de la esfera es continuo, seco, muy delgado y el humor acuoso no se mueve, por supuesto, esto es falso, por lo que Goldman calculó que el valor de Aplanación es 3.06mm; el dispositivo consiste de un prisma doble en el centro con forma de cono, con una cabeza de plástico de 3.06mm, tiene un resorte que permite el ajuste de la cabeza, cuando el biprisma de la cabeza entra en contacto con la córnea, se ve dos semicírculos verde amarillentos en la película de lágrimas de menisco y, aplica una

fuerza 10 veces el valor de lectura en mm Hg. Se pueden presentar errores en el GAT debido a: el examinador presiona el globo sosteniendo los párpados, excesiva fluoresceína o lágrimas, desalineación de los semi círculos, córnea rígida, paciente inquieto, mantener la respiración, ropa poco cómoda, mirada excéntrica, en ocasiones se sobreestima una verdadera PIO debido a: menisco delgado, córnea delgada, edema corneal, masaje ocular, astigmatismo, pobre o nula calibración del instrumento, variabilidad del observador, sin embargo, se puede presentar otros problemas como: riesgo de infección, uso de prismas desechables y abrasión corneal [25].

Tonómetro sin Contacto: Usa un soplo de aire para aplanar la córnea, cuya fuerza es proporcional a la PIO, la anestesia utópica no es necesaria, aunque este método es menos preciso [25].

Tonómetro de Perkins: Se utiliza una batería de mano para alimentarlo, tiene poca precisión, pero se puede usar en posición boca arriba [25].

Tono-Pen: es un tonómetro de mano con batería, tiene un transductor de galga extensométrica, que mide la fuerza de la placa central, esta se tapa con una manga de látex para evitar infecciones, es útil para ojos con distorsión y corneas edematosas [25].

Tonómetro de rebote iCare: También es un tonómetro de mano con un fino gatillo desechable y una sonda de acero inoxidable de 50mm de longitud, calcula la PIO al momento de desacelerar la presión sobre la córnea, que es de 4 a 8mm, se utiliza para mediciones en niños, pacientes con demencia y en ocasiones donde no se puede utilizar el GAT [25].

Tonometría de Contorno Dinámico: Usa una lámpara de hendidura, usa un sensor de presión muy pequeño, se toman 100 lecturas de la PIO por segundo para luego ser grabadas [25].

PAQUIMETRÍA

Este método se utiliza para medir el grosor central de la córnea (GCC), permite predecir el desarrollo del Glaucoma, si es menor a 555um hay tres veces de riesgo de Glaucoma (ver Figura 12), se puede usar dos métodos, por ultrasonido o por técnicas ópticas, en el caso del **ultrasonido**, este viaja hasta interferir con el tejido y luego regresa a la sonda, tiene un rango de 280 - 1000um, durante el procedimiento se anestesia localmente el ojo, el paciente debe mirar al frente, y se coloca una luz hacia el centro de la córnea [26].



Figura 12. Paquimetría Fuente: <https://www.salud.mapfre.es/pruebas-diagnosticas/oftalmologicas-pruebas/paquimetria>

PERIMETRÍA

Se trata de una prueba **psicofísica**, que evalúa el campo visual del paciente, el Glaucoma daña las células ganglio retinales lo que produce pérdida de la visión, en [25], se describe el campo visual como: “una isla de visión rodeada por un mar de oscuridad”, la agudeza visual se reduce a 60° superiormente, 60° nasalmente, 80° inferiormente, y 90° temporalmente [26].

Perimetría de Goldman: Este es el más usado, útil en el caso de pacientes que son incapaces de realizar el examen, para realizar el examen el paciente presiona un botón cuando mira un estímulo, el examinador califica en números romanos 0-V, representando el tamaño del objeto, números arábigos del 1-4, para la intensidad y minúsculas para el uso de filtros de luz y oscuridad, la ventaja de este examen es la velocidad, la desventaja depende de las habilidades del examinador, ya que no se realiza un análisis estadístico [26].

Perimetría Estática: se realiza de forma automatizada, con ayuda de máquinas como: Henson, Octopus y Perímetro de Humphrey, este último muestra mediciones en tres dimensiones en los límites del campo visual [26].

Entre las razones de un examen fallido esta la fiabilidad pobre del paciente, datos incorrectos de fecha de nacimiento, miosis, se dilata menos de 3mm, opacidad de los medios, error incorrecto de refracción, la cantidad de muestras, anatomía facial, y blefarocalosis, defectos en el campo superior [26].

OFTALMOSCOPIA

El examen se realiza sobre la parte posterior del ojo, es decir, la retina, el disco óptico, la coroides y los vasos sanguíneos, puede llevar un lapso entre 5-10 minutos. Existe tres formas en las que se realiza el examen [8], [26], [27]:



Figura 13. Oftalmoscopia indirecta Fuente: <https://oncologiaocular.org/oftalmoscopia-indirecta/>

Oftalmoscopia directa: Se proyecta un rayo de luz a través de la pupila en donde el medico con la ayuda de lentes diminutos puede ver la parte posterior del globo ocular. Esto se realiza dentro de una habitación oscura mientras el paciente se encuentra sentado [26], [27].

Oftalmoscopia Indirecta: El medico lleva puesto un instrumento con la forma de una linterna, y el mismo usa un lente que lo coloca de forma manual en el instrumento, también puede aplicar presión en el ojo con la ayuda de un romo y le pedirá que mire en diferentes direcciones (Ver Figura 13), [26], [27].

Oftalmoscopia con lámparas de hendidura: En este caso el paciente deberá apoyar la barbilla y la frente en un soporte y el resultado es el mismo que una oftalmoscopia indirecta, pero con mayor amplificación [26], [27].

1.4. Inteligencia Artificial (IA).

Marvin Minsky, uno de los pioneros de la IA dice: “*La inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos requerirían inteligencia*”[28], cuyo propósito es volver computacional el conocimiento humano a través de procedimientos simbólicos y conexionistas, indican dos paradigmas: **IA clásica o simbólica**, basado en el supuesto de manipulación de símbolos; **IA conexionista**, autoprogramable por aprendizaje y donde el conocimiento viene representado en la estructura de la red neuronal [29], [30].

Un problema está compuesto de: varios datos y condiciones en los estados iniciales, así como, datos desconocidos. Una de las llaves en los pasos que facilita el entendimiento del problema es: representar el problema en un formato que claramente presente los distintos estados o fases que tiene el problema para la solución propuesta y la transición entre ellas. De esta manera se reduce la complejidad de este, algunos se deben representar gráficamente con diagramas o bosquejos gráficos, mientras otros es mejor mostrarlos en formato de árbol. Un

importante aspecto en la representación del conocimiento es el nivel de detalle, puede tener **representación extensional**, se muestra cada paso, mientras que una **representación intensional**, son muestras pequeñas e implícitas, se debe elegir la representación que vaya mejor con el problema. En [31], Polya indica, que es importante separar las partes principales de los datos, así como las condiciones, y las variables desconocidas del problema; examinarlos individualmente y en relación con lo demás, esto ayuda a entender mejor dicho problema.

1.4.1. Redes Neuronales (NN).

En un inicio se las llamaba redes neuronales artificiales, inspiradas en el cerebro de los animales y el sistema nervioso, los estímulos se transmiten en todo el cuerpo debido a que las neuronas están conectadas unas con otras. Cuando una neurona recibe información de sus vecinos rápidamente dispara una señal por el axón a cada neurona conectada y luego está a la siguiente, de esta manera la información viaja por todo el sistema (ver Figura 14); el cerebro humano tiene cerca de 100 Billones de neuronas todas juntas, y alrededor de 100 Trillones de conexiones [32].

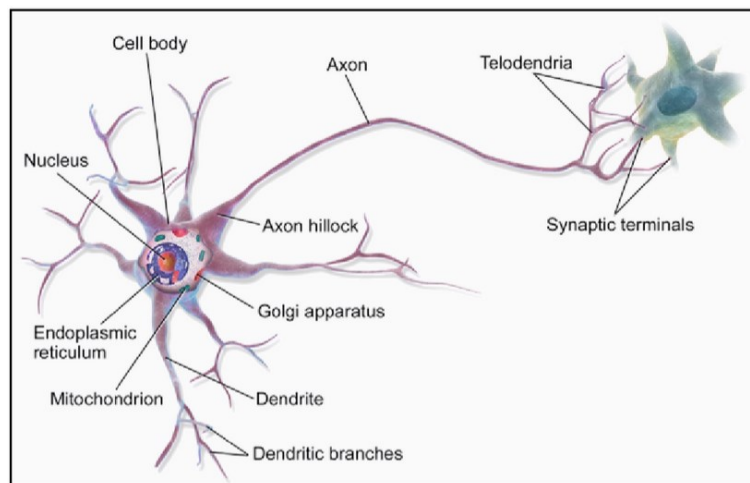


Figura 14. Componentes de una neurona [32].

El modelo neuronal más común utiliza la suma de los pesos de la entrada con funciones no lineales [33]:

$$y = f(w^T x + b) \quad (1)$$

Donde w , es el vector de pesos, x es vector de entrada, b es el bias, y es la salida escalar de la neurona (Ver Figura 15), [33].

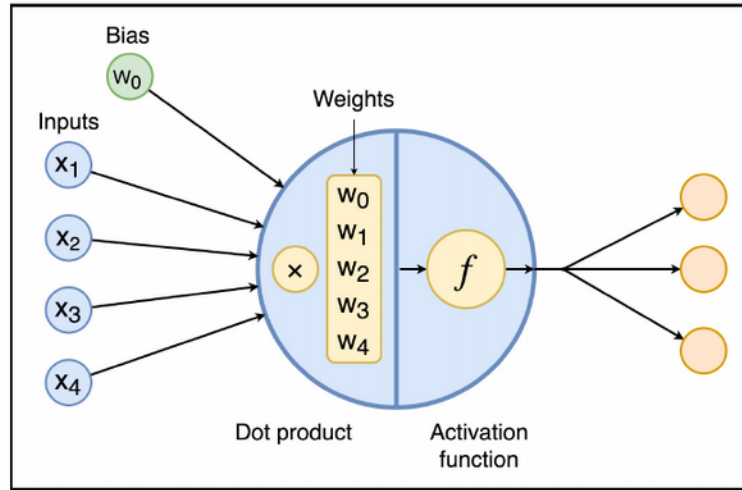


Figura 15. Diagrama de Neurona Artificial, donde las entradas son x , bias es w , y la salida es f [33].

Una neurona artificial procesa las entrada en tres pasos: **Toma la suma ponderada de las entradas**, en este punto cada neurona tiene un vector de pesos de la misma longitud que las entradas, luego se introduce un peso más que recibe el nombre de *bias*, que siempre tiene el valor de 1, la suma ponderada es un producto puntual del vector de entrada y el vector de pesos; **El resultado atraviesa una función no lineal**, se le llama también función de activación o función de transferencia, por último, **El resultado continua hacia la siguiente neurona** [33].

FUNCIONES DE ACTIVACION.

Una función de activación mapea las entradas ponderadas de una neurona para producir la salida, algunas propiedades de las redes neuronales dependen de la elección de función de activación incluyendo la generalización, así como, la velocidad de convergencia en el proceso de entrenamiento [33], para el presente trabajo se utilizan estas dos:

Tabla 1. Funciones de Activación [33].

Nombre	Formula	Derivada
ReLU	$f(x) = \begin{cases} 0, & \wedge x < 0 \\ x, & \wedge x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & \wedge x < 0 \\ 1, & \wedge x \geq 0 \end{cases}$
Leaky ReLU	$f(x) = \begin{cases} ax, & \wedge x < 0 \\ x, & \wedge x \geq 0 \end{cases}$	$f'(x) = \begin{cases} a, & \wedge x < 0 \\ 1, & \wedge x \geq 0 \end{cases}$

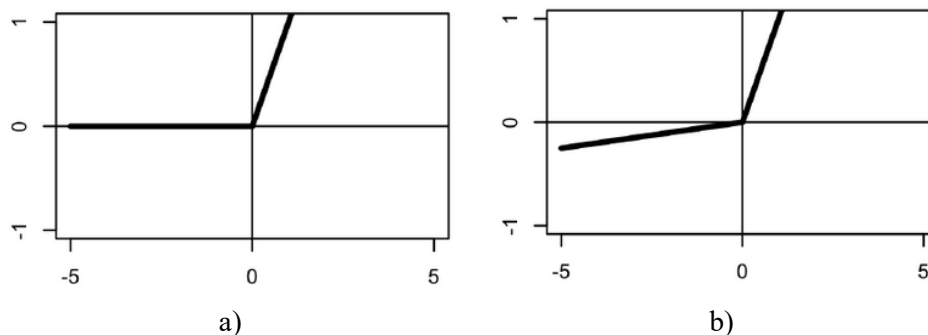


Figura 16. a) función ReLU, b) Leaky ReLU $\alpha=0,05$.

FUNCIONES DE ACTIVACIÓN RELU

Este tipo de funciones son conocidas como **Unidad Lineal Rectificada (ReLU)**, estas aplican un umbral $\max(0, x)$, de esta manera la neurona no se satura en el extremo superior, evita una predicción pobre, inclusive en situaciones difíciles, los recursos computacionales son mínimos, elevando la convergencia en una velocidad de hasta seis veces más rápida que al usar funciones sigmoideas (ver Figura 16.a), aunque tienen algunos inconvenientes [33], también se han propuesto varias soluciones:

Leaky ReLU: en lugar de 0 para todos los valores, menos que 0 regresa una minúscula fracción de su entrada, por ejemplo, 0,01; el tamaño esta dado por α , esto previene la saturación del extremo inferior, aunque en la práctica no ayuda mucho (ver Figura 16.b).

MÉTODOS DE APRENDIZAJE PROFUNDO.

Son métodos que se usan en aplicaciones específicas, entre ellas: capas convolucionales y pooling, las funciones de activación son simples y rápidas tales como ReLU, regresa los pesos a la neurona, suma si es positivo y, coloca un cero si es negativo, se usan técnicas de regularización como: dropout, ignora los pesos de forma aleatoria durante la actualización de pesos para prevenir el overfitting¹⁵, se usa las GPU para un rápido entrenamiento con velocidad de esta has 50 veces, ayuda a optimizar la matriz de cálculos [32].

La salida de cada neurona depende de la función de activación, las conexiones de entrada son aleatorias, que luego son ajustadas durante el entrenamiento, puede ser cientos, miles o incluso más épocas para lograr una precisión satisfactoria de la red; luego de esto se tiene una estructura de red definida, y todos los pesos que se han aprendido durante el entramiento, pasan una validación

¹⁵ Overfitting: También llamado sobreajuste, es el sobre entrenamiento del algoritmo de aprendizaje, que da como resultado datos incorrectos en la clasificación de nuevas muestras. <https://es.wikipedia.org/wiki/Sobreajuste>

en donde se extrae un 20% de los datos, estos no deben estar entre los datos de entrenamiento; en caso de que la red no esté bien diseñada se tendrá overfitting [32].

OPTIMIZADORES DE APRENDIZAJE PROFUNDO.

Adagrad: Ajusta la tasa de aprendizaje de acuerdo con la red, realizando ajustes mayores en los parámetros que cambian muy rara vez, mientras que aquellos que varían constantemente reciben un cambio mucho menor [34].

$$C = C + \Delta W^2 \quad (1)$$

$$W = W - \frac{\partial * \Delta W}{\sqrt{C + \epsilon}} \quad (2)$$

RMSprop: Se enfoca en corregir los efectos negativos de las caches, que son la suma de los cuadrados gradientes almacenados en una variable C, de tal manera que los convierte en un promedio movable exponencial con pesos [34].

$$C = C + d * C * (1 - d) * \Delta W^2 \quad (3)$$

$$W = W - \frac{\partial * \Delta W}{\sqrt{C + \epsilon}} \quad (4)$$

Adam: En esencia es un RMSprop con momentum, reemplaza los caches por las variables de momentum [34].

$$m = \beta_1 * m + (1 - \beta_1) * \Delta W \quad (5)$$

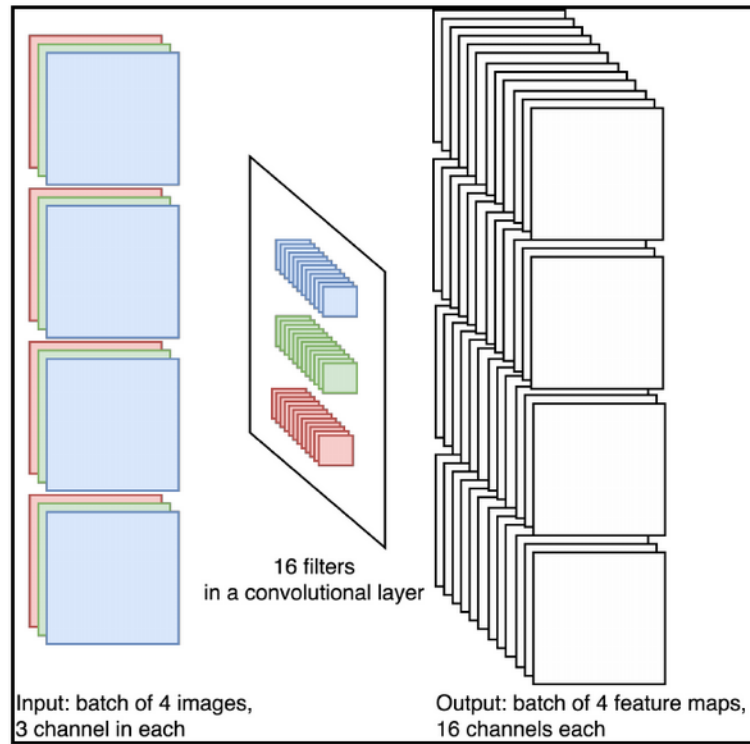
$$v = \beta_2 * v + (1 - \beta_2) * \Delta W^2 \quad (6)$$

$$W = W - \frac{\partial * m}{\sqrt{v + \epsilon}} \quad (7)$$

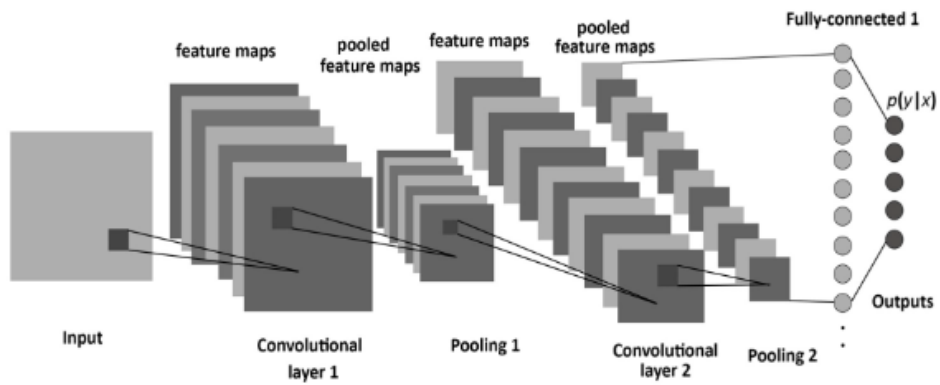
1.4.2. Red Neuronal Convolutacional (CNN)

Este tipo de redes neuronales se usan con frecuencia en el campo de la visión computarizada en aplicaciones de: **Reconocimiento de objetos**, con tareas de clasificación en las imágenes con etiquetas. **Localización de Objetos**, encontrar en un cuadro de imagen el objeto. **Detección de Objetos**, encontrar varios objetos en los cuadros de una imagen. **Segmentación Semántica**, cada píxel pertenece a una clase, por ejemplo, en un cuadro con varios animales, los píxeles en donde se encuentren gatos pertenecen a la clase gatos. **Segmentación de Instancia**, cada píxel pertenece a una instancia de clase, por ejemplo, en un cuadro con varios animales, los píxeles en donde se encuentren gatos pertenecen a un segmento separado de la clase gatos. **Estimación de pose**, determina la orientación de los objetos en el espacio. **Seguimiento de Objetos**, analiza videos para encontrar la trayectoria de movimiento. **Segmentación de Imagen**, encuentra los bordes entre diferentes objetos dentro de una imagen. **Búsqueda y restauración de imágenes, Restauración 3D** [33], [35].

Una red Convolutiva (CNN) está compuesta de: **Capa de Entrada**, primera capa en donde están las entradas, es decir, los tres canales Red, Blue and Green (RGB). **Capas Convolutivas**, en donde sucede las convoluciones, también se la conoce como *mapa de características*, debido a que muestra características específicas que se encuentran en las imágenes de entrada, estas se convierten en entradas de la siguiente capa y se repite en las capas siguientes, las características se almacenan en tensores, término denominado en el aprendizaje profundo como el array multidimensional, luego se crean **Capas totalmente conectadas**, se refiere a las conexiones entre las capas, puede ser vista como varios perceptrones multicapa conectados, con la diferencia que no posee función de activación. (ver Figura 17.a) **Capas no Lineales**, en esta se aplican las funciones de activación previas a la capa de salida, ya sea, sigmoidea, ReLU, tangencial, softmax, entre otras, para la generalización de una función logística en vectores se usa softmax, para valores escalares entre 0 – 1, la función logística cuadrada es útil para clasificaciones cuando la variable del objetivo es discreta. **Capas de Pooling**, se encarga de reducir la muestra de entrada, esta se realiza a través de tensores antes de pasar a la siguiente capa. **Capas de Regularización**, esta capa controla el sobreajuste, incrementando así la velocidad de entrenamiento, dos técnicas muy usadas son el *dropout*, la idea es deshabilitar algunas neuronas en capas anteriores con una misma probabilidad predefinida en cada paso de entrenamiento, como consecuencia durante el entrenamiento estas neuronas se apagan, sin embargo, regresan la próxima vez con sus pesos originales y el *batch*, este ofrece solución al *desplazamiento covariable interno*, que son pequeños cambios en las capas de parámetros que afectan a cada capa y que luego se amplifica generando un problema serio en el aprendizaje profundo, para ello se usa mini capas de batch, entre el producto puntual y no linealidad, esto eleva la tasa de aprendizaje, se puede tener menor cuidado en los pesos iniciales, funciona con la regularización, en el mismo modelo se puede entrenar 14 veces más rápido (Ver Figura 17.b), [33].



a)



b)

Figura 17. a) Red Neuronal Convolutacional, b) Extracción de Características [33], [35].

CONVOLUCIÓN Y POOLING

Las Convoluciones ocurren por canal, una imagen está compuesta de tres canales, rojo, verde y azul, para tratar la imagen primero se separan estos colores, una convolución en realidad es un kernel, por ejemplo, se aplica un kernel de 3x3 (ver Figura 18.a), contiene un número de pesos, que se desplaza a lo largo de la imagen; se suma los pesos de cada píxel, cada uno multiplicado por su correspondiente peso (ver Figura 18.b). El **bias** es un número simple con valor de uno. Los pesos comienzan con valores aleatorios que cambian en la fase de entrenamiento; diferentes pesos, producen diferentes variaciones en la imagen (ver Figura 18.c), el centro del kernel se mueve a la derecha y hacia abajo, pasando por cada píxel, al mismo tiempo puede superponerse sobre otros kernel (ver Figura 18.d). Para definir los límites del kernel se utiliza el padding, que es un píxel extra que siempre será **cero**, este no se suma a los pesos en el kernel, en definitiva, la convolución ayuda a reducir las dimensiones de la imagen (ver Figura 18.e), Para conocer las dimensiones de salida de la convolución [32], se utiliza la fórmula:

$$D = 1 + \frac{W-K+2*P}{S} \quad (8)$$

Donde:

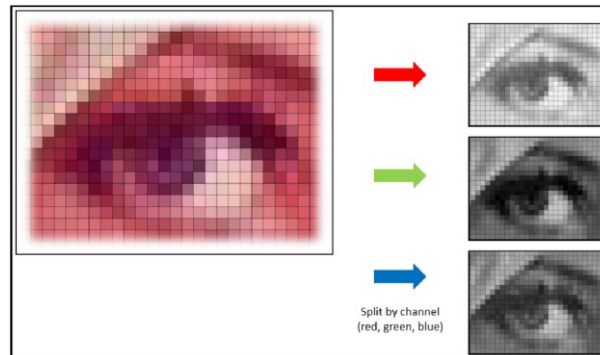
W = ancho de entrada

K = Tamaño del kernel

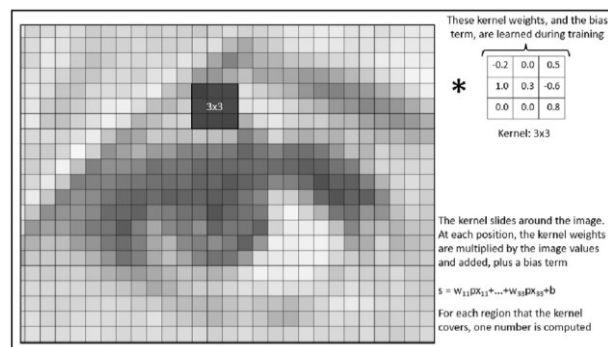
P = Tamaño de padding

S = Tamaño del paso

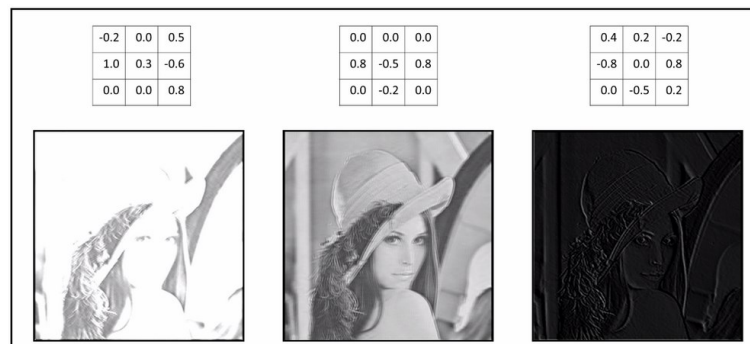
Ya que existe píxeles en donde el kernel no llega, en lugar de variar la longitud del paso, se utiliza el pooling que es una ventana de 2x2, como no usa pesos, no entra en el entrenamiento. La región de pooling es del mismo del tamaño de pool, para evitar el overlapping o solapamiento [32].



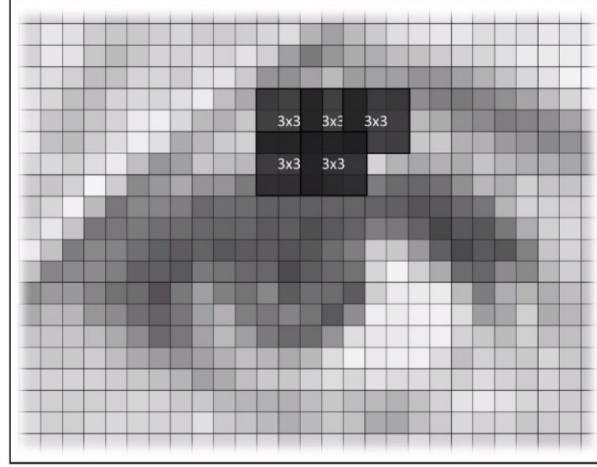
a)



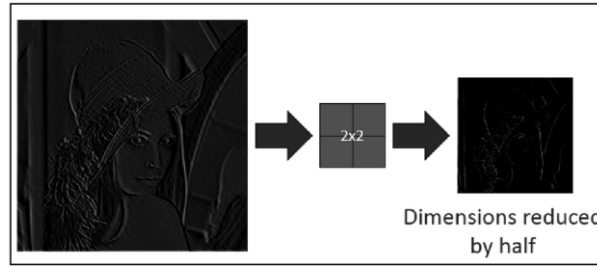
b)



c)



d)



e)

Figura 18. a) Separación de colores, b) Pesos del kernel, c) Kernel diferentes pesos, d) Deslizamiento de kernel, e) Dimensión reducida a la mitad [32], [33].

1.5. Especificidad y Sensibilidad

Este método se utiliza para determinar la validez de la red neuronal. Con ayuda de la matriz de confusión es posible obtener la especificidad y sensibilidad para cada red entrenada.

$$Sensibilidad = \left(\frac{VP}{enfermos} \right) * 100 = \left(\frac{VP}{VP+FN} \right) * 100 \quad (9)$$

$$Espeficidad = \left(\frac{VN}{sanos} \right) * 100 = \left(\frac{VN}{VN+FN} \right) * 100 \quad (10)$$

$$VPP = \left(\frac{VP}{totalTest+} \right) * 100 = \left(\frac{VP}{VP+FP} \right) * 100 \quad (11)$$

$$VPN = \left(\frac{VN}{totalTest-} \right) * 100 = \left(\frac{VN}{VN+FN} \right) * 100 \quad (12)$$

Donde:

VP = Verdaderos Positivos.

VN = Verdaderos Negativos.

FP = Falsos Positivos.

FN = Falsos Negativos.

VPP = Valor Predictivo Positivo

VPN = Valor Predictivo Negativo

Una red 100% sensible indica que se descarta la enfermedad, mientras que una red 100% específica, confirma el diagnóstico, para este trabajo se busca la especificidad.

Valor Predictivo Positivo: Es la probabilidad de que un paciente con un resultado positivo tenga dicha enfermedad.

Valor Predictivo Negativo: Es la probabilidad de que un paciente con un resultado negativo no tenga dicha enfermedad.

1.6. Transferencia de Conocimiento.

La transferencia de conocimiento se refiere a usar una red pre entrenada, con el fin de obtener una mejor predicción del clasificador que se está desarrollando, similar al momento que un profesor transfiere sus conocimientos a un estudiante, de esta manera se toma un modelo ya desarrollado para luego modificar ciertas capas y posteriormente entrenarlo con las muestras. Cada año se realiza la competencia “ImageNET Large Scale Visual Recognition Challenge (ILSVRC)”, el ganador de dicha competición se determina de acuerdo con la mejor tasa de entrenamiento, de aquí nacen varios de los modelos que se usan hoy en día cuando se trata de transferencia de conocimiento, entre ellos están:

Resnet 50: Residual Network o abreviado Resnet, esta estructura fue ganadora de la competencia ILSVRC en 2015 desarrollada por Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Su, está compuesta de 34 capas, principalmente inspirada de la VGG-19 [36]

Inception: Presentada en 2014 por Min Lin et al, en aquel año fue la red más larga y profunda que se había desarrollado, compuesta de módulos repetidos Inception, es decir, capa de entrada, capa convolucional 1x1, capa convolucional 3x3, capa convolucional 5x5, capa de max pooling y capa de concatenación, siendo esta la representación de un módulo sencillo [37].

Xception: Presentada por Google, siendo una versión mejorada al reemplazar los módulos Inception por convoluciones separables profundas, su principal ventaja es que reduce la carga computacional, en desafíos clásicos llega a superar otros modelos como VGG16, Resnet e Inception V3 [38].

VGG16: Fue presentada en la competición de ILSVRC de 2014, desarrollada por K. Simonyan y Zisserman de la Universidad de Oxford, compuesta de 16 capas, llegando a tener una precisión de 92.7% [39], [40].

CAPÍTULO 2

2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

El objetivo de esta sección es dar a conocer el tratamiento aplicado en las Imágenes de Fondo de Ojo adquiridas de diversas fuentes [41], así como el desarrollo del algoritmo de entrenamiento para la red neuronal.

2.1. Base de Datos

La mayor parte de las imágenes destinadas al entrenamiento de la red Neuronal han sido tomadas de diferentes fuentes, entre ellas están:

Dristhi – GS1

Consiste en 101 imágenes retinales, la edad de los pacientes oscila entre 40 – 80 años, con un número igual entre hombres y mujeres, tienen una resolución de 2896 x 1944 px [41].

HRF

High Resolution Fundus, contiene 45 imágenes, 15 de pacientes con retinopatía diabética, 15 de pacientes sanos y 15 de pacientes con Glaucomatoma, las imágenes tienen una resolución de 3504 x 2336 px [41].

CHASEDB

Base de Datos de Child Heart and Health Study in England (CHASE), de 200 escuelas primarias de Londres, Brimingham, y Leicester, tiene una resolución de 1280 x 960 px; las imágenes no disponen de iluminación uniforme y contraste pobre de los vasos sanguíneos, tiene un total de 28 imágenes [41].

DRIONS – DB

Digital Images for Optic Segmentation, tiene un total de 110 imágenes retinales de 600 x 400 px, revisadas por dos especialistas, la edad de los pacientes fue entre 53 años con un 46.2% de hombres y 53.8% de mujeres [41].

RIM ONE

Consta de imágenes de evaluación del nervio óptico y detección del Glaucoma, compuesta de 772 imágenes separadas en pacientes sanos y aquellos que existe la sospecha que tiene Glaucoma, se encuentran agrupadas en 3 carpetas RIM ONE r1, RIM ONE r2, y RIM ONE r3 [41].

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

ACRIMA

Es una base de datos publicada reciente que consta de 396 imágenes de pacientes con Glaucoma y 309 sanos. Las anotaciones fueron hechas por expertos con 8 años de experiencia [42].

Tabla 2. Cantidad de Imágenes en la Base de Datos.

Base de Datos Publicas.			
Nombre	Sospecha	Sano	Total
Drishti-GS1	70	31	101
HRF	15	15	30
CHASEDB	-	-	28
DRIONS-DB	-	-	110
RIM ONE r1	40	118	158
RIM ONE r2	200	255	455
RIM ONE r3	74	85	159
ACRIMA	396	309	705
	795	813	1618

Al revisar las imágenes de cada Base de Datos, algunas de ellas fueron descartadas debido a:

- Imagen Borrosa
- Imagen oscura

Las Imágenes descartadas pertenecen a:

RIM ONE r3.

Contiene algunas imágenes borrosas y otras muy oscuras, lo que llevo a descartar un total de 5 imágenes de pacientes sanos y 8 de pacientes con sospecha de Glaucoma.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

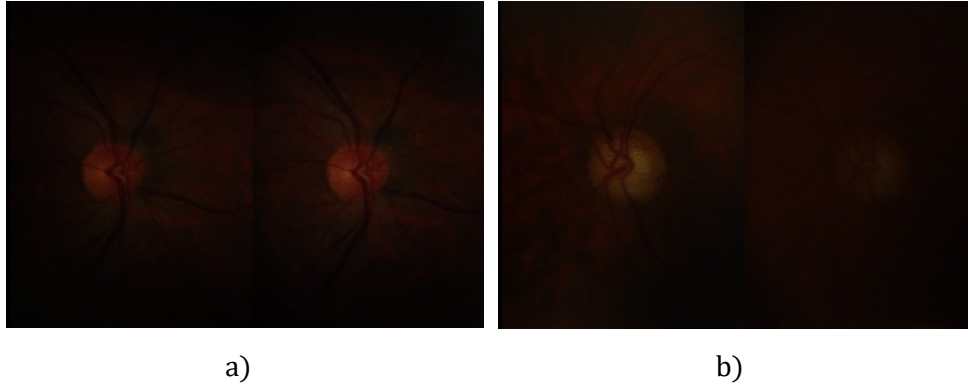


Figura 19. a) Imagen de RIM ONE Sano Descartada, b) Imagen RIM ONE Sospecha Descartada [32], [33].

2.2. Software y Hardware

El software que se ha tomado para realizar el programa es Python, brinda varias herramientas relacionadas con machine learning y Deep learning, además de varias librerías como: OpenCV, Keras entre otras.

Dado que el software necesita un interpretador, se optó por usar *Jupyter Notebook*, este es un software que puede correr en el navegador, además es una extensión de Python, por lo que no necesita descargar programas adicionales en el pc, y no necesita licencia, por lo que es software libre y cumple con lo que se busca.

Las características del computador utilizado para el desarrollo del proyecto son las siguientes:

- **CPU:** Intel Core i7-7700HQ
- **GPU:** NVIDIA GeForce GTX 1060 (3GB GDDR5)
- **DISPLAY:** 15.6", Full HD (1920 x 1080), IPS + G-Sync
- **ALMACENAMIENTO:** 1000GB HDD
- **RAM:** 1x 4GB + 8GB DDR4 (soldada)
- **HDD:** 1TB
- **OS:** Windows 10 Home

2.3. Pre-Procesamiento

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

TRATAMIENTO DE LAS IMÁGENES

Todas las imágenes se han agrupado de acuerdo con la base de datos de la que fue tomada, dentro de cada carpeta se encuentra separadas por imágenes de pacientes sanos y por sospecha de Glaucoma (ver Figura 20).

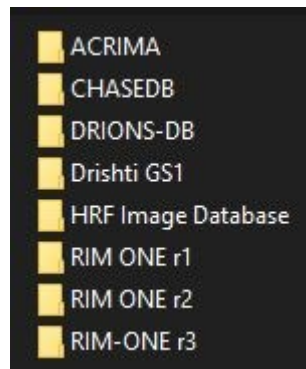


Figura 20. Base de Datos para CNN.

REGIÓN DE INTERÉS.

Las bases de datos contienen imágenes que pertenecen a un Tomógrafo de Coherencia óptica (OCT), dado que contienen todo el globo ocular, se debe recortar el disco óptico, ya que es nuestra región de interés.

En varias de las imágenes, el disco óptico se encuentra en distintas ubicaciones (ver Figura 21), ya sea, en el centro, izquierda, y derecha, se ha decidido crear un programa para realizar estos recortes de manera más rápida, se ha separado por carpetas que corresponde a cada base de datos y se aplica dos recortes a todas las imágenes, el primero tiene como objetivo normalizar la imagen a dimensiones cercanas a un cuadrado, y el segundo recorte está centrado en las ubicación de la región de interés.

El algoritmo funciona de la siguiente manera, Inicialmente se toma el directorio con todas las imágenes en su interior, luego se recorta para eliminar las zonas negras alrededor del globo ocular, se continua con un segundo recorte, pero esta vez enfocándose en el disco óptico, este proceso se realiza para cada carpeta (ver Figura 20), dado que el disco se encuentra en diferentes posiciones se varían los valores en el segundo recorte según sea necesario. El código utilizado para obtener estas imágenes se encuentra en el Anexo A.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

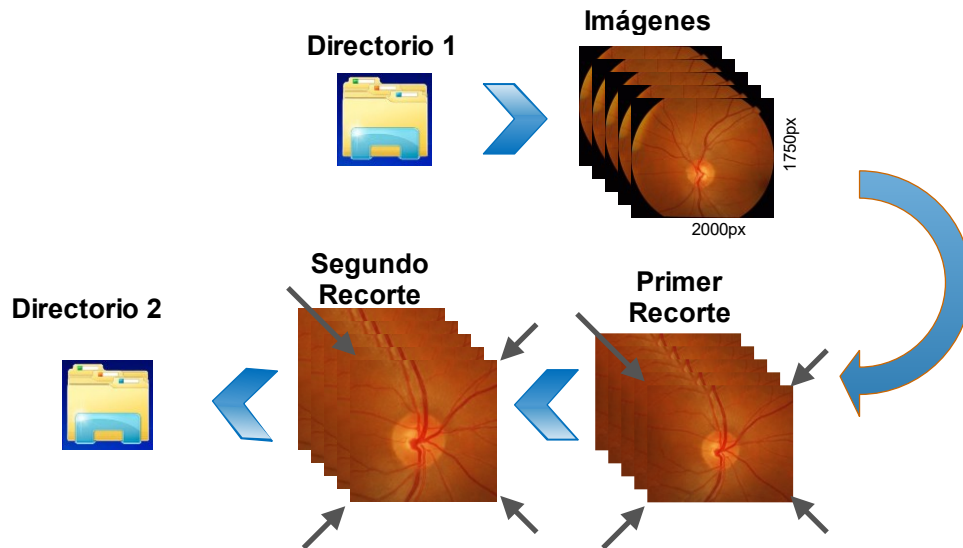
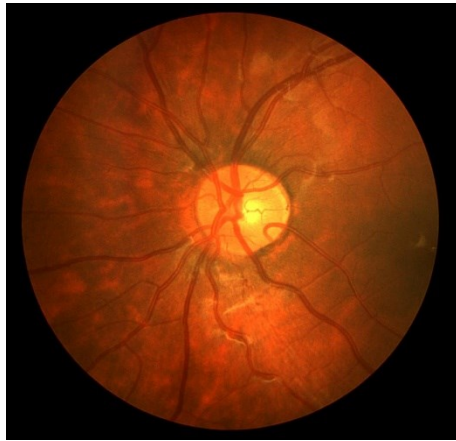


Figura 21. Algoritmo utilizado para recortar las imágenes.

A continuación, se muestran la imagen original (imágenes de la izquierda), y el resultado del doble recorte sobre la zona del disco óptico (imágenes de la derecha).

Los recortes antes mencionados fueron realizados en el mismo orden que se organizó las carpetas (ver Figura 20), las imágenes de CHADSEB (ver Figura 22), DRIONS (ver Figura 23), DRISTHI, se realizó dos programas ya que las imágenes fueron separadas por sano (ver Figura 24.a, Figura 24.b) y sospecha (ver Figura 24.c, Figura 24.d), HRF Imagen Data, a diferencia del anterior existen imágenes en donde el disco óptico está ubicado a la derecha e izquierda en los pacientes sanos (ver Figura 25.a, Figura 25.c); de igual forma para los pacientes con sospecha (ver Figura 25.e, Figura 25.g) y, por último, RIM-ONE r3, las imágenes de esta contenían dos discos ópticos en cada una, correspondiendo al mismo globo ocular, por esta razón se toma la imagen izquierda, se han separado en pacientes sano (ver Figura 26.a) y pacientes con sospecha (ver Figura 26.c). No se aplica en las carpetas RIM ONE r1 y RIM ONE r2, debido a que las imágenes en ellas ya se encuentran recortadas.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE



a)

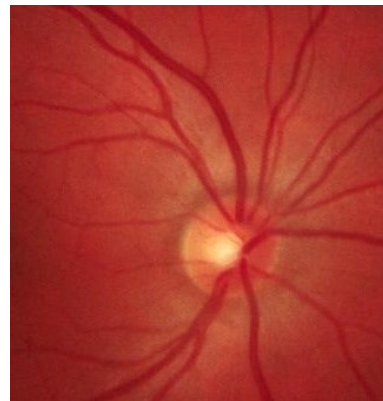


b)

Figura 22. Base de Imágenes de CHADSEB a) Imagen Original, b) Imagen Recortada.



c)



d)

Figura 23. Base de Imágenes de DRIONS a) Imagen Original, b) Imagen Recortada.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

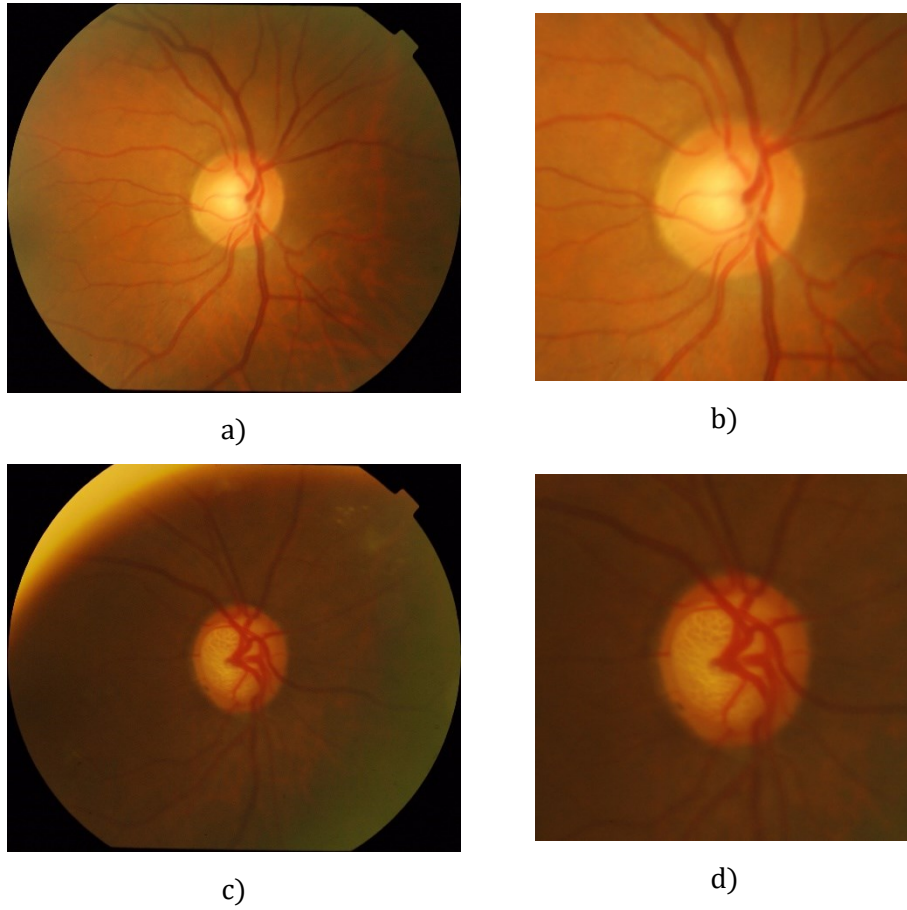
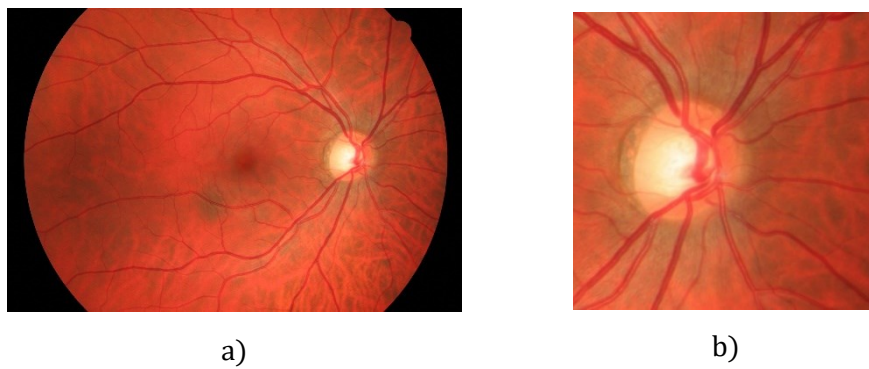


Figura 24. Base de Imágenes de Drishti- G1, Paciente Sano a) Imagen Original, b) Imagen Recortada; Paciente con Sospecha c) Imagen Original, d) Imagen Recortada.



CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

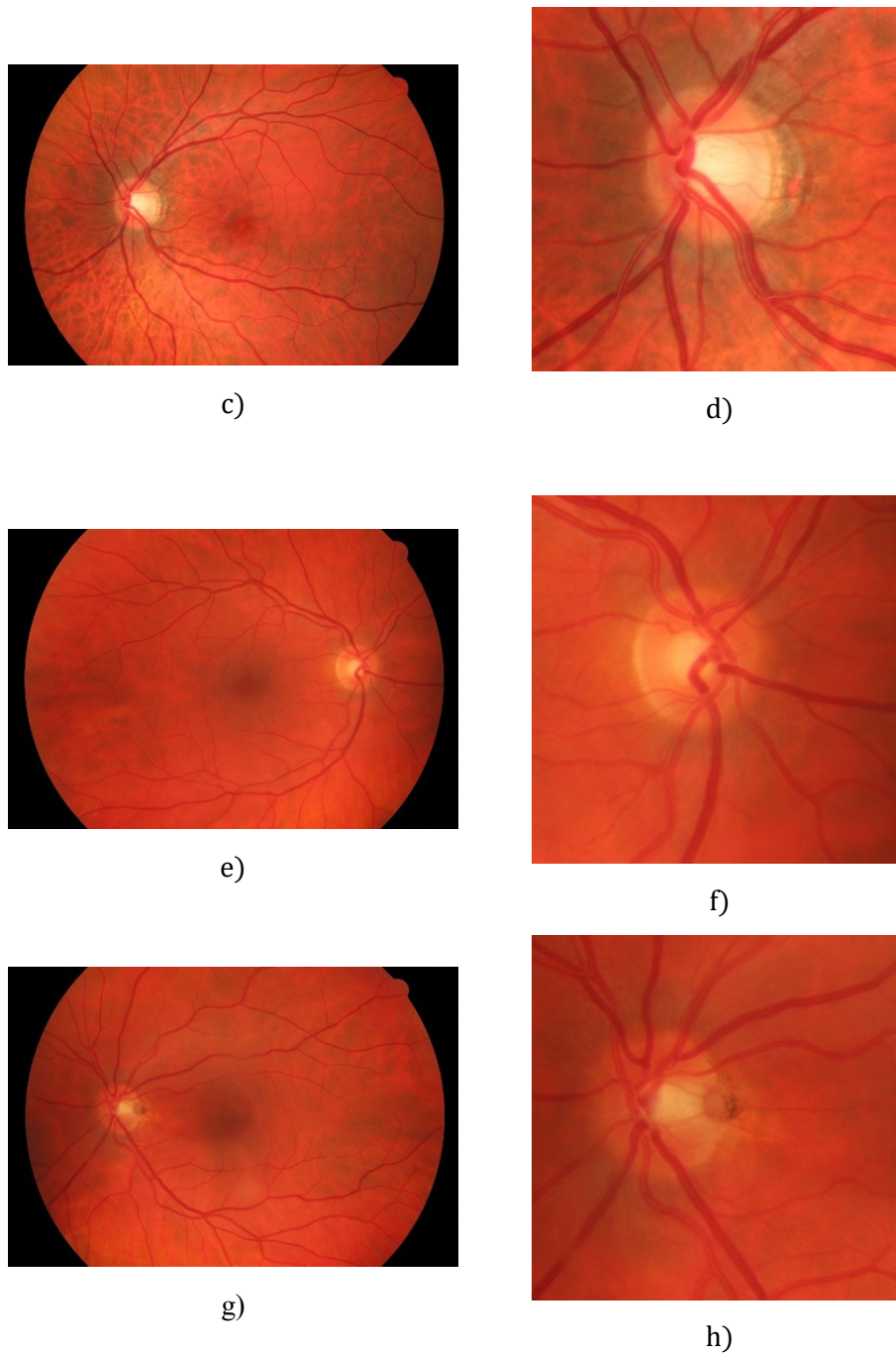


Figura 25. Base de Imágenes de HRF Imagen Data, Derecha Paciente Sano a) Imagen Original, b) Imagen Recortada; Izquierda Paciente Sano c) Imagen Original, d) Imagen Recortada, Derecha Paciente con Sospecha e) Imagen Original, f) Imagen Recortada, Izquierda Paciente con Sospecha g) Imagen Original, h) Imagen Recortada.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

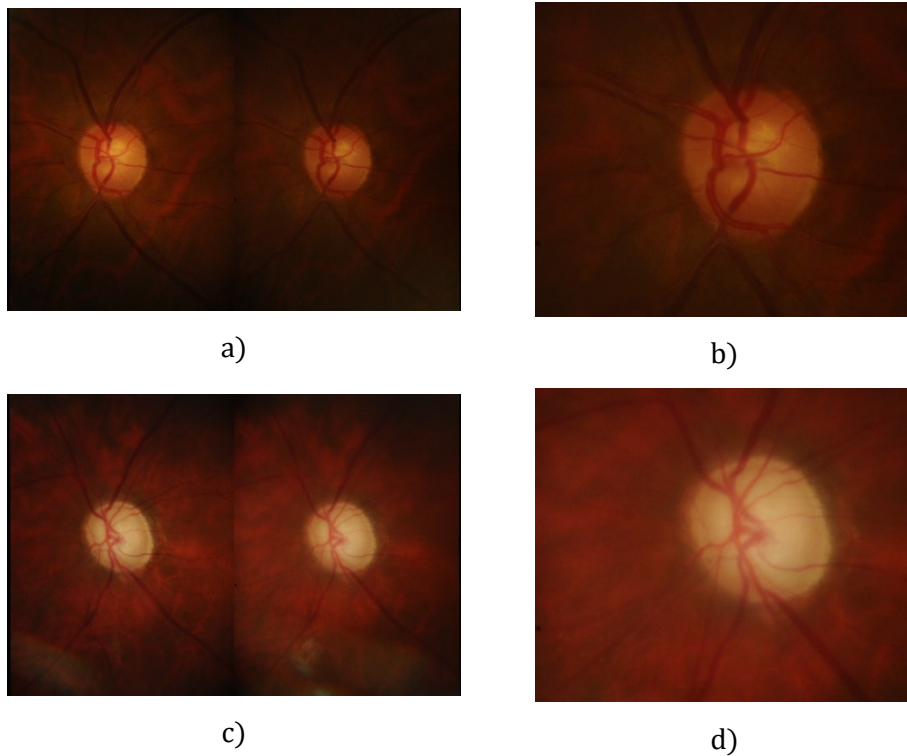


Figura 26. Base de Imágenes de RIM-ONE r3, Paciente Sano a) Imagen Original, b) Imagen Recortada; Paciente con Sospecha c) Imagen Original, d) Imagen Recortada.

REDIMENSIONAMIENTO DE IMÁGENES

Para que sean muestras optimas al momento de realizar el entrenamiento de la red neuronal, en primera instancia deben estar todas con las mismas dimensiones, ya que estas se convertirán en nuestras muestras de entrada, para ello se ha elegido que todas sean de **224 x 224**, con este tamaño la carga sobre el CPU del PC será menor, ya que mientras mayor sea el tamaño, incrementa el número de muestras de entrada, por ende, el tiempo de entrenamiento también aumenta. Para ello se utilizó el código en el Anexo B.

CAPÍTULO 2. PROCESAMIENTO DE IMÁGENES Y SOFTWARE

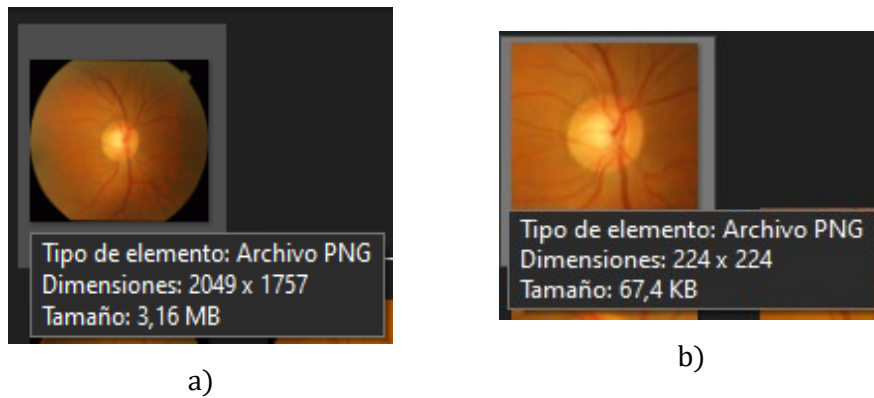


Figura 27. Imágenes de recorte de región de interés, a) imagen Original, imagen recortada y redimensionada.

Si se desea cambiar las dimensiones basta con cambiar los valores de ancho y largo, lo mismo para la ubicación de otra dirección en donde se guardarán las imágenes basta con agregar una nueva dirección.

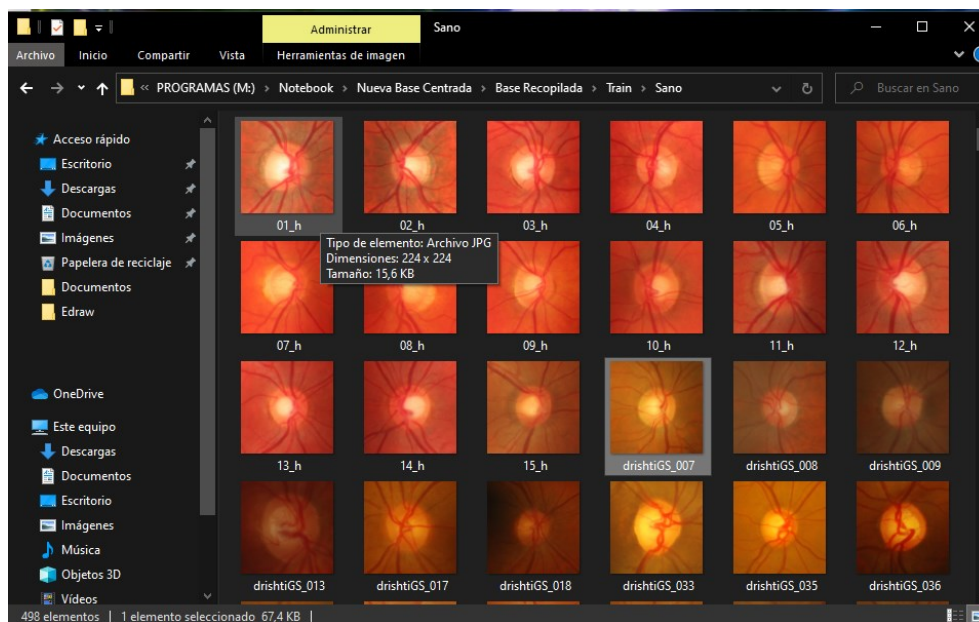


Figura 28. Imágenes redimensionadas.

CAPÍTULO 3

3. IMPLEMENTACIÓN Y ANÁLISIS DE RESULTADOS

Las imágenes se han clasificado en dos carpetas, **736** en la carpeta de Sospecha y **782** en la de Sano, que posteriormente serán usadas para el entrenamiento de la red convolucional. Reservando **60** imágenes, de las cuales la mitad son de pacientes sanos y las restantes con Glaucoma, serán usadas para la prueba en la matriz de confusión.

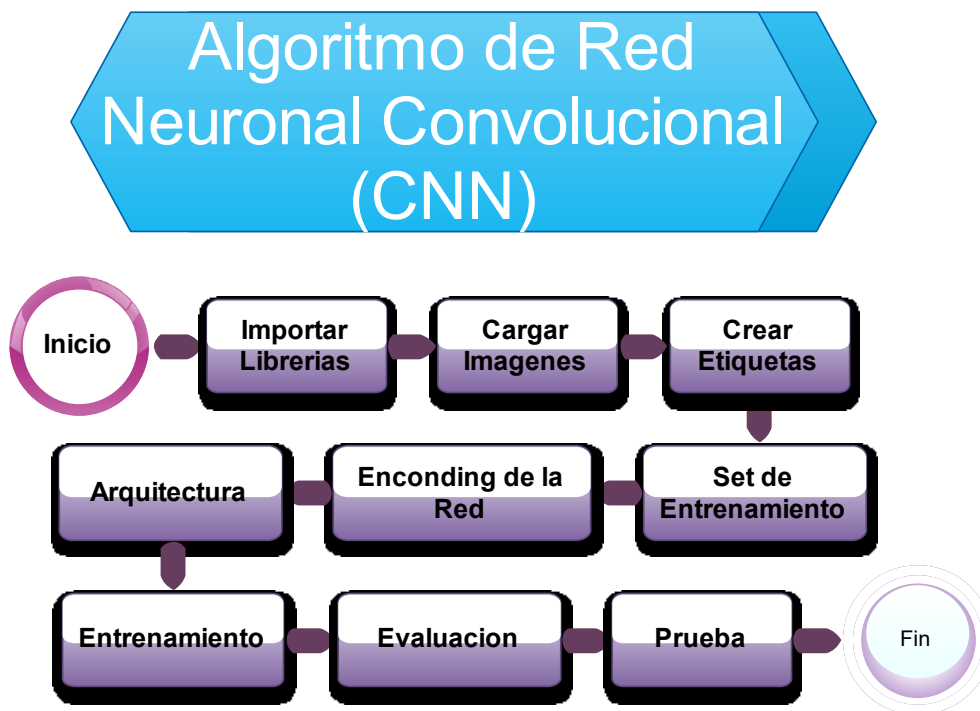


Figura 29. Algoritmo de Red Neuronal Convolucional (CNN).

Inicialmente se **importan las librerías** a utilizar: numpy (librería especializada en cálculo numérico y análisis de datos en grandes cantidades), matplotlib (permite crear gráficos en dos dimensiones), cv2 (librería de OpenCv), keras (permite la creación de redes neuronales), seguido se **carga las imágenes**, para ello se usa un contador que indique la cantidad de imágenes contenidas dentro de la carpeta, con ello se determinan las cantidad de datos, en este caso la cantidad de imágenes serían 1518, luego se designa las **etiquetas** para los grupos de clasificación

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

que son dos: Sano y Sospecha, seguido en el **set de entrenamiento** se debe separar los datos 80% para entrenamiento y, el 20% restante para validación, posteriormente inicia el **encoding de red**, primero se normalizan los datos para que estén entre 0 - 1, seguido se realiza una conversión de las etiquetas categóricas a etiquetas binarias (one hot), es decir, se les asigna un vector de clasificación, para Sano sería [1,0] y, para Sospecha [0,1]. Ahora se elige la **arquitectura de la red**, se designa el tamaño de kernel, dropout, tamaño de pool, batch, así como, la función de activación para cada capa convolucional, luego se procede al **entrenamiento de la red**, el tiempo de entrenamiento dependerá de la cantidad de datos de entrada, capas convolucionales y cantidad de conexiones en la red, antes de terminar, se realiza una evaluación de la red entrenada, para finalmente realizar una **prueba** al clasificar una imagen y también aplicar la matriz de confusión con las 60 imágenes que fueron separadas (Ver Figura 29).

En los apartados siguientes se muestran varias redes convolucionales, inicialmente se compone de una capa convolucional, seguido de dos capas, luego cuatro capas y finalmente cinco capas convolucionales; de esta manera se puede conocer como responden estas arquitecturas en el entrenamiento de Imágenes de Fondo de Ojo, adicionalmente se usa dos optimizadores el primero es Adagrad, un optimizador bastante usado para este tipo de redes, sin embargo, últimamente existe otro optimizador llamado Adam que ha tomado bastante acogida, debido a que es más preciso que el antes mencionado, por ello se compara los resultados de estos dos optimizadores.

El código completo se encuentra en el Anexo C, y en los siguientes apéndices se muestran solamente la arquitectura de red para cada caso.

3.1. CNN con Una Capa Convolucional

Esta red convolucional (CNN) está conformada de una capa convolucional, a continuación, se mencionan los diferentes parámetros utilizados (Ver Tabla 3).

Tabla 3 Parámetros para una sola capa Convolucional

Parámetro	Valor
Datos Entrenamiento	1214
Datos Prueba	304
Épocas	100
Tamaño Imagen	224x224
Batch	16

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Conv1	(3,3)
Tamaño Conv1	32
Tamaño Pool	(2,2)
Optimizador1	Adagrad
Optimizador2	Adam
Dropout	0.25
Conexiones Capa Densa	256

La cantidad de Imágenes del entrenamiento son 1518 (Ver Figura 30), compuesta de dos clases: Sano (782) y Sospecha (736), luego se separan las imágenes para entrenamiento (1214) y para validación (304), se puede ver las primeras imágenes del set entrenamiento y validación (ver Figura 30), luego de asignar las respectivas etiquetas a cada imagen se procede a la arquitectura de la red (Anexo C), inicialmente las imágenes son de: 224x224, que luego de pasar por el max pooling y el dropout, se reducen a la mitad (112x112), seguido se aplanan en la salida y se asigna 256 conexiones, como resultado se tiene 102.762.114 de parámetros (ver Figura 31).

Para el optimizador de Adagrad el entrenamiento finaliza luego de 6 minutos 40 segundos (ver Figura 32.a), entonces se guarda con el nombre: “biomedical_Adagrad_Acrima1.hdf5”, además al evaluar la red se tiene una precisión de 0.87 y una pérdida de 0.33 (ver Figura 33.a), es posible observar el comportamiento de la precisión (ver Figura 34.a) y pérdida de la red (ver Figura 35.a), luego se obtiene las imágenes que se han etiquetado dando como resultado 266 correctas (ver Figura 36.a), y 38 incorrectas (ver Figura 37.a), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 38.a), y se obtiene sano para una imagen de paciente sano.

En cambio en el optimizador de Adam el entrenamiento finaliza luego de 8 minutos 20 segundos (ver Figura 32.b), entonces se guarda con el nombre: “biomedical_Adam_Acrima1.hdf5”, además al evaluar la red se tiene una precisión de 0.81 y una pérdida de 1.48 (ver Figura 33.b), es posible observar el comportamiento de la precisión (ver Figura 34.b) y pérdida de la red (ver Figura 35.b), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 249 correctas (ver Figura 36.b), y 55 incorrectas (ver Figura 37.b), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 38.b), y se obtiene sano para una imagen de paciente sano.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

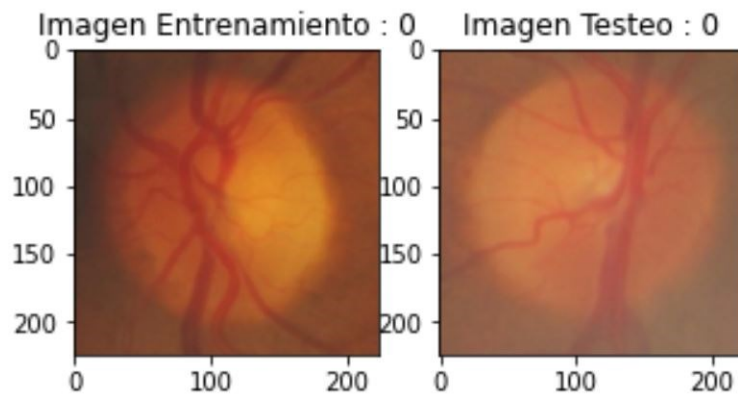


Figura 30. Primeras imágenes de Entrenamiento y prueba o validación.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
leaky_re_lu (LeakyReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
flatten (Flatten)	(None, 401408)	0
dense (Dense)	(None, 256)	102760704
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514
Total params: 102,762,114		
Trainable params: 102,762,114		
Non-trainable params: 0		

Figura 31. Arquitectura de la Red de una sola capa Convolutiva.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```

8395
Epoch 95/100
61/61 [=====] - 4s 70ms/step - loss: 0.2557 - accuracy: 0.9082 - val_loss: 0.3275 - val_accuracy: 0.
8601
Epoch 96/100
61/61 [=====] - 4s 71ms/step - loss: 0.2208 - accuracy: 0.9138 - val_loss: 0.3119 - val_accuracy: 0.
8642
Epoch 97/100
61/61 [=====] - 4s 71ms/step - loss: 0.2527 - accuracy: 0.9004 - val_loss: 0.3186 - val_accuracy: 0.
8560
Epoch 98/100
61/61 [=====] - 4s 71ms/step - loss: 0.2550 - accuracy: 0.8986 - val_loss: 0.3082 - val_accuracy: 0.
8477
Epoch 99/100
61/61 [=====] - 4s 71ms/step - loss: 0.2430 - accuracy: 0.8986 - val_loss: 0.3223 - val_accuracy: 0.
8601
Epoch 100/100
61/61 [=====] - 4s 71ms/step - loss: 0.2580 - accuracy: 0.8938 - val_loss: 0.3126 - val_accuracy: 0.
8683

```

a)

```

7901
Epoch 95/100
61/61 [=====] - 5s 75ms/step - loss: 0.0278 - accuracy: 0.9881 - val_loss: 0.9917 - val_accuracy: 0.
8436
Epoch 96/100
61/61 [=====] - 5s 75ms/step - loss: 0.0674 - accuracy: 0.9814 - val_loss: 0.9039 - val_accuracy: 0.
8519
Epoch 97/100
61/61 [=====] - 5s 76ms/step - loss: 0.0307 - accuracy: 0.9847 - val_loss: 0.9103 - val_accuracy: 0.
8230
Epoch 98/100
61/61 [=====] - 5s 77ms/step - loss: 0.1407 - accuracy: 0.9700 - val_loss: 1.7338 - val_accuracy: 0.
7984
Epoch 99/100
61/61 [=====] - 5s 76ms/step - loss: 0.3504 - accuracy: 0.9289 - val_loss: 1.1861 - val_accuracy: 0.
8436
Epoch 100/100
61/61 [=====] - 5s 76ms/step - loss: 0.0254 - accuracy: 0.9906 - val_loss: 1.4650 - val_accuracy: 0.
8354

```

b)

Figura 32. Últimas épocas de entrenamiento a) Optimizador Adagrad, b) Optimizador Adam.

```

10/10 [=====] - 0s 32ms/step - loss: 0.3396 - accuracy: 0.8750

```

a)

```

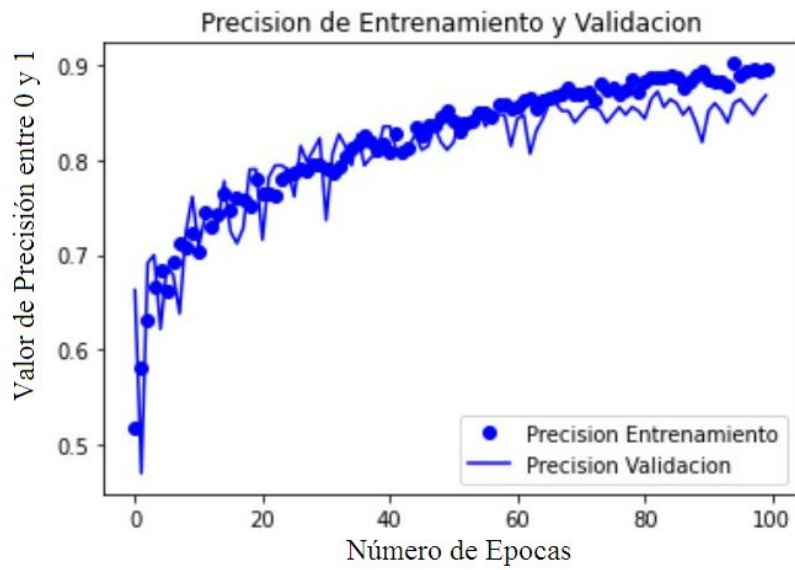
10/10 [=====] - 0s 33ms/step - loss: 1.4876 - accuracy: 0.8191

```

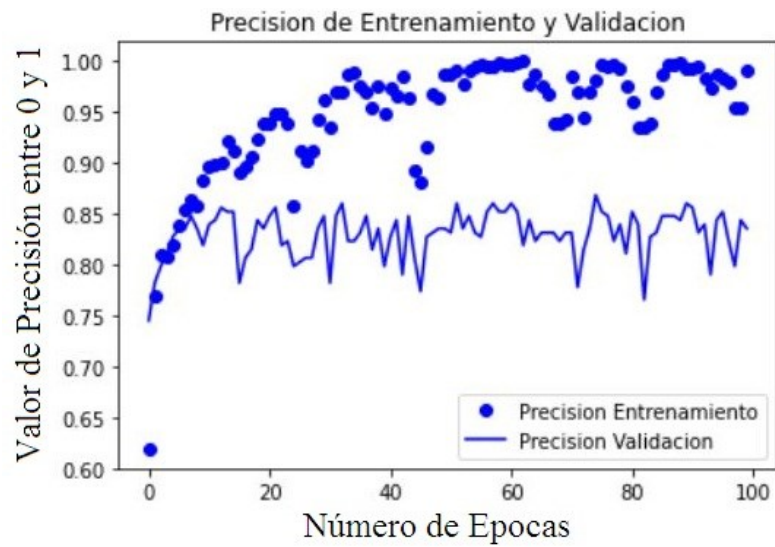
b)

Figura 33. Valores de evaluación de la CNN con a) Adagrad, b) Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



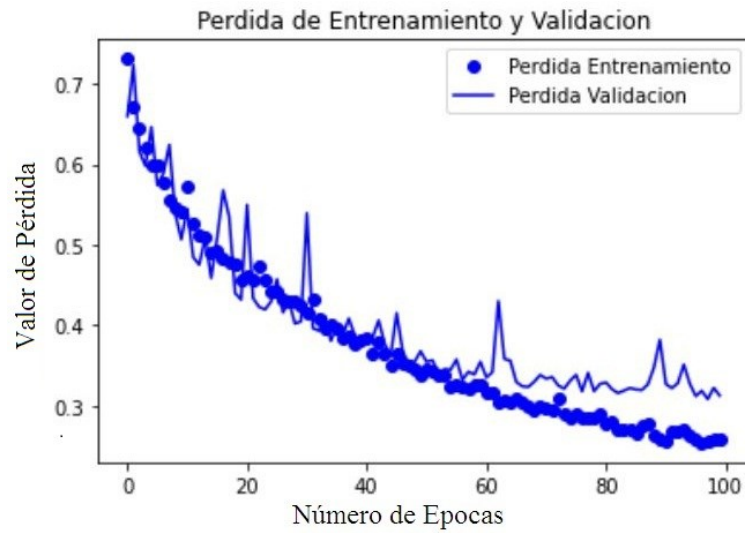
a)



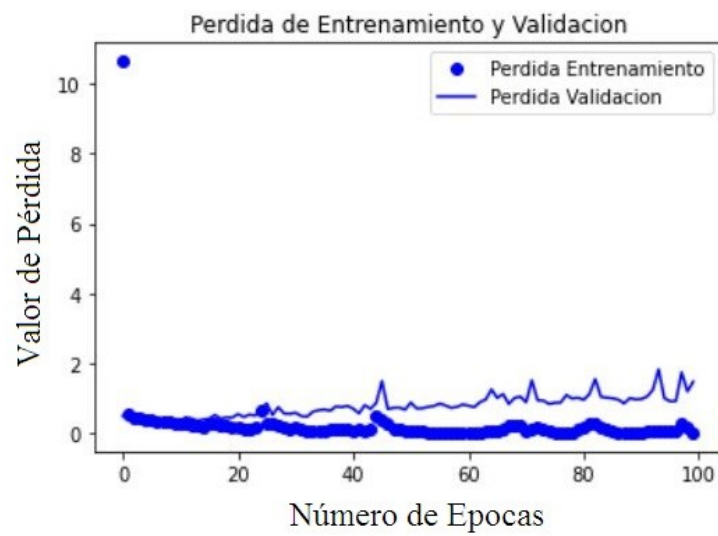
b)

Figura 34. Gráfica de precisión de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

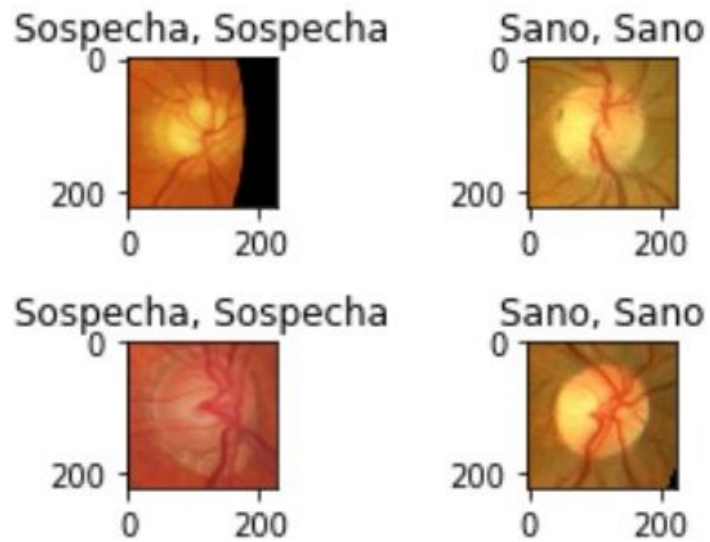


b)

Figura 35. Gráfica de Pérdida de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

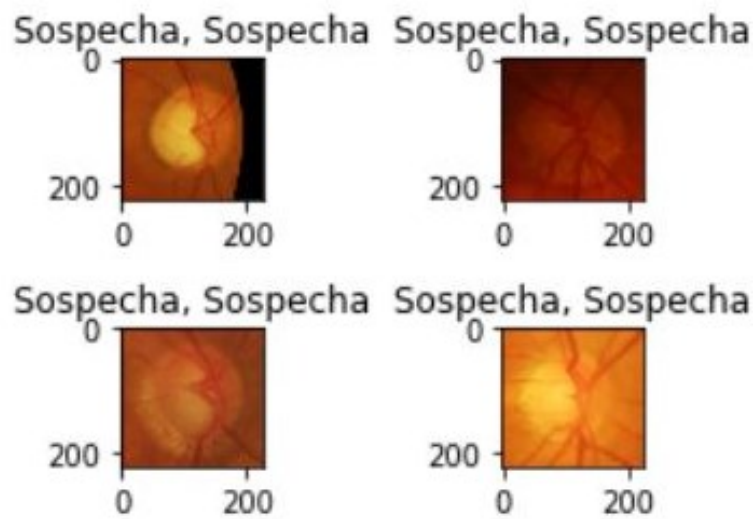
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 266 etiquetas correctas



a)

Encontradas 249 etiquetas correctas

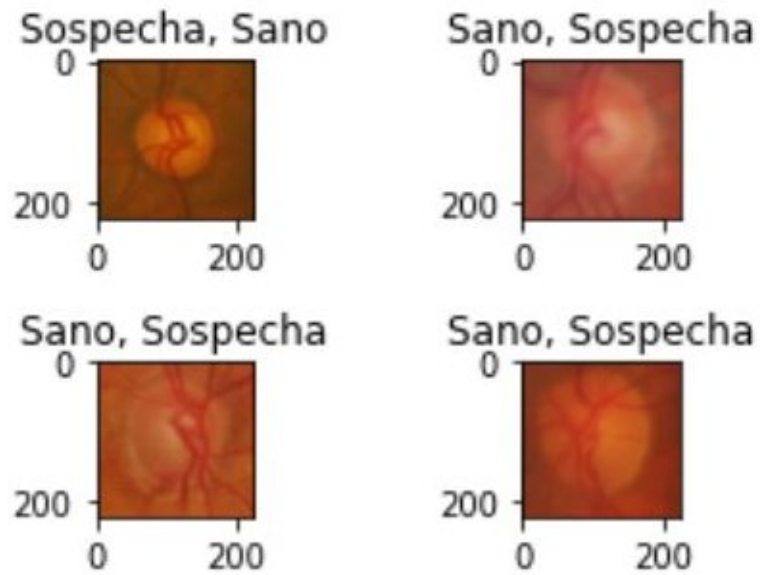


b)

Figura 36. Etiquetas Clasificadas Correctamente, a) Optimizador Adagrad, b) Optimizador Adam.

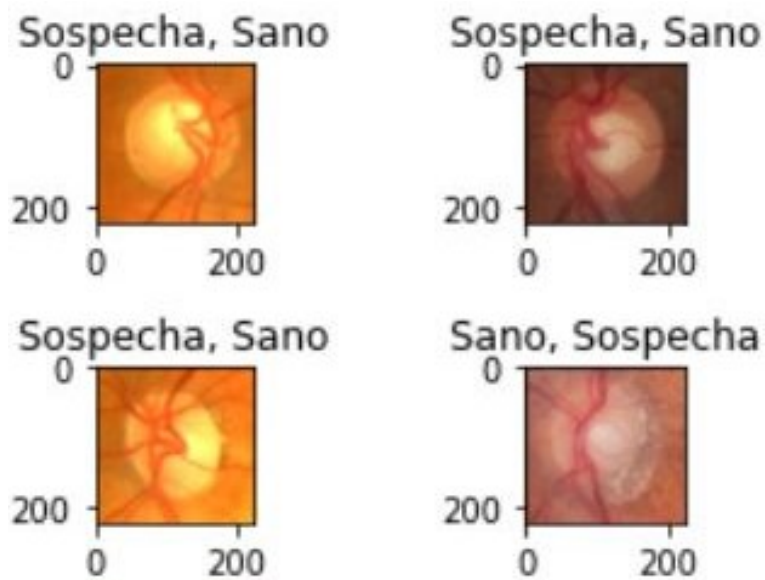
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 38 etiquetas incorrectas



a)

Encontradas 55 etiquetas incorrectas



b)

Figura 37. Etiquetas Clasificadas Incorrectamente, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Nueva Base Centrada/Base 224 Acrima/Test/Sano/2.jpg Sano

a)

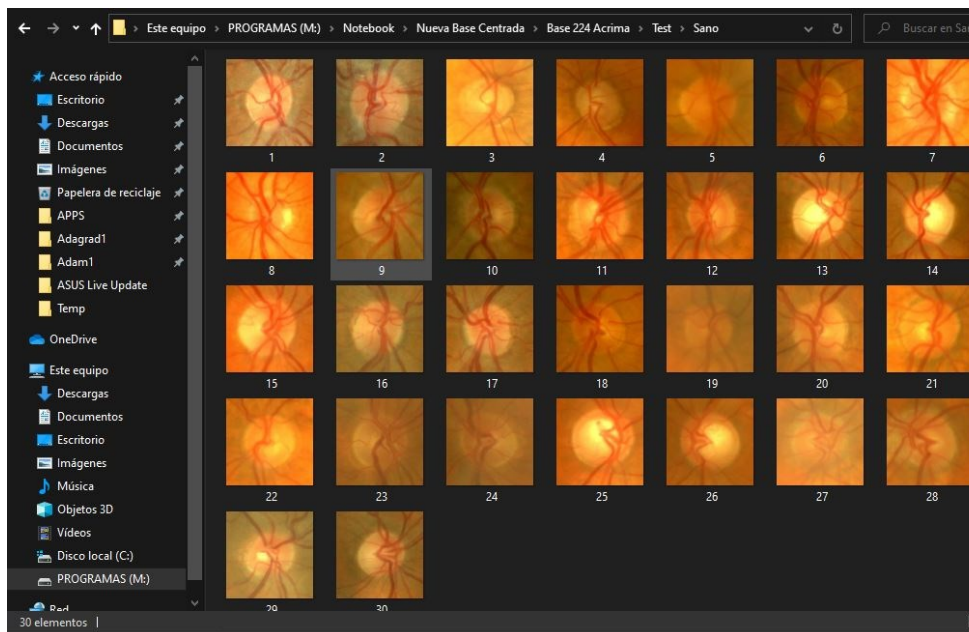
Nueva Base Centrada/Base 224 Acrima/Test/Sano/1.jpg Sano

b)

Figura 38. Prueba de la red en una imagen, a) Optimizador Adagrad, b) Optimizador Adam.

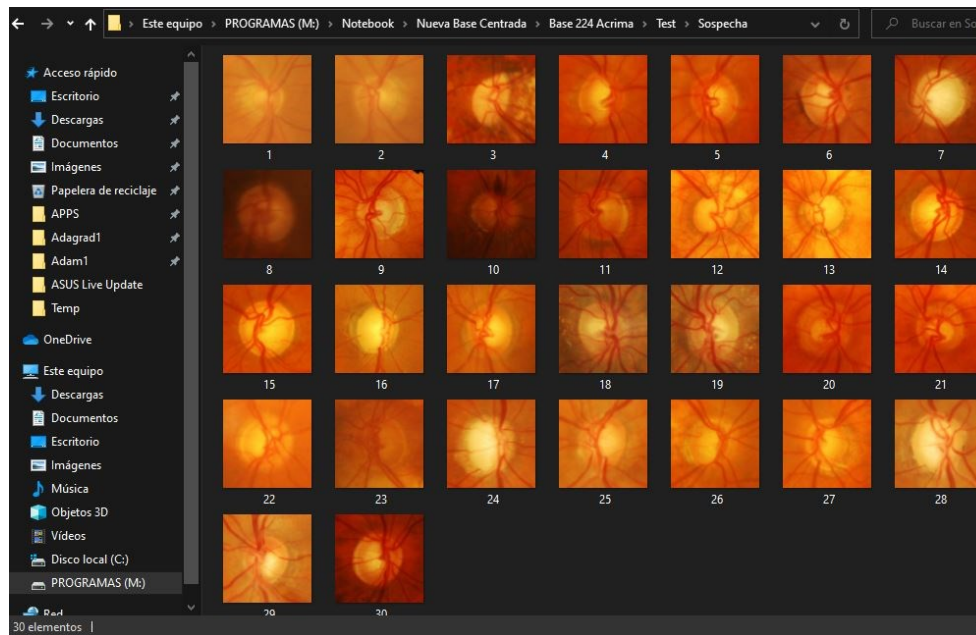
Matriz de Confusión

La Matriz de Confusión permite tener una mejor idea de la precisión que tiene la red neuronal para clasificar la imagen biomédica del paciente en sano o enfermo. Para ello se aplica la predicción con 30 imágenes para Sano (ver Figura 39.a) y 30 para sospecha (ver Figura 39.b).



a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



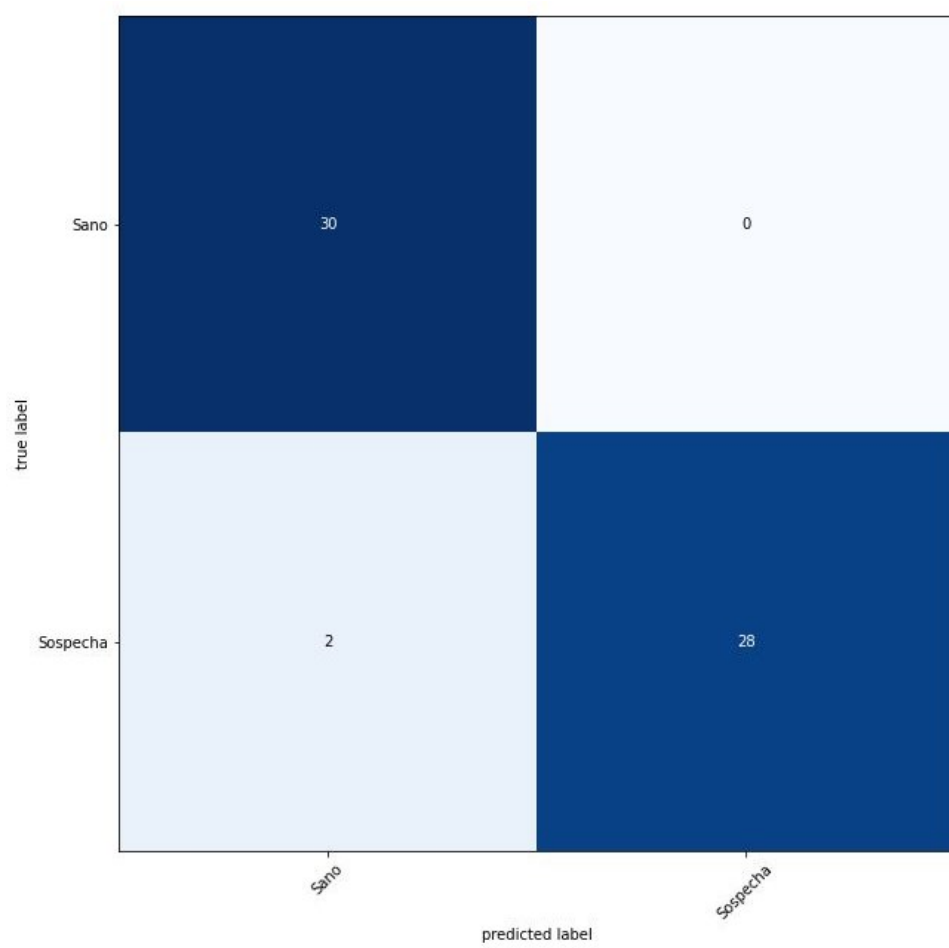
b)

Figura 39. a) Carpeta test Sano, b) Carpeta test Sospecha.

En la representación gráfica de la Matriz de confusión del Optimizador de Adagrad, se puede ver las imágenes de la clase sospecha han tenido una predicción correcta de 28 imágenes, sin embargo, en sano todas las 30 imágenes han sido correctas (ver Figura 40.a).

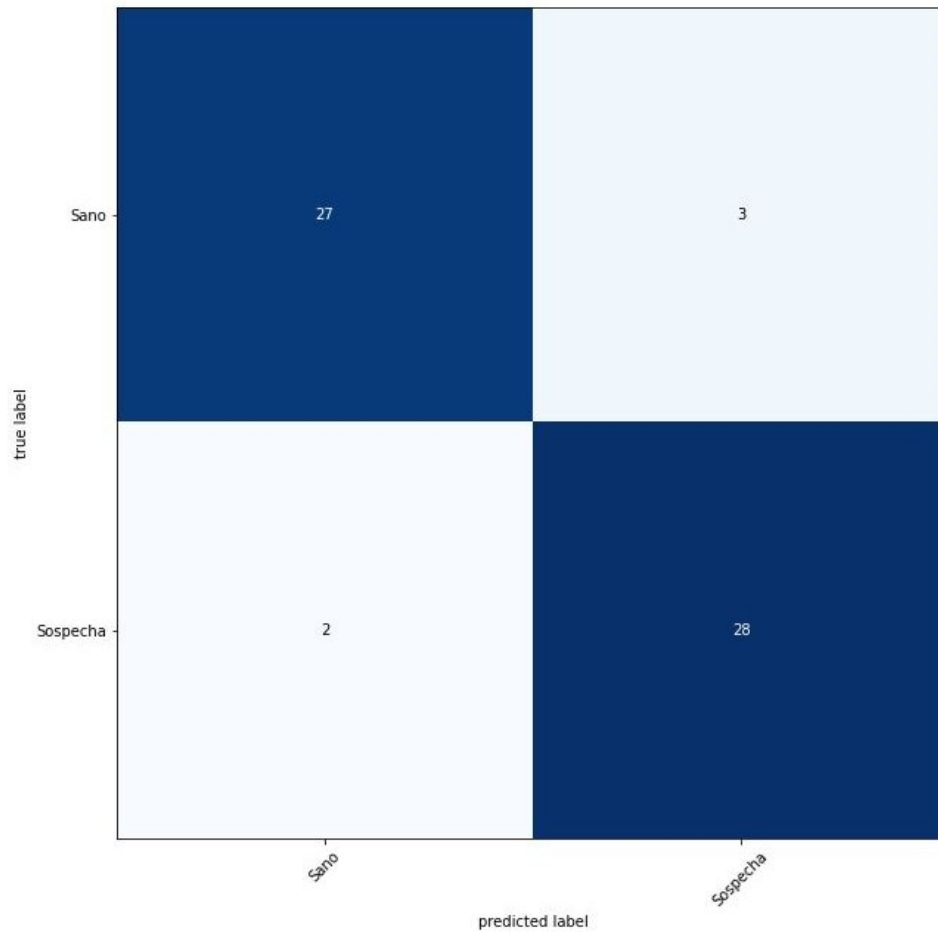
Por otro lado, la representación gráfica de la Matriz de confusión del Optimizador de Adam, la clase sospecha ha tenido una predicción correcta de 28 imágenes de manera similar en sano 27 imágenes han sido correctas (ver Figura 40.b).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



b)

Figura 40. Matriz de Confusión a) Optimizador Adagrad, b) Optimizador Adam.

3.2. CNN con Dos Capas Convolucionales

Esta red convolucional (CNN) está conformada de dos capas convolucionales, a continuación, se mencionan los diferentes parámetros utilizados (Ver Tabla 4).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Tabla 4. Parámetros para una sola capa Convolutacional.

Parámetro	Valor
Datos Entrenamiento	1214
Datos Prueba	304
Épocas	100
Tamaño Imagen	224x224
Batch	16
Conv1	(5,5)
Conv2	(3,3)
Tamaño Conv1	32
Tamaño Conv2	64
Tamaño Pool	(2,2)
Optimizador1	Adagrad
Optimizador2	Adam
Dropout	0.25
Conexiones Capa Densa	512

La cantidad de Imágenes del entrenamiento son 1518 (Ver Figura 30), compuesta de dos clases: Sano (782) y Sospecha (736), luego se separan las imágenes para entrenamiento (1214) y para validación (304), las primeras imágenes del set entrenamiento y validación varían cada vez que se entrena, luego de asignar las respectivas etiquetas a cada imagen se procede a la arquitectura de la red (Anexo D), inicialmente las imágenes son de: 224x224, que luego de pasar por el max pooling y el dropout, se reducen a la mitad (112x112), al pasar el max pooling de la primera capa, luego debido al max pooling de la segunda capa se reduce nuevamente a la mitad (56x56), seguido se aplanan en la salida y se asigna 512 conexiones, como resultado se tiene 102.782.914 de parámetros (ver Figura 41).

Para el optimizador de Adagrad el entrenamiento finaliza luego de 8 minutos 20 segundos (ver Figura 42.a), entonces se guarda con el nombre: “biomedical_Adagrad_Acrima2.hdf5”, además al evaluar la red se tiene una precisión de 0.82 y una pérdida de 0.99 (ver Figura 43.a), es posible observar el comportamiento de la precisión (ver Figura 44.a) y pérdida de la red (ver Figura

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

45.a), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 252 correctas (ver Figura 46.a), y 52 incorrectas (ver Figura 47.a), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 48.a), y se obtiene sospecha para una imagen de paciente con sospecha.

En cambio en el optimizador de Adam el entrenamiento finaliza luego de 8 minutos 20 segundos (ver Figura 42.b), entonces se guarda con el nombre: “biomedical_Adam_Acrima1.hdf5”, además al evaluar la red se tiene una precisión de 0.81 y una pérdida de 1.48 (ver Figura 43.b), es posible observar el comportamiento de la precisión (ver Figura 44.b) y pérdida de la red (ver Figura 45.b), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 249 correctas (ver Figura 46.b), y 55 incorrectas (ver Figura 47.b), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 48.b), y se obtiene sano para una imagen de paciente sano.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	2432
leaky_re_lu (LeakyReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18496
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_1 (Dropout)	(None, 56, 56, 64)	0
flatten (Flatten)	(None, 200704)	0
dense (Dense)	(None, 512)	102760960
leaky_re_lu_2 (LeakyReLU)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026
Total params: 102,782,914		
Trainable params: 102,782,914		
Non-trainable params: 0		

Figura 41. Arquitectura de la Red de dos capas convolucionales

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```

8560
Epoch 95/100
61/61 [=====] - 5s 83ms/step - loss: 0.2404 - accuracy: 0.8982 - val_loss: 0.2960 - val_accuracy: 0.
8724
Epoch 96/100
61/61 [=====] - 5s 83ms/step - loss: 0.2380 - accuracy: 0.9019 - val_loss: 0.2967 - val_accuracy: 0.
8765
Epoch 97/100
61/61 [=====] - 5s 83ms/step - loss: 0.2303 - accuracy: 0.9068 - val_loss: 0.3055 - val_accuracy: 0.
8642
Epoch 98/100
61/61 [=====] - 5s 83ms/step - loss: 0.2195 - accuracy: 0.9131 - val_loss: 0.3295 - val_accuracy: 0.
8477
Epoch 99/100
61/61 [=====] - 5s 84ms/step - loss: 0.2450 - accuracy: 0.9002 - val_loss: 0.3113 - val_accuracy: 0.
8642
Epoch 100/100
61/61 [=====] - 5s 83ms/step - loss: 0.2273 - accuracy: 0.9211 - val_loss: 0.3046 - val_accuracy: 0.
8683

```

a)

```

8395
Epoch 95/100
61/61 [=====] - 5s 87ms/step - loss: 0.0110 - accuracy: 0.9956 - val_loss: 1.1620 - val_accuracy: 0.
8683
Epoch 96/100
61/61 [=====] - 5s 87ms/step - loss: 0.0541 - accuracy: 0.9869 - val_loss: 1.4645 - val_accuracy: 0.
8313
Epoch 97/100
61/61 [=====] - 5s 87ms/step - loss: 0.0492 - accuracy: 0.9871 - val_loss: 1.0283 - val_accuracy: 0.
8272
Epoch 98/100
61/61 [=====] - 5s 87ms/step - loss: 0.4532 - accuracy: 0.8976 - val_loss: 1.2821 - val_accuracy: 0.
8354
Epoch 99/100
61/61 [=====] - 5s 87ms/step - loss: 0.2978 - accuracy: 0.9177 - val_loss: 1.8831 - val_accuracy: 0.
7325
Epoch 100/100
61/61 [=====] - 5s 86ms/step - loss: 0.4086 - accuracy: 0.9142 - val_loss: 1.1120 - val_accuracy: 0.
8395

```

b)

Figura 42. Últimas épocas de entrenamiento a) Optimizador Adagrad, b) Optimizador Adam.

```

10/10 [=====] - 1s 36ms/step - loss: 0.2968 - accuracy: 0.8717

```

a)

```

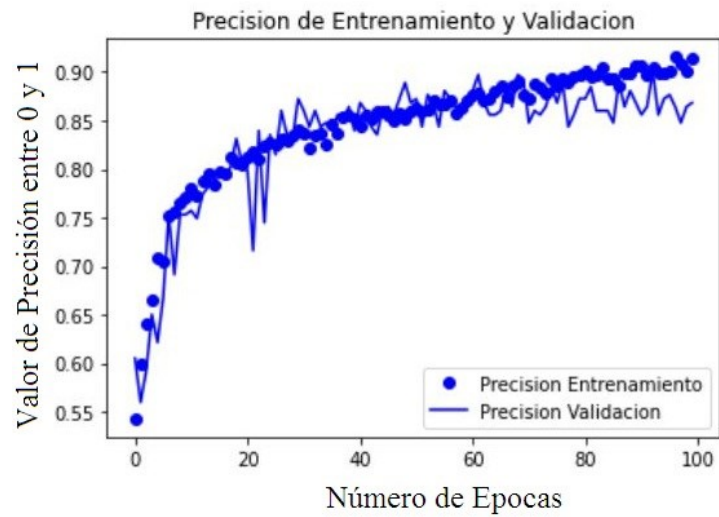
10/10 [=====] - 1s 38ms/step - loss: 0.9934 - accuracy: 0.8289

```

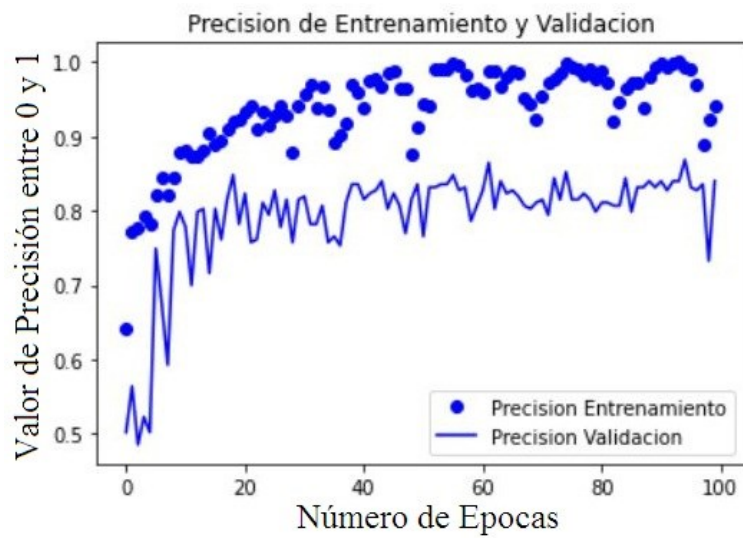
b)

Figura 43. Valores de evaluación de la CNN con a) Adagrad, b) Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



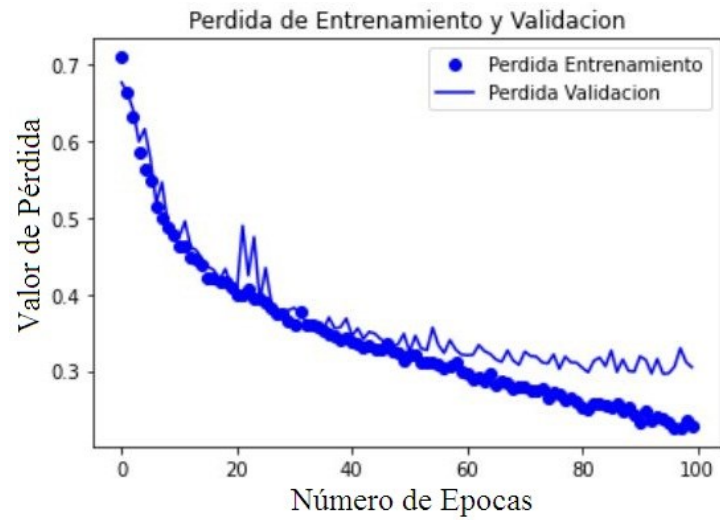
a)



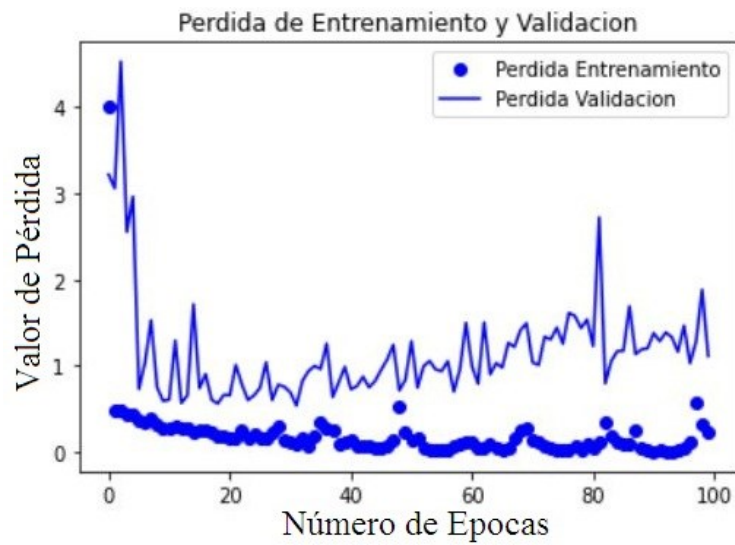
b)

Figura 44. Gráfica de precisión de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

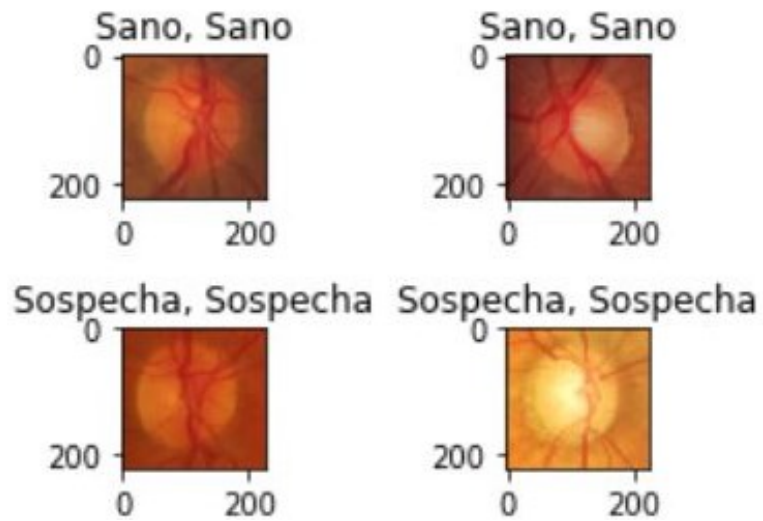


b)

Figura 45. Gráfica de Pérdida de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

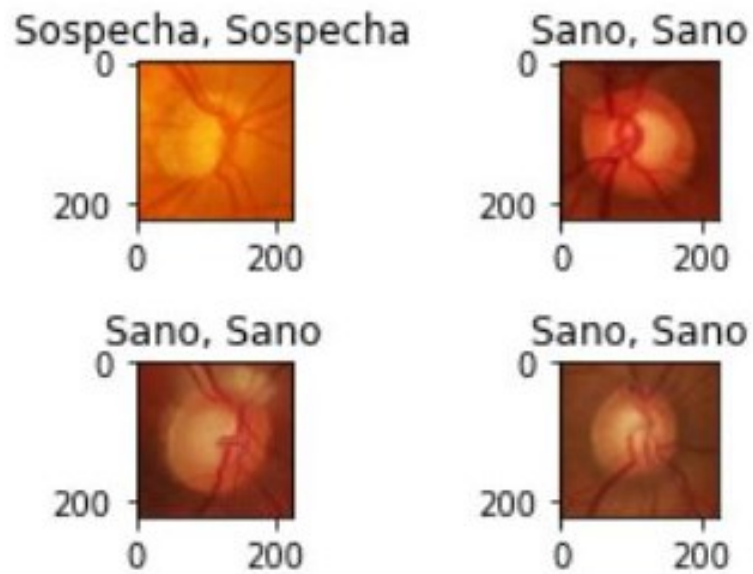
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 265 etiquetas correctas



a)

Encontradas 252 etiquetas correctas

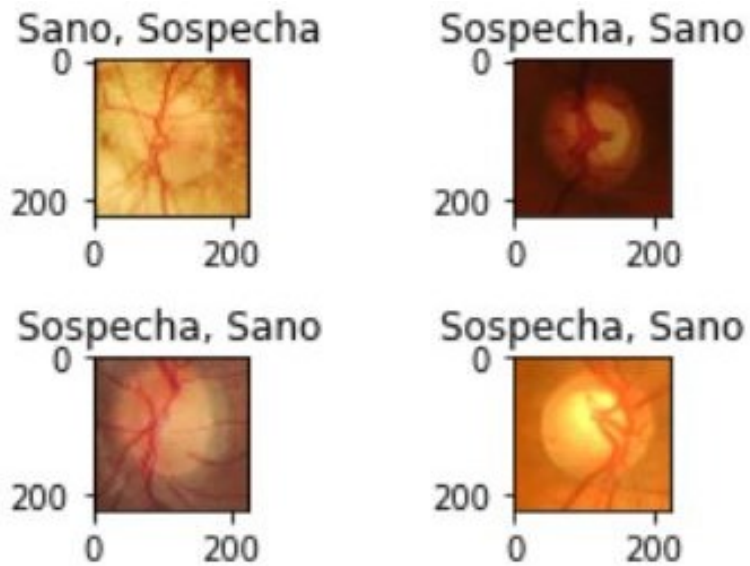


b)

Figura 46. Etiquetas Clasificadas Correctamente, a) Optimizador Adagrad, b) Optimizador Adam.

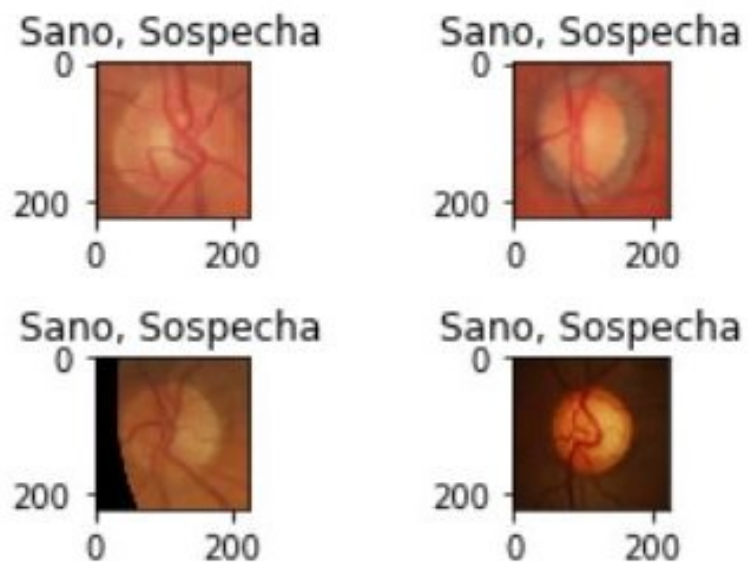
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 39 etiquetas incorrectas



a)

Encontradas 52 etiquetas incorrectas



b)

Figura 47. Etiquetas Clasificadas Incorrectamente, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

a)

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

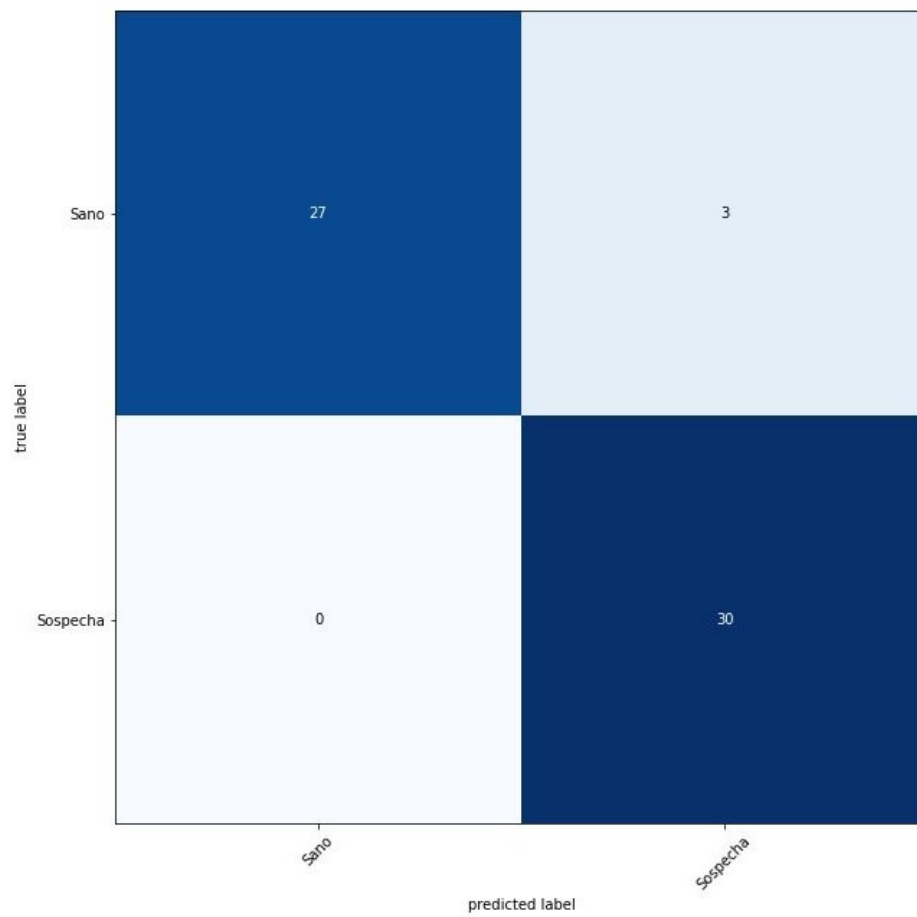
b)

Figura 48. Prueba de la red en una imagen, a) Optimizador Adagrad, b) Optimizador Adam.

En la representación gráfica de la Matriz de confusión del Optimizador de Adagrad, se puede ver las imágenes de la clase sospecha han tenido una predicción correcta de todas las imágenes, sin embargo, en sano 27 de las imágenes han sido correctas. (ver Figura 49.a)

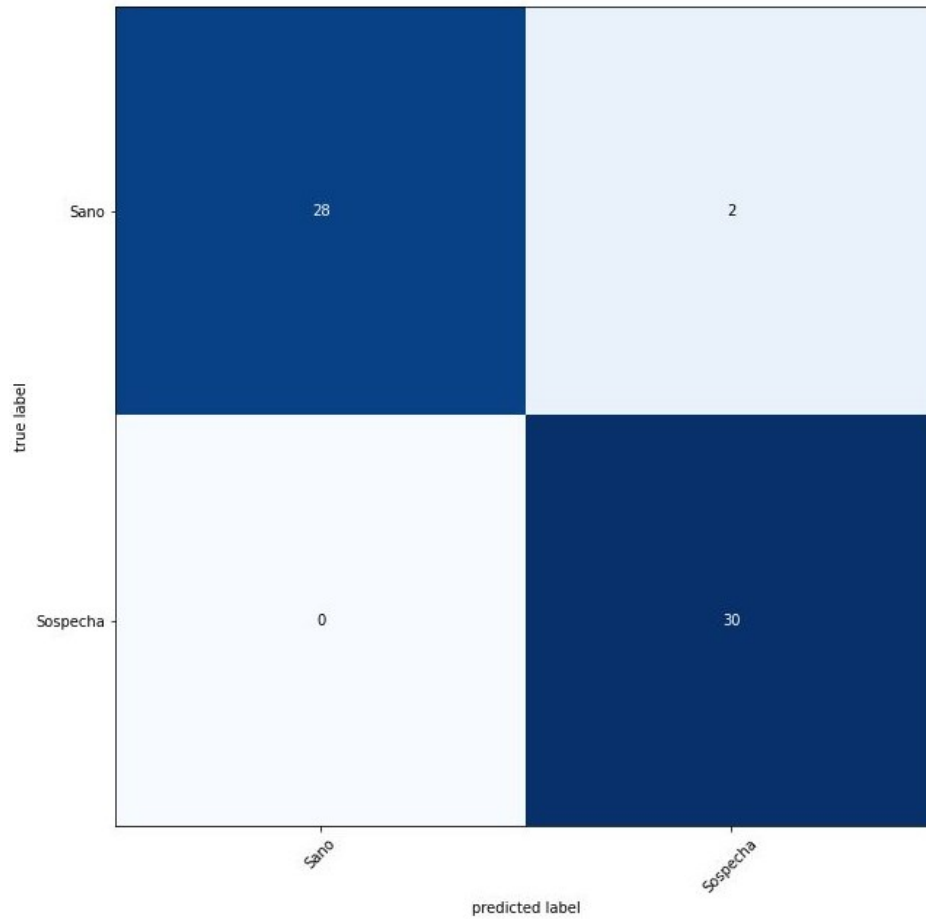
Por otro lado, la representación gráfica de la Matriz de confusión del Optimizador de Adam, la clase sospecha ha tenido una predicción correcta de todas las imágenes, sin embargo, en sano 28 imágenes han sido correctas. (ver Figura 49.b)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



b)

Figura 49. Matriz de Confusión a) Optimizador Adagrad, b) Optimizador Adam.

3.3. CNN con Tres Capas Convolucionales

Esta red convolucional (CNN) está conformada de tres capas convolucionales, a continuación, se mencionan los diferentes parámetros utilizados (Ver Tabla 5).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Tabla 5. Parámetros para una sola capa Convolucional.

Parámetro	Valor
Datos Entrenamiento	1214
Datos Prueba	304
Épocas	100
Tamaño Imagen	224x224
Batch	16
Conv1	(7,7)
Conv2	(5,5)
Conv3	(3,3)
Tamaño Conv1	32
Tamaño Conv2	64
Tamaño Conv3	128
Tamaño Pool	(2,2)
Optimizador1	Adagrad
Optimizador2	Adam
Dropout	0.25
Conexiones Capa Densa	512

La cantidad de Imágenes del entrenamiento son 1518 (Ver Figura 30), compuesta de dos clases: Sano (782) y Sospecha (736), luego se separan las imágenes para entrenamiento (1214) y para validación (304), las primeras imágenes del set entrenamiento y validación varían cada vez que se entrena, luego de asignar las respectivas etiquetas a cada imagen se procede a la arquitectura de la red (Anexo E), inicialmente las imágenes son de: 224x224, al haber 3 capas convolucionales su tamaño se divide a la mitad por cada una, siendo la siguiente a 112x112, luego 56x56 y 28x28; finalmente se aplanan los parámetros en la salida y se asignan 512 conexiones, como resultado se tiene 51.511.618 de parámetros (ver Figura 50).

Para el optimizador de Adagrad el entrenamiento finaliza luego de 10 minutos (ver Figura 51.a), entonces se guarda con el nombre: “biomedical_Adagrad_Acrima3.hdf5”, además al evaluar la red se tiene una

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

precisión de 0.86 y una pérdida de 0.31 (ver Figura 52.a), es posible observar el comportamiento de la precisión (ver Figura 53.a) y pérdida (ver Figura 54.a) de la red, luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 264 correctas (ver Figura 55.a), y 40 incorrectas (ver Figura 56.a), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 57.a), y se obtiene sospecha para una imagen de paciente con sospecha.

En cambio en el optimizador de Adam el entrenamiento finaliza luego de 10 minutos (ver Figura 51.b), entonces se guarda con el nombre: “biomedical_Adam_Acrima3.hdf5”, además al evaluar la red se tiene una precisión de 0.86 y una pérdida de 0.85 (ver Figura 52.b), es posible observar gráficamente la precisión (ver Figura 53.b) y pérdida de la red (ver Figura 54.b), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 262 correctas (ver Figura 55.b), y 42 incorrectas (ver Figura 56.b), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 57.b), y se obtiene sospecha para una imagen de paciente con sospecha de Glaucoma.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	4736
leaky_re_lu (LeakyReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	51264
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_1 (Dropout)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73856
leaky_re_lu_2 (LeakyReLU)	(None, 56, 56, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
dropout_2 (Dropout)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 512)	51380736
leaky_re_lu_3 (LeakyReLU)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026
Total params: 51,511,618		
Trainable params: 51,511,618		
Non-trainable params: 0		

Figura 50. Arquitectura de la Red de tres capas convolucionales.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```

8354
Epoch 95/100
61/61 [=====] - 6s 92ms/step - loss: 0.1957 - accuracy: 0.9216 - val_loss: 0.3848 - val_accuracy: 0.8189
Epoch 96/100
61/61 [=====] - 6s 92ms/step - loss: 0.1969 - accuracy: 0.9286 - val_loss: 0.3952 - val_accuracy: 0.8148
Epoch 97/100
61/61 [=====] - 6s 93ms/step - loss: 0.1777 - accuracy: 0.9268 - val_loss: 0.3842 - val_accuracy: 0.8436
Epoch 98/100
61/61 [=====] - 6s 93ms/step - loss: 0.2034 - accuracy: 0.9242 - val_loss: 0.3833 - val_accuracy: 0.8313
Epoch 99/100
61/61 [=====] - 6s 93ms/step - loss: 0.2004 - accuracy: 0.9201 - val_loss: 0.3814 - val_accuracy: 0.8354
Epoch 100/100
61/61 [=====] - 6s 92ms/step - loss: 0.2018 - accuracy: 0.9111 - val_loss: 0.3881 - val_accuracy: 0.8272

```

a)

```

8395
Epoch 95/100
61/61 [=====] - 7s 109ms/step - loss: 0.2327 - accuracy: 0.9417 - val_loss: 1.0279 - val_accuracy: 0.8189
Epoch 96/100
61/61 [=====] - 6s 100ms/step - loss: 0.1251 - accuracy: 0.9579 - val_loss: 1.5514 - val_accuracy: 0.7984
Epoch 97/100
61/61 [=====] - 6s 99ms/step - loss: 0.1927 - accuracy: 0.9394 - val_loss: 1.0160 - val_accuracy: 0.8107
Epoch 98/100
61/61 [=====] - 6s 102ms/step - loss: 0.0647 - accuracy: 0.9746 - val_loss: 1.0361 - val_accuracy: 0.7819
Epoch 99/100
61/61 [=====] - 6s 105ms/step - loss: 0.0581 - accuracy: 0.9823 - val_loss: 1.2112 - val_accuracy: 0.7901
Epoch 100/100
61/61 [=====] - 6s 101ms/step - loss: 0.1535 - accuracy: 0.9468 - val_loss: 0.9284 - val_accuracy: 0.8230

```

b)

Figura 51. Últimas épocas de entrenamiento a) Optimizador Adagrad, b) Optimizador Adam.

```

10/10 [=====] - 1s 49ms/step - loss: 0.3123 - accuracy: 0.8684

```

a)

```

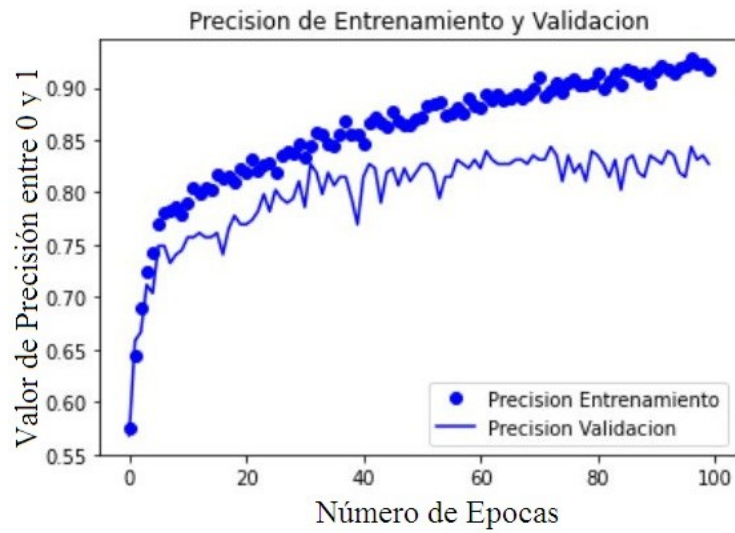
10/10 [=====] - 1s 59ms/step - loss: 0.8543 - accuracy: 0.8618

```

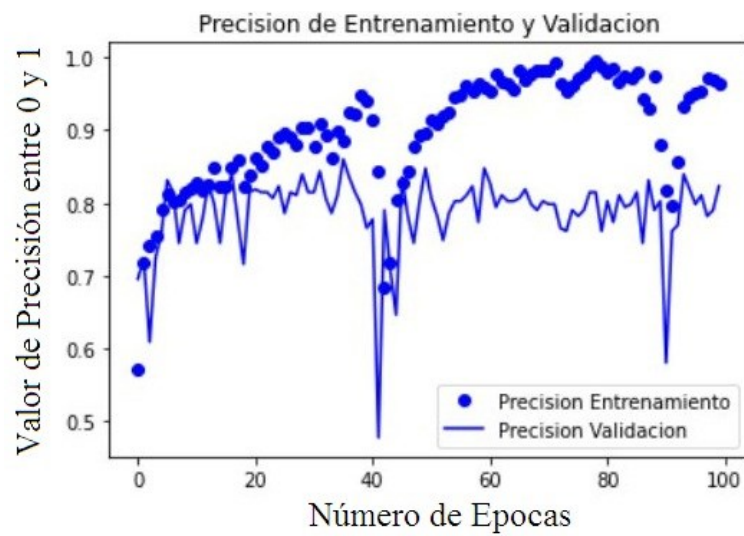
b)

Figura 52. Valores de evaluación de la CNN con a) Adagrad, b) Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



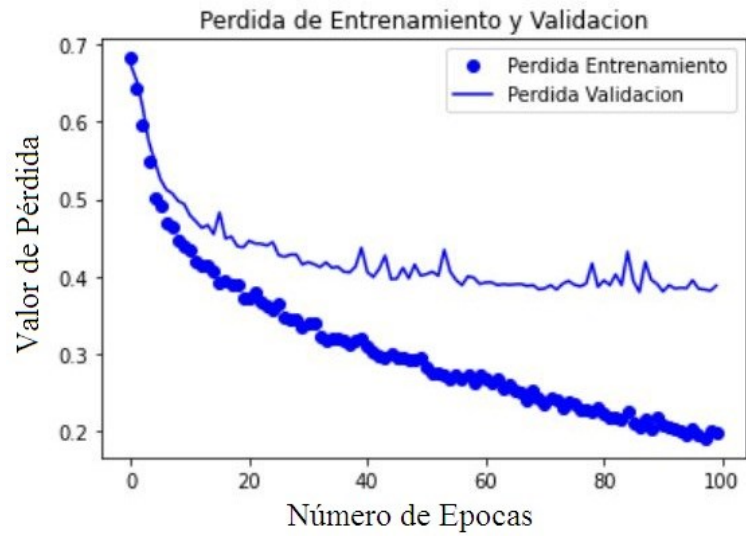
a)



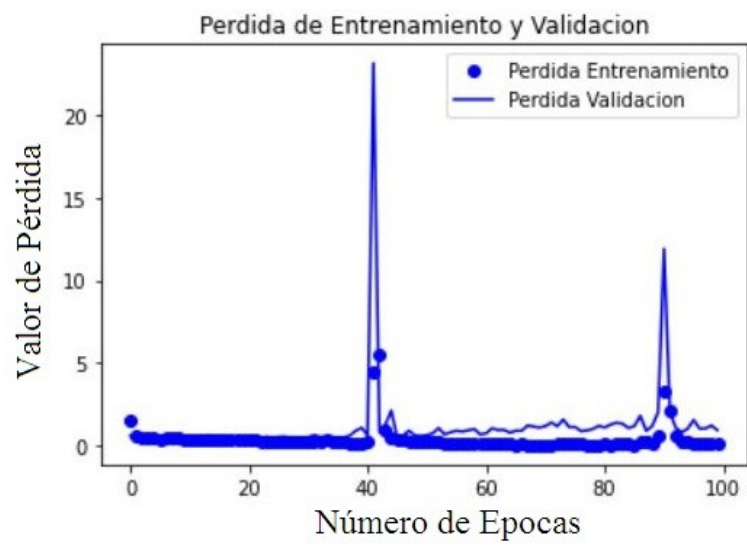
b)

Figura 53. Gráfica de precisión de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

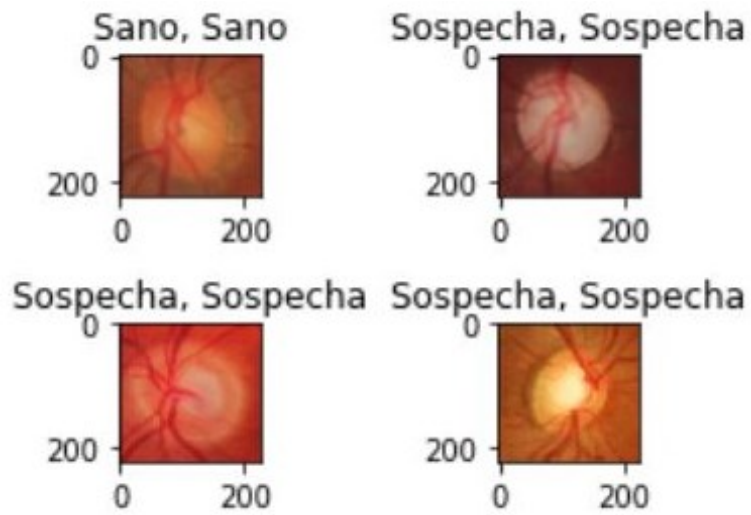


b)

Figura 54. Gráfica de Pérdida de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

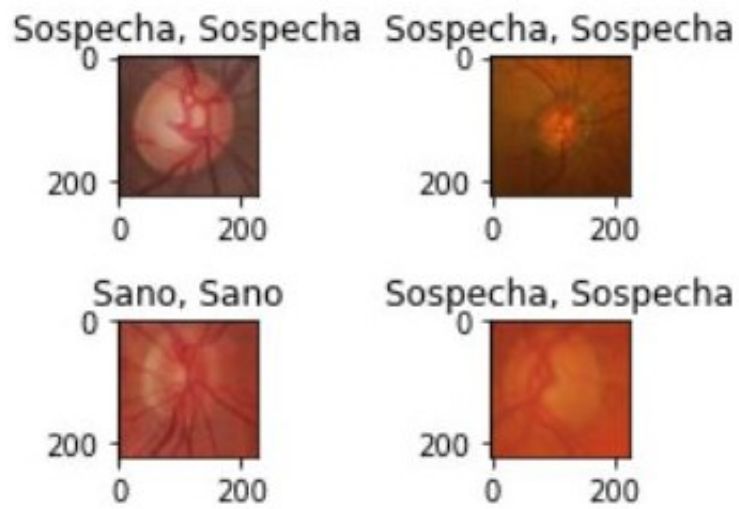
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 264 etiquetas correctas



a)

Encontradas 262 etiquetas correctas

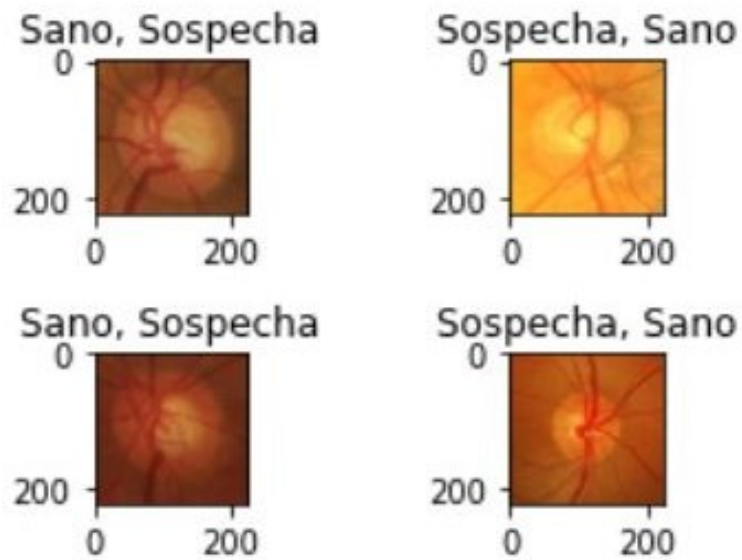


b)

Figura 55. Etiquetas Clasificadas Correctamente, a) Optimizador Adagrad, b) Optimizador Adam.

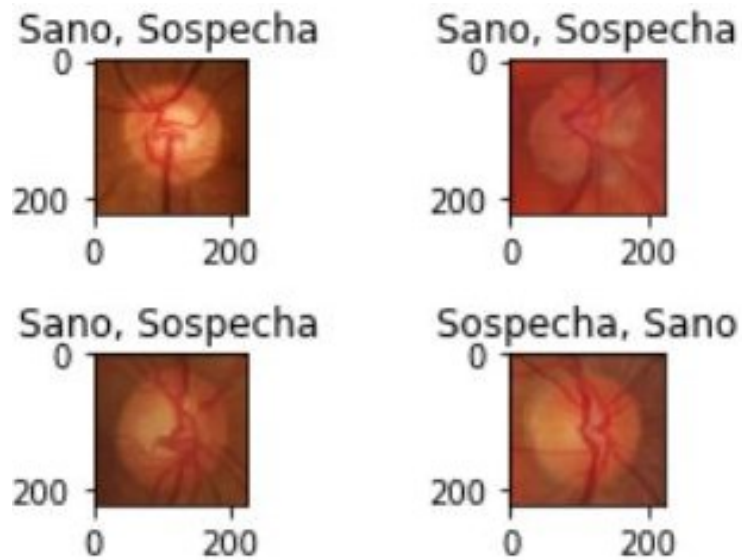
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 40 etiquetas incorrectas



a)

Encontradas 42 etiquetas incorrectas



b)

Figura 56. Etiquetas Clasificadas Incorrectamente, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

a)

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

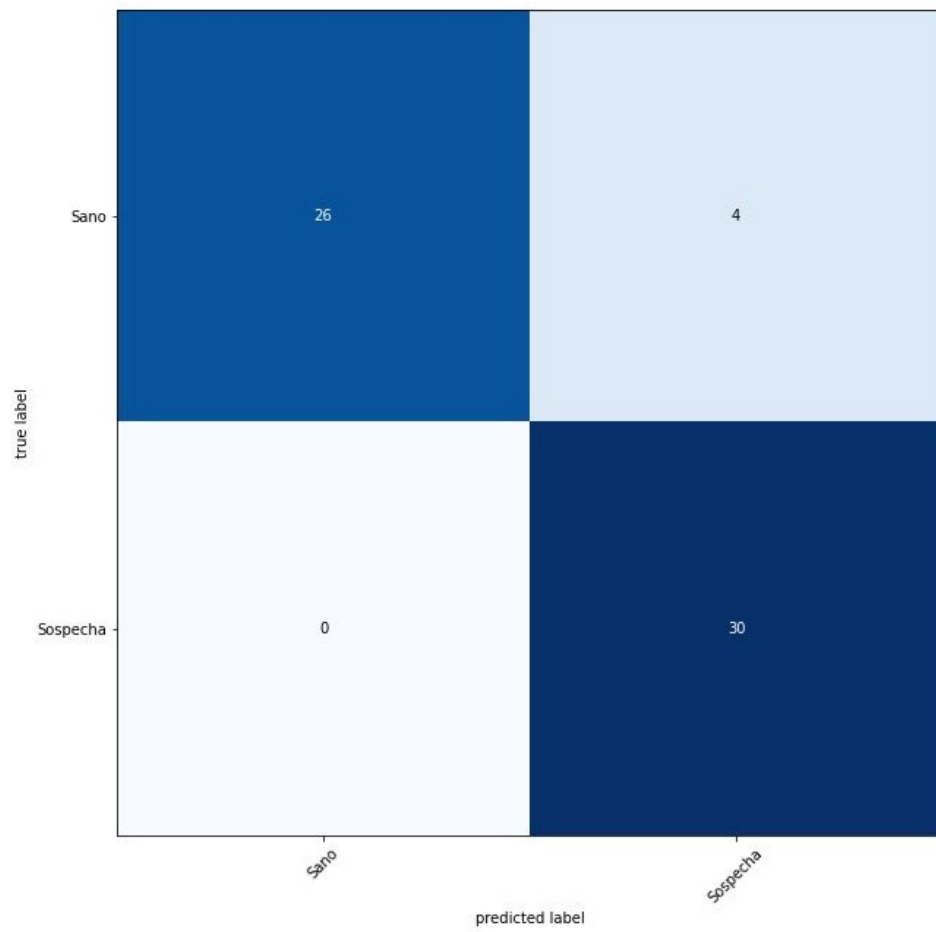
b)

Figura 57. Prueba de la red en una imagen, a) Optimizador Adagrad, b) Optimizador Adam.

En la representación gráfica de la Matriz de confusión del Optimizador de Adagrad, se puede ver que todas las imágenes de la clase sospecha han tenido una predicción correcta, sin embargo, en sano 26 de las imágenes han sido correctas (ver Figura 58.a).

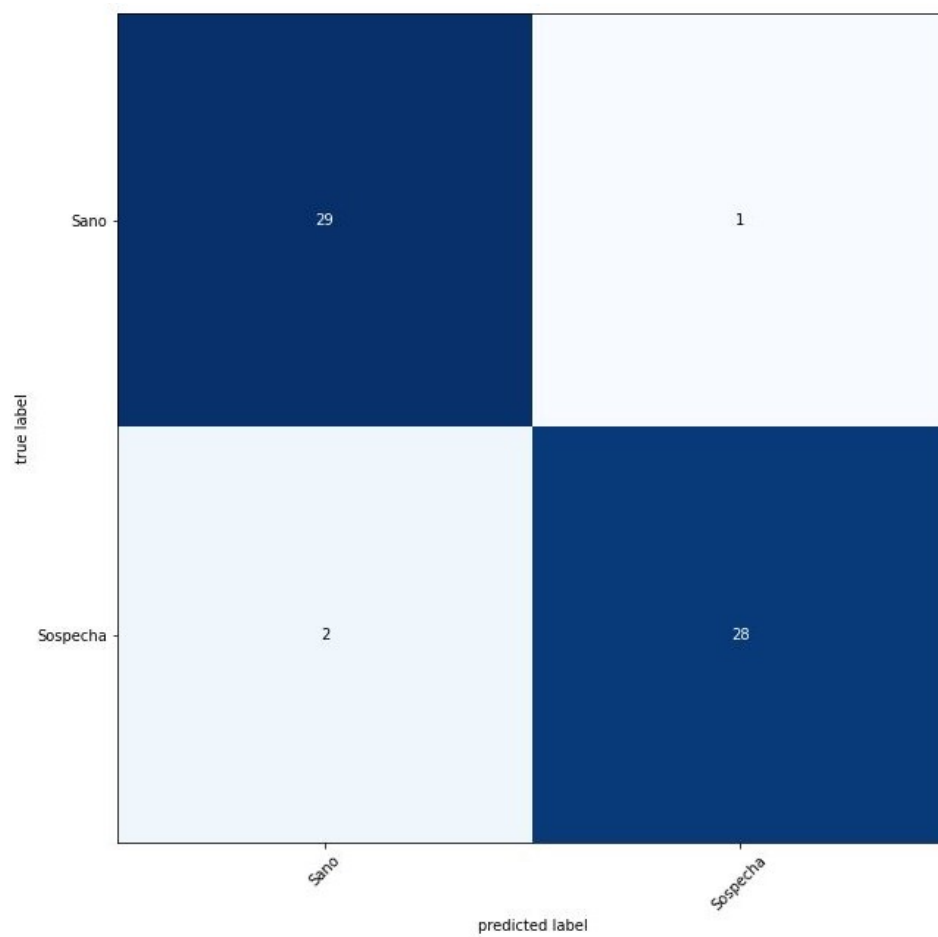
Por otro lado, la representación gráfica de la Matriz de confusión del Optimizador de Adam, la clase sospecha ha tenido una predicción de 28 imágenes correctas, sin embargo, en sano 29 imágenes han sido correctas (ver Figura 58.b).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



b)

Figura 58. Matriz de Confusión a) Optimizador Adagrad, b) Optimizador Adam.

3.4. CNN con Cuatro Capas Convolucionales

Esta red convolucional (CNN) está conformada de cuatro capas convolucionales, a continuación, se mencionan los diferentes parámetros utilizados (Ver Tabla 6).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Tabla 6. Parámetros para una sola capa Convolucional.

Parámetro	Valor
Datos Entrenamiento	1214
Datos Prueba	304
Épocas	100
Tamaño Imagen	224x224
Batch	16
Conv1	(9,9)
Conv2	(7,7)
Conv3	(5,5)
Conv4	(3,3)
Tamaño Conv1	32
Tamaño Conv2	64
Tamaño Conv3	128
Tamaño Conv4	256
Tamaño Pool	(2,2)
Optimizador1	Adagrad
Optimizador2	Adam
Dropout	0.25
Conexiones Capa Densa	512

La cantidad de Imágenes del entrenamiento son 1518 (Ver Figura 30), compuesta de dos clases: Sano (782) y Sospecha (736), luego se separan las imágenes para entrenamiento (1214) y para validación (304), las primeras imágenes del set entrenamiento y validación varían cada vez que se entrena, luego de asignar las respectivas etiquetas a cada imagen se procede a la arquitectura de la red (Anexo F), inicialmente las imágenes son de: 224x224, al haber 4 capas convolucionales su tamaño se divide a la mitad por cada una, siendo la siguiente a 112x112, luego 56x56 seguido 28x28 y 14x14; finalmente se aplanan los parámetros en la salida y se asignan 512 conexiones, como resultado se tiene 26.299.970 de parámetros (ver Figura 59).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Para el optimizador de Adagrad el entrenamiento finaliza luego de 11 minutos 40 segundos (ver Figura 60.a), entonces se guarda con el nombre: “biomedical_Adagrad_Acrima4.hdf5”, además al evaluar la red se tiene una precisión de 0.84 y una pérdida de 0.34 (ver Figura 61.a), es posible observar gráficamente la precisión (ver Figura 62.a) y pérdida de la red (ver Figura 63.a), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 257 correctas (ver Figura 64.a), y 47 incorrectas (ver Figura 65.a), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 66.a), y se obtiene sospecha para una imagen de paciente con sospecha.

En cambio en el optimizador de Adam el entrenamiento finaliza luego de 10 minutos (ver Figura 60.b), entonces se guarda con el nombre: “biomedical_Adam_Acrima4.hdf5”, además al evaluar la red se tiene una precisión de 0.79 y una pérdida de 0.46 (ver Figura 61.b), es posible observar gráficamente la precisión (ver Figura 62.b) y pérdida de la red (ver Figura 63.b), luego se compara las imágenes para determinar cuales se han etiquetado correctamente dando como resultado 243 correctas (ver Figura 64.b), y 61 incorrectas (ver Figura 65.b), además, se realiza la prueba con una imagen de la carpeta test (ver Figura 66.b), y se obtiene sospecha para una imagen de paciente con sospecha.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	7808
leaky_re_lu (LeakyReLU)	(None, 224, 224, 32)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	100416
leaky_re_lu_1 (LeakyReLU)	(None, 112, 112, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_1 (Dropout)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	204928
leaky_re_lu_2 (LeakyReLU)	(None, 56, 56, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
dropout_2 (Dropout)	(None, 28, 28, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 256)	295168
leaky_re_lu_3 (LeakyReLU)	(None, 28, 28, 256)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 256)	0
dropout_3 (Dropout)	(None, 14, 14, 256)	0
flatten (Flatten)	(None, 50176)	0
dense (Dense)	(None, 512)	25690624
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dropout_4 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 2)	1026
Total params: 26,299,970		
Trainable params: 26,299,970		
Non-trainable params: 0		

Figura 59. Arquitectura de la Red de cuatro capas convolucionales.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```

0.8519
Epoch 95/100
61/61 [=====] - 7s 114ms/step - loss: 0.2202 - accuracy: 0.9049 - val_loss: 0.3321 - val_accuracy:
0.8683
Epoch 96/100
61/61 [=====] - 7s 113ms/step - loss: 0.2304 - accuracy: 0.9108 - val_loss: 0.3392 - val_accuracy:
0.8560
Epoch 97/100
61/61 [=====] - 7s 111ms/step - loss: 0.2410 - accuracy: 0.9104 - val_loss: 0.3263 - val_accuracy:
0.8724
Epoch 98/100
61/61 [=====] - 7s 112ms/step - loss: 0.2368 - accuracy: 0.8955 - val_loss: 0.3674 - val_accuracy:
0.8395
Epoch 99/100
61/61 [=====] - 7s 110ms/step - loss: 0.2412 - accuracy: 0.9017 - val_loss: 0.3258 - val_accuracy:
0.8683
Epoch 100/100
61/61 [=====] - 7s 111ms/step - loss: 0.2223 - accuracy: 0.9053 - val_loss: 0.3217 - val_accuracy:
0.8683

```

a)

```

0.7325
Epoch 95/100
61/61 [=====] - 8s 127ms/step - loss: 0.4920 - accuracy: 0.7727 - val_loss: 0.4848 - val_accuracy:
0.7449
Epoch 96/100
61/61 [=====] - 7s 122ms/step - loss: 0.5239 - accuracy: 0.7610 - val_loss: 0.6598 - val_accuracy:
0.7984
Epoch 97/100
61/61 [=====] - 7s 122ms/step - loss: 0.4765 - accuracy: 0.7868 - val_loss: 0.4377 - val_accuracy:
0.7984
Epoch 98/100
61/61 [=====] - 8s 133ms/step - loss: 0.5364 - accuracy: 0.7924 - val_loss: 0.7518 - val_accuracy:
0.6996
Epoch 99/100
61/61 [=====] - 8s 131ms/step - loss: 0.7360 - accuracy: 0.7013 - val_loss: 0.9524 - val_accuracy:
0.6337
Epoch 100/100
61/61 [=====] - 7s 121ms/step - loss: 0.5839 - accuracy: 0.7167 - val_loss: 0.4413 - val_accuracy:
0.7737

```

b)

Figura 60. Últimas épocas de entrenamiento a) Optimizador Adagrad, b) Optimizador Adam.

```

10/10 [=====] - 1s 61ms/step - loss: 0.3423 - accuracy: 0.8454

```

a)

```

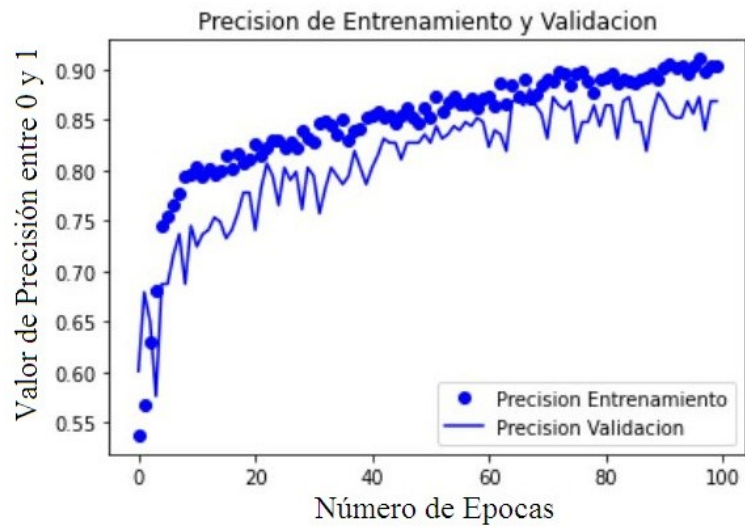
10/10 [=====] - 1s 69ms/step - loss: 0.4647 - accuracy: 0.7993

```

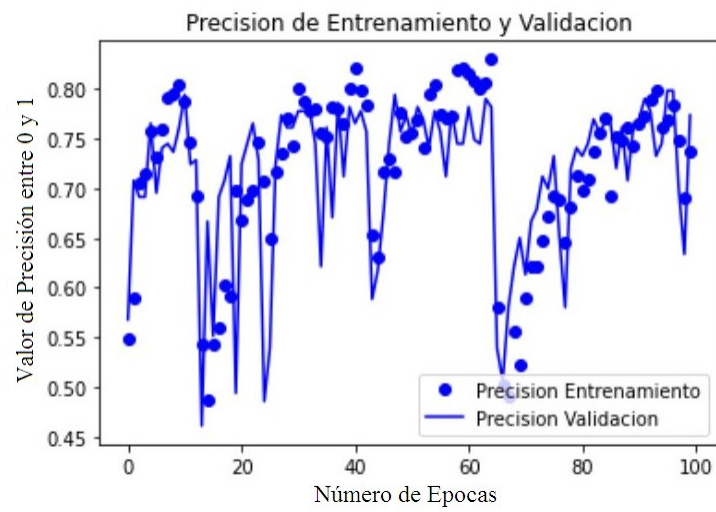
b)

Figura 61. Valores de evaluación de la CNN con a) Adagrad, b) Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



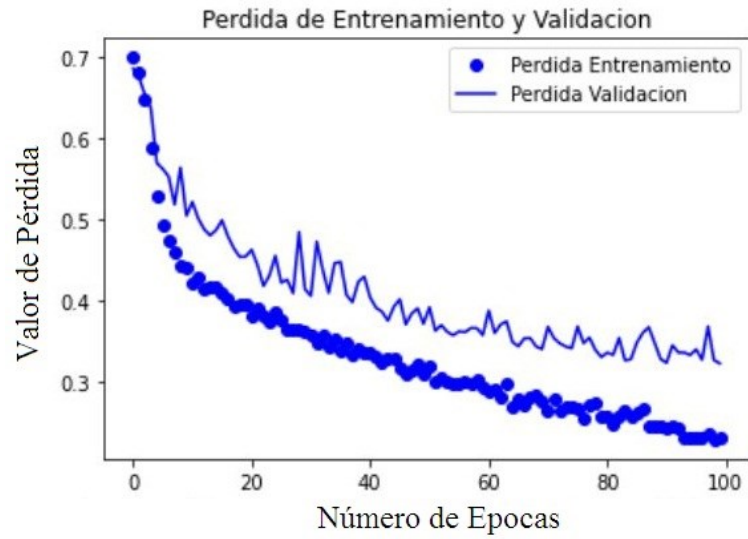
a)



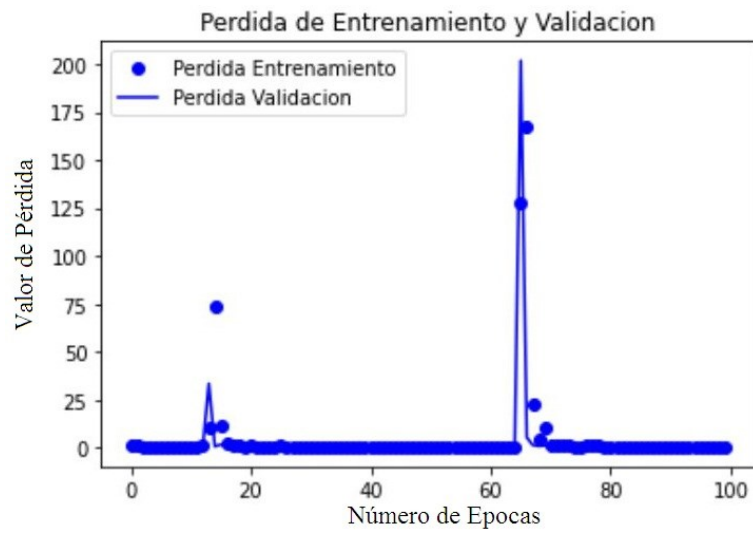
b)

Figura 62. Gráfica de precisión de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

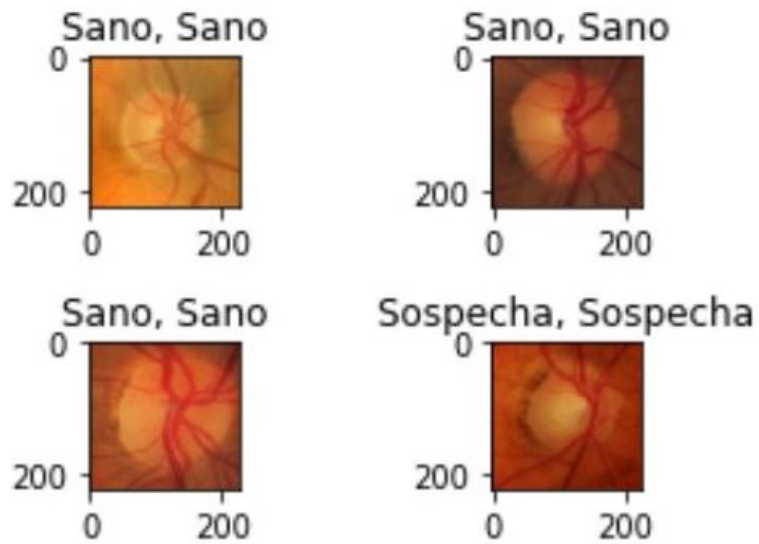


b)

Figura 63. Gráfica de Pérdida de Entrenamiento y Validación, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 257 etiquetas correctas



a)

Encontradas 243 etiquetas correctas

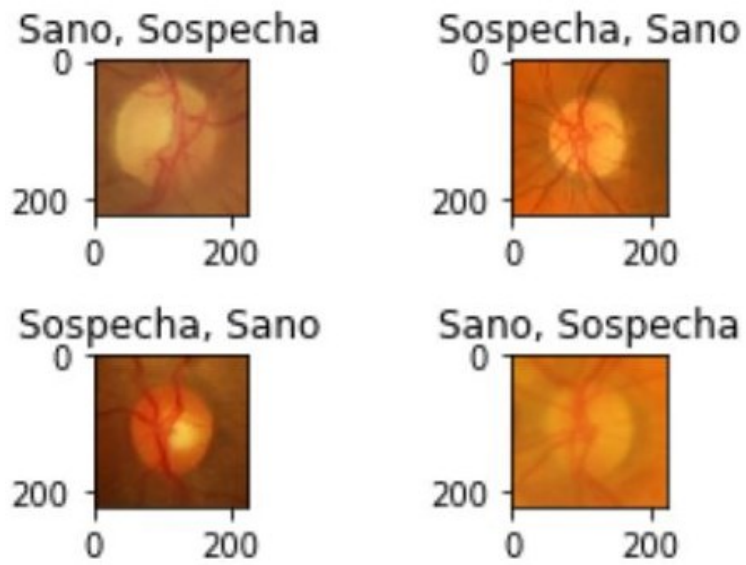


b)

Figura 64. Etiquetas Clasificadas Correctamente, a) Optimizador Adagrad, b) Optimizador Adam.

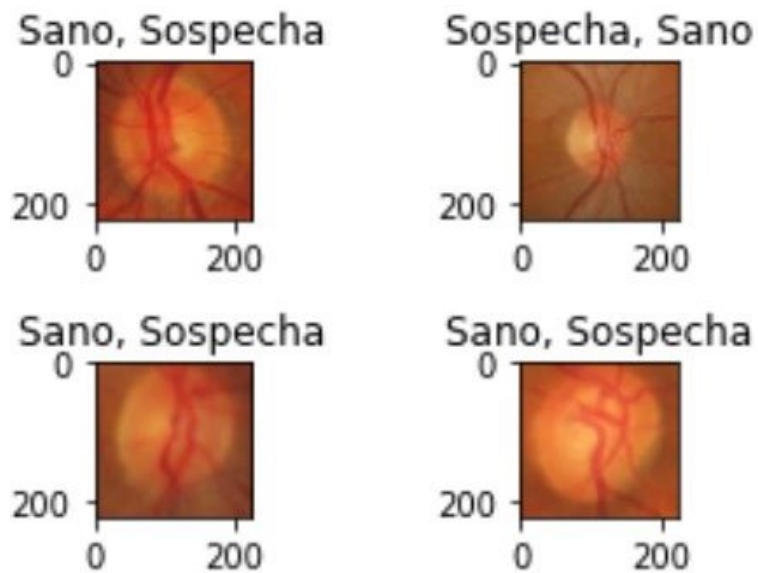
CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Encontradas 47 etiquetas incorrectas



a)

Encontradas 61 etiquetas incorrectas



b)

Figura 65. Etiquetas Clasificadas Incorrectamente, a) Optimizador Adagrad, b) Optimizador Adam.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

a)

Nueva Base Centrada/Base 224 Acrima/Test/Sospecha/2.jpg Sospecha

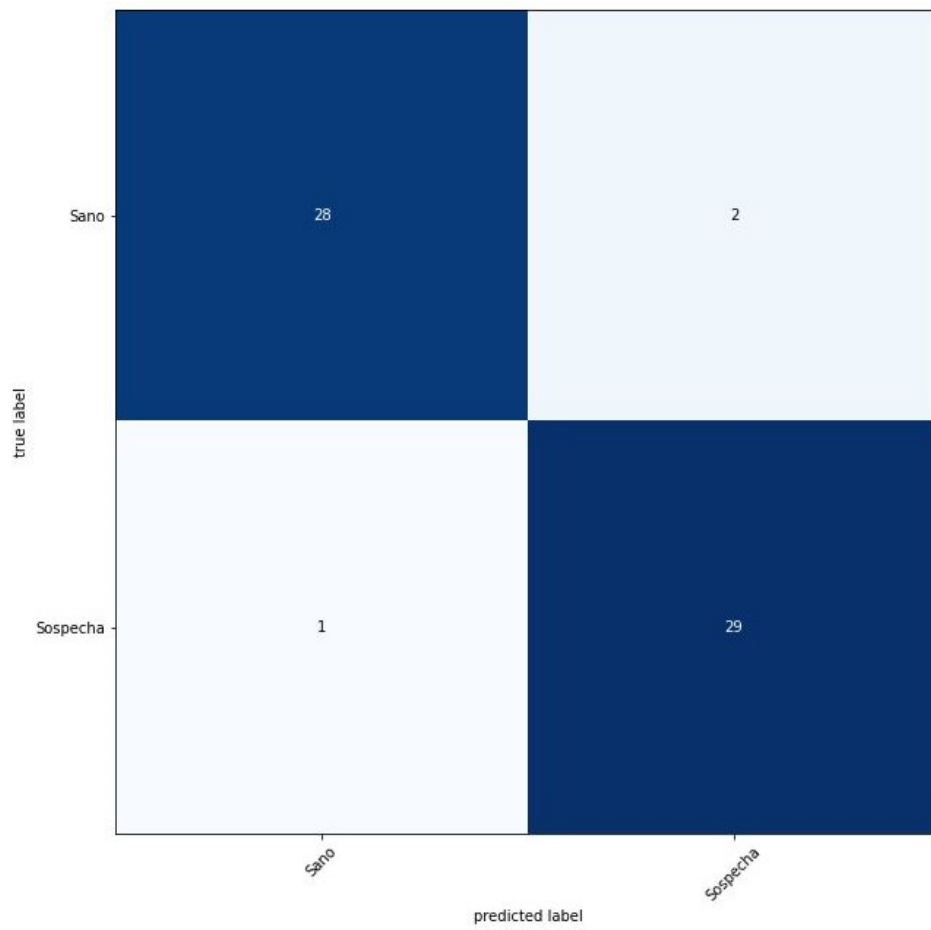
b)

Figura 66. Prueba de la red en una imagen, a) Optimizador Adagrad, b) Optimizador Adam.

En la representación gráfica de la Matriz de confusión del Optimizador de Adagrad, se puede ver 29 imágenes de la clase sospecha han tenido una predicción correcta, sin embargo, en sano 28 de las imágenes han sido correctas (ver Figura 67.a).

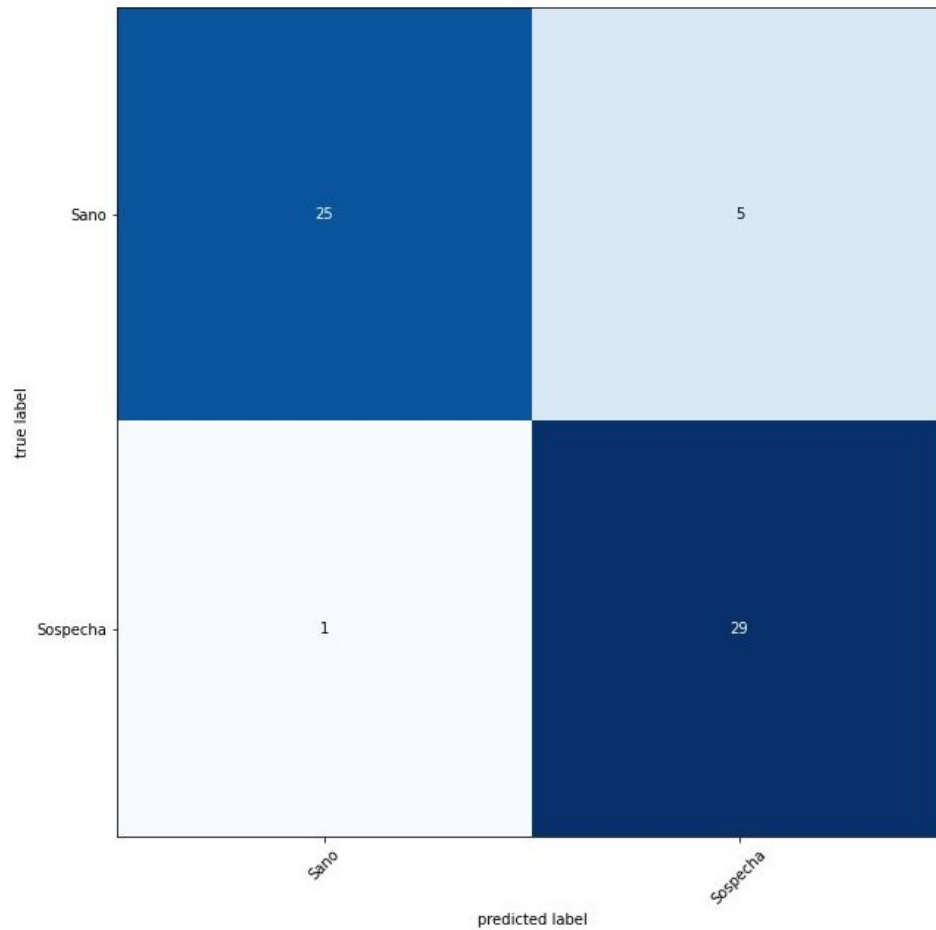
Por otro lado, la representación gráfica de la Matriz de confusión del Optimizador de Adam, la clase sospecha ha tenido una predicción de 29 imágenes correctas, sin embargo, en sano 25 imágenes han sido correctas (ver Figura 67.b).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



b)

Figura 67. Matriz de Confusión a) Optimizador Adagrad, b) Optimizador Adam.

3.5. Transferencia de Conocimiento.

Esta red convolucional (CNN) está conformada de tres capas convolucionales, a continuación, se mencionan los diferentes parámetros utilizados (Ver Tabla 7).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Tabla 7. Parámetros para una sola capa Convolutacional.

Parámetro	Valor
Datos Entrenamiento	1214
Datos Prueba	304
Épocas	100
Tamaño Imagen	224x224
Batch	16
Conv1	(7,7)
Conv2	(5,5)
Conv3	(3,3)
Tamaño Conv1	256
Tamaño Conv2	512
Tamaño Conv3	1024
Tamaño Pool	(2,2)
Optimizador	RMSprop
Dropout Capas	0.25
Conexiones Capa Densa1	4096
Conexiones Capa Densa2	4096
Conexiones Capa Densa3	4096
Dropout Capa Densa	0.1

La cantidad de Imágenes del entrenamiento son: 1518 (Ver Figura 30), compuesta de dos clases: Sano (782) y Sospecha (736), luego se separan las imágenes para entrenamiento (1214) y para validación (304). La estructura para cada uno de los cuatro modelos que se presenta es la misma que la usada en 3.3, la mayor diferencia es el optimizador y las tres capas en la salida de 4.096; con ayuda de la plataforma Kaggle, es posible el uso de una mayor cantidad de memoria RAM, dado que estos entrenamientos son demasiado para el computador utilizado, se utiliza los pesos de las redes pre-entrenadas: VGG16, Inception V3, Resnet50 y Xception; al momento de entrenar los datos con la red presentada, se actualizan los pesos de las ultimas 3 capas convolucionales.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Modelo VGG16.

El entrenamiento del modelo finaliza luego de 41 minutos 40 segundos, el modelo se guarda con el nombre: “vgg16_model.hdf5”, el mejor valor de precisión que se obtiene es 0.93403 (ver Figura 68.a), es posible observar gráficamente la precisión (ver Figura 69.a) y pérdida de entrenamiento de la red (ver Figura 70.a), por último, la representación gráfica de la Matriz de confusión indica que la clase sospecha ha tenido una predicción de 28 imágenes correctas, sin embargo, en sano 26 imágenes han sido correctas (ver Figura 71.a).

Modelo Inception V3.

El entrenamiento del modelo finaliza luego de 46 minutos 40 segundos, el modelo se guarda con el nombre: “inceptionV3_model.hdf5”, el mejor valor de precisión que se obtiene es 0.92014 (ver Figura 68.b), es posible observar gráficamente la precisión (ver Figura 69.b) y pérdida de entrenamiento de la red (ver Figura 70.b), por último, la representación gráfica de la Matriz de confusión indica que la clase sospecha ha tenido una predicción de 1 imagen correcta, sin embargo, en sano 27 imágenes han sido correctas (ver Figura 71.b).

Modelo Resnet50

El entrenamiento del modelo finaliza luego de 48 minutos 20 segundos, el modelo se guarda con el nombre: “resnet50_model.hdf5”, el mejor valor de precisión que se obtiene es 0.93403 (ver Figura 68.c), es posible observar gráficamente la precisión (ver Figura 69.c) y pérdida de entrenamiento de la red (ver Figura 70.c), por último, la representación gráfica de la Matriz de confusión indica que la clase sospecha ha tenido una predicción de 30 imágenes correctas, sin embargo, en sano 27 imágenes han sido correctas (ver Figura 71.c).

Modelo Xception.

El entrenamiento del modelo finaliza luego de 51 minutos 40 segundos, el modelo se guarda con el nombre: “xception_model.hdf5”, el mejor valor de precisión que se obtiene es 0.92014 y una pérdida de 0.0075 (ver Figura 68.c), es posible observar gráficamente la precisión (ver Figura 69.c) y pérdida de entrenamiento de la red (ver Figura 70.c), por último, la representación gráfica de la Matriz de confusión indica que la clase sospecha ha tenido una predicción de 14 imágenes correctas, sin embargo, en sano 13 imágenes han sido correctas (ver Figura 71.c).

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```
Epoch 97/100
75/75 [=====] - ETA: 0s - loss: 0.0045 - accuracy: 0.9983
Epoch 00097: val_accuracy did not improve from 0.93403
75/75 [=====] - 25s 338ms/step - loss: 0.0045 - accuracy: 0.9983 -
val_loss: 1.4726 - val_accuracy: 0.8924
Epoch 98/100
75/75 [=====] - ETA: 0s - loss: 0.0253 - accuracy: 0.9942
Epoch 00098: val_accuracy did not improve from 0.93403
75/75 [=====] - 25s 330ms/step - loss: 0.0253 - accuracy: 0.9942 -
val_loss: 0.7414 - val_accuracy: 0.9097
Epoch 99/100
75/75 [=====] - ETA: 0s - loss: 0.0043 - accuracy: 0.9992
Epoch 00099: val_accuracy did not improve from 0.93403
75/75 [=====] - 26s 343ms/step - loss: 0.0043 - accuracy: 0.9992 -
val_loss: 0.9679 - val_accuracy: 0.8993
Epoch 100/100
75/75 [=====] - ETA: 0s - loss: 0.0116 - accuracy: 0.9967
Epoch 00100: val_accuracy did not improve from 0.93403
75/75 [=====] - 24s 317ms/step - loss: 0.0116 - accuracy: 0.9967 -
val_loss: 0.8461 - val_accuracy: 0.9167
```

a)

```
Epoch 00096: val_accuracy did not improve from 0.92014
75/75 [=====] - 29s 388ms/step - loss: 0.0423 - accuracy: 0.9942 -
val_loss: 0.7939 - val_accuracy: 0.9028
Epoch 97/100
75/75 [=====] - ETA: 0s - loss: 0.0247 - accuracy: 0.9933
Epoch 00097: val_accuracy did not improve from 0.92014
75/75 [=====] - 25s 332ms/step - loss: 0.0247 - accuracy: 0.9933 -
val_loss: 0.7701 - val_accuracy: 0.8819
Epoch 98/100
75/75 [=====] - ETA: 0s - loss: 0.0334 - accuracy: 0.9933
Epoch 00098: val_accuracy did not improve from 0.92014
75/75 [=====] - 24s 326ms/step - loss: 0.0334 - accuracy: 0.9933 -
val_loss: 0.6476 - val_accuracy: 0.9167
Epoch 99/100
75/75 [=====] - ETA: 0s - loss: 0.0111 - accuracy: 0.9967
Epoch 00099: val_accuracy did not improve from 0.92014
75/75 [=====] - 28s 367ms/step - loss: 0.0111 - accuracy: 0.9967 -
val_loss: 1.0698 - val_accuracy: 0.8924
Epoch 100/100
75/75 [=====] - ETA: 0s - loss: 0.0247 - accuracy: 0.9925
Epoch 00100: val_accuracy did not improve from 0.92014
75/75 [=====] - 25s 334ms/step - loss: 0.0247 - accuracy: 0.9925 -
val_loss: 0.6736 - val_accuracy: 0.9028
```

b)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

```

75/75 [=====] - ETA: 0s - loss: 0.3361 - accuracy: 0.8549
Epoch 00096: val_accuracy did not improve from 0.93403
75/75 [=====] - 25s 339ms/step - loss: 0.3361 - accuracy: 0.8549 -
val_loss: 0.3102 - val_accuracy: 0.8854
Epoch 97/100
75/75 [=====] - ETA: 0s - loss: 0.3667 - accuracy: 0.8382
Epoch 00097: val_accuracy did not improve from 0.93403
75/75 [=====] - 30s 404ms/step - loss: 0.3667 - accuracy: 0.8382 -
val_loss: 0.3646 - val_accuracy: 0.9132
Epoch 98/100
75/75 [=====] - ETA: 0s - loss: 0.3382 - accuracy: 0.8490
Epoch 00098: val_accuracy did not improve from 0.93403
75/75 [=====] - 25s 339ms/step - loss: 0.3382 - accuracy: 0.8490 -
val_loss: 0.2795 - val_accuracy: 0.9062
Epoch 99/100
75/75 [=====] - ETA: 0s - loss: 0.3581 - accuracy: 0.8424
Epoch 00099: val_accuracy did not improve from 0.93403
75/75 [=====] - 26s 346ms/step - loss: 0.3581 - accuracy: 0.8424 -
val_loss: 0.2533 - val_accuracy: 0.9132
Epoch 100/100
75/75 [=====] - ETA: 0s - loss: 0.3776 - accuracy: 0.8274
Epoch 00100: val_accuracy did not improve from 0.93403
75/75 [=====] - 30s 395ms/step - loss: 0.3776 - accuracy: 0.8274 -
val_loss: 0.4116 - val_accuracy: 0.7326

```

c)

```

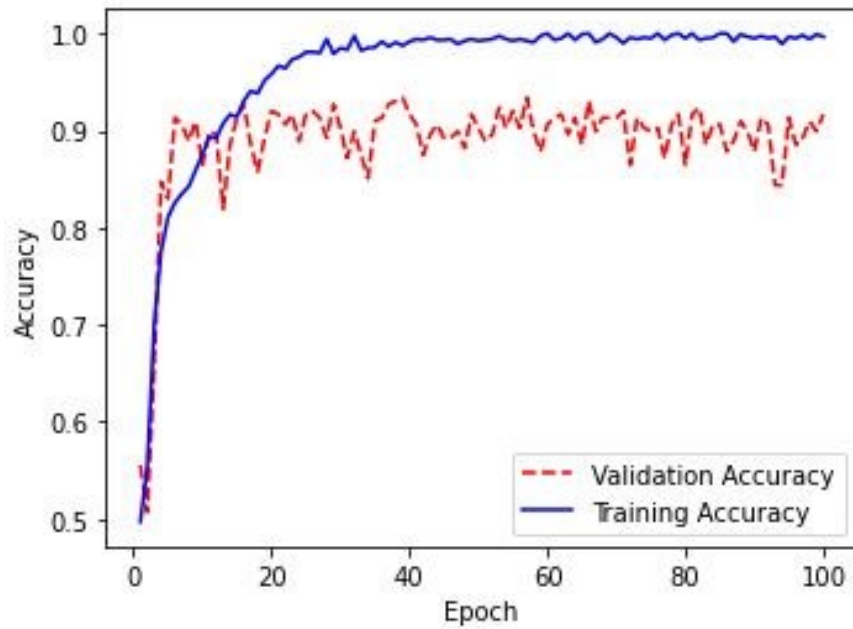
Epoch 97/100
75/75 [=====] - ETA: 0s - loss: 4.0017e-04 - accuracy: 1.0000
Epoch 00097: val_accuracy did not improve from 0.90625
75/75 [=====] - 29s 392ms/step - loss: 4.0017e-04 - accuracy: 1.000
0 - val_loss: 1.3621 - val_accuracy: 0.8889
Epoch 98/100
75/75 [=====] - ETA: 0s - loss: 0.0466 - accuracy: 0.9942
Epoch 00098: val_accuracy did not improve from 0.90625
75/75 [=====] - 34s 458ms/step - loss: 0.0466 - accuracy: 0.9942 -
val_loss: 1.2396 - val_accuracy: 0.8889
Epoch 99/100
75/75 [=====] - ETA: 0s - loss: 0.0212 - accuracy: 0.9933
Epoch 00099: val_accuracy did not improve from 0.90625
75/75 [=====] - 30s 406ms/step - loss: 0.0212 - accuracy: 0.9933 -
val_loss: 1.1151 - val_accuracy: 0.8819
Epoch 100/100
75/75 [=====] - ETA: 0s - loss: 0.0075 - accuracy: 0.9975
Epoch 00100: val_accuracy improved from 0.90625 to 0.92014, saving model to xception_model.h
df5
75/75 [=====] - 32s 424ms/step - loss: 0.0075 - accuracy: 0.9975 -
val_loss: 0.9190 - val_accuracy: 0.9201

```

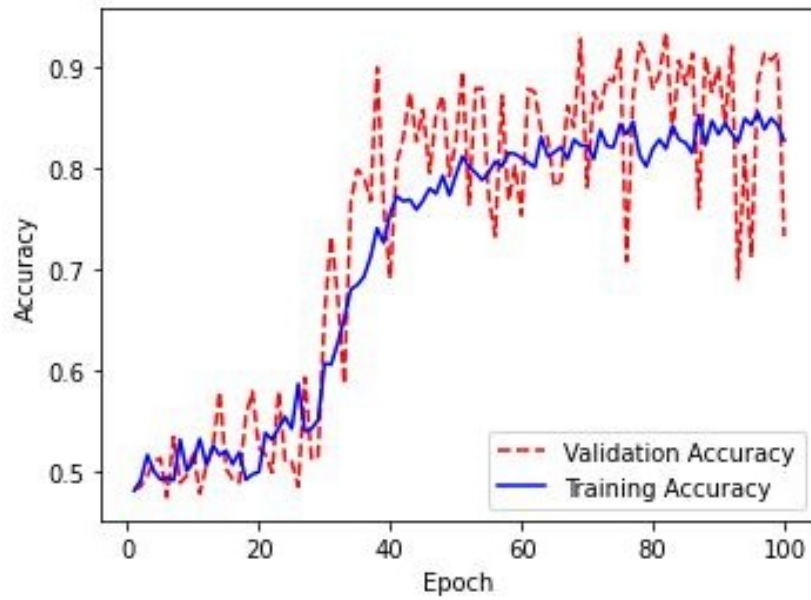
d)

Figura 68. Mejor valor de precisión de la red, a) Modelo VGG16, b) Modelo Inception V3, c) Modelo Resnet50, d) Modelo Xception.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

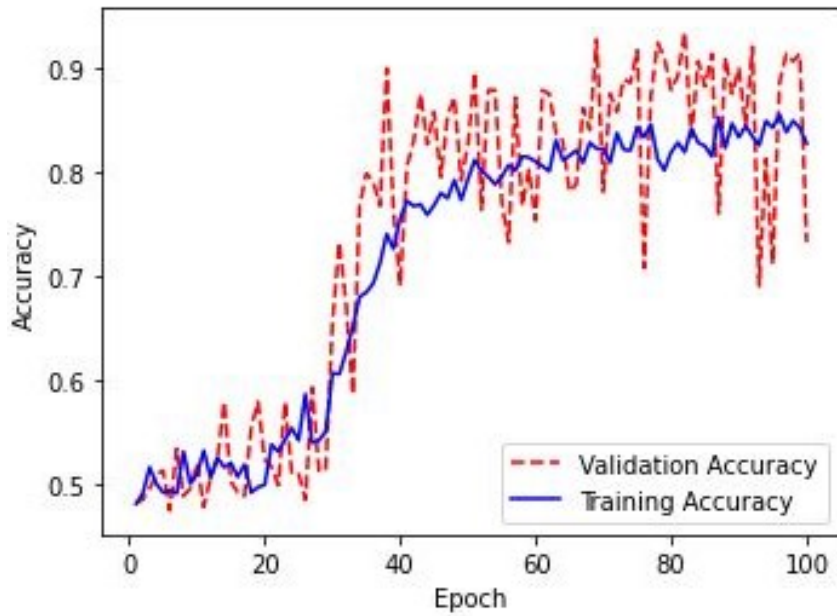


a)

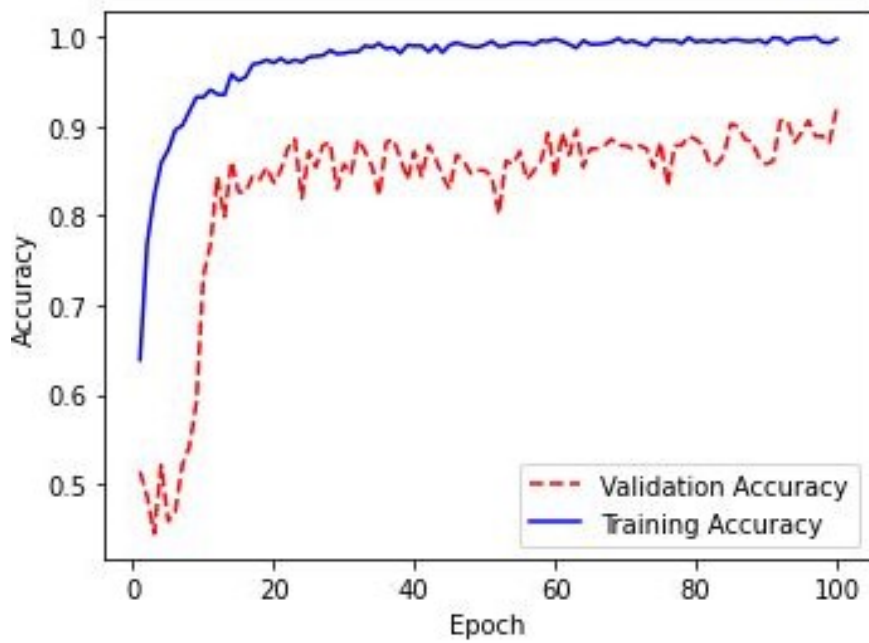


b)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



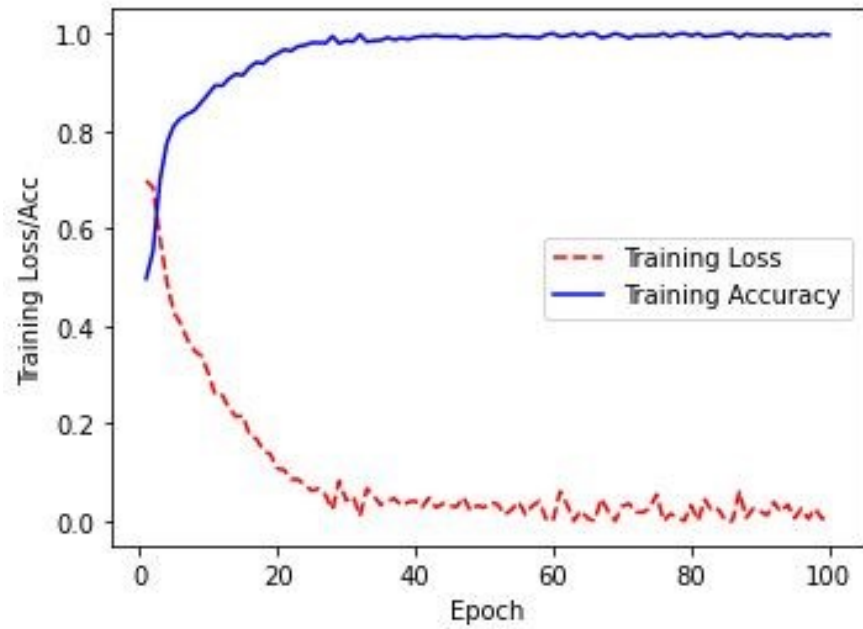
c)



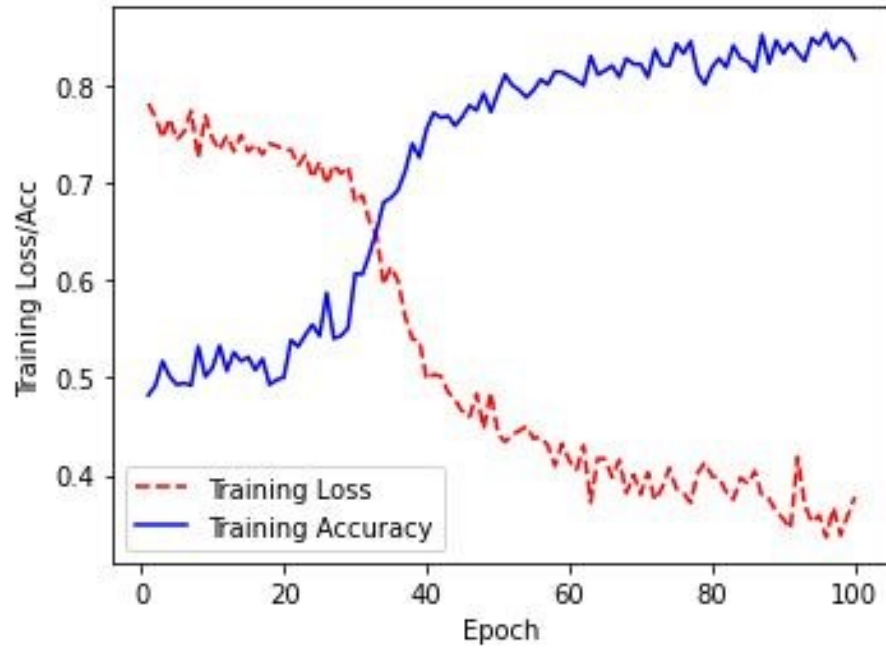
d)

Figura 69. Gráfica de precisión de Entrenamiento y Validación, a) Modelo VGG16, b) Modelo Inception V3, c) Modelo Resnet50, d) Modelo Xception.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

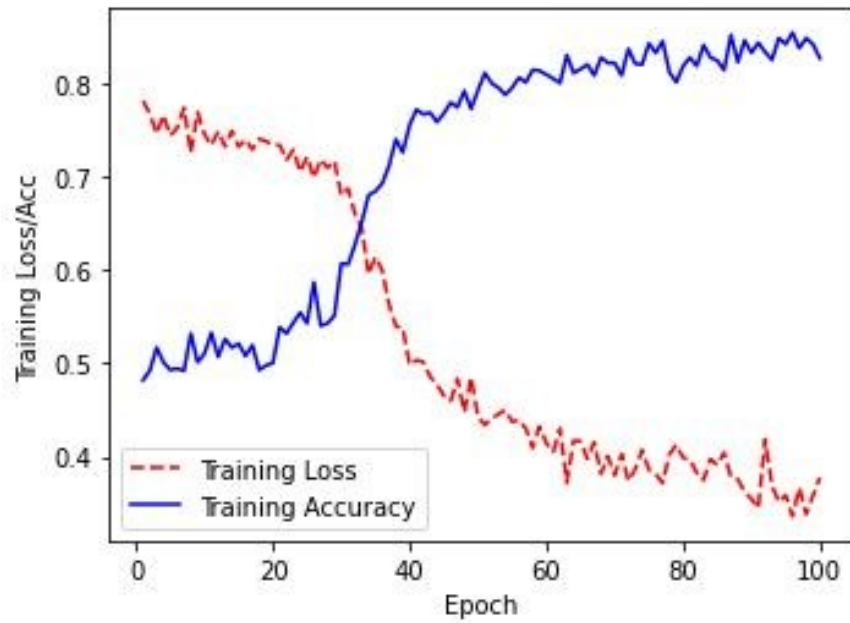


a)

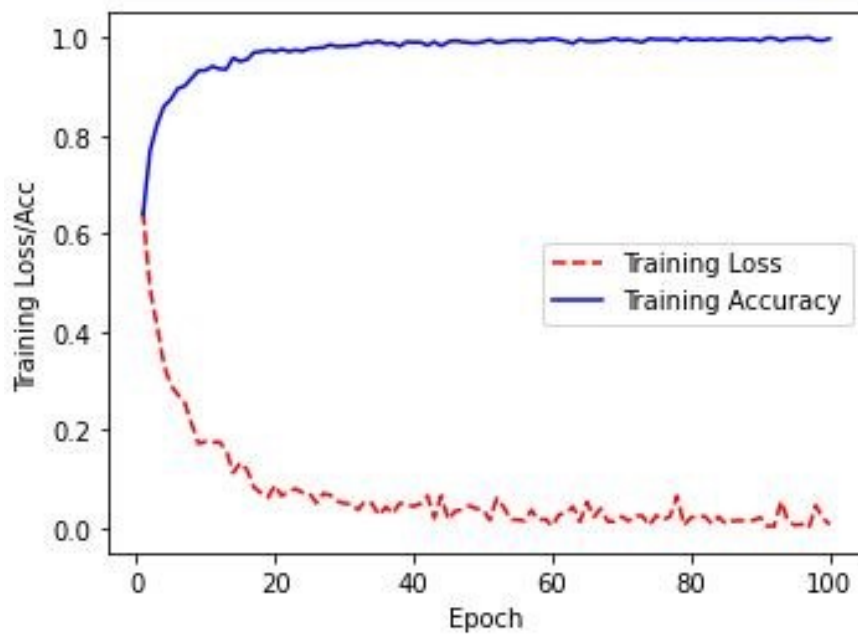


b)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



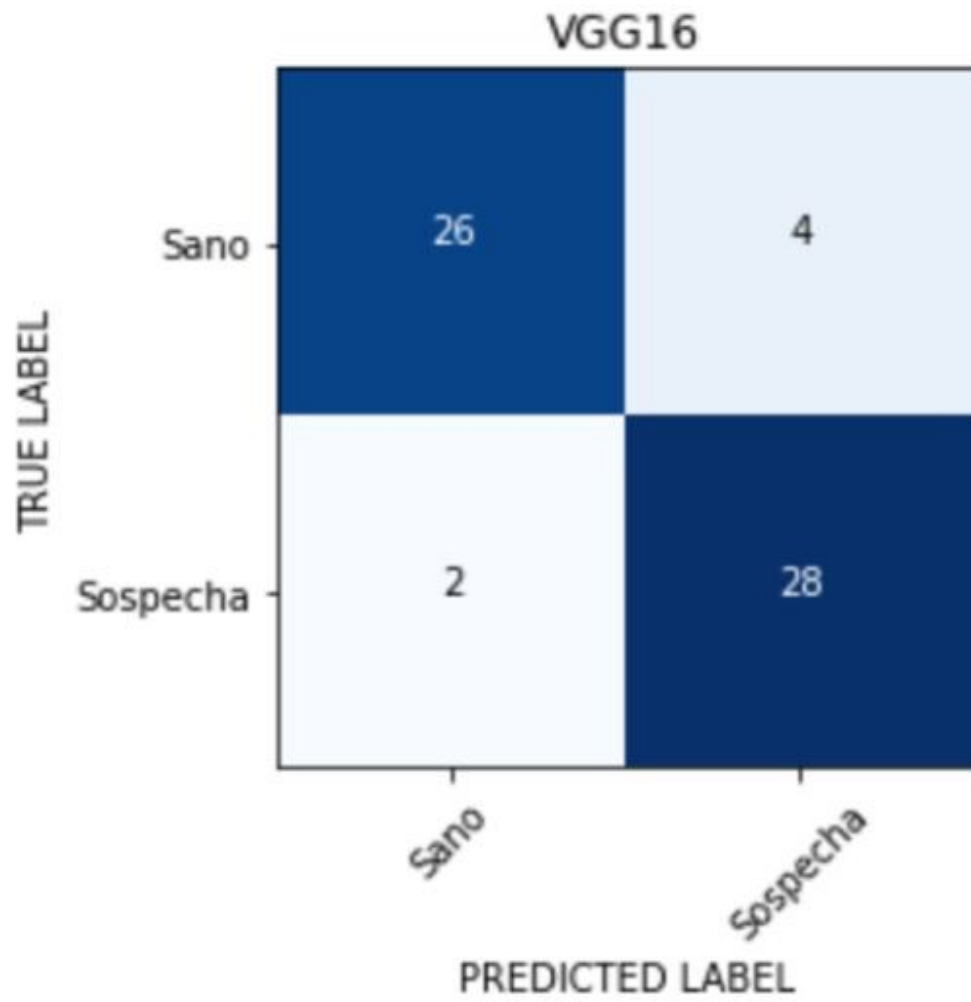
c)



d)

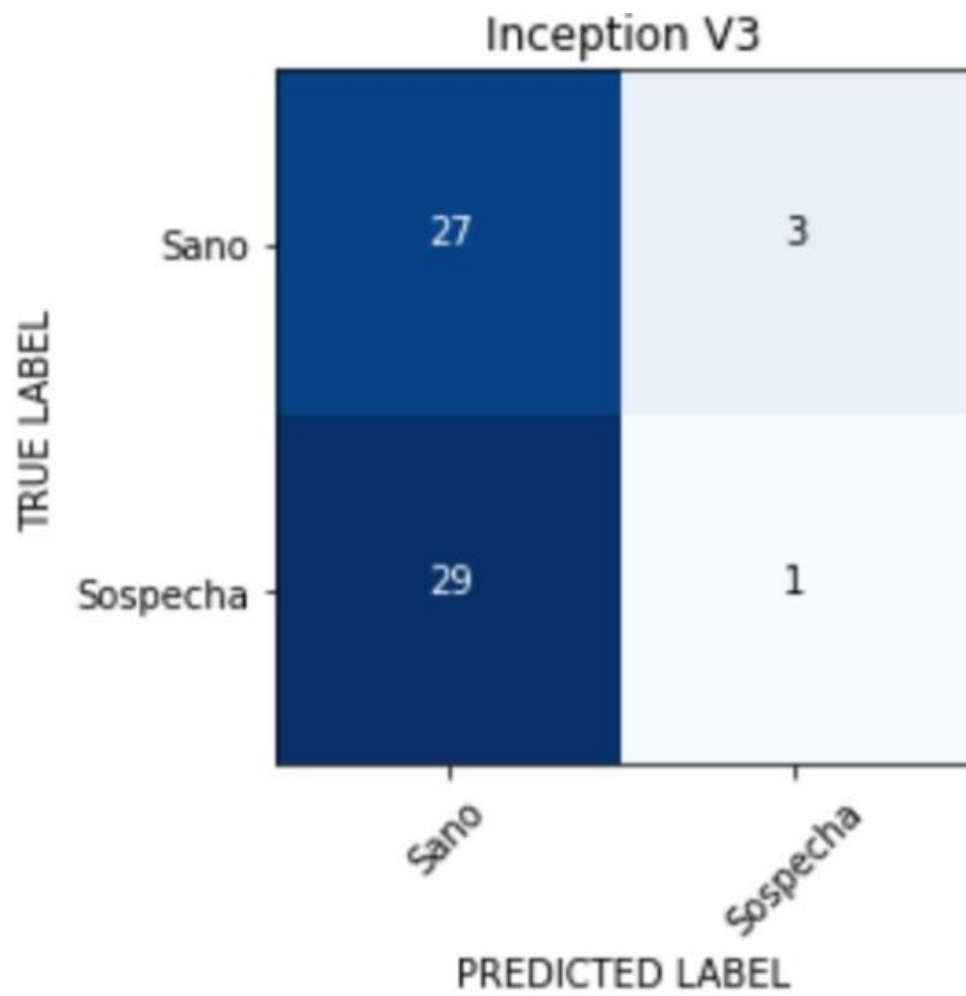
Figura 70. Gráfica de Precisión y Pérdida de Entrenamiento, a) Modelo VGG16, b) Modelo Inception V3, c) Modelo Resnet50, d) Modelo Xception.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



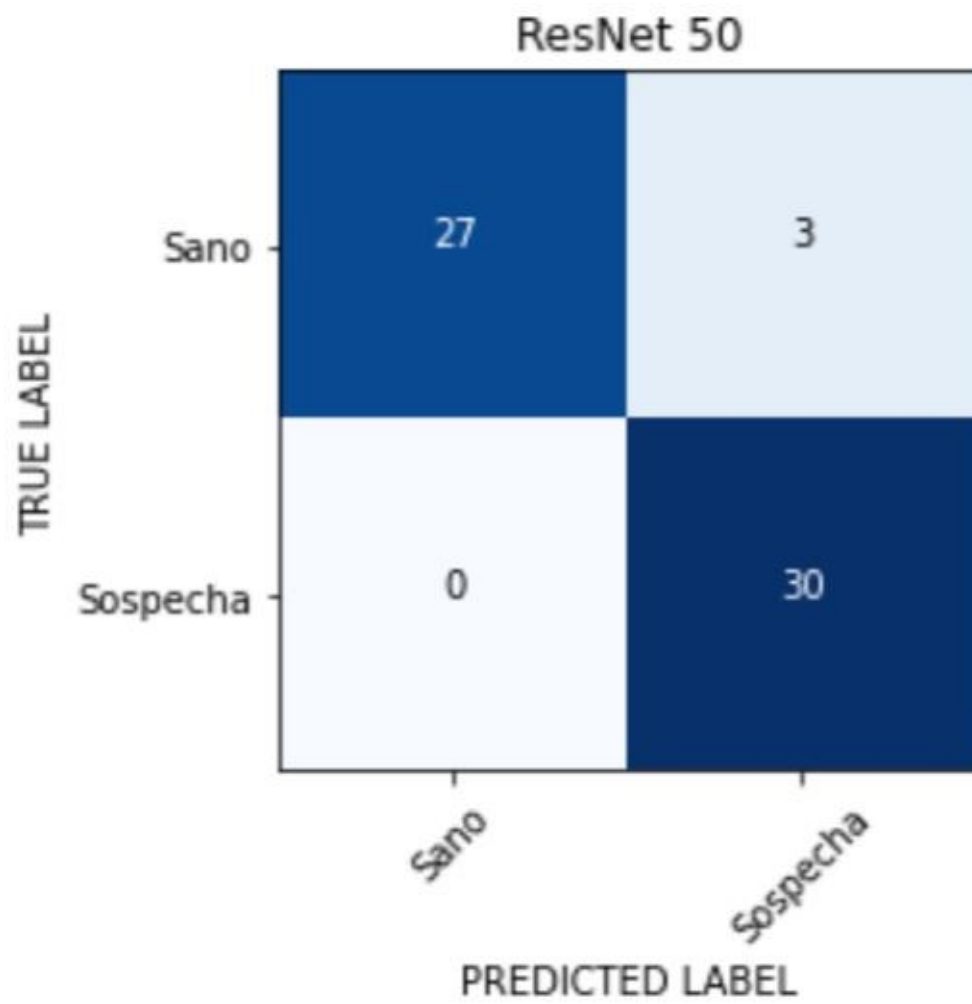
a)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



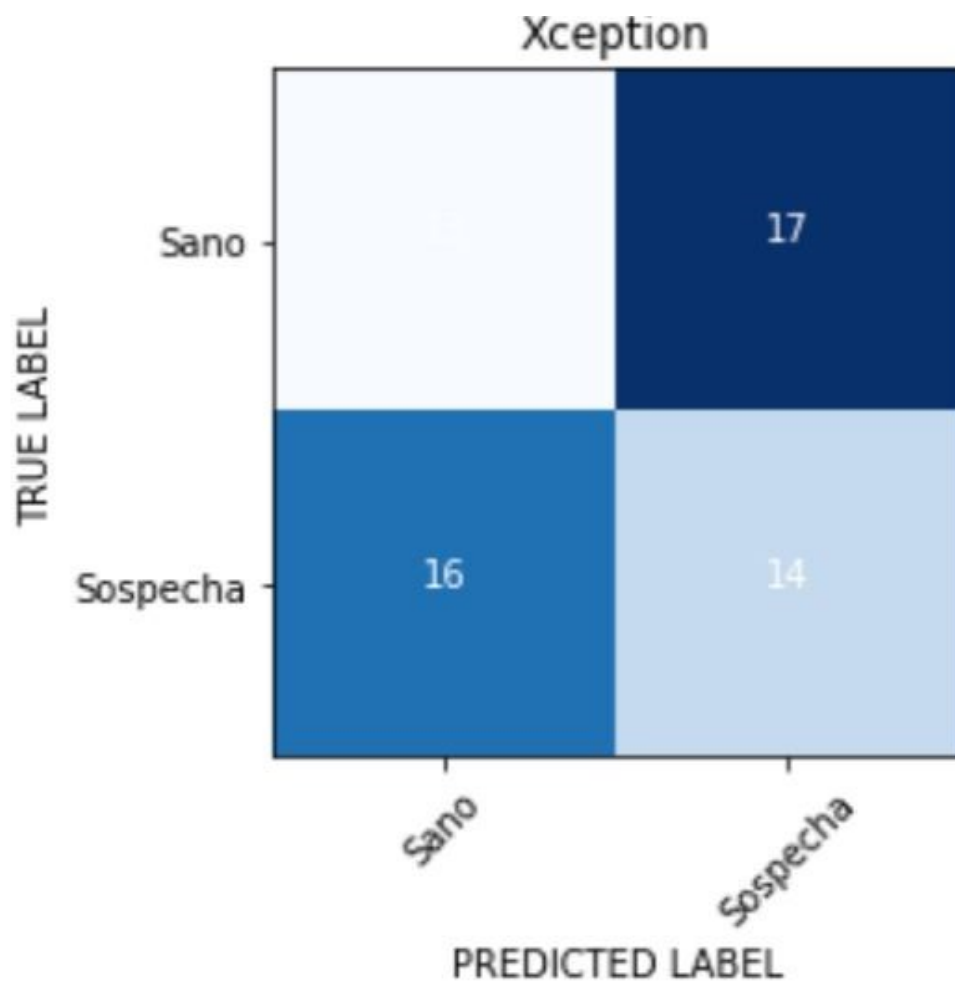
b)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



c)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS



d)

Figura 71. Matriz de Confusión: a) Modelo VGG16, b) Modelo Inception V3, c) Modelo Resnet50, d) Modelo Xception.

3.6. Desarrollo de la Interfaz de la Herramienta de Soporte.

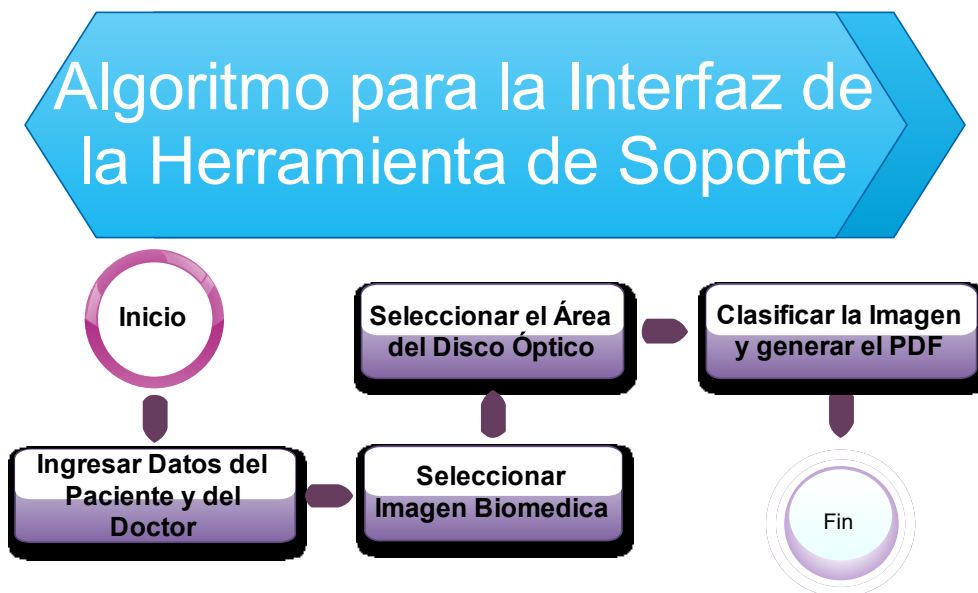


Figura 72. Algoritmo para la interfaz de la Herramienta de Soporte.

Para el desarrollo de la herramienta de soporte es necesario un esquema general (ver Figura 72), la interfaz gráfica fue realizada con la librería TkInter, dado que permite generar ventanas de dialogo, botones, cajas de texto, etiquetas, listas entre otros, el código completo se encuentra en **Anexo H**.

A continuación, se describe cada uno de los elementos presentes en la interfaz (ver Figura 73).

- **Etiquetas de Entrada:** En estos campos se ingresa los nombres y apellidos tanto del paciente como del doctor encargado.
- **Botón Elegir Imagen:** Al pulsar este botón se escoge del directorio la imagen biomédica y se visualiza en Imagen de Entrada.
- **Botón Recortar Imagen:** Este botón al presionarlo abre una ventana con la imagen biomédica elegida y mediante el ratón se arrastra dentro de ella el área del Disco Óptico y se visualiza la imagen en Imagen Recortada.
- **Botón Clasificar Imagen:** Una vez pulsado se aplica la predicción de la red entrenada, genera un PDF con la información ingresada

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

en el directorio actual (ver Figura 74), y se visualiza el resultado en la parte inferior.

- **Botón Acerca de:** Brinda información sobre el uso del programa, el autor y la institución.
- **Botón Limpiar:** Al presionarlo inmediatamente limpia todos los campos de texto como de imagen para el ingreso de nueva información.
- **Botón Salir:** Cierra la Interfaz de la Herramienta de soporte.



Figura 73. Diseño de la interfaz de la Herramienta de Soporte.




 Edison Buri_Paciente	12/11/2021 19:32	Archivo PDF	66 KB
 Gabriel Sarmiento_Paciente	12/11/2021 18:02	Archivo PDF	78 KB
 Maria de Lurdes_Paciente	12/11/2021 18:03	Archivo PDF	80 KB

Figura 74. Archivos PDF generados con la información ingresada en la Herramienta de Soporte.

3.7. Sensibilidad y Especificidad

Para realizar los cálculos de especificidad y sensibilidad se utiliza las ecuaciones (8) y (9), dado que se busca que la red sea lo más cercana a 100% de especificidad, es decir, una red centrada en diagnóstico presuntivo. Además para el cálculo de los valores predictivos positivos y negativos se requieren las ecuaciones (10) y (11), respectivamente. (ver Figura 75)

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

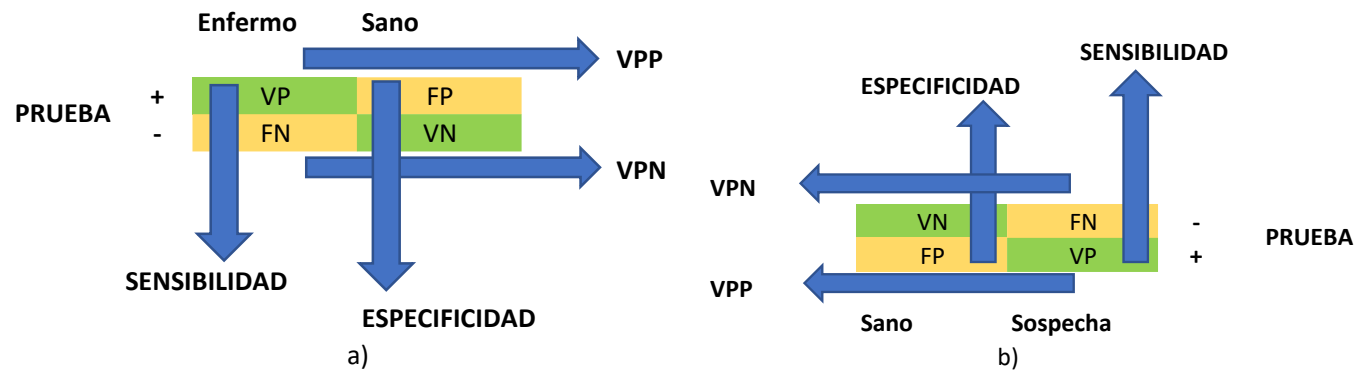


Figura 75. Gold Standard a) Original, b) Reorganizada para el proyecto.

Tabla 8 Sensibilidad, Especificidad, VPP y VPN de los Modelos Propuestos.

	3,1 (a)	3,2 (a)	3,3 (a)	3,4 (a)	3,1 (b)	3,2 (b)	3,3 (b)	3,4 (b)	3,5 (a)	3,5 (b)	3,5 (c)	3,5 (d)
VP	28	30	30	29	28	30	28	29	28	1	30	14
FN	0	3	4	2	3	2	1	5	4	3	3	17
FP	2	0	0	1	2	2	2	1	2	29	0	16
VN	30	27	26	28	27	28	29	25	26	27	27	13
Sensibilidad	100,00	90,91	88,24	93,55	90,32	93,75	96,55	85,29	87,50	25,00	90,91	45,16
Especificidad	93,75	100,00	100,00	96,55	93,10	93,33	93,55	96,15	92,86	48,21	100,00	44,83
VPP	93,33	100,00	100,00	96,67	93,33	93,75	93,33	96,67	93,33	3,33	100,00	46,67
VPN	100,00	90,00	86,67	93,33	90,00	93,33	96,67	83,33	86,67	90,00	90,00	43,33

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

3.8. Análisis de Resultados

Se ha extraído los datos mencionados en las secciones 3.1 – 3.5 (ver Tabla 9).

Tabla 9 Datos de los modelos Propuestos.

	3,1 (a)	3,2 (a)	3,3 (a)	3,4 (a)	3,1 (b)	3,2 (b)	3,3 (b)	3,4 (b)	3,5 (a)	3,5 (b)	3,5 (c)	3,5 (d)
<i>Tiempo</i>	6' 40"	8' 20"	10' 00"	11' 40"	8' 20"	8' 20"	10' 00"	10' 00"	41' 40"	46' 40"	48' 20"	51' 40"
<i>Precisión</i>	0,875	0,8717	0,8684	0,8458	0,8191	0,8289	0,8618	0,7993	0,93403	0,92014	0,93403	0,90625
<i>Pérdida</i>	0,3396	0,2968	0,3123	0,3423	1,4876	0,9934	0,8543	0,4647	0,4236	0,3937	0,2382	1,237
<i>Etiquetas Correctas</i>	266	265	264	257	249	252	262	243				
<i>Etiquetas Incorrectas</i>	38	39	40	47	55	52	42	61				
<i>Predicción Sano</i>	30	27	26	28	27	28	29	25	26	27	27	13
<i>Predicción Sospecha</i>	28	30	30	29	28	30	28	29	28	1	30	14

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

Los modelos de **3.1 – 3.4**, se han separado de acuerdo con los optimizadores a) para Adagrad y b) para Adam, los modelos de **3.5** corresponden a los de transferencia de conocimiento, siendo a) **VGG16**, b) **Inception V3**, c) **Resnet 50** y c) **Xception**.

En los Modelos de **3.1** la diferencia de tiempo es de 1 minuto 40 segundos, en cuanto a la precisión se obtiene una diferencia de 0.0559, siendo el mayor de 0.875 para Adagrad, sin embargo, el modelo proporcionado para el optimizador de Adam presentó mayor pérdida que el de Adagrad, lo cual es notable en las gráficas de precisión (ver Figura 34) y pérdida (ver Figura 35), la cantidad de etiquetados correctos es mayor en Adagrad que en Adam. En la matriz de confusión ambos tuvieron la misma cantidad de aciertos para sospecha, pero en Sanos, Adagrad tuvo mayor cantidad de aciertos, claramente Adagrad es el mejor de los dos.

En los Modelos de **3.2** el tiempo de entrenamiento es el mismo para los dos, se presenta una diferencia de precisión de 0.0428, siendo nuevamente Adagrad el que tiene mayor precisión, de igual forma, las gráficas de precisión (ver Figura 44), pérdida (ver Figura 45) y cantidad de etiquetas correctas confirman lo antes mencionado, sin embargo, en la matriz de confusión ambos optimizadores tuvieron una predicción muy similar, acertando correctamente todas las imágenes de sospecha y habiendo apenas la diferencia de una imagen en los pacientes sanos, dado que Adam tiene mayor pérdida, Adagrad es la mejor opción.

En los Modelos de **3.3** el tiempo de entrenamiento para ambos fue el mismo, de igual manera la precisión de ambos es similar, siendo de 0.86, sin embargo, la pérdida fue mayor en el optimizador Adam, se puede notar estos cambios en sus gráficas de precisión (ver Figura 53) y pérdida (ver Figura 54), respecto a la cantidad de etiquetas correctas, ambos optimizadores tienen cifras similares, la mayor diferencia radica en la matriz de confusión, ya que Adagrad tuvo 30 aciertos en la predicción de sospecha, a diferencia de 28 para Adam.

En los Modelos de **3.4** se tiene una diferencia de 1 minuto 40 segundos en el tiempo de entrenamiento, en el valor de precisión se tiene una diferencia de 0.0465 siendo el valor más alto de precisión con 0.8458 para Adagrad, sin embargo, Adam presenta un valor de perdida mayor, en sus gráficas de precisión (ver Figura 62) y pérdida (ver Figura 63), sin duda Adagrad es mucho más estable, en cuanto a cantidad de etiquetas Adagrad tiene una cantidad más alta con una diferencia de 14 correctas, naturalmente presenta una menor cantidad de etiquetas incorrectas, y en la matriz de confusión ambos tienen la misma cantidad de imágenes correctas para sospecha, nuevamente Adagrad es la mejor opción.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

En los modelos de **3.5** que corresponde a transferencia de conocimiento el más rápido en entrenarse fue VGG16, seguido de Inception V3, Resnet 50 y Xception, con una diferencia de 10 minutos entre VGG16 y Xception, las redes con mejor precisión fueron VGG16 y Resnet 50, seguidos de Inception v3 y Xception con 0.92014 y 0.90625 respectivamente, la red con menor pérdida fue Resnet 50 con 0.2382, seguido de Inception V3 con 0.3937, luego VGG16 con 0.4236 y Xception con 1.237, por último la red que mejor clasificó las imágenes fue Resnet 50, logrando 30 imágenes correctas en sospecha y 27 correctas en sano, seguido de VGG16 con 28 correctas en sospecha y 26 en sano, la red de Inception solo tuvo 1 imagen correcta en sospecha con 27 correctas en sano, por último Xception tuvo 14 correctas en sospecha y 13 en sano.

Para aplicar sensibilidad y especificidad a los modelos presentados es necesario usar una Gold Estándar (ver Figura 75), de esta manera se obtiene que:

En los modelos de **3.1** el valor de sensibilidad es de un 100% para Adagrad y 90.32% para Adam, es decir, que Adagrad puede descartar la sospecha; en especificidad Adagrad tuvo el valor más alto con 93.75%, mientras que Adam tuvo una diferencia de 0.65%, indica que Adagrad tiene el porcentaje más alto de confirmar el diagnóstico, ambos tienen el mismo valor predictivo positivo (VPP), es decir, la misma probabilidad de que el paciente con resultado positivo tenga la sospecha de Glaucoma, mientras que en los valores predictivos negativos Adagrad tiene el máximo valor, es decir, tiene más probabilidad de que un paciente con resultado negativo no tenga sospecha de Glaucoma.

En los modelos de **3.2** el valor de sensibilidad es de 93.75% para Adam, y 90.91% para Adagrad, de esta manera Adam tiene la probabilidad más alta de descartar la enfermedad, el valor de especificidad más alto es de Adagrad, de esta manera tiene un 100% de confirmar el diagnóstico, en los valores predictivos positivos (VPP) Adagrad tiene un 100% de probabilidad de que un paciente con resultado positivo tenga sospecha de Glaucoma, mientras que en los valores predictivos negativos (VPN) Adam tiene un 93.33% de probabilidad de que un paciente con resultado negativo no tenga sospecha de Glaucoma.

En los modelos de **3.3** el valor de sensibilidad es de 96.55% de descartar la enfermedad, siendo el valor más alto para Adam, mientras que en especificidad Adagrad tiene un 100% de confirmar el diagnóstico, en los valores predictivos positivos (VPP) Adagrad tiene un 100% de que un paciente con resultado positivo tenga sospecha de Glaucoma, y en valores predictivos negativos (VPN) Adam tiene un 96.67% de que un paciente con resultado negativo no tenga sospecha de Glaucoma.

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

En los modelos de 3.4 el valor de sensibilidad es de 93.55% de probabilidad de descartar la enfermedad para Adagrad, siendo el más alto con respecto a Adam, de igual manera tiene un 96.55% de especificidad, es decir tiene el valor más alto para confirmar el diagnóstico, el resultado de los valores predictivos positivos (VPP) es el mismo para ambos, es decir, ambos tienen un 96.67% de que un paciente con resultado positivo tenga sospecha de Glaucoma, mientras que Adagrad tiene 93.33% de probabilidad de que un paciente con resultado negativo no tenga sospecha de Glaucoma, a diferencia del 83.33% de Adam.

En los modelos de 3.5 que corresponden a la transferencia de conocimiento, se obtiene que el mejor valor de sensibilidad es de 90.91% para descartar la enfermedad que pertenece a Resnet 50, seguido de 87.50% para VGG16, luego 45.16% para Xception y 25% para Inception V3, se obtiene un 100% de especificidad para Resnet 50, seguido de 92.86% para VGG16, luego 48.21% para Inception V3 y 44.83% para Xception, en los valores predictivos Resnet 50 tiene un 100% de que el paciente con resultado positivo tenga sospecha de Glaucoma, seguido de 93.33% de VGG16, luego 46.47% de Xception y 3.33% para Inception V3, por último en los valores predictivos negativos (VPN) tanto Inception V3 como Resnet 50, tienen un 90% de que el paciente con resultado negativo no tenga sospecha de Glaucoma, seguido de 86.67 para VGG16 y 43.33% para Xception.

4. CONCLUSIONES Y RECOMENDACIONES

El Glaucoma en la actualidad es una enfermedad que no tiene cura. Sin embargo, puede ser tratada si es diagnosticada en una etapa temprana; la mayor parte de personas empiezan a notar los síntomas cuando ya se encuentra en una etapa crítica.

La preparación de la base de datos de imágenes del Fondo de Ojo Humano fue tomada de bases de datos públicas, en cada una se indica que los doctores encargados tienen varios años de experiencia en el diagnóstico de Glaucoma, las imágenes contienen anotaciones de los especialistas en donde se cataloga si el paciente fue diagnosticado sano o existe la sospecha de padecer Glaucoma.

En el presente proyecto, El Disco Óptico es la región de interés, debido a que ocupa la mayor parte de la imagen de Fondo de Ojo, además en este lugar se interconectan los vasos sanguíneos, debido a ello es una importante zona de análisis.

La región de interés se encontraba en diferentes posiciones: en el centro, izquierda, derecha y en algunos casos a los extremos, por esta razón fue necesario realizar un preprocesamiento previo al entrenamiento, posteriormente se normalizan a un tamaño de 224x224 (píxeles), dado que será el tamaño usado para el entrenamiento de los modelos.

La arquitectura de una red neuronal no es una ciencia verídica ya que esta depende en gran medida de la base de datos; la precisión de una red neuronal varía de acuerdo al: tamaño de batch, dimensiones de las imágenes, valor exponencial, número de épocas de entrenamiento, cantidad de capas convolucionales, así como el número de conexiones en la capa densa, tamaño de pool, tamaño de kernel, tamaño de los filtros de cada capa, por ello, es necesario realizar varias pruebas de entrenamiento con diferentes arquitecturas. Es un proceso largo que lleva bastante tiempo, ya que al incrementar o reducir cualquiera de estos valores afecta directamente el tiempo de entrenamiento, el cual puede extenderse de minutos hasta horas, en gran medida depende del hardware utilizado, por lo que se recomienda usar la GPU, activar todos los núcleos del computador, y disponer de suficiente memoria RAM.

Al realizar las diferentes pruebas se recomienda un tamaño de batch de 16, ya que será la cantidad de imágenes que ingresa en cada época, todas las imágenes deben tener el mismo tamaño, la cantidad de épocas con un valor de 100 funciona bien, ya que un valor demasiado alto genera sobre entrenamiento, el tamaño de filtro indica el valor de kernel, es decir, el tamaño del cuadrado que barrera toda la imagen, que forma parte del proceso convolucional. Los números de capas convolucionales, así como el valor de cada una estará limitado por la Memoria Ram del computador,

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

con 12Gb (ddr4), se logró generar un máximo de 4 capas convolucionales y 1 capa de salida, al disponer de mayor capacidad de RAM es posible generar modelos mucho más grandes.

En cada modelo, conforme aumentan las capas convolucionales, el tamaño de kernel también se incrementa, es decir, para la primera arquitectura (3.1) el tamaño de kernel es de valor 3, debido a que posee una capa convolucional, cuando se incrementa a 2 capas convolucionales, el tamaño de kernel de la primera capa es de 5, mientras que en la segunda capa su valor es de 3, de esta manera mientras más capas tenga su valor se incrementa en 2, sea así, 3 - 5 - 7 - 9, y así sucesivamente, esto afecta directamente a las imágenes, ya que realizar una convolución significa que el tamaño de la imagen se reduce, como inicialmente fue de 224x224, en la salida la imagen tendrá un tamaño reducido a la mitad por cada convolución aplicada en ella, lo que indica que se tendría 112x112, 56x56, 28x28, 14x14, respectivamente a los tamaños de kernel antes mencionados, cabe indicar que la cantidad de parámetros de entrenamientos resultantes de ello se reducen notoriamente siendo de aproximadamente 102.7Mb a 26.2Mb, para el modelo 3.1 y 3.4 respectivamente, el archivo generado del entrenamiento también reduce su tamaño considerablemente siendo de aproximadamente 802Mb a 205Mb para Adagrad, en Adam es de aproximadamente 1.2Gb a 308Mb para los modelos mencionados anteriormente.

El tamaño de filtro para cada capa convolucional se ve limitado por la capacidad de Memoria RAM del computador, en las arquitecturas propuestas el primer modelo tiene una capa con un tamaño de 32, en los siguientes modelos se decidió duplicar el valor anterior quedando 32, 64, 128, 256, para los modelos 3.1 – 3.2 – 3.3 – 3.4, respectivamente, siendo cada uno de estos el valor más alto para las capas de dichas arquitecturas, y en la capa densa, es decir, la capa de salida se asignó un valor de 512, dado que es el valor máximo permitido por el hardware.

En la transferencia de conocimiento se utilizó la plataforma Kaggle, ya que permite el desarrollo de arquitecturas mucho más robustas, lo cual es un beneficio para implementar los modelos pre-entrenados: VGG16, Inception V3, Resnet 50 y Xception. La arquitectura tomada de plantilla fue de tres capas convolucionales de las propuestas con el modelo de Adagrad, de este modo, los valores de kernel utilizados son: 3 – 5 – 7, la primera capa tiene el valor más alto; el tamaño de las capas son: 256, 512, 1024, siendo la primera capa el valor más bajo, por último, la capa densa de salida que está conformada de 3 secciones de 4096 cada una, para cada modelo el tamaño del archivo generado es diferente, es así que aproximadamente tienen: 510Mb, 712Mb, 725Mb y 704Mb, para los modelos mencionados anteriormente.

Al obtener la matriz de confusión de cada uno de los modelos propuestos es posible obtener la sensibilidad (indica que el sistema está funcionando) y

CAPÍTULO 3. IMPLEMENTACION Y ANALISIS DE RESULTADOS

especificidad (enfocada en confirmar el diagnóstico presuntivo), para realizar dichos cálculos se usa una Gold Estándar, la cual se trata de una prueba de diagnóstico utilizada dentro de la medicina para determinar la fiabilidad al momento de diagnosticar una enfermedad, dado que el objetivo del proyecto es la especificidad y considerando la transferencia de conocimiento. El modelo Resnet 50 (Modelo 3.5 - c), es el que brindo mejores resultados con un 100% de especificidad y 90.91% de sensibilidad, además tiene un 100% (VPP) de que un paciente con resultado positivo tenga un diagnóstico de sospecha de Glaucoma acertado y un 90% (VPN) de que un paciente con resultado negativo no tenga dicha enfermedad.

La herramienta de soporte fue desarrollada bajo software libre con el objetivo de brindarle al médico apoyo adicional para poder determinar un diagnóstico más acertado de la enfermedad del Glaucoma, entre las librerías más utilizadas se encuentra OpenCV, Keras, sklearn y tkinter, donde las tres primeras son muy importantes al momento de desarrollar una red neuronal. Mientras que la última mencionada brinda el apoyo para realizar una interfaz gráfica dentro del lenguaje de Python.

El software desarrollado es intuitivo, permite el ingreso de imágenes en formato jpg, jpeg y png a color, además se puede ingresar los datos del paciente y el médico encargado, luego de clasificar la imagen automáticamente se genera un PDF con la información brindada anteriormente, de esta manera el médico podrá llevar un mejor control de los pacientes y sus respectivos diagnósticos.

Para que el resultado sea lo más acertado posible se recomienda que en las imágenes de Fondo de Ojo utilizadas, el disco óptico no se encuentre demasiado a los extremos, ya que al momento de recortar en la imagen habrá una parte negra, y dentro de la base de datos utilizada para los entrenamientos no se cuenta con este tipo de imágenes, dado que fueron descartadas.

La tabla de Gold Estándar es necesaria para el cálculo de sensibilidad y especificidad, dado que esta tabla tiene Enfermo y Sano, a diferencia de los datos a clasificar en el presente proyecto que son Sano y Sospecha fue necesario realizar una transposición de los valores en dicha tabla, ya que se encontraba en orden inverso quedando como se muestra en Figura 75.b.

TRABAJOS FUTUROS

Como continuación de este trabajo de tesis se propone lo siguiente:

- Recopilar una nueva base de datos con Imágenes de Fondo de Ojo de Fibras Nerviosas, para posteriormente diseñar una red neuronal artificial que permita la detección de pacientes sanos y de pacientes con sospecha de Glaucoma.
- La base de datos debe comprender un mayor número de muestras de entrada, en la presente tesis se logró recopilar 1518 imágenes, sin embargo, esta cantidad sigue siendo pequeña, por lo que se recomienda un aproximado de 5000 imágenes.
- Al desarrollar la red neuronal convolucional desde cero o realizar transferencia de conocimiento con modelos pre entrenados, la mayor limitante es la capacidad de memoria RAM del computador, por lo que nuevos dispositivos como los Biosistemas o dispositivos similares que permitan el uso de una cantidad similar de capacidad de memoria RAM, abre la puerta al desarrollo de arquitecturas mucho más robustas.

ANEXOS

ANEXO A

Región de Interés CHADSEB

```
#Recorte Imagenes CHADSEB
for i in os.listdir(dirBase):
    img = cv2.imread(os.path.join(dirBase,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/5.5)
    ly = int(img.shape[1]/5.5)
    recor = img[lx: -lx, ly: -ly]

    #segundo Recorte
    lx = int(recor.shape[0]/4.6)
    lxx = int(recor.shape[0]/4)
    lyy = int(recor.shape[1]/5)
    ly = int(recor.shape[1]/4.7)
    recor2 = recor[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot+"\\"+i,recor2)

print(recor.shape)
print(recor2.shape)
```

Región de Interés DRIONS

```
#Recorte Imagenes DRIONS
for i in os.listdir(dirBase2):
    img = cv2.imread(os.path.join(dirBase2,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/60)
    ly = int(img.shape[1]/8)
    recor = img[lx: -lx, ly: -ly]

    #segundo Recorte
    lx = int(recor.shape[0]/15)
    lxx = int(recor.shape[0]/15)
    lyy = int(recor.shape[1]/7)
    ly = int(recor.shape[1]/7)
    recor2 = recor[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot2+"\\"+i,recor2)

print(recor.shape)
print(recor2.shape)
```

ANEXOS

Región de Interés DRISTHI – GS1 Sano

```
#Recorte Imagenes DRISTHI-GS1 SANO
for i in os.listdir(dirBase3):
    img = cv2.imread(os.path.join(dirBase3,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/7.6)
    ly = int(img.shape[1]/5)
    recor = img[lx: -lx, ly: -ly]

    #segundo Recorte
    lx = int(recor.shape[0]/10)
    lxx = int(recor.shape[0]/15)
    lyy = int(recor.shape[1]/15)
    ly = int(recor.shape[1]/15)
    recor2 = recor[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot3+"\\"+i,recor2)

print(recor.shape)
print(recor2.shape)
```

Región de Interés DRISTHI – GS1 Sospecha

```
#Recorte Imagenes DRISTHI-GS1 SOSPECHA
for i in os.listdir(dirBase4):
    img = cv2.imread(os.path.join(dirBase4,i))
    #Normaliza la imagen
    lx4 = int(img.shape[0]/7.6)
    ly4 = int(img.shape[1]/5)
    recor = img[lx4: -lx4, ly4: -ly4]

    #segundo Recorte
    lx = int(recor.shape[0]/10)
    lxx = int(recor.shape[0]/15)
    lyy = int(recor.shape[1]/15)
    ly = int(recor.shape[1]/15)
    recor4 = recor[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot4+"\\"+i,recor4)

print(recor.shape)
print(recor4.shape)
```

ANEXOS

Región de Interés HRF Sano Izquierda

```
#Recorte HRF Sano Izquierda
for i in os.listdir(dirBase5):
    img = cv2.imread(os.path.join(dirBase5,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/5.5)
    ly = int(img.shape[1]/10)
    recor = img[lx:-lx, ly:-ly]

    #segundo Recorte
    lx2 = int(recor.shape[0]/6)
    lxx2 = int(recor.shape[0]/5)
    lyy2 = int(recor.shape[1]/1.7)
    ly2 = int(recor.shape[1]/15)
    recor5 = recor[lx2:-lxx2, ly2:-lyy2]
    cv2.imwrite(dirNot5+"\\"+i,recor5)

print(recor.shape)
print(recor5.shape)
```

Región de Interés HRF Sano Derecha

```
#Recorte HRF Sano Derecho
for i in os.listdir(dirBase6):
    img = cv2.imread(os.path.join(dirBase6,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/5.5)
    ly = int(img.shape[1]/25)
    recor = img[lx:-lx, ly:-ly]

    #segundo Recorte
    lx = int(recor.shape[0]/6.2)
    lxx = int(recor.shape[0]/5)
    lyy = int(recor.shape[1]/30)
    ly = int(recor.shape[1]/1.5)
    recor2 = recor[lx:-lxx, ly:-lyy]
    cv2.imwrite(dirNot6+"\\"+i,recor2)

print(recor.shape)
print(recor2.shape)
```


ANEXOS

Región de Interés RIM ONE r3 Sano

```
#Recorte Imagenes RIM ONE r3 Sano
for i in os.listdir(dirBase9):
    img = cv2.imread(os.path.join(dirBase9,i))
    #segundo Recorte
    lx = int(img.shape[0]/4)
    lyy = int(img.shape[1]/10)
    ly = int(img.shape[1]/2)
    recor9 = img[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot9+"\\"+i, recor9)

print(img.shape)
print(recor9.shape)
```

Región de Interés RIM ONE r3 Sospecha

```
#Recorte Imagenes RIM ONE r3 Sospecha
for i in os.listdir(dirBase10):
    img = cv2.imread(os.path.join(dirBase10,i))
    #Normaliza la imagen
    lx = int(img.shape[0]/4)
    lyy = int(img.shape[1]/10)
    ly = int(img.shape[1]/2)
    recor10 = img[lx: -lx, ly: -lyy]
    cv2.imwrite(dirNot10+"\\"+i, recor10)

print(img.shape)
print(recor10.shape)
```

ANEXOS

ANEXO B

Redimensionamiento de Imágenes

```
import numpy as np
import cv2
import os
```

```
#Imagenes test 224x224
enfermo_folder2 = 'Nueva Base Centrada/Base Normalizada 224/Valid/Sospecha'
enfermo_folder = 'Nueva Base Centrada/Base Normalizada 200/Valid/Sospecha'
enfermo=[]
largo=224
ancho=224
for i in os.listdir(enfermo_folder):
    img = cv2.imread(os.path.join(enfermo_folder,i))
    img_resize = cv2.resize(img,(largo,ancho))
    cv2.imwrite(enfermo_folder+"\\"+i,img_resize)
    enfermo.append(img_resize)
enfermo = np.array(enfermo)
print(enfermo.shape)
```

ANEXO C

```
import numpy as np
import os
import re
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import keras
from keras.utils import to_categorical
from keras.models import Sequential, Input, Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
```

ANEXOS

Cargar Imagenes ¶

```
dirname = os.path.join(os.getcwd(), 'Nueva Base Centrada\Base Normalizada 224\Train')
imgpath = dirname + os.sep

images = []
directories = []
dircount = []
prevRoot=''
cant=0

print("leyendo imagenes de ",imgpath)

for root, dirnames, filenames in os.walk(imgpath):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant=cant+1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images.append(image)
            b = "Leyendo..." + str(cant)
            print (b, end="\r")
            if prevRoot !=root:
                print(root, cant)
                prevRoot=root
                directories.append(root)
                dircount.append(cant)
                cant=0
    dircount.append(cant)

dircount = dircount[1:]
dircount[0]=dircount[0]+1
print('Directorios leidos:',len(directories))
print("Imagenes en cada directorio", dircount)
print('suma Total de imagenes en subdirs:',sum(dircount))
```

Crear Etiquetas

```
labels=[]
indice=0
for cantidad in dircount:
    for i in range(cantidad):
        labels.append(indice)
        indice=indice+1
print("Cantidad etiquetas creadas: ",len(labels))
```

```
Biomedic=[]
indice=0
for directorio in directories:
    name = directorio.split(os.sep)
    print(indice , name[len(name)-1])
    Biomedic.append(name[len(name)-1])
    indice=indice+1
```

ANEXOS

```
y = np.array(labels)
X = np.array(images, dtype=np.uint8) #convierto de lista a numpy

# Numero de Etiquetas
classes = np.unique(y)
nClasses = len(classes)
print('Numero de clases de Salida: ', nClasses)
print('Clases: ', classes)
```

Set Entrenamiento

```
#Mezclar todo y crear los grupos de entrenamiento y testing
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Características Datos de entrenamiento : ', train_X.shape, train_Y.shape)
print('Características Datos de Test : ', test_X.shape, test_Y.shape)
```

```
plt.figure(figsize=[5,5])

# Primera Imagen de set de Entrenamiento
plt.subplot(121)
plt.imshow(train_X[0, :, :], cmap='gray')
plt.title("Imagen Entrenamiento : {}".format(train_Y[0]))

# Primera imagen de set de testeo
plt.subplot(122)
plt.imshow(test_X[0, :, :], cmap='gray')
plt.title("Imagen Testeo : {}".format(test_Y[0]))
```

Preprocesamiento

```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
```

Encoding para la Red

```
#Mezclar todo y crear los grupos de entrenamiento y testing
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Forma de datos de entrenamiento : ', train_X.shape, train_Y.shape)
print('Forma de datos de Test : ', test_X.shape, test_Y.shape)

train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.

# Cambios en Las etiquetas desde categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Cambios en Las etiquetas de categoria using one-hot encoding
print('Etiqueta Original:', train_Y[0])
print('Despues de la conversion:', train_Y_one_hot[0])

train_X, valid_X, train_label, valid_label = train_test_split(train_X, train_Y_one_hot, test_size=0.2, random_state=13)
print(train_X.shape, valid_X.shape, train_label.shape, valid_label.shape)
```

ANEXOS

Set de Entrenamiento

```
INIT_LR = 1e-3
epochs = 100
batch_size = 16
width, height = 224, 224
filtrosConv1 = (5,5)
filtrosConv2 = (3,3)
tamano_filtro1 = 32
tamano_filtro2 = 64
tamano_pool = (2,2)

biomedic_model = Sequential()

biomedic_model.add(Conv2D(tamano_filtro1,
                           kernel_size=filtrosConv1,
                           activation='linear',
                           padding='same',
                           input_shape=(width,height,3)))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Conv2D(tamano_filtro2,
                           kernel_size=filtrosConv2,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Flatten())
biomedic_model.add(Dense(512, activation='linear'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(Dropout(0.1))
biomedic_model.add(Dense(nClasses, activation='softmax'))

biomedic_model.summary()

biomedic_model.compile(loss=keras.losses.categorical_crossentropy,
                       optimizer=keras.optimizers.Adagrad(lr=INIT_LR,
                                                            decay=INIT_LR / 100),
                       metrics=['accuracy'])
```

Entrenamiento de la Red ¶

```
biomedic_train = biomedic_model.fit(train_X, train_label,
                                     batch_size=batch_size,
                                     epochs=epochs,
                                     verbose=1,
                                     validation_data=(valid_X, valid_label))
```


ANEXOS

Guardar

```
# guardamos la red, para reutilizarla en el futuro, sin tener que volver a entrenar
biomedic_model.save("biomedical_Adagrad_Acrima1.hdf5")
```

Evaluar la Red

```
test_eval = biomedic_model.evaluate(test_X, test_Y_one_hot, verbose=1)
```

Evaluación la Red

```
accuracy = biomedic_train.history['accuracy']
val_accuracy = biomedic_train.history['val_accuracy']
loss = biomedic_train.history['loss']
val_loss = biomedic_train.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Precision Entrenamiento')
plt.plot(epochs, val_accuracy, 'b', label='Precision Validacion')
plt.title('Precision de Entrenamiento y Validacion')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Perdida Entrenamiento')
plt.plot(epochs, val_loss, 'b', label='Perdida Validacion')
plt.title('Perdida de Entrenamiento y Validacion')
plt.legend()
plt.show()
```

```
predicted_classes2 = biomedic_model.predict(test_X)
```

```
predicted_classes=[]
for predicted_biomedic in predicted_classes2:
    predicted_classes.append(predicted_biomedic.tolist().index(max(predicted_biomedic)))
predicted_classes=np.array(predicted_classes)
```

```
predicted_classes.shape, test_Y.shape
```

```
((163,), (163,))
```

```
correct = np.where(predicted_classes==test_Y)[0]
print("Encontradas %d etiquetas correctas" % len(correct))
for i, correct in enumerate(correct[0:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[correct].reshape(width,height,3), cmap='gray', interpolation='none')
    plt.title("{} , {}".format(Biomedic[predicted_classes[correct]],
                               Biomedic[test_Y[correct]]))

plt.tight_layout()
```

ANEXOS

```
incorrect = np.where(predicted_classes!=test_Y)[0]
print("Encontradas %d etiquetas incorrectas " % len(incorrect))
for i, incorrect in enumerate(incorrect[0:9]):
    plt.subplot(3,3,i+1)
    plt.imshow(test_X[incorrect].reshape(width,height,3), cmap='gray', interpolation='none')
    plt.title("{} , {}".format(Biomedic[predicted_classes[incorrect]],
                               Biomedic[test_Y[incorrect]]))
plt.tight_layout()
```

```
target_names = ["Clase {}".format(i) for i in range(nClasses)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

```
target_names = ["Clase {}".format(i) for i in range(nClasses)]
print(classification_report(test_Y, predicted_classes, target_names=target_names))
```

```
from skimage.transform import resize

images=[]
# Carga la imagen
filenames = ['Nueva Base Centrada/Base Normalizada 224/Test/Sano/sano1.jpg']

for filepath in filenames:
    image = plt.imread(filepath,0)
    image_resized = resize(image, (width, height),anti_aliasing=True,clip=False,preserve_range=True)
    images.append(image_resized)

X = np.array(images, dtype=np.uint8) #conversion a numpy
test_X = X.astype('float32')
test_X = test_X / 255.

predicted_classes3 = biomedic_model.predict(test_X)

for i, img_tagged in enumerate(predicted_classes3):
    print(filenames[i], Biomedic[img_tagged.tolist().index(max(img_tagged))])
```

Matriz de Confusión

```
from sklearn.metrics import confusion_matrix, f1_score, roc_curve,
from sklearn.metrics import precision_score, recall_score, accuracy_score, roc_auc_score
from sklearn import metrics
from mlxtend.plotting import plot_confusion_matrix
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```


ANEXOS

```
width_shape = 224
height_shape = 224
batch_size = 16

names = ['Sano', 'Sospecha']

test_data_dir = 'Nueva Base Centrada/Base 224 Acrima/Test'
test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(width_shape, height_shape),
    batch_size = batch_size,
    class_mode='categorical',
    shuffle=False)

custom_Model= load_model("biomedical_Adam_Acrima1.hdf5")

predictions = custom_Model.predict_generator(generator=test_generator)
y_pred = np.argmax(predictions, axis=1)
y_real = test_generator.classes

matc=confusion_matrix(y_real, y_pred)

plot_confusion_matrix(conf_mat=matc, figsize=(9,9), class_names = names, show_normed=False)
plt.tight_layout()

print(metrics.classification_report(y_real,y_pred, digits = 4))
```

ANEXO D

Set de Entrenamiento

```
INIT_LR = 1e-3
epochs = 100
batch_size = 16
width, height = 224, 224
filtrosConv1 = (5,5)
filtrosConv2 = (3,3)
tamano_filtro1 = 32
tamano_filtro2 = 64
tamano_pool = (2,2)

biomedic_model = Sequential()

biomedic_model.add(Conv2D(tamano_filtro1,
                          kernel_size=filtrosConv1,
                          activation='linear',
                          padding='same',
                          input_shape=(width,height,3)))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Conv2D(tamano_filtro2,
                          kernel_size=filtrosConv2,
                          activation='linear',
                          padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Flatten())
biomedic_model.add(Dense(512, activation='linear'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(Dropout(0.1))
biomedic_model.add(Dense(nClasses, activation='softmax'))
```

ANEXO E

Set de Entrenamiento

```
INIT_LR = 1e-3
epochs = 100
batch_size = 16
width, height = 224, 224
filtrosConv1 = (7,7)
filtrosConv2 = (5,5)
filtrosConv3 = (3,3)
tamano_filtro1 = 32
tamano_filtro2 = 64
tamano_filtro3 = 128
tamano_pool = (2,2)

biomedic_model = Sequential()

biomedic_model.add(Conv2D(tamano_filtro1,
                           kernel_size=filtrosConv1,
                           activation='linear',
                           padding='same',
                           input_shape=(width,height,3)))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))
```

ANEXOS

```
biomedic_model.add(Conv2D(tamano_filtro2,
                           kernel_size=filtrosConv2,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool, padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Conv2D(tamano_filtro3,
                           kernel_size=filtrosConv3,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool, padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Flatten())
biomedic_model.add(Dense(512, activation='linear'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(Dropout(0.1))
biomedic_model.add(Dense(nClasses, activation='softmax'))
```

ANEXO F

Set de Entrenamiento

```
INIT_LR = 1e-3
epochs = 100
batch_size = 16
width, height = 224, 224
filtrosConv1 = (9,9)
filtrosConv2 = (7,7)
filtrosConv3 = (5,5)
filtrosConv4 = (3,3)
tamano_filtro1 = 32
tamano_filtro2 = 64
tamano_filtro3 = 128
tamano_filtro4 = 256
tamano_pool = (2,2)

biomedic_model = Sequential()

biomedic_model.add(Conv2D(tamano_filtro1,
                           kernel_size=filtrosConv1,
                           activation='linear',
                           padding='same',
                           input_shape=(width,height,3)))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))
```


ANEXOS

```
biomedic_model.add(Conv2D(tamano_filtro2,
                           kernel_size=filtrosConv2,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Conv2D(tamano_filtro3,
                           kernel_size=filtrosConv3,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Conv2D(tamano_filtro4,
                           kernel_size=filtrosConv4,
                           activation='linear',
                           padding='same'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(MaxPooling2D(tamano_pool,padding='same'))
biomedic_model.add(Dropout(0.25))

biomedic_model.add(Flatten())
biomedic_model.add(Dense(512, activation='linear'))
biomedic_model.add(LeakyReLU(alpha=0.1))
biomedic_model.add(Dropout(0.1))
biomedic_model.add(Dense(nClasses, activation='softmax'))
```

ANEXOS

ANEXO G

```
import os
print(os.listdir("../input"))

data_dir = '../input/glaucoma/Glaucoma10/Glaucoma10'
vgg16weight = '../input/keras-pretrained-models/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5'
resnet50weight = '../input/keras-pretrained-models/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
inceptionV3weight = '../input/keras-pretrained-models/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'
xceptionweight = '../input/keras-pretrained-models/xception_weights_tf_dim_ordering_tf_kernels_notop.h5'

from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model, Sequential
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization, GlobalAveragePooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.optimizers import RMSprop, SGD
from keras import backend as K
from keras.layers.advanced_activations import LeakyReLU
import keras
from keras.applications.vgg16 import VGG16 as VGG16, preprocess_input as preprocess_input_VGG16
from keras.applications.xception import Xception as Xception, preprocess_input as preprocess_input_Xception
from keras.applications.inception_v3 import InceptionV3 as InceptionV3, preprocess_input as preprocess_input_InceptionV3
from keras.applications.resnet50 import ResNet50 as ResNet50, preprocess_input as preprocess_input_ResNet50

import matplotlib.pyplot as plt

img_width, img_height = 224, 224

train_data_dir = os.path.join(data_dir, 'Train')
nb_train_samples = 1215
nb_validation_samples = 303
epochs = 100
batch_size = 16
numclasses = 2
```

ANEXOS

```
def generar_dataset_train(func_preprocess):

    train_datagen = ImageDataGenerator(
        rescale=1. / 255,
        rotation_range=10, # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.1, # Randomly zoom image
        width_shift_range=0.1, # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1, # randomly shift images vertically (fraction of total height)
        shear_range=0.2,
        vertical_flip=False,
        horizontal_flip=False,
        validation_split=0.2,
        preprocessing_function=func_preprocess,
    )

    train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical',
        subset='training')

    validation_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode='categorical',
        subset='validation')

    return train_generator, validation_generator

if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)
```


ANEXOS

```
def vgg16CNN(input_shape, outclass, sigma='sigmoid'):

    base_model = VGG16(weights=vgg16weight, include_top=False, input_shape=input_shape)

    top_model = Sequential()
    top_model.add(Conv2D(256,
                        kernel_size=(7,7),
                        activation='linear',
                        padding='same',
                        input_shape=base_model.output_shape[1:]))
    top_model.add(MaxPooling2D((2,2),padding='same'))
    top_model.add(Dropout(0.25))
    top_model.add(Conv2D(512,
                        kernel_size=(5,5),
                        activation='linear',
                        padding='same'))
    top_model.add(MaxPooling2D((2,2),padding='same'))
    top_model.add(Dropout(0.25))
    top_model.add(Conv2D(1024,
                        kernel_size=(3,3),
                        activation='linear',
                        padding='same'))
    top_model.add(MaxPooling2D((2,2),padding='same'))
    top_model.add(Dropout(0.25))

    top_model.add(Flatten())

    for i in range(3):
        top_model.add(Dense(4096, activation='relu'))
        top_model.add(Dropout(0.1))
    top_model.add(Dense(outclass, activation=sigma))

    return Model(inputs=base_model.input, outputs=top_model(base_model.output))
```

ANEXOS

```
modelvgg16 = vgg16CNN(input_shape, numclasses, 'softmax')
lr = 1e-5
decay = 1e-8 #0.0
optimizer = RMSprop(lr=lr, decay=decay)
modelvgg16.compile(loss='categorical_crossentropy',
                   optimizer=optimizer,
                   metrics=['accuracy'])

checkpoint_vgg16 = keras.callbacks.ModelCheckpoint("vgg16_model.hdf5",
                                                  monitor='val_accuracy',
                                                  verbose=1,
                                                  save_best_only=True,
                                                  mode='auto',
                                                  save_weights_only=False)

modelinceptionV3 = inceptionV3(input_shape, numclasses, 'softmax')
lr = 1e-5
decay = 1e-8 #0.0
optimizer = RMSprop(lr=lr, decay=decay)
modelinceptionV3.compile(loss='categorical_crossentropy',
                        optimizer=optimizer,
                        metrics=['accuracy'])

checkpoint_inceptionV3 = keras.callbacks.ModelCheckpoint("inceptionV3_model.hdf5",
                                                         monitor='val_accuracy',
                                                         verbose=1,
                                                         save_best_only=True,
                                                         mode='auto',
                                                         save_weights_only=False)
```

ANEXOS

```
def inceptionV3(input_shape, outclass, sigma='sigmoid'):  
  
    base_model = InceptionV3(weights=inceptionV3weight, include_top=False, input_shape=input_shape)  
  
    top_model = Sequential()  
    top_model.add(Conv2D(256,  
                        kernel_size=(7,7),  
                        activation='linear',  
                        padding='same',  
                        input_shape=base_model.output_shape[1:]))  
    top_model.add(MaxPooling2D((2,2),padding='same'))  
    top_model.add(Dropout(0.25))  
    top_model.add(Conv2D(512,  
                        kernel_size=(5,5),  
                        activation='linear',  
                        padding='same'))  
    top_model.add(MaxPooling2D((2,2),padding='same'))  
    top_model.add(Dropout(0.25))  
    top_model.add(Conv2D(1024,  
                        kernel_size=(3,3),  
                        activation='linear',  
                        padding='same'))  
    top_model.add(MaxPooling2D((2,2),padding='same'))  
    top_model.add(Dropout(0.25))  
  
    top_model.add(Flatten())  
  
    for i in range(3):  
        top_model.add(Dense(4096, activation='relu'))  
        top_model.add(Dropout(0.1))  
    top_model.add(Dense(outclass, activation=sigma))  
  
    return Model(inputs=base_model.input, outputs=top_model(base_model.output))
```

ANEXOS

```
modelvgg16 = vgg16CNN(input_shape, numclasses, 'softmax')
lr = 1e-5
decay = 1e-8 #0.0
optimizer = RMSprop(lr=lr, decay=decay)
modelvgg16.compile(loss='categorical_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])

checkpoint_vgg16 = keras.callbacks.ModelCheckpoint("vgg16_model.hdf5",
                                                  monitor='val_accuracy',
                                                  verbose=1,
                                                  save_best_only=True,
                                                  mode='auto',
                                                  save_weights_only=False)

modelinceptionV3 = inceptionV3(input_shape, numclasses, 'softmax')
lr = 1e-5
decay = 1e-8 #0.0
optimizer = RMSprop(lr=lr, decay=decay)
modelinceptionV3.compile(loss='categorical_crossentropy',
                        optimizer=optimizer,
                        metrics=['accuracy'])

checkpoint_inceptionV3 = keras.callbacks.ModelCheckpoint("inceptionV3_model.hdf5",
                                                         monitor='val_accuracy',
                                                         verbose=1,
                                                         save_best_only=True,
                                                         mode='auto',
                                                         save_weights_only=False)
```

ANEXOS

```
def graficas_perdida(history):
    training_loss = history.history['loss']
    training_acc = history.history['accuracy']

    # Create count of the number of epochs
    epoch_count = range(1, len(training_loss) + 1)

    fig=plt.figure(figsize=(12, 4))
    # Visualize loss history
    fig.add_subplot(121)
    plt.plot(epoch_count, training_loss, 'r--')
    plt.plot(epoch_count, training_acc, 'b-')
    plt.legend(['Training Loss', 'Training Accuracy'])
    plt.xlabel('Epoch')
    plt.ylabel('Training Loss/Acc')

    # Get training and test loss histories
    val_acc = history.history['val_accuracy']
    training_acc = history.history['accuracy']

    # Create count of the number of epochs
    epoch_count = range(1, len(val_acc) + 1)

    # Visualize loss history
    fig.add_subplot(122)
    plt.plot(epoch_count, val_acc, 'r--')
    plt.plot(epoch_count, training_acc, 'b-')
    plt.legend(['Validation Accuracy', 'Training Accuracy'])
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')

    plt.show();
```

ANEXOS

```
#modelvgg16 checkpoint_vgg16 preprocess_input_VGG16
#modelinceptionV3 checkpoint_inceptionV3 preprocess_input_Xception
#modelresnet50 checkpoint_resnet50 preprocess_input_ResNet50
#modelxception checkpoint_xception preprocess_input_InceptionV3

train_generator, validation_generator = generar_dataset_train(preprocess_input_VGG16)
history_vgg16 = modelvgg16.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
    callbacks = checkpoint_vgg16)
```

```
graficas_perdida(history_vgg16)
```

```
from keras.models import load_model
```

```
modelLoadVgg16 = load_model('./vgg16_model.hdf5')
modelLoadInceptionV3 = load_model('./inceptionV3_model.hdf5')
modelLoadResNet50 = load_model('./resnet50_model.hdf5')
modelLoadXception = load_model('./xception_model.hdf5')
```

```
test_datagen = ImageDataGenerator(
    rescale=1./255,
    preprocessing_function=preprocess_input_VGG16,)
```

```
test_generator= test_datagen.flow_from_directory(
    directory='../input/glaucoma/GlaucomaTest/GlaucomaTest/Test',
    color_mode="rgb",
    shuffle = False,
    class_mode='categorical',
    batch_size=1)
```

```
filenames = test_generator.filenames
nb_samples = len(filenames)
```

```
pred_vgg16 = modelLoadVgg16.predict_generator(test_generator, steps = nb_samples)
pred_inceptionV3 = modelLoadInceptionV3.predict_generator(test_generator, steps = nb_samples)
pred_resnet50 = modelLoadResNet50.predict_generator(test_generator, steps = nb_samples)
pred_xception = modelLoadXception.predict_generator(test_generator, steps = nb_samples)
```


ANEXOS

```
import numpy as np

predicted_class_indices_vgg16= np.argmax(pred_vgg16,axis=1)
predicted_class_indices_inceptionV3= np.argmax(pred_inceptionV3,axis=1)
predicted_class_indices_resnet50= np.argmax(pred_resnet50,axis=1)
predicted_class_indices_xception= np.argmax(pred_xception,axis=1)

# ETIQUETAMOS
labels= (test_generator.class_indices)
labels= dict((v,k) for k,v in labels.items())

predictions_vgg16= [labels[k] for k in predicted_class_indices_vgg16]
predictions_inceptionV3= [labels[k] for k in predicted_class_indices_inceptionV3]
predictions_resnet50= [labels[k] for k in predicted_class_indices_resnet50]
predictions_xception= [labels[k] for k in predicted_class_indices_xception]

filenames= test_generator.filenames
real_class_indices=[]
for i in range (0,len(filenames)):
    your_path= filenames[i]
    path_list= your_path.split(os.sep)
    if ("Sano" in path_list[0]):
        real_class_indices.append(0)
    else:
        real_class_indices.append(1)
real_class_indices= np.array(real_class_indices)

from sklearn.metrics import confusion_matrix
import seaborn as sns

def display_cm(y_test, y_pred):
    sns.heatmap(confusion_matrix(y_test, y_pred, normalize = 'true'),
                annot = True, cmap = 'Blues', fmt='g', cbar=False)
    plt.title('Real vs pred')
    plt.xlabel('Predicted values')
    plt.ylabel('Real values');

    plt.show()
```

ANEXOS

```
display_cm(real_class_indices, predicted_class_indices_vgg16)

cm= confusion_matrix(real_class_indices, predicted_class_indices_vgg16)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap= plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks= np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm= cm.astype('float')/cm.sum(axis=1)[:, np.newaxis]
        print("Normalizacion matriz confusion")
    else:
        print('Matriz Confusion, con normalizacion')

    print(cm)

    thresh= cm.max()/2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j,i, cm[i,j],
                     horizontalalignment="center",
                     color="white" if cm[i,j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('TRUE LABEL')
    plt.xlabel('PREDICTED LABEL')
cm_plot_labels= sorted(train_generator.class_indices)
plot_confusion_matrix(cm, cm_plot_labels, title='VGG16')
```


ANEXOS

ANEXO H

A continuación el Código del Software de soporte.

```
# -*- coding: utf-8 -*-
```

```
"""
```

Herramienta de soporte para Diagnostico de Sospecha de Glacuoma
Interfaz para recortar la imagen y obtener una predicción mediante el uso
de redes neuronales convolucionales, red preentrenada Resnet50
imagenes de base de datos 1518
imagenes de test 60, 30 sano y 30 enfermo

Creado 18 /08 /2021 10:08

```
@author: Edison Buri
```

```
"""
```

```
from tkinter import *  
from tkinter import filedialog  
from PIL import Image  
from PIL import ImageTk  
import cv2  
import numpy as np  
import tkinter.messagebox  
import tkinter as tk  
from datetime import datetime  
from keras.models import load_model  
from keras.preprocessing.image import ImageDataGenerator  
import keras
```

ANEXOS

```
from keras.applications.vgg16 import VGG16 as VGG16, preprocess_input as
preprocess_input_VGG16

from keras.applications.xception import Xception as Xception, preprocess_input as
preprocess_input_Xception

from keras.applications.inception_v3 import InceptionV3 as InceptionV3,
preprocess_input as preprocess_input_InceptionV3

from keras.applications.resnet50 import ResNet50 as ResNet50, preprocess_input as
preprocess_input_ResNet50

from fpdf import FPDF
```

```
cropping = False
stop = False
x_start, y_start, x_end, y_end = 0, 0, 0, 0
```

```
def resize(img,w,h):
    width = w
    height = h
    return cv2.resize(img,(width,height), interpolation = cv2.INTER_CUBIC)
```

```
def elegir_imagen():
    # Especificar los tipos de archivos, para elegir solo a las imágenes
    lblClassification.config(text="")
    path_image = filedialog.askopenfilename(filetypes = [
        ("image", ".jpeg"),
        ("image", ".png"),
        ("image", ".jpg")])
    if len(path_image) > 0:
        global image
        # Leer la imagen de entrada y la redimensionamos
```

ANEXOS

```
inputImg = cv2.imread(path_image)
image = resize(inputImg, 880, 880)
imageToShow= resize(image, 480, 480)
imageToShow = cv2.cvtColor(imageToShow, cv2.COLOR_BGR2RGB)
im = Image.fromarray(imageToShow )
img = ImageTk.PhotoImage(image=im)
lblInputImage.configure(image=img)
lblInputImage.image = img
```

```
def mouse_crop(event, x, y, flags, param):
```

```
    global x_start, y_start, x_end, y_end, cropping, stop, oriImage, recorImage
    if event == cv2.EVENT_LBUTTONDOWN:
        x_start, y_start, x_end, y_end = x, y, x, y
        cropping = True
    elif event == cv2.EVENT_MOUSEMOVE:
        if cropping == True:
            x_end, y_end = x, y
    elif event == cv2.EVENT_LBUTTONUP:
        x_end, y_end = x, y
        cropping = False
        stop = True
        refPoint = [(x_start, y_start), (x_end, y_end)]
        if len(refPoint) == 2:
            recorImage = oriImage[refPoint[0][1]:refPoint[1][1],
refPoint[0][0]:refPoint[1][0]]
```

```
def recortar_imagen():
```

ANEXOS

```
global image, oriImage, x_start, y_start, x_end, y_end, cropping, stop
lblClassification.config(text="")
if 'image' in globals():
    cv2.namedWindow("Recortar imagen")
    cv2.setMouseCallback("Recortar imagen", mouse_crop)
    oriImage = image.copy()
    while True:
        i = image.copy()
        if not cropping:
            cv2.imshow("Recortar imagen", image)
        elif cropping:
            cv2.rectangle(i, (x_start, y_start), (x_end, y_end), (255, 0, 0), 2)
            cv2.imshow("Recortar imagen", i)
        k = cv2.waitKey(1) & 0xFF
        if stop:
            cv2.destroyAllWindows()
            break
    # close all open windows
    stop = False
    mostrar_imagen()

def mostrar_imagen():
    global recorImage, imageToShowOutput
    imageToShowOutput = resize(cv2.cvtColor(recorImage,
cv2.COLOR_BGR2RGB), 480, 480)
    # Para visualizar la imagen en lblOutputImage en la GUI
    im = Image.fromarray(imageToShowOutput)
    img = ImageTk.PhotoImage(image=im)
    lblOutputImage.configure(image=img)
```

ANEXOS

```
lblOutputImage.image = img
```

```
def imagen_default():
```

```
    image = cv2.imread('./fondo.png')
    imageToShow= resize(image, 480, 480)
    imageToShow = cv2.cvtColor(imageToShow, cv2.COLOR_BGR2RGB)
    im = Image.fromarray(imageToShow )
    img = ImageTk.PhotoImage(image=im)
    lblInputImage.configure(image=img)
    lblInputImage.image = img
    lblOutputImage.configure(image=img)
    lblOutputImage.image = img
```

```
def imagen_default2():
```

```
    image2 = cv2.imread('./ups.png')
    imageToShow2= resize(image2, 250, 100)
    imageToShow2 = cv2.cvtColor(imageToShow2, cv2.COLOR_BGR2RGB)
    im2 = Image.fromarray(imageToShow2)
    img2 = ImageTk.PhotoImage(image=im2)
    lblInputImage.configure(image=img2)
    lblInputImage.image = img2
```

```
def guardar_pdf(diagnostico):
```

```
    fecha = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font('Arial', "", 11)
    pdf.cell(15, 7, 'Paciente: '+entPaciente.get(), ln=1)
```

ANEXOS

```
pdf.cell(15, 7, 'Doctor: '+entDoctor.get(), ln=1)
pdf.cell(15, 7, 'Diagnostico: '+diagnostico, ln=1)
pdf.cell(15, 7, 'Fecha: ' +fecha, ln=1)
pdf.image('./Imagen Recortada/Predict/recor.png', x = None, y = None, w = 100,
h = 100, type = "")
pdf.output(entPaciente.get()+'_diagnostico.pdf, 'F')
```

```
def acerca_de():
```

```
    tkinter.messagebox.showinfo('Acerca de',"Herramienta de soporte para  
diagnóstico de sospecha de glaucoma con el"
```

```
        "uso de redes neuronales convolucionales \n\n"
```

```
        "1. Ingresar datos de paciente y doctor\n"
```

```
        "2. seleccionar la imagen biomedica\n"
```

```
        "3. Recortar el Area del Disco Óptico\n"
```

```
        "4. Finalmente, click en clasificar para obtener la  
predicción\n\n"
```

```
        "Autor: Edison Buri Abad\n"
```

```
        "Institución: Universidad Politecnica Salesiana")
```

```
def limpiar():
```

```
    entPaciente.delete(0, tk.END)
```

```
    entDoctor.delete(0, tk.END)
```

```
    lblClassification.config(text="")
```

```
    imagen_default()
```

```
def salir():
```

```
    root.destroy()
```

```
def analizar_imagen():
```

ANEXOS

```
global modelLoadCNN, recorImage
cv2.imwrite('./Imagen Recortada/Predict/recor.png', recorImage)

test_datagen =
ImageDataGenerator(rescale=1./255,preprocessing_function=preprocess_input_ResNet50,)

test_generator= test_datagen.flow_from_directory(
    directory='./Imagen Recortada/',
    color_mode="rgb",
    shuffle = False,
    class_mode='categorical',
    batch_size=1)

prediction = modelLoadCNN.predict(test_generator)
if np.argmax(prediction[0]) == 0:
    lblClassification.config(text="Resultado: La imagen analizada no presenta Sospecha de Glaucoma")
    guardar_pdf('Sano')
else:
    lblClassification.config(text="Resultado: La imagen analizada presenta Sospecha de Glaucoma")
    guardar_pdf('Sospecha de Glaucoma')

print(keras.__version__)

modelLoadCNN = load_model('./resnet50_model.hdf5')
print('Modelo leído y cargado satisfactoriamente')

root = Tk()
root.title('ANALISIS PRESUNTIVO DEL GLAUCOMA BASADO EN EL DISCO OPTICO CON REDES NEURNALES CONVOLUCIONALES')
```

ANEXOS

Label donde se presentará el logo

```
lblInputImage = Label(root)
```

```
lblInputImage.grid(column=0, columnspan=1, row=0, rowspan=2, padx=5, pady=10)
```

```
imagen_default2()
```

Label donde se presentará la imagen de entrada

```
lblInputImage = Label(root)
```

```
lblInputImage.grid(column=2, row=2, rowspan=5, padx=5, pady=10)
```

```
lblInfo1 = Label(root, text="IMAGEN DE ENTRADA", font=(10))
```

```
lblInfo1.grid(column=2, row=1, padx=5, pady=5, sticky='S')
```

Label donde se presentará la imagen de salida

```
lblOutputImage = Label(root)
```

```
lblOutputImage.grid(column=3, row=2, rowspan=5, padx=5, pady=10)
```

```
lblInfo3 = Label(root, text="IMAGEN RECORTADA", font=(10))
```

```
lblInfo3.grid(column=3, row=1, padx=5, pady=5, sticky='S')
```

```
imagen_default()
```

Label del titulo

```
lblTitulo = Label(root, text="ANALISIS PRESUNTIVO DEL GLAUCOMA  
BASADO EN EL", font=14)
```

```
lblTitulo.grid(column=1, columnspan=4, row=0, padx=20, sticky='N')
```

```
lblTitulo = Label(root, text="DISCO ÓPTICO CON REDES NEURNALES  
CONVOLUCIONALES", font=14)
```

```
lblTitulo.grid(column=1, columnspan=4, row=0, padx=20, sticky='S')
```

Label del Paciente

```
lblPaciente = Label(root, text=" Nombre del Paciente:", font=5)
```


ANEXOS

```
lblPaciente.grid(column=0, row=2, padx=20, sticky='N')
entPaciente = Entry(root, bd =5, font=8)
entPaciente.grid(column=0, row=2, padx=20)

# Label del Doctor
lblDoctor = Label(root, text="Nombre del Doctor:", font=5)
lblDoctor.grid(column=0, row=2, padx=20, sticky='S')
entDoctor = Entry(root, bd =5,font=8)
entDoctor.grid(column=0, row=3, padx=20, sticky='N')

# Creamos el botón para elegir la imagen de entrada
btnElegir = Button(root, text="Elegir imagen", width=25, command=elegir_imagen,
font=5)
btnElegir.grid(column=0, columnspan=2, row=3, padx=20, sticky='S')

# Creanos ek boton para recortar la imagen de entrada
btnRecortar = Button(root, text="Recortar imagen", width=25,
command=recortar_imagen, font=5)
btnRecortar.grid(column=0, columnspan=2, row=4, padx=20)

#Creamos el boton para clasificar la imagen de entrada
btnClasificar = Button(root, text="Clasificar imagen", width=25,
command=analizar_imagen, font=5)
btnClasificar.grid(column=0, columnspan=2, row=5, padx=20, sticky='N')
lblClassification = Label(root, text="", font=(14))
lblClassification.grid(column=2, row=7, columnspan=2, padx=5, pady=10)

# Creamos el boton Acerca de
```

ANEXOS

```
btnAcerca = Button(root, text="Acerca de", width=25, command=acerca_de,  
font=5)
```

```
btnAcerca.grid(column=0, columnspan=2, row=6, padx=20, sticky='N')
```

```
# Creamos el boton Limpiar
```

```
btnLimpiar = Button(root, text="Limpiar", width=25, command=limpiar, font=5)
```

```
btnLimpiar.grid(column=0, columnspan=2, row=6, padx=20, sticky='S')
```

```
#Creamos el boton Salir
```

```
btnSalir = Button(root, text="Salir", width=25, command=salir, font=5)
```

```
btnSalir.grid(column=0, columnspan=2, row=7, padx=20, sticky='N')
```

```
root.mainloop()
```

REFERENCIAS BIBLIOGRAFICAS

- [1] F. B. Garrido, F. A. H. Simanca, P. E. H. Carreño, y M. A. Moncada, “Development of Accessible Website as support for people with visual disability”, *Iber. Conf. Inf. Syst. Technol. Cist.*, vol. 2019-June, núm. June, pp. 19–22, 2019, doi: 10.23919/CISTI.2019.8760793.
- [2] A. Saucedo Castillo y S. Rojas Juarez, “Glaucoma”, en *Oftalmologia*, C. Mendoza, Ed. Mexico: Editorial El Manual Moderno, 2014, p. 437.
- [3] CONADIS, “Estadísticas de Discapacidad – Consejo Nacional para la Igualdad de Discapacidades”, 2019. <https://www.consejodiscapacidades.gob.ec/estadisticas-de-discapacidad/> (consultado jul. 02, 2020).
- [4] WHO, *World report on vision*, vol. 214, núm. 14. 2019.
- [5] L. R. Menéndez Bueyes, *Medicina, enfermedad y muerte en la España tardoantigua: un acercamiento histórico a las patologías de las poblaciones de la época tardorromana e hispanovisigoda (siglos IV-VIII)*. Salamanca, SPAIN: Ediciones Universidad de Salamanca, 2013.
- [6] F. T. y S. Real academia de Medicina, “XI conferencia: ‘Innovacion Tecnologica en Oftalmologia’”, *Real Acad. Med. Fund. Tecnol. y Salud*.
- [7] M. Landín Sorí, A. Sigler Villanueva, y R. Avilés Merens, *Estrategia de intervención sanitaria para el diagnóstico y tratamiento en pacientes con glaucoma neovascular en Cuba*. La Habana, CUBA: Editorial Universitaria, 2012.
- [8] I. de M. Ocular, “El glaucoma, a través de la historia”. <https://www.imo.es/es/noticias/el-glaucoma-a-traves-de-la-historia> (consultado jul. 02, 2020).
- [9] F. C. de O. Revista Mision Milagro, “The technological development and its impact in the diagnosis of glaucoma.”, *Mision Milagro V.3 N.2*, 2009, Consultado: jul. 02, 2020. [En línea]. Disponible en: <http://www.misionmilagro.sld.cu/vol3no2/articulos/inv3210.php>.
- [10] R. Del Río, “Desarrollo tecnológico en oftalmología durante el siglo xx”, vol.

REFERENCIAS BIBLIOGRÁFICAS

- 3, pp. 14–21, 2006.
- [11] T. Today, “Gonioscopy : Past , Present , and Future”, pp. 31–34, 2010.
- [12] N. del R. Álvarez Cárdenas y C. S. Torres Ríos, “Simulación y modelamiento del ojo humano como herramienta para la prevención del glaucoma a través la medición de la presión intraocular”, 2015, [En línea]. Disponible en: <https://dspace.ups.edu.ec/handle/123456789/8963>.
- [13] F. M. A. Herrera Deysi, “Implementación de Técnicas aplicadas para la identificación y prevención del glaucoma mediante el uso de imágenes médicas y software”, 2016, [En línea]. Disponible en: <https://dspace.ups.edu.ec/bitstream/123456789/12674/1/UPS-CT006488.pdf>.
- [14] M. S. Encalada Ojeda y E. G. Gamboa Vinueza, *Desarrollo de un sistema de soporte a la detección del glaucoma a través de procesamiento digital de imágenes biomédicas del fondo de ojo y uso de software libre*. 2016.
- [15] A. Cabrera y A. Curay, *Modelamiento y simulación del ojo humano para establecer una correlación entre el aumento de la presión intraocular y el grosor central de la córnea*. 2018.
- [16] C. M. Cajamarca Buneo y M. F. Chazi Solis, “Desarrollo de una herramienta para la identificación y prevención del glaucoma a través de procesamiento digital de imágenes biomédicas mediante la relación de volumen de fibras nerviosas”, p. 87, 2018, [En línea]. Disponible en: <https://dspace.ups.edu.ec/bitstream/123456789/16368/1/UPS-CT007968.pdf>.
- [17] C. Shea, “Anatomy and Physiology of the Eye - Distance Learning Course”, 2012, [En línea]. Disponible en: <http://www.vetmed.vt.edu/education/curriculum/vm8054/eye/EYEDEMO.HTM>.
- [18] F. Franckowiak, “Fisiología Ocular”, 2007. [En línea]. Disponible en: http://sisbib.unmsm.edu.pe/bibVirtualData/libros/Medicina/cirugia/Tomo_IV/archivospdf/03fisio_ocular.pdf.
- [19] H. M. Spinelli, A. B. Lewis, y E. Elahi, “Anatomía”, *Atlas cirugía estética periocular y del párpado*, pp. 2–27, 2017, doi: 10.1016/b978-84-8174-835-2.50001-2.
- [20] I. Cánovas, M. Moreno, y S. Mangas, “Anatomía del ojo”, *Man. Urgencias Oftalmológicas.*, p. 12, 2016, [En línea]. Disponible en: <https://www.faeditorial.es/capitulos/urgencias-oftalmologicas.pdf>.
- [21] M. Azcona-Cruz, M. Ríos Lobo, y S. Amador-Jiménez, “Glaucoma: Aspectos relevantes para la detección oportuna”, *Salud y Adm.*, vol. 2, núm.

REFERENCIAS BIBLIOGRÁFICAS

- 4, pp. 23–35, 2015, [En línea]. Disponible en: http://www.unsis.edu.mx/revista/doc/vol2num4/A3_Glaucoma.pdf.
- [22] “Cinco pruebas comunes para el glaucoma | Glaucoma Research Foundation”. <https://www.glaucoma.org/es/cinco-pruebas-comunes-para-el-glaucoma.php> (consultado jul. 11, 2020).
- [23] “¿Qué es la gonioscopia? - American Academy of Ophthalmology”. <https://www.aaof.org/salud-ocular/tratamientos/que-es-la-gonioscopia> (consultado jul. 11, 2020).
- [24] A. Baldwin, N. Hjelde, y C. Goumalatsou, *Manual Oxford de especialidades médicas*. Ciudad de México, MEXICO: Editorial El Manual Moderno, 2018.
- [25] V. Sundaram, A. Barsam, L. Barker, y P. T. Khaw, *Training in Ophthalmology*. Oxford, UNITED KINGDOM: Oxford University Press, Incorporated, 2016.
- [26] C. Del Portillo, “Mi deseo para el futuro es que nadie tenga que vivir con el miedo de perder la vista”, *Cómo Entender y Vivir con Glaucoma*, p. 36, 2007, [En línea]. Disponible en: <http://envejecimiento.csic.es/documentos/documentos/grf-glaucoma-01.pdf>.
- [27] “eL Ojo Humano”, *Univ. Sevilla*, pp. 1, 12, [En línea]. Disponible en: <http://bibing.us.es/proyectos/abreproy/12018/fichero/Memoria%252F5+-+El+ojo+humano.pdf>.
- [28] C. S. Catarina, “Inteligencia artificial”, vol. 52, núm. 55, pp. 1–6, 2018.
- [29] U. Nebrija, “Introducción a la Inteligencia Artificial ¿Qué es la Inteligencia Artificial?”, 2018. https://www.nebrija.es/~cmalagon/ia/transparencias/introduccion_IA.pdf.
- [30] M. A. ALFONSO GALIPIENSO, MARIA ISABEL, CAZORLA QUEVEDO, MIGUEL ANGEL, COLOMINA PARDO, OTTO, ESCOLANO RUIZ, FRANCISCO, LOZANO ORTEGA, *Inteligencia artificial: modelos, técnicas y áreas de aplicación*. 2003.
- [31] D. Kopec, C. Pileggi, D. Ungar, y S. Shetty, *Artificial Intelligence and Problem Solving*. Bloomfield, UNITED STATES: Mercury Learning & Information, 2016.
- [32] J. Eckroth, *Python Artificial Intelligence Projects for Beginners : Get up and Running with Artificial Intelligence Using 8 Smart and Exciting AI Applications*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2018.
- [33] O. Sosnovshchenko y O. Baiev, *Machine Learning with Swift : Artificial Intelligence for iOS*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2018.

REFERENCIAS BIBLIOGRÁFICAS

- [34] Jesús, “¿Qué es un Optimizador y Para Qué Se Usa en Deep Learning?”, *Datasmarts*, 2020. <https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/#>.
- [35] M. S. Martínez, D. D. E. Personas, y C. Sampedro, “Deteccion de Personas Mediante Técnicas De Aprendizaje Automático : Svm Y Cnn”, 2018.
- [36] H. Mujtaba, “What is Resnet or Residual Network | How Resnet Helps?”, *Great Learning*, sep. 28, 2020. <https://www.mygreatlearning.com/blog/resnet/#sh31> (consultado jul. 11, 2021).
- [37] A. Richmond, “Deep Learning: Understanding The Inception Module | by Richmond Alake | Towards Data Science”, *Towards data science*, dic. 22, 2020. <https://towardsdatascience.com/deep-learning-understand-the-inception-module-56146866e652> (consultado jul. 11, 2021).
- [38] M. Fabien, “XCception Model and Depthwise Separable Convolutions -”, *Mael Fabien*, mar. 20, 2019. <https://maelfabien.github.io/deeplearning/xception/#> (consultado jul. 11, 2021).
- [39] M. Hassan, “VGG16 - Convolutional Network for Classification and Detection”, *Neurohive*, nov. 20, 2018. <https://neurohive.io/en/popular-networks/vgg16/> (consultado jul. 11, 2021).
- [40] R. Thakur, “Step by step VGG16 implementation in Keras for beginners | by Rohit Thakur | Towards Data Science”, *Towards data science*, ago. 06, 2019. <https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c> (consultado jul. 11, 2021).
- [41] A. Y. Díaz Pinto, “Machine learning for glaucoma assessment using fundus images”, núm. June, p. 1, 2019, [En línea]. Disponible en: <http://hdl.handle.net/10251/124351>.
- [42] A. Diaz-Pinto, S. Morales, V. Naranjo, T. Köhler, J. M. Mossi, y A. Navea, “CNNs for automatic glaucoma assessment using fundus images: An extensive validation”, *Biomed. Eng. Online*, vol. 18, núm. 1, pp. 1–19, 2019, doi: 10.1186/s12938-019-0649-y.