Facultad de Ingeniería

Ingeniería de Sistemas e Informática

Tesis:

“Arquitectura de Red Neuronal Convolucional para Diagnóstico de Cáncer de Piel”

Giorzinio Maikol Tejada Layme

Renzo Pascual Gonzales Chama

Para optar el Título Profesional de:
Ingeniero de Sistemas e Informática

Asesor:
Mg. Wilder Nina Choquehuayta

Arequipa – Perú
2020

AGRADECIMIENTOS

El agradecimiento de esta tesis, lo dedicamos a todas las personas que siempre nos han apoyado a culminar y siempre estuvieron en el proceso. A nuestros padres, por su amor, trabajo y sacrificio en todos estos años, gracias a ustedes hemos llegado hasta aquí y convertirnos en lo que somos, cumplir con un sueño más y gracias por siempre inculcar en nosotros el ejemplo de esfuerzo y valentía.

Finalmente dedicar esta tesis a mis abuelos, ya que son una inspiración y un apoyo de lo que más se necesita, agradecer por siempre estar en momentos difíciles y por el amor brindado.

RESUMEN

El aprendizaje automático ha sido la técnica más usada últimamente en diferentes aplicativos, el *Deep Learning* que se encuentra dentro del aprendizaje automático, es uno de los más aplicados para el análisis de imágenes médicas, facilitando el diagnóstico de enfermedades en los pacientes y así tomar mejores decisiones acertadas sobre su salud.

El presente trabajo describe el problema para la detección de cáncer de piel a partir de imágenes ya clasificadas en melanoma maligno y benigno utilizando un modelo de aprendizaje profundo. La solución a este problema, se evaluó diferentes redes neuronales convolucionales, las cuales puedan obtener una mejor exactitud de la imagen tomada. El modelo establecido para el problema está basado en una clasificación binaria utilizando los valores 1 en caso de maligno y 0 para benigno, así se podrá detectar de forma temprana el melanoma siendo de gran utilidad. La solución propuesta es una nueva arquitectura para el entrenamiento y validación de las imágenes.

El proyecto finalmente realiza una comparativa de los resultados que se han realizado en otra investigación, donde las métricas de nuestro proyecto mejoran considerablemente al tener 3 capas. Estos resultados se evaluaron utilizando repositorios de imágenes, que están validados por centros especiales de salud de cáncer de piel.

ABSTRACT

Machine learning has been the most used technique in different applications, deep learning is in machine learning, is the most used technique for analysis of medical images, facilitate the diagnosis of diseases in patients and thus make better decisions Decisions successful on your health.

The present work describes the problem for the detection of skin cancer and the images classified in malignant and benign melanoma in a deep learning model. The solution to this problem, different convolutional neuronal networks were evaluated, so that they to be able to best accuracy of the image taken. The model established for the problem is based on a binary classification using the values 1 in case of malignancy and 0 for more information. The solution is a new architecture for the training and validation of images.

The project finally makes a comparison of the results that have been made in another investigation, where the metrics of our project improve the results of having 3 layers. These results are evaluated using image repositories, which are validated by special skin cancer health centers.

ÍNDICE

ÍNDICE DE FIGURAS/ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ABREVIATURAS

- Convolución (CONV)
- Digito Binario (BIT)
- Gigabyte (GB)
- International Skin Imaging Collaboration (ISIC)
- International Symposium on Biomedical Imaging (ISBI)
- Irradiación (IR)
- K-Nearest Neighbor (K-NN)
- Python Package Index (PYPI)
- Reconocimiento Visual a Gran Escala de ImageNet (ILSVRC)
- Rectified Linear Units (RELU)
- Red Green Blue (RGB)
- Red Neuronal Artificial (RNA)
- Redes Neuronales Artificiales (RNAs)
- Redes Neuronales Convolucionales (CNN)
- Redes Residulaes Totalmente Convolucionales (FCRN)
- Support Vector Machines (SVM)
- Unidad de Procesamiento Grafico (GPU)
- Unidad Central de Procesamiento (CPU)
- Unidad de Cálculo del Índice de Lesiones (LICU)

INTRODUCCIÓN

La piel está considerada la más importante dentro del cuerpo humano ya que protege las partes internas del mundo exterior. La dermatología en la medicina se encarga del diagnóstico de enfermedades de la piel, el cabello y las uñas. El cáncer de piel es la más importante enfermedad que puede estar creciendo en cualquier parte del cuerpo y ocurre a partir de células no pigmentadas. El cáncer de piel es el más peligroso en el mundo, existen dos tipos de estos tumores maligno y benigno, los cuales han aumentado las tasas de melanoma maligno durante al menos 30 años [1].

La piel es el órgano más grande en el cuerpo humano. Su composición consistente está dividida en tres capas: la epidermis, la dermis y el tejido subcutáneo. La epidermis constituye dos partes de células, los melanocitos y los queratinocitos, que producen melanina, un pigmento que protege contra los daños efectos de la luz solar. Algunas veces los melanocitos pueden crear acumulaciones más oscuras, que se conocen como nevi (o nevus). Nevi aparece en la superficie de la piel y en algunas ocasiones puede transformarse en melanoma [2]. Algunos de estas causas son:

- Edad. Las estadísticas muestran que a medida que la edad aumenta, el peligro para alguien desarrollar melanoma es más grande.

- Enfermedades anteriores. Si alguien en el pasado enfermó de melanoma, hay altas probabilidades de volver a enfermarse ya que esto está sucediendo con todos los tipos de cáncer.
- Lunares. A veces, en la superficie de la piel humana hay demasiados lunares de un tipo diferente.
- Fuertes antecedentes familiares de melanoma. Como sucedió en todos los tipos de cáncer, si alguien tiene antecedentes de enfermedad para su familia de 2 o más parientes en primer grado afectada, las probabilidades sentirse afectado son altos.
- Piel blanca que se quema fácilmente. Cuando el color de la piel es más claro, la radiación ultravioleta es más dañina.

CAPÍTULO 1

GENERALIDADES

1.1 Planteamiento del problema

Durante varios años y décadas, el cáncer de piel se ha ido presentando con mucha más ocurrencia en Australia que en los EE.UU., Inglaterra y Canadá, con más de 10000 casos de diagnóstico y una mortalidad anual de 1250 personas [3]. En el Perú se encuentra también varios casos y no son pocos ya que, cada año se detectan en el país 1200 pacientes nuevos con cáncer de piel; además cada año 500 personas mueren a consecuencias de este melanoma [4].

El aumento persistente de este cáncer en todo el mundo, el alto costo médico y la tasa de mortalidad han priorizado el diagnóstico precoz de este cáncer. La anticipación y la cura del melanoma son estrictamente relevantes, si se detecta temprano, la tasa de supervivencia se incrementará [1].

Hay dos razones que hacen que el diagnóstico precoz de este cáncer sea muy importante, en primer lugar, el diagnóstico de cáncer de piel es muy simple debido a

su localización en la piel y en segundo lugar la anticipación del melanoma es estrictamente relevante para su grosor. En etapas tempranas en las que el grosor es inferior a 1 mm, el melanoma se puede curar con éxito. Sin embargo, no es muy obvio para detectar el melanoma en las primeras etapas, incluso por dermatólogos con experiencia [5].

Se han realizado muchos esfuerzos para desarrollar la detección de cáncer de piel. Incluye diferentes algoritmos de detección como el análisis de patrones, el método de Menzies, el algoritmo CASH y la regla de dermastoscopias, la lista de comprobación de 7 puntos y muchas más técnicas basadas en tecnologías de imágenes [5].

Hasta enero del año 2017 se ha desarrollado una investigación el cual se centra en detectar el cáncer de piel basado en un modelo de aprendizaje profundo [6]. Es por eso que se pretende mejorar estos valores a partir de una red neuronal convolucional realizando un nuevo modelo.

1.1.1 Pregunta principal de la investigación

Reconocer e identificar el cáncer de piel a las personas a partir de una imagen del melanoma

1.2 Objetivos

1.2.1 Objetivo general

Proponer una red neuronal *Deep Learning* para la detección de cáncer de piel.

1.2.2 Objetivos específicos

- Analizar y comparar arquitecturas para la detección de cáncer de piel en *Deep Learning*.

- Diseñar e implementar un prototipo que diagnostique en cáncer maligno o benigno automáticamente.
- Analizar y comparar los resultados para recomendar el uso de esta tecnología para la detección de cáncer de piel.

1.3 Justificación

1.3.1 Aspecto a justificar

La detección temprana, la reorganización y el tratamiento son críticos para la terapia del melanoma. Puede ayudar a salvar a más del 95% de las personas [4]. La dermatoscopia es una de las técnicas más importantes para examinar lesiones de la piel y puede capturar imágenes de alta resolución de la piel que escapan de la interrupción de los reflejos de la superficie. Los médicos especialmente entrenados usan esta imagen de alta resolución para evaluar la posibilidad del melanoma desde el principio y pueden obtener una precisión diagnostica tan alta como 80%; sin embargo, no hay suficientes dermatólogos experimentados en todo el mundo [3].

Para resolver este problema, se pretende realizar un prototipo de análisis de imágenes basado en *Deep Learning* para detectar si es cáncer maligno o benigno utilizando bases de datos de imágenes en dermatoscopia en la comunidad de investigación académica.

1.4 Alcances y limitaciones

1.4.1 Alcances de la investigación

El alcance de la investigación desarrollada es demostrar que se puede realizar un diagnóstico de cáncer de piel mediante *DeepLearning* de una manera confiable y efectiva, usando una nueva arquitectura propuesta por la tesis, para para lograr dicha demostración se realizaron pruebas empíricas sobre bases de datos de imágenes de cáncer de piel.

1.4.2 Limitaciones de la Investigación

A pesar de la utilidad de las redes convolucionales, existe un potencial riesgo para la arquitectura de la red neuronal, debido a la cantidad de imágenes que se puedan obtener. El análisis de estas imágenes va permitir tener un mayor amplio entrenamiento y validación para clasificar en maligno o benigno.

Por otra parte, no se cuenta con una infraestructura computacional para que se pueda realizar múltiples pruebas instantáneas, ya que el proceso de entrenamiento demora entre 8 a 12 días, ya que se realizara un proceso de *data augmentation* sobre las imágenes obtenidas, para tener mejores resultados.

1.4.3 Delimitaciones de la Investigación

El estudio de este proyecto, abarca repositorios de imágenes dermatoscópicas de internet, los cuales deben ser validados por centros de salud especializados en la detección de cáncer de piel, en la presente tesis se utilizó la base de datos de ISIC Archive [7] y PH2 Dataset [8].

Los tipos de redes neuronales convolucionales, a comparar, serán aquellas probadas en otras tesis o revistas indexadas referido al tema de investigación,

menores a 5 años del 2019 y a la actualidad. También especificar que, el proceso de detección de cáncer de piel, se basa en solo conocer si es maligno o benigno.

CAPÍTULO 2

FUNDAMENTACIÓN TEÓRICA

2.1 Revisión histórica

La piel como habíamos señalado anteriormente en el capítulo 1; es el órgano más extenso del cuerpo humano y nuestra protección para los diferentes órganos, músculos y huesos. Es, además, una superficie que nos protege contra micro organismos, u regulador térmico y un órgano sensorial increíble que permite obtener sensaciones de tacto, calor, presión o frio. La piel se caracteriza por 3 capas epidermis, dermis y tejido celular subcutáneo. La epidermis es la capa exterior de la piel, la dermis se encuentra debajo de la epidermis y la supera en grosor y el tejido celular subcutáneo se encuentra debajo de la dermis contiene fibras de colágeno y células grasas [9].

Estas células se dividen continuamente de una manera programada y controlada para formar nuevas células y reemplazar las que han ido envejeciendo y muriendo. Esto genera un equilibrio en la producción de nuevas células y la eliminación de células muertas, permitiendo que la piel funcione de forma equilibrada. Cuando se pierde este equilibrio, causado por mutaciones en los genes, ocurre un crecimiento descontrolado

y una división excesiva de un determinado tipo de célula, produciéndose cáncer. En general el cáncer de piel se divide en melanoma benigno y maligno [9].

Para el diagnostico de los tumores de la piel se inicia con el examen clínico del médico a simple vista de la lesión. Este examen se acompaña habitualmente de una exploración con dermatoscopia, técnica no invasiva que permite la exploración de los tumores y su diagnóstico con mucha precisión. Un factor que influye son características que pueden afectar posiblemente las causas de padecer alguna enfermedad, en este caso, cáncer de piel. Cada tipo de cáncer tiene diferentes factores de riesgo. Algunos factores de riesgo, como la exposición al sol, se pueden controlar. Otros como la historia personal, la edad o las características genéticas, no pueden ser modificados [10].

Acciones tendientes a prevenir el melanoma es dispensarización y observación periódica de aquellos pacientes con nevos hipercrómicos congénitos, nevos displásicos, o que estén sometidos a traumatismos repetidos y sospechar melanoma maligno ante las siguientes contingencias en un nevo hipercrómico.

a) Aumento rápido de tamaño.
b) Cambio de coloración: más oscuro o más claro.
c) Alteraciones en la superficie o forma.
d) Irregularidad del borde.
e) Signos de irritación o halo inflamatorio.
f) Sangrado espontaneo.
g) Ulceración.
h) Brote de macula pigmentarias y/o nódulos.
i) Prurito o dolor.

j) Adenopatías satélites.

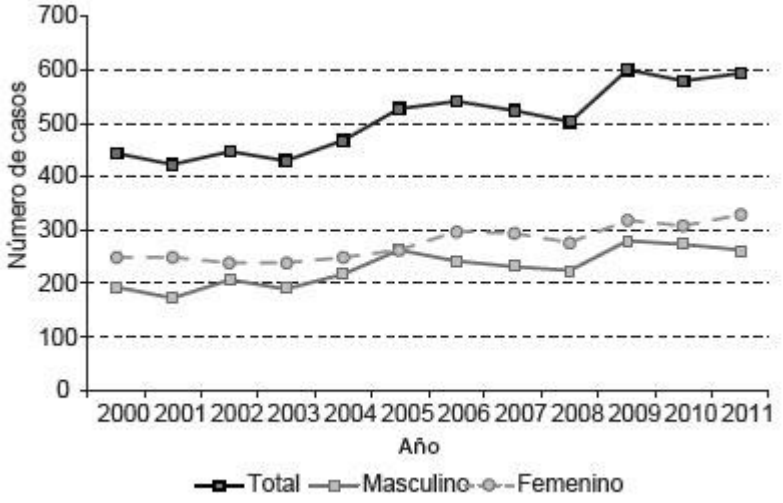
2.2 Epidemiologia del cáncer de piel en el Perú

En el caso para Perú no se cuenta con datos sobre la incidencia de las masas anormales de tejidos o también conocidas como neoplasias de piel. La Dirección General de Epidemiología (DGE) ha realizado un análisis con respecto a la población con respecto a la situación de cáncer a nivel nacional, es por eso que la vigilancia epidemiológica de cáncer, ha encontrado que en el periodo comprendido entre los años 2006 y 2010 se registró un total de 5975 casos de cáncer de piel los cuales 2744 fueron varones y 3231 fueron mujeres, para la DGE representan el 6,6% del total de cánceres registrados durante estos periodos. El informe emitido, sobre el cáncer de piel ocupa el cuarto lugar de frecuencia a nivel nacional (superado por el cáncer de cérvix, estómago y mama). Las regiones con mayor distribución de cáncer de piel respecto al total de neoplasias registradas son superiores al promedio nacional en un 6,6%; de la cuales en La Libertad (10,7%), Cajamarca (9,5%), Madre de Dios (9,2%), San Martín (8,0%), Amazonas (7,9%), Lima (7,9%), Arequipa (7,8%), Ayacucho (7,3%) y Ucayali (7,1%). Según las medidas de frecuencia de la enfermedad en la población, sí es importante mencionar la importancia del cáncer de piel la detección temprana. En el año 2011 se registró un total de 1208 fallecidos debido al cáncer maligno en la piel de los cuales fueron 725 hombres y 483 mujeres, tasa de 2,1 por ciento por cada 100 000 hombres y 1,6 por ciento por cada 100 000 mujeres [4].

Es fundamental conocer también la aproximación al conocimiento de la situación de cáncer de piel en el país, según las estadísticas registradas en el Departamento de Epidemiología y Estadística del Cáncer del Instituto Nacional de Enfermedades

Neoplásicas (INEN), indican que en los periodos 2000 y 2011, se han atendido en el INEN 6048 casos de cáncer de piel, lo cual representa el 5,8% del total de cáncer de piel atendidas. Al pasar el tiempo la evolución se ha incrementado de 439 casos al año 2000 de los cuales 191 hombres y 248 mujeres; a 592 casos el año 2011 de los cuales 262 hombres y 330 mujeres (Figura 1). Esta información, no es una incidencia, son un indicador de que el número de casos de cáncer de piel se ha incrementado a nivel poblacional. Se observa que a mayor edad es mayor el número de casos atendidos; así, el 1,5% de los casos de cáncer de piel atendidos en el INEN entre los periodos 2000 y 2011 son pacientes menores de 20 años, el 7,7% tuvieron entre 20 y 39 años, el 26,8% entre 40 y 59 años y el 63,8% fueron personas de 60 años y más [4].

Figura 1. Casos de neoplasias cutáneas atendidas en el Instituto Nacional de Enfermedades Neoplásicas, 2000-2011.



Fuente: *[4]*.

2.3 Experiencia en la educación prevención y detección temprana del cáncer de piel

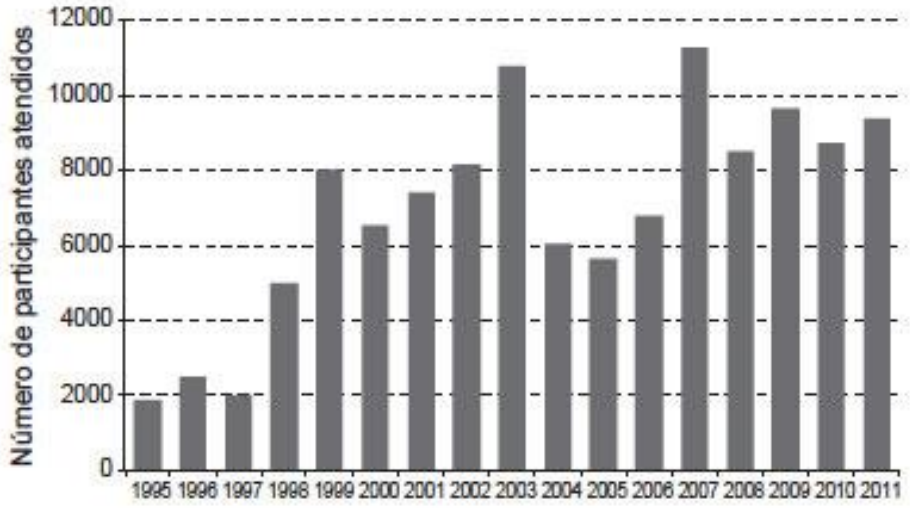
En la medicina, existen especialistas que se encargan de identificar las lesiones cutáneas benignas o malignas, el dermatólogo o especialista realiza el examen completo de la piel, para luego extirparlas o biopsiarlas, a fin de remitirlas al patólogo

para su estudio e identificación correspondiente y, si el caso lo requiere, derivar al paciente para su tratamiento oportuno [11].

Para los dermatólogos existe una regla practica la cual es de forma visual, esto sirve para identificar aquellas lesiones sospechosas, esta regla es conocida como el ABCDE de los lunares; A de asimetría, B de bordes, C de color, D de diámetro y E de elevación o evolución y, posteriormente, esto sirve para prevenir cualquier tipo de cáncer de piel a futuro. Las primeras pruebas se realizaron en USA y luego le siguieron las experiencias en américa latina en los países de Chile, Argentina y Perú [11].

Para el año 1995 el Perú mediante El Círculo Dermatológico del Perú (CIDERM) realiza la primera Campaña de Educación, Prevención y Detección Temprana de Cáncer de Piel, no es solo detectar tempranamente las lesiones sospechosas de cáncer de piel sino educar a la población sobre los peligros que hay por esta enfermedad, y los efectos negativos derivados de la sobreexposición al sol, creando conciencia y fomentando una cultura de prevención [11].

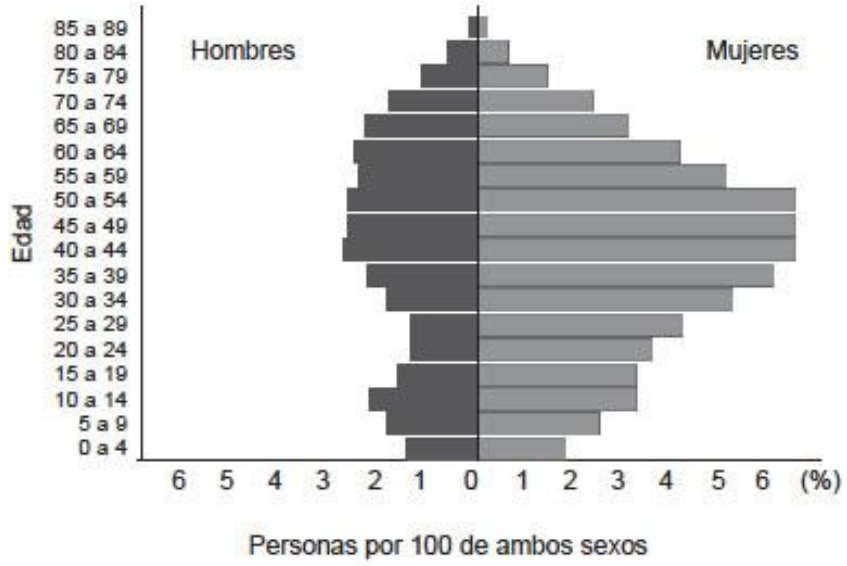
Figura 2. Atendidos en las campañas "El Día de Lunar", 1995-2011.



Fuente: *[11].*

CIDERM registra entre 1995 y 2011, 118 092 personas atendidas, elevando en el año 1877 atendidos al año 1995 en doce sedes ubicadas solamente en Lima, a 9355 atendidos el año 2011 en 18 ciudades de todo el país (Figura 2). En la Figura 3 existe una mayoría de participantes mujeres (66,0%). Cuyo promedio de edad ha sido 41,4 ± 20,4 años, con un rango que va desde participantes menores de un año hasta personas mayores de 90. En el caso de los varones su promedio es entre 42,2 ± 22,5 años, mientras que las mujeres 41,0 ± 19,2 años. La distribución según grupos de edad y sexo se observa en la Figura 3.

Figura 3. Distribución de los participantes en las campañas, según edad y sexo.



Fuente: *[11]*.

De las casi 120 000 personas atendidas, en el 2,8% se identificó alguna lesión cutánea sospechosa de cáncer de piel, y entre estas lesiones la distribución porcentual fue carcinoma basoceluar 64,9%, melanoma cutáneo 26,7% y carcinoma espinocelular 8,4%.

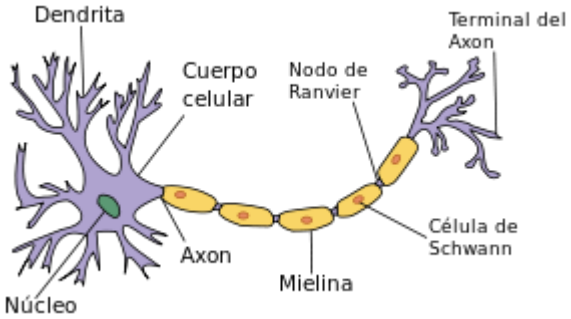
2.4 Red neuronal artificial

Una parte importante y destacada del campo científico de la Inteligencia Artificial es la que corresponde a las Redes Neuronales Artificiales (RNA) las cuales son aquellas redes en las que existen elementos de procesamiento de información, que realizan interacciones locales y estás dependen en el comportamiento del conjunto del sistema [6].

Las RNA intentan simular la conducta del cerebro humano, el cual se caracteriza por el aprendizaje a través de la experiencia y la extracción de conocimiento genérico a partir de un conjunto de datos. Estos sistemas simulan de forma esquemátizada la estructura neuronal del cerebro, a través de un programa de ordenador, o mediante su modelado de estructuras de procesamiento con cierta capacidad de cálculo paralelo, o también mediante la construcción física de sistemas cuya arquitectura se aproxima a la estructura de la red neuronal biológica [6].

La parte fundamental en los sistemas neuronales biológicos es la neurona, una célula viva que, como tal, contiene todos los elementos que integran las células biológicas, si bien incorpora otros elementos que la diferencian. En términos generales, la neurona es el cuerpo célula o soma más o menos esféricos, e inicia a partir de una principal o axón y un denso árbol de ramificaciones más cortas, compuesto por dendritas. También, el terminal del axón puede ramificarse en su punto de arranque, y con frecuencia presenta múltiples ramas en su extremo. La forma final de la neurona depende de la función que cumple, esto es, de la posición que ocupa en el conjunto del sistema y de los estímulos que recibe como se muestra en la siguiente figura 1 [6].

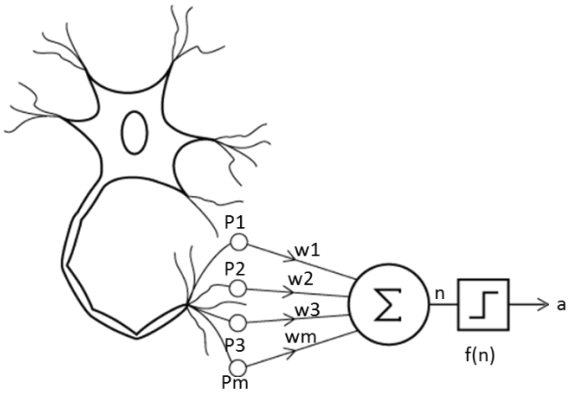
Figura 4. Estructuras típicas de las neuronas biológicas



Fuente: *[12]*.

La forma más habitual de representación de una neurona artificial está caracterizado por nodos a través de círculos, con conexiones como líneas o flechas, la cual obtiene una estrecha relación con la representación tradicional de una red neuronal artificial (RNA), recogida por la Figura 5. También es corriente la representación del grafo por la matriz de conexiones, donde será simétrica debido al grafo si no es dirigido [12].

Figura 5.Estructuras de un sistema de neuronas artificiales



Fuente: *[12]*.

2.4.1 Perceptrón simple

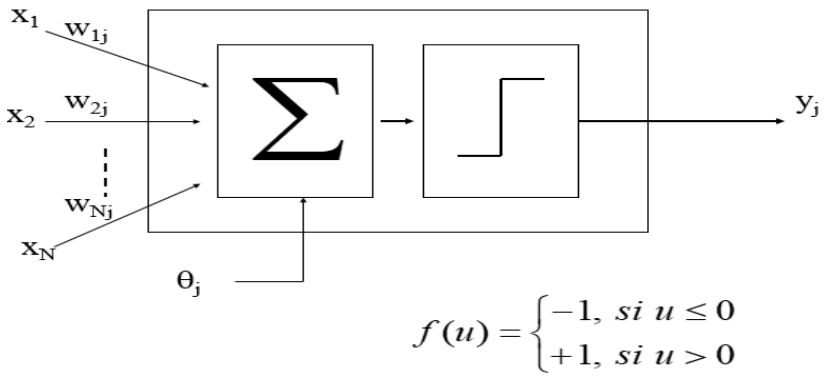
El Perceptrón Simpe fue elaborado por Frank Rosenblatt, es un pionero para los sistemas neuronales más simples, los cuales son usados como eficientes procesadores de información [13].

Dado que algunas de sus características, son extensibles a otros sistemas neuronales más complicados. El Perceptrón es usado para resolver problemas linealmente separables, utiliza el modelo de McCullot-Pitts como neurona [13].

Las características del Perceptrón son las siguientes:

- Aprendizaje supervisado.
- Aprendizaje por corrección de error.
- Reconocimiento de patrones sencillos.
- Clasificación de patrones linealmente separables.

Figura 6. Características del Perceptrón simple



Fuente: [13].

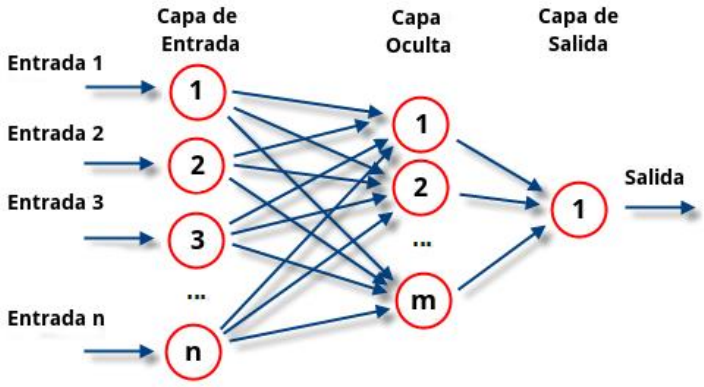
2.4.2 Perceptrón multicapa

Este modelo de RNA más utilizado en la práctica, como para resolver problemas de clasificación y de regresión, al haber demostrado su condición de aproximador universal de funciones, lo que justifica su estudio individual y detallado [12].

Perceptrón Multicapa es un modelo neuronal con propagación hacia adelante, basado en su organización de capas de celdas disjuntas, esta forma realiza que ninguna salida neuronal constituya una entrada en las neuronas de la misma capa o de capas previas, evitándose así las conexiones hacia atrás o auto recurrentes [12].

De esta forma, esta arquitectura presente viene a coincidir con la del Perceptrón Simple, la diferencia es que esta incluye una o varias capas ocultas para una mayor complejidad [12].

Figura 7. Arquitectura del perceptrón multicapa



Fuente: *[12].*

Se caracteriza por presentar el siguiente proceso de aprendizaje:

- Proceso de aprendizaje mediante la minimización del error cuadrático por el método del gradiente desentiende.

$$\Delta W_{ij} = \frac{n\,\partial E}{\partial W_{ij}}$$

- Cada peso sináptico i de la neurona j es actualizado proporcionalmente al negativo de la derivada parcial del error de esta neurona con relación al peso.

- Donde el error cuadrático de la neurona j es definido por:

$$\mathrm{E_j} = \frac{1}{2}(\mathrm{d_j} - \mathrm{y_j})^2$$

- Donde d es el valor deseado de la salida para la respectiva neurona; y es la salida de la neurona j.
- Se debe minimizar cada error para todas las neuronas dentro la capa de la salida para todos los patrones.

$$\mathrm{E_{SSE}} \rightarrow \mathrm{SumSquaredErrors}$$

$$\mathrm{E_{SSE}} = \frac{1}{2}\sum \mathrm{p} \sum \mathrm{j} (\mathrm{d_{pj}} - \mathrm{y_{pj}})^2$$

- $\mathrm{d_{pj}}$ es el valor deseado de salida para el patrón p para la neurona j.
- $\mathrm{y_{pj}}$ es salida de la neurona j cuando es presentado el patrón p.

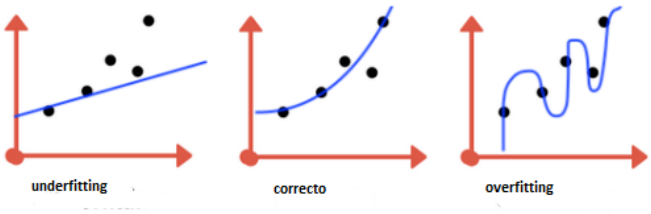
  El algoritmo del Perceptrón Multicapa inicializa con los pesos con valores aleatorios $W_{ij}| < 0{,}1$. El entrenamiento es repetir hasta que el error de cada neurona de salida esta deseada para todos los patrones de un conjunto de entrenamiento [14].
- Presentar un patrón de entrada x_i y su respectivo valor de salida deseada.
- Calcule las salidas de las neuronas comenzando en la primera capa escondida hasta la capa de salida.
- Calcule el error para cada neurona de cada salida. Si el error es menor que la tolerancia para todas las neuronas, retorne para 1.
- Actualiza los pesos de cada procesador comenzando por cada salida hasta la capa de entrada.

2.5 Machine learning

El aprendizaje automático es un algoritmo que funciona con nuevas entradas, normalmente cuando entrenamos un modelo de aprendizaje automático, tenemos acceso a un conjunto de entrenamiento y podemos calcular alguna medida de error en el conjunto de entrenamiento, llamado training error para luego reducir este error de entrenamiento. Lo que separa al aprendizaje automático de la optimización es que queremos que el error de generalización, también llamado error de prueba, sea bajo. El error de generalización se define como el valor esperado del error en una nueva entrada. Aquí la expectativa se toma a través de diferentes entradas posibles, extraídas de la distribución de entradas que esperamos que el sistema encuentre en la práctica [15].

Los algoritmos de aprendizaje automático generalmente se desempeñan mejor cuando su capacidad es apropiada para la verdadera complejidad de la tarea que necesitan realizar y la cantidad de datos de entrenamiento que reciben. Los modelos con capacidad insuficiente no pueden resolver tareas complejas. Los modelos con alta capacidad pueden resolver tareas complejas, pero cuando su capacidad es mayor que la necesaria para resolver la tarea actual, es posible que se encuentren en exceso [15].

Figura 8. Ejemplo de predictor lineal

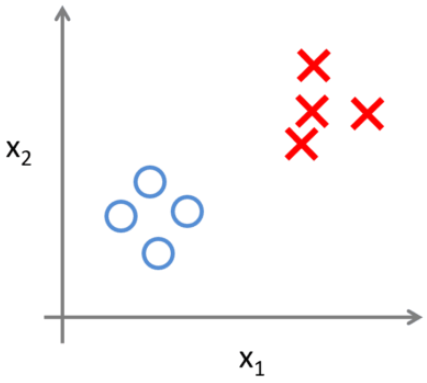


Fuente: [15].

2.5.1 Aprendizaje supervisado

Básicamente este algoritmo sirve para clasificar elementos a partir de una serie de variables. Este tipo de algoritmo es útil cuando se tiene una propiedad conocida en un elemento concreto. Estos algoritmos nos ayudaran a predecir cuál es esa propiedad que se desconoce [16].

Figura 9. Aprendizaje supervisado

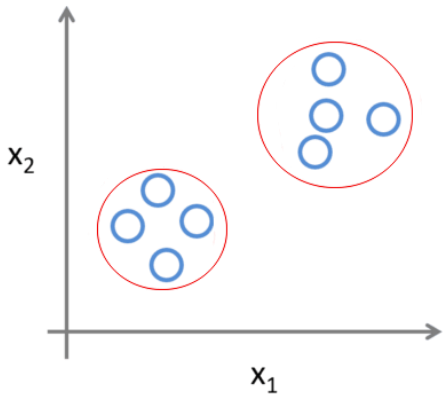


Fuente: *[16].*

2.5.2 Aprendizaje no supervisado

Este tipo de algoritmo es útil para encontrar las relaciones que existen en un conjunto de datos pero que no son conocidas. Es decir, permite considerar que varios elementos pertenecen a un mismo grupo o a diferentes grupos al estudio de sus características [17].

Figura 10. Aprendizaje no supervisado

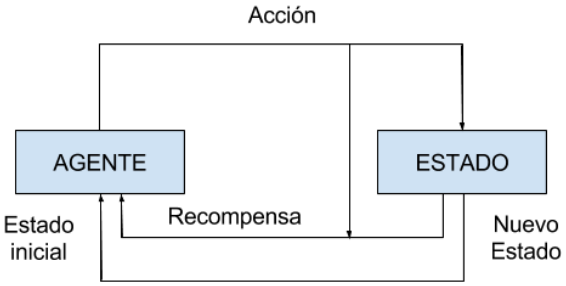


Fuente: *[17].*

2.5.3 Aprendizaje por refuerzo

Los algoritmos por refuerzo son una mezcla entre el aprendizaje supervisado y no supervisado. Este modelo toma la forma del agente sintiendo el ambiente, basado en la entrada de sensores, escoge una acción a ser desempeñada [18].

Figura 11. Aprendizaje por refuerzo



Fuente: *[18].*

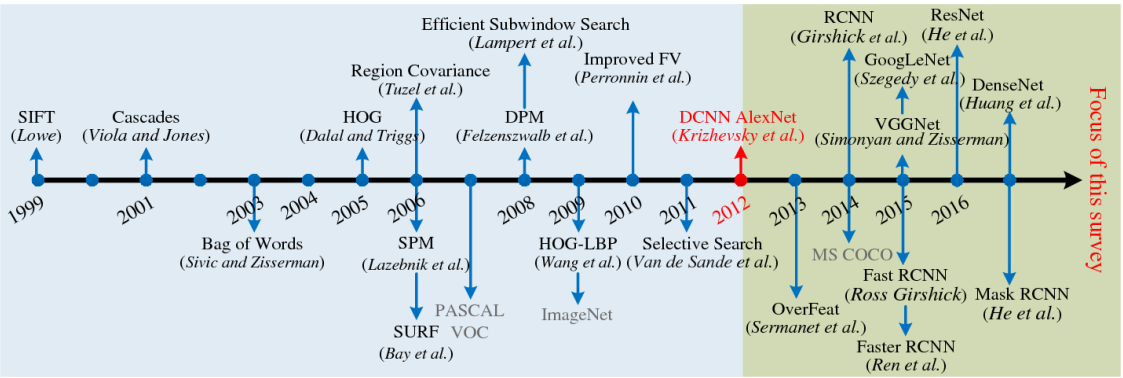
2.6 Deep learning

Es llamado como aprendizaje profundo ya que esta nueva técnica es conocida para aplicar el aprendizaje automático, ya que se baja en el aprendizaje de representaciones de datos, en eso hace uso del Patrón Multicapa como un ejemplo de

entrada como imagen, pueden ser de muchas formas diferentes, como un vector de valores de fuerza por pixel, o en una forma más abstracta como un conjunto de áreas. Muchas de las representaciones son mejores que otras para implicar la tarea de aprendizaje [15].

Una de las garantías de Deep Learning es que las características de algo diferente, serán reemplazados con la función no supervisada o semi supervisada. La exploración de este tema, avanza para hacer mejores representaciones y modelos de construcción de datos sin etiqueta a gran escala. Algunas de las representaciones están inspiradas en avances en la neurociencia y se basan libremente en la interpretación de en formación de procesamiento y patrones de comunicación en un sistema nervioso, tales como codificación neuronal, que intenta definir la relación entre varios estímulos y respuestas neuronales asociadas en el cerebro [19].

Figura 12. Detección y clasificación de imágenes



Fuente: *[20].*

2.6.1 Red neuronal convolucional

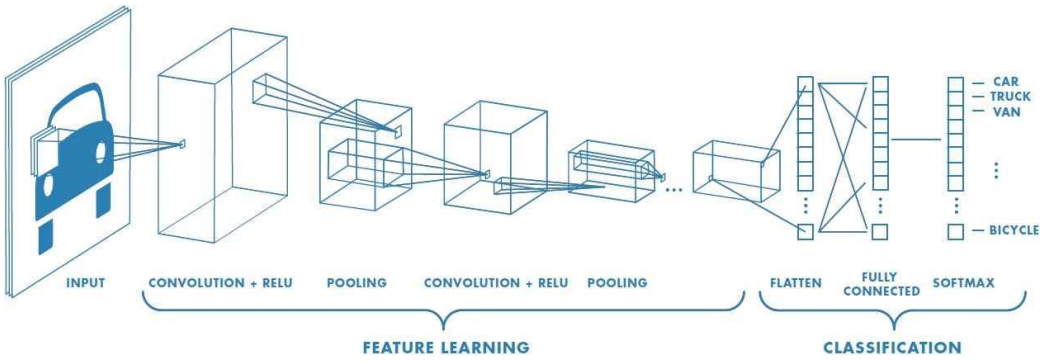
Una red neuronal convolucional (*Convolutional Neural Networks*) es un análogo de la inteligencia artificial tradicional porque están formadas por neuronas que se auto optimizan a través del aprendizaje. Cada neurona va recibir una entrada

y esta realizara una operación escalar producto seguido a una función no lineal, bajo la base de innumerables de la inteligencia artificial. Desde los vectores de imagen en bruto de entrada para la salida final en la puntuación para la clase, la totalidad de la red seguirá expresando una sola función de puntuación preceptiva al peso para que puedan obtener la mejor precisión de la imagen tomada [21].

Una de las mayores limitaciones de las formas tradicionales de RNA es que tienden a lucha con la complejidad computacional requerida para calcular los datos de la imagen. Conjuntos de datos de evaluación comparativa de aprendizaje automático comunes, como la base de datos MNIST de dígitos escritos a mano son adecuados para la mayoría de las formas de RNA, debido a su relativamente dimensionalidad de imagen pequeña de solo 28 × 28. Con este conjunto de datos, una sola neurona en la primera capa oculta contendrá 784 pesos (28 × 28 × 1 donde 1 tiene en cuenta que MNIST está normalizado a solo valores en blanco y negro), lo cual es manejable para la mayoría de las formas de RNA [21].

En realidad, podemos verlo como un clasificador equivalente al presentado en el capítulo 2, que predice una probabilidad de que la imagen de entrada sea un auto [22].

Figura 13. Convolución en cada pixel

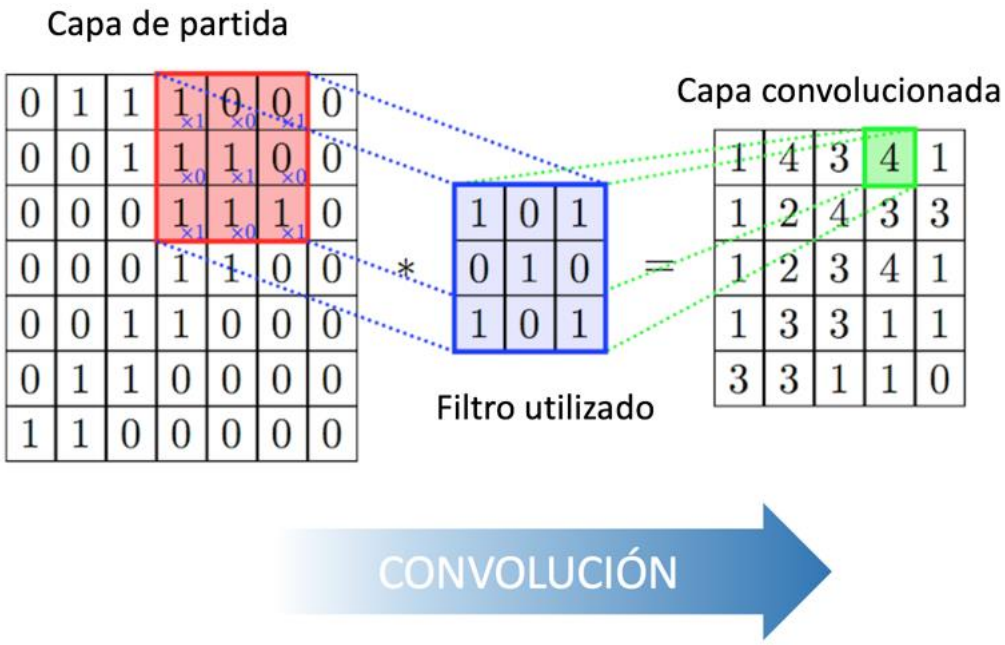


Fuente: *[23].*

Ahora que tenemos una visión intuitiva de cómo clasifican las redes neuronales convolucionales una imagen, vamos a presentar un ejemplo de reconocimiento de dígitos y a partir de él introduciremos las dos capas que definen a las redes convolucionales que pueden expresarse como grupos de neuronas especializadas en dos operaciones: convolución y pooling [22].

Operación de convolución. La convolución determinara la salida de neuronas de las cuales son conectado a regiones locales de la entrada a través del cálculo del escalar productor entre sus pesos y la región conectada al volumen de entrada. La unidad lineal rectificada llamada ReLu tiene como objetivo aplicar una función de activación por elementos, como sigmoide, a la salida de la activación producida por la capa anterior [22].

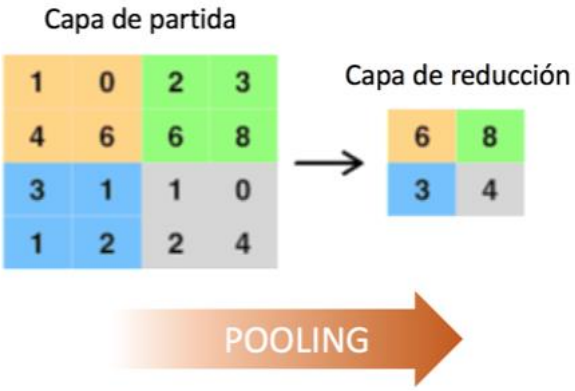
Figura 14.Ejemplo de convolución



Fuente: *[24].*

Operación pooling. La capa pooling o capa de agrupación es simplemente la que realizara un muestreo descender a lo largo de la dimensionalidad espacial de la entrada dada, reduciendo aún más el número de parámetros dentro de esa activación [21].

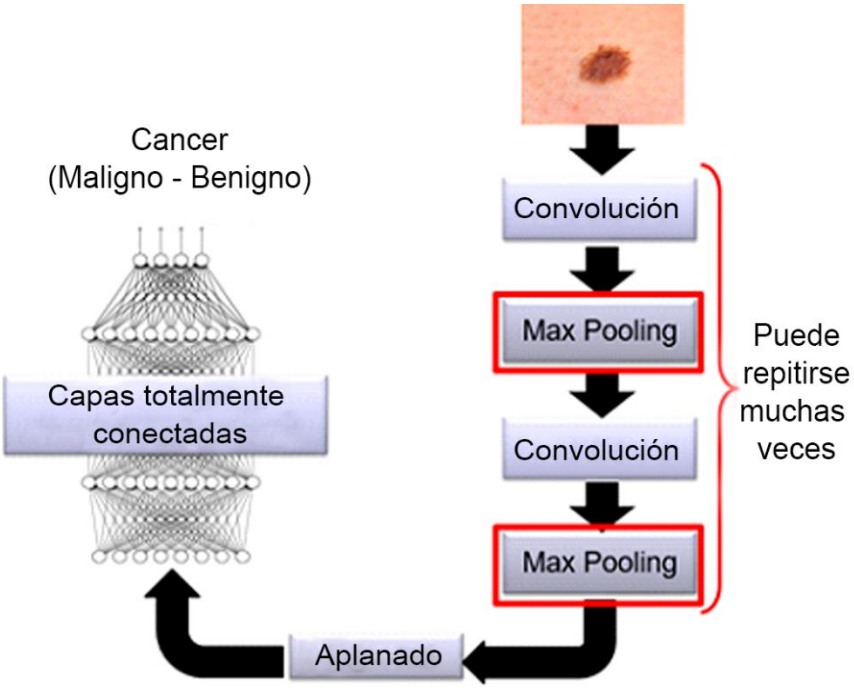
Figura 15. Ejemplo de reducción



Fuente: *[24].*

Capas totalmente conectadas. Realizaran las mismas tareas que se encuentran en las RNA y va intentar producir puntajes de clase a partir de las activaciones para ser utilizado para la clasificación. También se sugiere que ReLu pueda usarse entre estas capas, para mejorar el rendimiento [25].

Figura 16. Capas totalmente conectadas



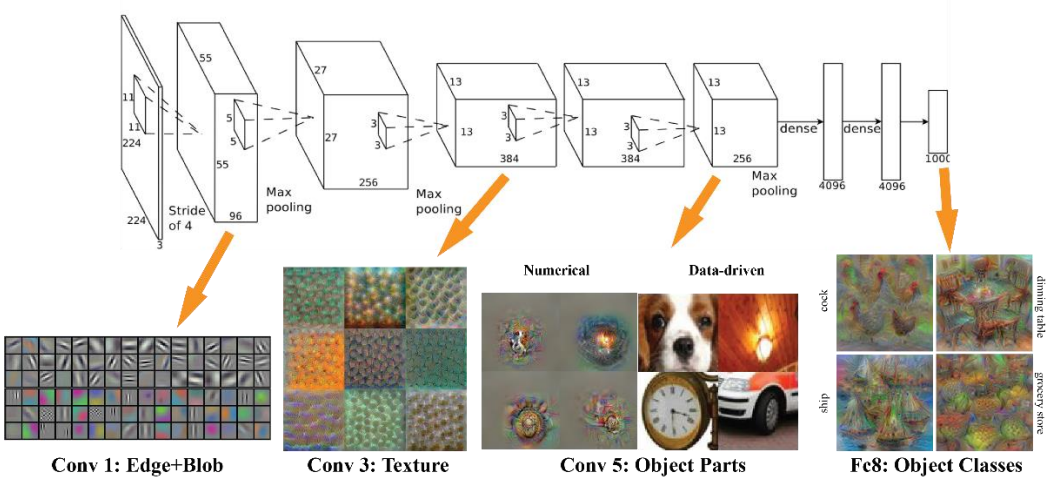
Fuente: *[19].*

El funcionamiento de una red neuronal convolucional tiene dos componentes:

Las capas ocultas: En esta parte, la red realizara una serie de convoluciones y operaciones de pooling durante las cuales se detectan las características [19].

Clasificador: Aquí, las capas totalmente conectadas servirán como un clasificador por encima de estas características extraídas [19].

Figura 17.Funcionamiento de una CNN



Fuente: *[26].*

2.6.2 Padding

El *Padding* nos permite conservar el tamaño de la matriz de entrada, para poder usar redes neuronales profundas, realmente necesitamos usar *padding*. Para darle una regla general, si a una matriz $n * n$ se le aplica un *filter* fxf, tenemos:

$$(n - f + 1)\, x\, (n - f + 1)$$

La operación de convolución reduce la matriz si $f > 1$. Si queremos aplicar la operación de convolución varias veces, pero si la imagen se reduce, perderemos muchos datos en este proceso. También los pixeles de los bordes se utilizan menos que otros pixeles en una imagen. En casi todos los casos los valores del *padding* son ceros. La regla general ahora, si una matriz nxn se le aplica un *filter* de fxf y un *padding* p, esto nos da una matriz [26]:

$$(n - 2p + f + 1)\, x\, (n - 2p - f + 1)$$

Si tenemos $n = 6, f = 3$ y $p = 1$, entonces la imagen de salida tendrá:

$$= n + 2p - f + 1$$

$$= 6 + 2 - 3 + 1$$

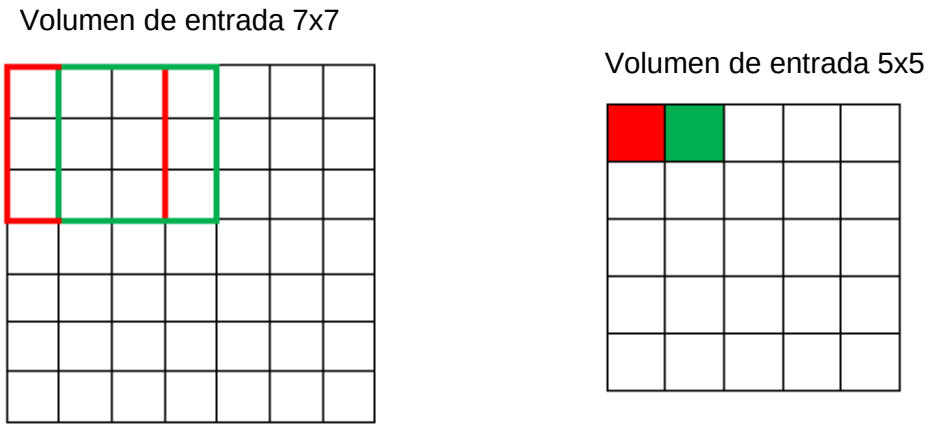
$$= 6$$

En este ejemplo vamos a mantener el tamaño de la imagen [26].

2.6.3 Stride

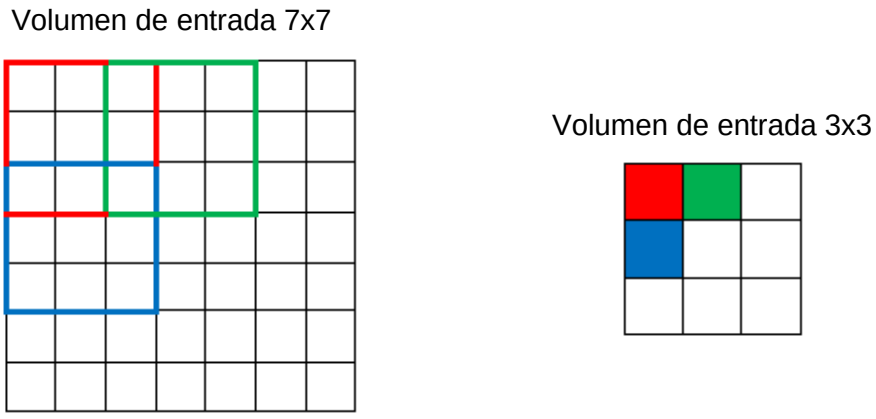
El s*tride* controla como el *filter* avanza sobre del volumen de entrada. Normalmente, los programadores aumentaran el valor del *stride* si desean que los campos receptivos se superpongan menos y si desean dimensiones espaciales más pequeñas [26].

Figura 18. Ejemplo de stride de 1



Fuente: *[26].*

Figura 19.Ejemplo de stride de 2

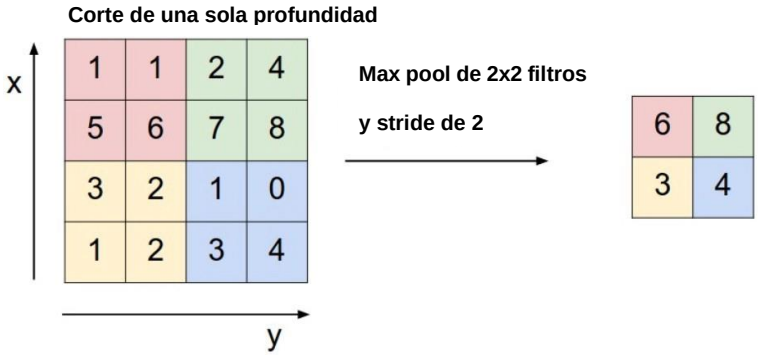


Fuente: *[26].*

Llamaremos al *stride* como s, cuando se realiza la operación de convolución, decimos que s representa la cantidad de pixeles que se recorrerá cuando se esté aplicando el *filter* y *kernel*. Entonces calculando el *stride* de una matriz nxn se le aplica un *filter* de $f * f$ y un *padding* p, y da un *stride* s, tenemos una matriz de [26]:

$$\frac{n+2p-f}{s}+1 \; x \; \frac{n+2p-f}{s}+1$$

Figura 20.Ejemplo de clasificador con stride 2



Fuente: *[19].*

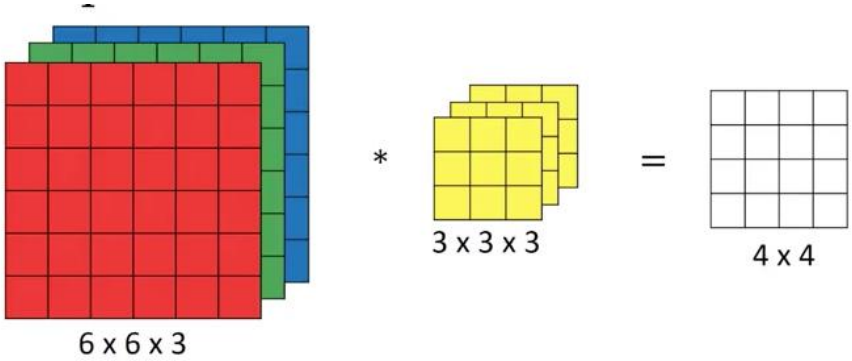
Escogiendo los híper parámetros, como sabemos cuántas capas usar, cuantas capas convolucionales, cuales son los tamaños de filtro o los valores del *Stride*

y *Padding*. No existe un estándar establecido que sea usado por todos los investigadores. Esto se debe a que la red dependerá en gran medida del tipo de datos que se tenga. Los datos pueden variar según el tamaño, la complejidad de la imagen, el tipo de tarea de procesamiento de imágenes y más [26].

2.6.4 Canales RGB

Ya hemos visto cómo funciona la convolución con imágenes en 2D, ahora analizaremos las imágenes en 3D(RGB). Aplicaremos la convolución a una imagen con altura, ancho, número de canales con un *filter* de altura, ancho, mismo número de canales. Esto nos indica que el número de canales de la imagen y el número de canales del *filter* son los mismos. Por ejemplo, si tenemos una imagen de entrada $6 * 6 * 3$, un *filter* de $3 * 3 * 3$ la imagen resultante será de $4 * 4 * 1$. En este caso el valor de $p = 0$ y $s = 1$ [26].

Figura 21.Convolución en imágenes 3D RGB

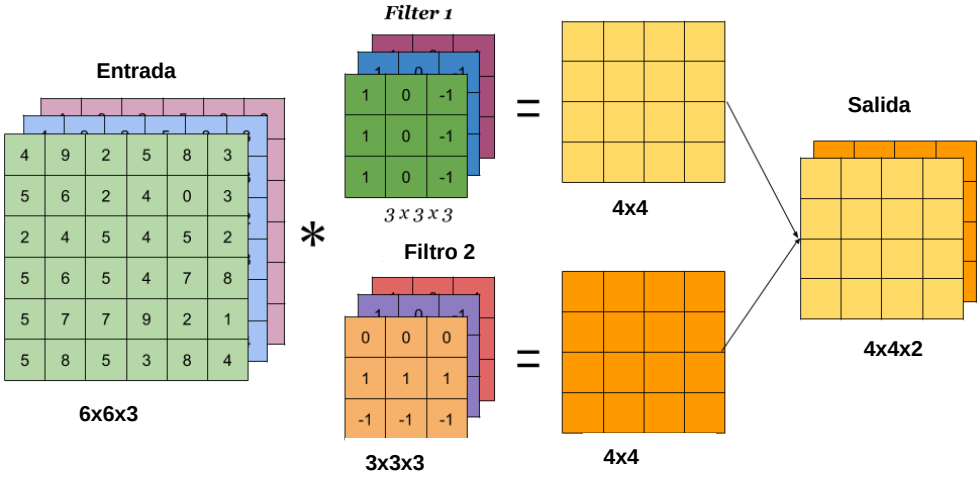


Fuente: *[26].*

Utilizando más filtros podemos detectar múltiples características o bordes. Por ejemplo, si tenemos una imagen de entrada de $6 * 6 * 3$, 10 *filter* de $3 * 3 * 3$ la imagen resultante será de $4 * 4 * 10$. En este caso el valor de $p = 0$ y $s = 1$ [26].

Figura 22. Múltiples filtros convolución de imágenes 3D RGB

Filtro 1



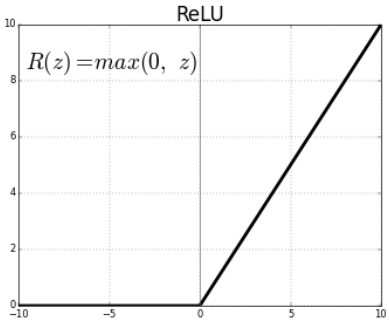
Fuente: *[26].*

2.6.5 ReLu (rectified linear units)

Después de cada capa CONV, se debe aplicar una capa no lineal o capa de activación inmediatamente después. El propósito de esta capa es introducir la no linealidad en un sistema que básicamente ha estado computando operaciones lineales durante las capas CONV, ya que estas hacían operaciones de suma y multiplicación. En el pasado, se utilizaron funciones no lineales como *tanh* y *sigmoid*, pero los investigadores descubrieron que las capas ReLu funcionan mucho mejor porque la red puede entrenar mucho más rápido debido a la eficiencia computacional sin hacer una diferencia significativa en la precisión.

La capa ReLu aplica la función $f(x) = \max(0, x)$ a todos los valores en el volumen de entrada. En términos básicos, esta capa solo cambia todas las activaciones negativas a 0, esta capa aumenta las propiedades no lineales del modelo y la red general sin afectar los campos receptivos de la capa CONV [26].

Figura 23.ReLU Layer

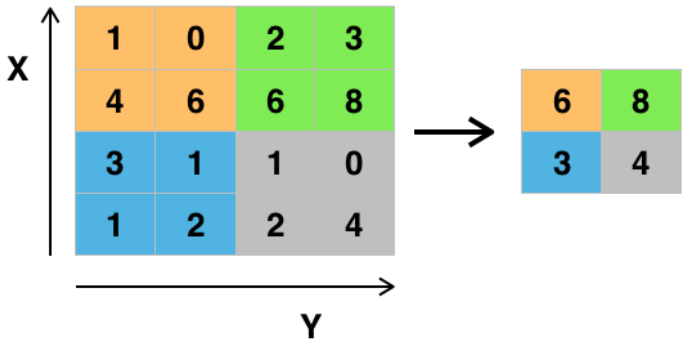


Fuente: *[26].*

2.6.6 Capa pooling

Aparte de las capas de CONV, las CNN a menudo usan capas de pooling para reducir el tamaño de las entradas, acelerar el cálculo y hacer algunas de las características que detecta sean más robustas. La capa de pooling también utiliza los parámetros f, s y p [26].

Figura 24. Capa pooling



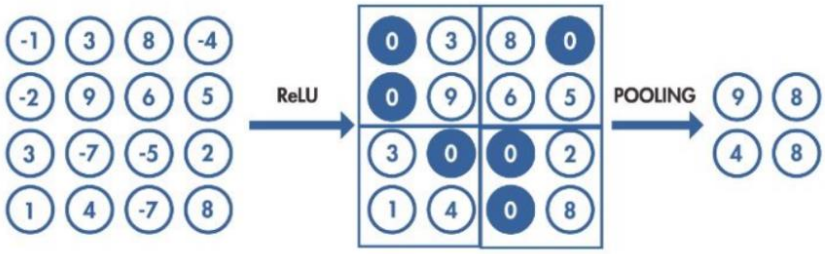
Fuente: *[26].*

2.6.7 Max pooling

El max pooling dice que, si la característica se detecta en cualquier lugar de este filtro, se mantiene un número alto. Pero la razón principal por la que las personas utilizan el pooling porque funciona bien en la práctica y reduce los cálculos. Max pooling no tiene parámetros para aprender. Después de algunas capas ReLu,

los programadores pueden optar por aplicar una pooling layer. Esto básicamente requiere un filtro normalmente de tamaño de $2 * 2$ y un *Stride* de la misma longitud. Luego lo aplica al volumen de entrada y genera el número máximo en cada subregión en la que el *filter* gira alrededor [26].

Figura 25.Max pooling después de la capa convolucional



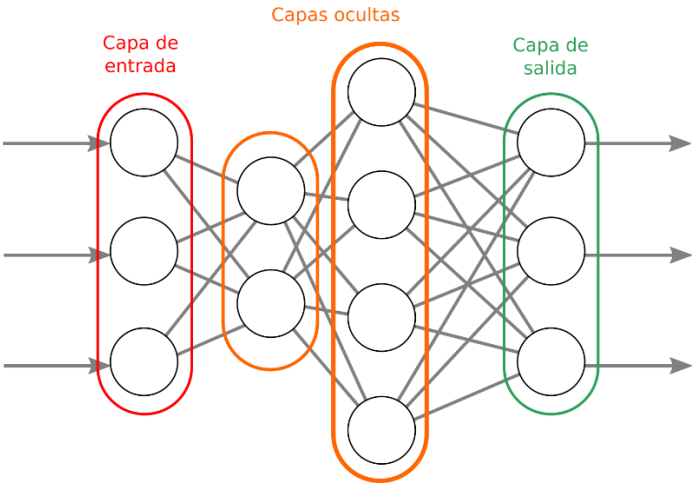
Fuente: *[26].*

2.6.8 Redes deep feedforward

Están compuestas de una o varias capas ocultas de neuronas, vinculadas a las entradas o a las capas anteriores, y de una capa de salida, vinculada a las neuronas ocultas. El usuario seleccionara el número de capas ocultas y el número de neuronas de cada capa generalmente haciendo pruebas sobre varios valores. Cuantas más capas posea la red, más compleja será, permitiendo así obtener un mejor aprendizaje más rápidamente que con una o dos capas ocultas. En efecto, cada capa puede verse como una etapa suplementaria en el algoritmo [27].

Son redes que son capaces de aprender relaciones entrada-salida a partir de una gran cantidad de ejemplos, están formadas por muchas neuronas simples interconectadas entre sí [27].

Figura 26. Ejemplo de red feed forward



Fuente: *[27].*

2.6.9 Transfer learning

El entrenamiento de toda una red neuronal convolucional en el campo de la imagen médica no siempre es posible, debido al hecho de que los conjuntos de datos a menudo no son lo suficientemente grandes. Alternativamente, la inicialización aleatoria de pesos se reemplaza por una red en grandes conjuntos de datos, es decir, un repositorio, que contiene 1.2 millones de imágenes etiquetadas con 1,000 clases [6]. Esta técnica se conoce como *Transfer Learning* y es muy común en los escenarios de aprendizaje automático. El objetivo principal es mejorar el aprendizaje en una nueva tarea mediante la transferencia de conocimientos de una tarea relacionada que ya se ha aprendido [27].

En la práctica, *Transfer Learning* es una red convolucional pre entrenada que se utiliza de dos formas diferentes [27].

- Extracción de características fijas que usa la salida de una capa intermedia para entrenar a un clasificador de aprendizaje automático.

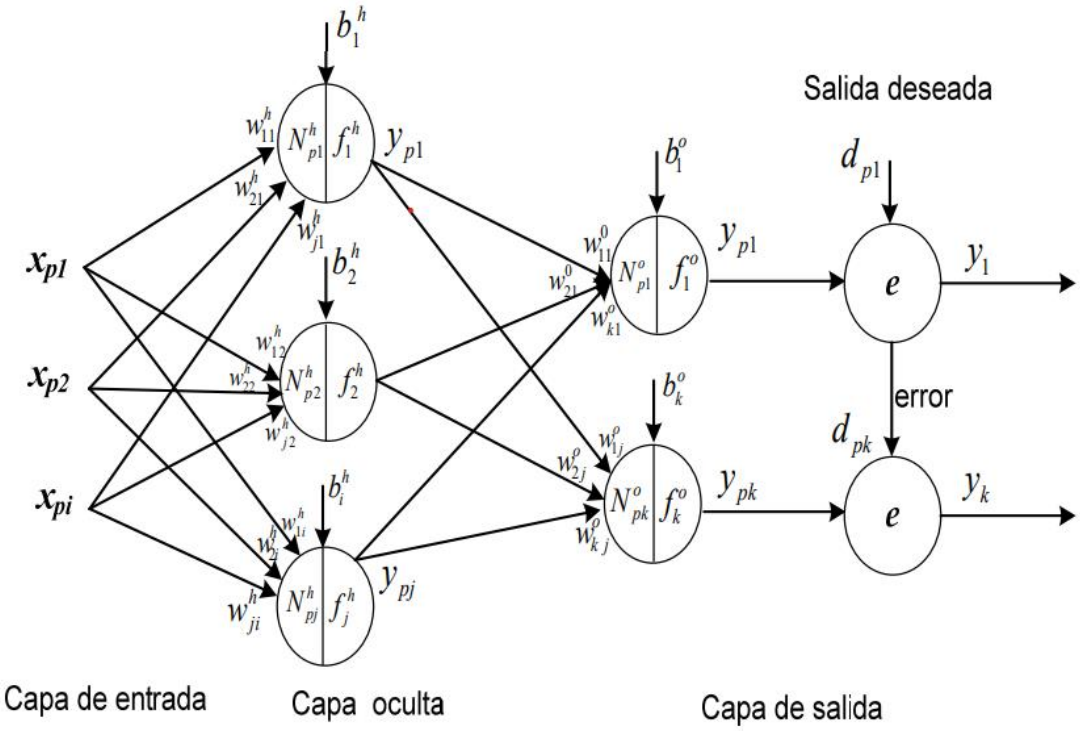
- *Tuning*, reemplaza la última capa completamente conectada de la red con una nueva tarea de clasificación y usa los datos de entrenamiento del nuevo dominio para actualizar los pesos de algunas capas [27].

2.7 Backpropagation

Retro propagación del error o más conocido como Backpropagation, es un tipo de red con aprendizaje supervisado, este método emplea el ciclo de la propagación y adaptación de dos fases [28].

El backpropagation esta basado a travez de patrones de entrenamiendo de una red neuronal, este se propaga desde la primera capa y continuamente a través de las capas siguientes de la red, esto como resultado va generar una salida, esta salida va ser comparada y calculada atra vez de la funcion de error, esto se va repetir para cada una de las salidas, a su vez esta es propagada hacia atrás, se empezara a partir de la capa de salida, y se ira propagando hacia atrás pasando hasta llegar a la capa de entrada, este proceso se realiza con el fin de actualizar los pesos de cada neurona hacer que la red converja a un estado que le permita clasificar correctamente todos los patrones de entrenamiento [28]. La estructura general se muestra en la Figura 24.

Figura 27. Modelo de la RNA Backpropagation.



Fuente: *[28]*.

2.7.1 Algoritmo de entrenamiento de la Red

El algoritmo de entrenamiento empleado para la RNA backpropagation se detalla a continuación [28].

1 Inicializar los pesos de la red (w) con valores aleatorios pequeños.

2 Mientras la condición de paro sea falsa realizar los pasos (3-6).

3 Se presenta un patrón de entrada, $(x_{p1}, x_{p2}, \dots, x_{pi})$ y se especifica la salida deseada que debe generar la red $(d_{p1}, d_{p2}, \dots, d_{pi})$.

4 Se calcula la salida actual de la red, para ello se presentan las entradas a la red y se va calculando la salida que presenta cada capa hasta llegar a la capa de salida $(y_1, y_2, \dots, y_k)$. Los pasos son los siguientes:

a) Se determinan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada.

$$N_{pj}^{h} = \sum_{i=1}^{m} w_{ji}^{h} x_{pi} + b_{i}^{h}$$

b) Aplicar la función de activación para cada entrada de la neurona y así obtener su salida.

$$y_{pj} = f_{j}^{h}(N_{pj}^{h} = \sum_{i=1}^{m} w_{ji}^{h} + b_{i}^{h})$$

c) Realizaremos cálculos para obtener las salidas de las neuronas ingresadas.

$$N_{pk}^{o} = \sum_{j=1}^{m} w_{ji}^{o}\, y_{pj} + b_{k}^{o};$$

$$Y_{pk} = f_{j}^{o}(N_{pj}^{o} = \sum_{j=1}^{m} w_{ji}^{o} y_{pj} + b_{k}^{o})$$

5 Determinación de los términos de error para todas las neuronas:

a) Calculo del error (salida deseada – salida obtenida).

$$e = (d_{pk} - y_{pk})$$

b) Obtención de la delta (producto del error con la derivada de la función de activación con respecto a los pesos de la red).

$$\delta_{pk}^{o} = e * f_{k}^{o}(N_{pk}^{o})$$

6 Para actualizar los pesos. Debemos emplear el siguiente algoritmo recursivo denominado gradiente descendente, empezando por la salida y propangando hacia atrás para llegar a la capa de entrada.

a) Para los pesos de las neuronas de la capa de salida:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \Delta w_{kj}^{o}(t+1);$$

$$\Delta w_{kj}^{o}(t+1) = miu\delta_{pk}^{o} y_{pj}$$

b) Para los pesos de las neuronas de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta w_{ji}^{h}(t+1);$$

$$\Delta w_{ji}^{h}(t+1) = miu\delta_{pj}^{h} y_{pi}$$

7 El error mínimo o número de iteraciones alcanzado logrado nos indica la condición de parar.

2.8 Python

Es un lenguaje de programación interpretado donde se basa en una sintaxis que favorezca un código legible y limpio, ya que es versátil multiplataforma y multiparadigma [29]. Para programar se necesita una instalación del interprete Python y de un editor de texto. Python viene con su propio editor integrado, que es muy agradable y totalmente suficiente para empezar a programar. A medida que se avance en programación, es probable que cambie a algunos otros editores.

En la página de descargas de Python encontraremos diferentes paquetes de instalación para diferentes plataformas. En este proyecto nos referimos a la versión 3.5 de Python [30].

2.9 Tensorflow

TensorFlow es una biblioteca de código libre para el aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar

patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos [31].

TensorFlow es un sistema de aprendizaje automático que funciona a gran escala y en entornos heterogéneos. Su modelo computacional se basa en gráficos de flujo de datos con estado mutable. Los nodos del gráfico se pueden asignar a diferentes máquinas en un clúster, y dentro de cada máquina a las CPU, GPU y otros dispositivos [23].

TensorFlow está disponible en Linux de 64 bits, macos, y plataformas móviles que incluyen Android e iOS. Los cómputos de TensorFlow están expresados como stateful dataflow graphs. El nombre TensorFlow deriva de las operaciones que tales redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como "tensores". En junio de 2016, Jeff Dean de Google declaró que 1,500 repositorios en GitHub mencionaron TensorFlow, de los cuales solo 5 eran de Google [32].

Los tensores son construcciones matemáticas fundamentales en campos como la física y la ingeniería. Históricamente, sin embargo, los tensores han hecho menos avances en la informática, Que tradicionalmente se ha asociado más con la matemática discreta y la lógica [33].

Este estado de cosas ha comenzado a cambiar significativamente con la llegada del aprendizaje de máquina. El aprendizaje y su fundamento en la matemática vectorial, continúa. El aprendizaje de maquina moderno se basa en la manipulación y el cálculo de los tensores [33].

El ejemplo más simple de un tensor es un escalar, un único valor constante extraído de los números reales (los números reales son números decimales de precisión

arbitraria, con números positivos y negativos permitidos). Matemáticamente, nosotros denotamos los números reales por R. Más formalmente, llamamos a un escalar un tensor de rango 0. Si los escalares son tensores de rango 0, ¿qué constituye un tensor de rango 1? formalmente, hablando, un tensor de rango 1 llamado vector. Tradicionalmente, los vectores se escriben como (a, b) ya sea vectores de columna. El tensor T de rango 3 es de forma (N, N, N). Un elemento arbitrario del tensor se seleccionaría especificando (i, j, k) como índices. Existe una vinculación entre tensores y formas. Un tensor de rango 1 tiene una forma de dimensión. 1, un tensor de rango 2 una forma de dimensión 2 y un tensor de rango 3 de dimensión 3. Los tensores se utilizan ampliamente en física para codificar cantidades físicas fundamentales. Por ejemplo, el tensor de tensión se usa comúnmente en la ciencia de los materiales para definir la tensión en un punto dentro de un material. Matemáticamente, el tensor de tensión es un tensor de forma de rango 2 (3, 3) [34].

El objetivo es demostrar que TensorFlow, se puede tomar como otro lenguaje para uso estadístico. Al concluir este artículo, y revisar la funcionalidad existente, tiene puntos que son más robustos, porque la visión de estadística que posee este framework, si bien suple las necesidades básicas como se demostró con el ejercicio de regresión lineal, se debe considerar que las funciones matemáticas que posee son en sí más robustas, en cuanto posee muchos más algoritmos orientados a análisis predictivos, y a la creación de sistemas inteligentes [33].

La unión de TensorFlow intrínseca con Python, y a su vez, observando que Python se integra muy bien con R, forman una trilogía interesante, que es capaz de suplir todos los puntos de análisis estadístico a todos los niveles, sean básicos, intermedios o avanzados.

El otro punto novedoso en cuanto a TensorFlow, es su capacidad, de además de usar el CPU del computador, puede usar el GPU del mismo, lo cual aumente muchísimo el poder de computo. Al tener este aprovechamiento excepcional de hardware, algoritmos de clasificación como manejo de clúster, o procesamientos más avanzados, los cuales necesitaban de servidores o bases de datos distribuidas, son posibles en esta arquitectura, dándole un valor extra de mucho peso para los investigadores [34].

2.10 Librería keras

Keras es una popular biblioteca de aprendizaje profundo con más de 250,000 desarrolladores al momento de escribir, un número que se duplica cada año. Más de 600 colaboradores lo mantienen activamente. Algunos de los ejemplos que usaremos han sido contribuidos al repositorio oficial de Keras GitHub [35].

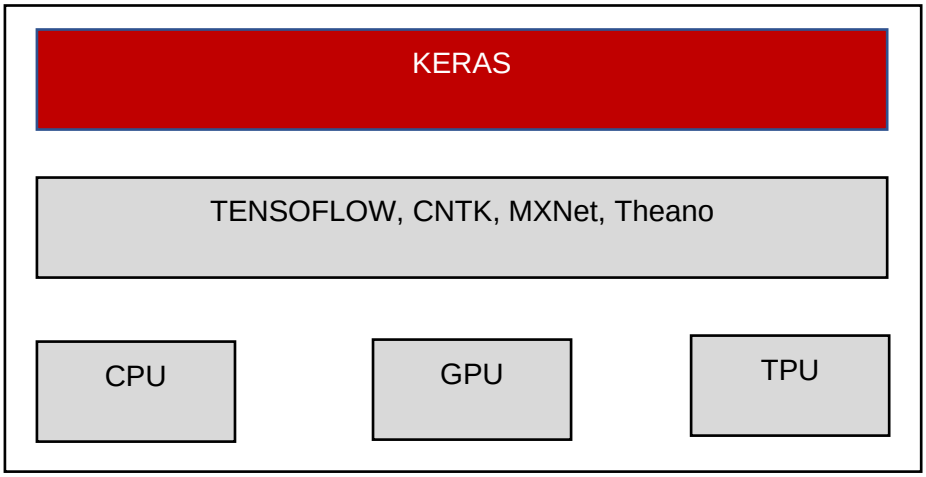
Keras nos permite construir y entrenar modelos de manera eficiente. En la biblioteca, las capas están conectadas entre sí como piezas de lego, lo que da como resultado un modelo limpio y fácil de entender. El entrenamiento de modelos es avanzado y solo requiere datos, varias épocas de entrenamiento y métricas para monitorear. El final de la nueva versión es que la mayoría de los modelos de aprendizaje profundo se pueden implementar con un número significativamente menor de líneas de código [35].

Keras no es una biblioteca de aprendizaje profundo independiente. Como se muestra en la figura, está construido sobre otra biblioteca de aprendizaje profundo o backend. Esto podría ser TensorFlow de Google, Theano de MILA o CNTK de Microsoft. El soporte para MXNet de Apache está casi completado [35].

Podemos cambiar fácilmente de un backend a otro editando el archivo de configuración de Keras. keras / keras.json en Líneas o macos. Ante las diferencias en

la forma en que se implementan los algoritmos de bajo nivel, las redes a menudo pueden tener diferentes velocidades en diferentes Backends [35]. En hardware, Keras se ejecuta en una CPU, GPU y TPU.

Figura 28.Keras soporta CPU, GPU y TPU



Fuente: *[35].*

2.10.1 Modelo secuencial

Keras es un modelo secuencial el cual es capaz de reconocer imágenes a partir de dígitos de la siguiente manera [36]:

$from\ keras.models\ import\ Sequential$

$from\ keras.layers.core\ import\ Dese, Activation$

$model = Sequential(\quad)$

$model.add(Dense\left(32, activation =' sigmoid', input_{shape} = (64,)\right))$

$model.add(Dense(32, activation =' softmax'))$

Para entender el código *input_shape* determina la primera capa donde declara los datos de entrada: Esto indica que tenemos 64 características en el modelo como entrada [22].

La librería Keras tiene como característica deducir automáticamente la forma de las entradas entre capas después de la primera. Para un programador esto es muy interesante ya que solo debe establecer la información para la primera capa. Además, como indicamos el número de neuronas o nodos que tiene para cada capa y su función de activación respectivamente, en este caso es de tipo *sigmoid* [22].

La segunda capa es una *softmax* de 32 neuronas, esto generara una matriz de 32 x 32 lo cual representa los 32 dígitos posibles [22]. Para cada imagen habrá una probabilidad de que pertenezca a los dígitos actuales, el método más simple que proporciona la Liberia Keras es de comprobar la arquitectura mediante la línea de código $summary()$ [22].

2.10.2 Clase model api

Una vez ya generado el modelo secuencial, será de forma más sencillo ya que podemos utilizar el método $add()$, para ingresar una nueva capa como en el ejemplo anterior. Al finalizar nuestro modelo, se debe siempre terminar el modelo dando unas entradas y salidas para inicializar de la siguiente forma [22]:

$$a = Input\big(shape = (64)\big)$$

$$b = Dense(64)(a)$$

$$model = Model(input = a, ouput = b)$$

Según el modelo que tengamos, podemos definir múltiples *inputs* y *ouputs*, y para inicializar estas variables serian de la siguiente forma [36].

$$model = Model(input = [a1, a2], ouput = [b1, b2, b3])$$

2.10.3 Capas

Para diseñas las capas de nuestro modelo, podemos utilizar las siguientes funciones [36]:

- Dense: Para conectar todas las capas.
- Activation: Esta función sus principales funciones es activar la última salida, Keras nos ofrece lo siguiente:
  - Sotfmax
  - ReLu
  - Tanh
  - Sigmoid
  - Linear
- Dropout: Consiste en establecer al azar una fracción de unidades de entrada a 0 en cada actualización durante la fase de entrenamiento. Esto ayuda a evitar el sobreajuste.
- Reshape: Transforma la salida en una forma concreta.

2.10.4 Capas pooling

Al igual que ocurre con las capas convolucionales, Keras ofrece un buen número de funciones para desarrollar capas pooling en redes convolucionales, dependiendo de las dimensiones de la entrada. Como en el caso anterior, una función representativa seria [36]:

$$MaxPooling2D(pool_{size} = (2,2), strides = None, border_{mode} =' valid')$$

Si en vez de utilizar la función máxima se requiere que nuestra capa utilice otro tipo de algoritmos, Keras también ofrece diferentes funciones como: AveragePooling2D, GlobalMaxPooling2D, etc. [36].

2.10.5 Funciones de perdida

La función de perdida se usa para evaluar el grado de error entre salidas calculadas y las salidas deseadas de los datos entrenamiento [22]:

- Mean_squared_error(y_true, y_pred).
- Mean_absolute_error.
- Binary_crossentropy.

2.10.6 Regulizadores

Los regularizadores permiten aplicar penalizaciones en las capas durante la optimización. Estas penalización se incorporan a la función de perdida que la red optimiza [36].

2.10.7 Optimizadores

Los optimizadores que veremos, es uno de los requisitos para poder compilar nuestro modelo ya que especifica el algoritmo según los datos de entrada y la función de costo o perdida ya definidos [22].

2.10.8 Datasets

La herramienta Keras tiene una gran variedad de dataset predefinidos para realizar diferentes pruebas [37].

2.11 CNN arquitecturas

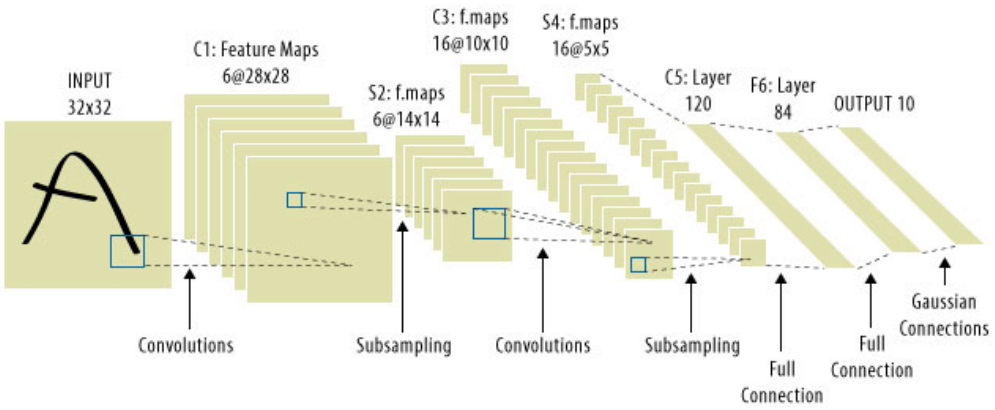
Una red neuronal convolucional es un tipo especial de redes neuronales multicapa, diseñadas para reconocer patrones visuales directamente desde imágenes de píxeles con un pre procesamiento mínimo. El proyecto *ImageNet* es una gran base de datos visual diseñada para uso en el reconocimiento visual de objetos de investigación de software. El proyecto *ImageNet* realiza un concurso anual de software, el Reto de

Reconocimiento Visual a Gran Escala de *ImageNet* (*ILSVRC*), donde los programas de software compiten para clasificar y detectar objetos y escenas correctamente [38].

2.11.1 LeNet-5

LeNet-5, es una red pionera de convolución de 7 niveles realizada por LeCun en 1998, que clasifica los dígitos, para reconocer números escritos a mano en cheques digitalizados en imágenes de escala de grises de 32x32 píxeles. La capacidad de procesar imágenes de mayor resolución requiere capas más grandes y más convolucionales, por lo que esta técnica está limitada por la disponibilidad de recursos informáticos [39].

Figura 29. Convoluciones LeNet-5



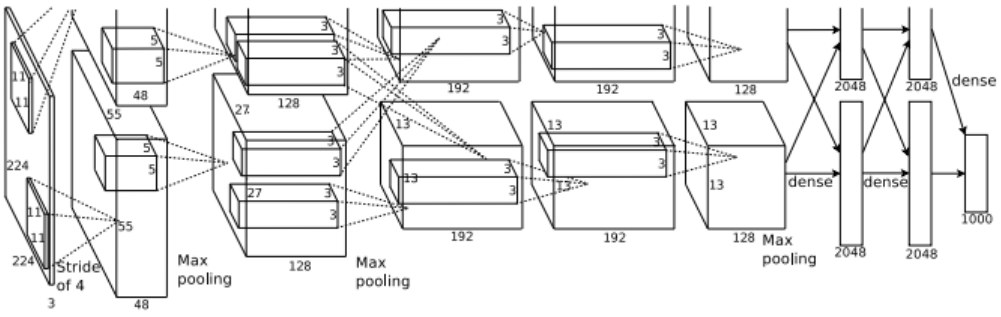
Fuente: *[39].*

2.11.2 AlexNet

La red tiene una arquitectura muy similar a la de LeNet por Yann LeCun et al, pero era más profunda, con más filtros por capa y con capas convolucionales apiladas. Consistió en 11x11, 5x5,3x3, convoluciones, agrupación máxima, abandono, aumento de datos, activaciones *ReLu*, *SGD* con impulso. Adjuntó activaciones *ReLu* después de cada capa convolucional y totalmente conectada. AlexNet recibió capacitación durante 6 días simultáneamente en dos GPU Nvidia

Geforce GTX 580, lo que explica la razón por la cual su red se divide en dos conductos. AlexNet fue diseñada por el grupo Supervisión, que consiste en Alex Krizhevsky, Geoffrey Hinton e Ilya Sutskever [39].

Figura 30. Convoluciones AlexNet

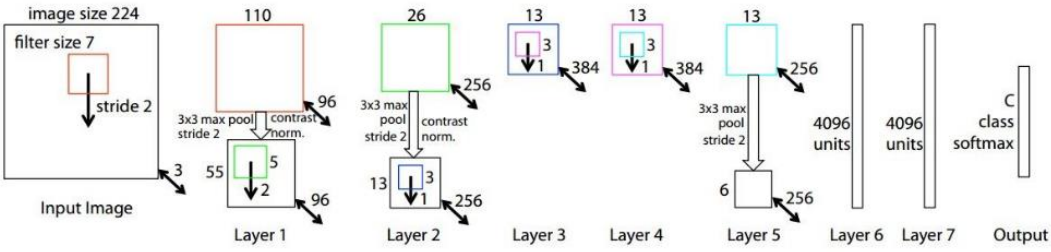


Fuente: *[39].*

2.11.3 ZFNet

Fue una mejora en AlexNet al ajustar los hiperparámetros de la arquitectura, en particular al expandir el tamaño de las capas convolucionales medias y hacer que la zancada y el tamaño del filtro en la primera capa sean más pequeños [38].

Figura 31.Convoluciones ZFNet



Fuente: *[39].*

2.11.4 GoogleNet

La red usa una CNN inspirada en LeNet, pero implementó un elemento novedoso que se denomina módulo de inicio. Se utilizó la normalización por lotes,

distorsiones de imagen y *RMSprop*. Este módulo se basa en varias convoluciones muy pequeñas para reducir drásticamente el número de parámetros. Su arquitectura consistía en una CNN de 22 capas de profundidad, pero redujo el número de parámetros de 60 millones (AlexNet) a 4 millones [39].

Figura 32. Vista tabular arquitectura GoogleNet

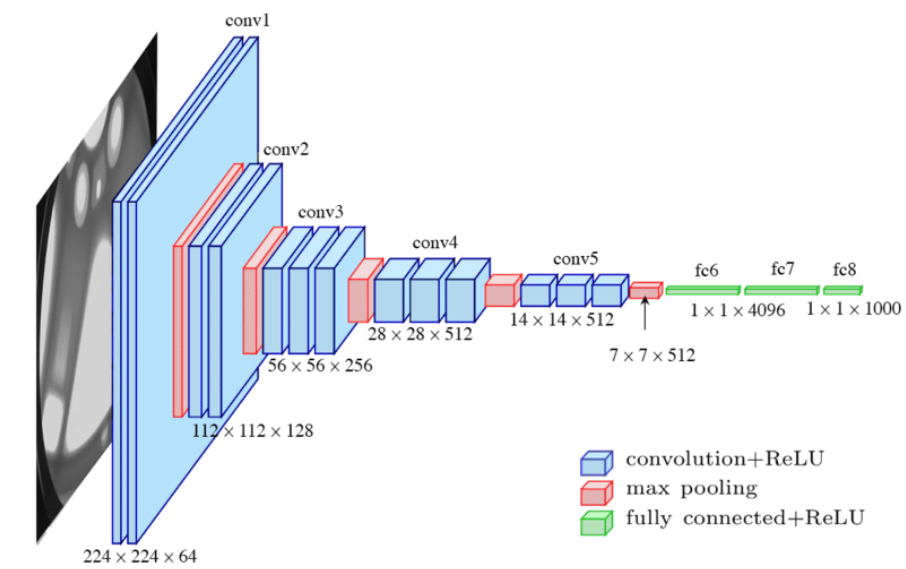
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fuente: *[39].*

2.11.5 VGGNet

Similar a AlexNet, solo convoluciones 3x3. Entrenado en 4 GPU durante 2 a 3 semanas. Actualmente es la opción más preferida en la comunidad para extraer características de las imágenes. La configuración de peso de VGGNet está disponible públicamente y se ha utilizado en muchas otras aplicaciones y desafíos como un extractor de características de línea de base [39]. VGG-16 siempre utiliza filtros 3x3 con un *stride* 1 y usa un mismo *Padding* en capas max pooling de 2x2 con *stride* de 2 [27].

Figura 33.Convoluciones VGGNet



Fuente: *[39].*

2.11.6 Comparación de arquitecturas

Las redes neuronales convolucionales, son modelos de vanguardia para clasificación de imágenes, segmentación, detección de objetos y muchas otras tareas de procesamiento de imágenes. Las arquitecturas de redes neuronales convolucionales representativos son AlexNet, ZFNet, VGGNet, GoogleNet, los resumiremos en la siguiente Tabla, y donde la mejora a lo largo del tiempo. La mejora de cada arquitectura se basa en la evolución de la profundidad: AlexNet Tiene 8 capas, VGGNet y GoogleNet que mostro que aumentar la profundidad puede mejorar la clasificación. Como se puede observar en la Tabla 2, las redes como AlexNet, ZFNet y VGGNet tienen arquitecturas de redes neuronales convolucionales en la que se usan para la clasificación de imágenes. La columna “Error de prueba” indica el error de clasificación Top 5 en ImageNet [20].

Tabla 1. Comparación de arquitecturas CNN.

CNN	Parms (millones)	Capas	Error de Prueba	Características
AlexNet	57	7	15.3%	Resulto efectivo para la clasificación de ImageNet; el punto de inflexión histórico; Ganando el ILSVRC2012 concurso de clasificación de imágenes.
ZFNet	58	7	14.8%	Similar a AlexNet, diferente en stride para cada convolución, tamaño de filtro y numero de profundidad de filtro para algunas capas.
GoogleNet	6	22	6.7%	Usa múltiples ramas de capas convolucionales con diferentes tamaños de filtro y luego concatena mapas de características producidos por estas ramas.
VGG Net	134	16	6.8%	Aumento significativo de la profundidad de la red al apilar filtro de convolución 3 x 3 y aumentando la profundidad de la red paso a paso.

Fuente: *[39].*

CAPÍTULO 3

ESTADO DEL ARTE

3.1 Antecedentes investigativos

Automatic skin cancer detection system [5]. El propósito de esta tesis es proponer un algoritmo para el diagnóstico de cáncer de piel que pueda clasificar las lesiones como malignas o benignas automáticamente. Los diferentes componentes en un diagnostico automatizado de cáncer de piel incluyen; pre procesamiento, segmentación, extracción y selección de características y clasificación. En esta tesis, después de seleccionar las mejores técnicas de mejora de imagen que se logran aplicando y comparando diferentes técnicas de eliminación de ruido y mejora de contraste en las imágenes, se realiza la etapa de segmentación. En esta etapa, se propone un algoritmo de segmentación completamente automatizado en dermatoscopia basado en algoritmos k-mean y level-set. En la siguiente etapa, después de extraer las diversas características de las imágenes, se propone un algoritmo de selección de funciones totalmente automatizado, Smart PSO-SVM, que optimiza la etapa de selección de características. En la etapa de clasificación, se propone el uso de SA-SVM como un nuevo clasificador en el área de los sistemas de detección de cáncer de piel. La

precisión promedio y el puntaje F se estiman en 87.0611% y 0.9167% respectivamente.

Skin cancer Detection by Temperature VariationAnalysis [40]. El propósito de este trabajo es investigar más a fondo otro método y aplicación de la tecnología IR aún no madurada en la detección de cáncer de piel para mejorar la capacidad de detección que se acompaña con un mayor nivel de seguridad. Se diseñó un extenso proyecto de investigación para utilizar dos animales de laboratorio inyectados con células cancerosas por vía subcutánea y dos sensores de radiación IR capaces de detectar longitudes de onda en el rango de 8 a 14 mu, que resultó ser un rango favorable para medir la temperatura de la piel. La recolección de datos se realizó utilizando dos animales de laboratorio como sujetos que formaron un proceso de investigación doble ciego. Se realizó un análisis de las observaciones tanto en enfoques cualitativos como cuantitativos. El análisis y la discusión revelaron el potencial de la radiación IR térmica para detectar la existencia de cáncer de piel. La tesis fue apoyada con evidencia significativa y logró su objetivo.

Convolutional Neural Network in classifying scanned document [2]. En esta tesis, se trata del procesamiento de imágenes en color que representar imágenes de pacientes con posible melanoma. El punto principal es construir un sistema para identificar casos que podrían ser potencialmente peligrosos. El sistema realiza extracción de características utilizando el algoritmo *SIFT* y *SURF* y estas características incorporadas en varios clasificadores como *Support Vector Machines*(*SVM)*, *K-Nearest Neighbor*(*K-NN)* y *Convolutional Neural Network*(*CNN)* y lograr una precisión del 94,51%.

Deep Learning and Convolutional Neural Networks for Medical Image Computing [24]. En este artículo proponen dos métodos de aprendizaje profundo para abordar tres tareas principales que surgen en el área del procesamiento de imágenes de lesiones cutáneas, es decir, segmentación de lesiones (tarea 1), extracción de características dermatoscópicas lesiónales (tarea 2) y clasificación de lesiones (tarea 3). Se propone un marco de aprendizaje profundo que consiste en dos Redes Residuales Totalmente Convolucionales(FCRN) para producir simultáneamente el resultado de la segmentación y el resultado de la clasificación aproximada. Se desarrolla una Unidad de Cálculo del Índice de Lesiones(LICU) para refinar los resultados de la clasificación aproximada calculando el mapa de calor de distancia. Se propone una CNN directa para la tarea de extracción de características dermatoscópicas. Los marcos de aprendizaje profundo propuestos se evaluaron en el conjunto de datos ISIC 2017. Los resultados experimentales muestran las exactitudes prometedoras de nuestros marcos, es decir, 0.753 para la tarea 1, 0.848 para la tarea 2 y 0.912 para la tarea 3.

Skin cancer reorganization and classification with deep neural network [3]. Presenta una estrategia novedosa basada en la técnica de aprendizaje profundo, y lograron una segmentación muy alta de la lesión cutánea y precisión del diagnóstico del melanoma:

- Construyeron una red neuronal de segmentación (skin_segnn), que logró una precisión de detección de límite de lesión muy alta.
- Construyeron otra red neuronal muy profunda basada en la red de inicio v3 de Google (skin_recnn) y su peso bien entrenado.

El novedoso diseño basado en aprendizaje de transferencia basado en la red neuronal profunda skin_inceptions_v3_nn ayuda a lograr una alta precisión de predicción.

Skin Cancer Detection using Arificial Neural Network [41]. En este trabajo, se desarrolla un sistema de clasificación de cáncer de piel automáticamente y se estudia la relación de la imagen de cáncer de piel a través de la red neuronal con diferentes tipos de pre procesamiento. La imagen recopilada se introduce en el sistema y el pre procesamiento de imágenes se utiliza para eliminar el ruido. Las imágenes son segmentadas por segmentos. Hay ciertas características únicas en la región del cáncer de piel que se extraen mediante la técnica de extracción de características. La descomposición de ondas múltiples Mltilevel 2-D se utiliza para la técnica de extracción de características. Estas características se dan a los nodos de entrada de la red neuronal. La red neuronal de propagación hacia atrás y la red neuronal básica radial se utilizan para fines de clasificación, que clasifica las imágenes dadas en cancerosas o no cancerosas.

Skin lesion detection from dermoscopic images using convolutional neural networks [42]. El presente trabajo se basa en la detección automática de lesiones de piel, en concreto en la detección del melanoma a partir de una segmentación semántica de la lesión seguida de una clasificación en forma de imágenes dermatoscópicas utilizando un modelo de aprendizaje profundo llamada U-Net es aplicada con la intención de obtener una exacta extracción de la lesión. Para el segundo modelo permite una generalización para cualquier clasificación de lesiones de piel. La solución propuesta utiliza la arquitectura de la red convolucional VGG-net e investiga el paradigma de la transferencia de aprendizaje. Finalmente, este proyecto realiza una evaluación comparativa del caso sencillo de clasificación sobre una combinación de dos ideas con la intención de averiguar cuál de ellas logra un mejor resultado en la clasificación.

CAPÍTULO 4

METODOLOGÍA

4.1 Metodología

4.1.1 Método

El método a realizar será de carácter descriptiva consiste en conocer actitudes predominantes, situaciones y costumbres que pueda describir de forma exacta las actividades.

4.1.2 Técnica

Las técnicas serán de recolección, análisis e interpretación de datos.

4.2 Descripción de la investigación

4.2.1 Estudio de caso

El estudio de caso se realizará en base la interpretación de los datos que serán en función a la exactitud de las técnicas analizadas.

4.2.2 Población

La población serán las bases de datos de cáncer de piel donde se obtengan imágenes disponibles en repositorios públicos de internet.

4.2.3 Muestra

Se deberán tener en cuenta por los menos 2 repositorios para validar los datos.

4.2.4 Técnicas de observación e instrumentos de colecta de datos

Se utilizará la matriz de confusión el cual nos va permitir ver el desempeño del modelo para diferenciar entre maligno y benigno.

4.3 Operacionalización de variables

Tabla 2.Operacionalización de variables.

Variable	Dimensiones	Indicadores
Independiente: Deep Learning	Arquitecturas de redes convolucionales	Exactitud Sensibilidad Precisión
Dependiente: Porcentaje de Reconocimiento	Bases de datos de validación	Cantidad de parámetros

Fuente: Autoría propia.

CAPÍTULO 5

PROPUESTA

Para el desarrollo de este proyecto se hará un estudio en el campo de aprendizaje profundo sobre el impacto de la imagen de la piel mediante la clasificación de imágenes. Con el fin de alcanzar el objetivo del proyecto se ha adoptado por la propuesta de una arquitectura de red convolucional de clasificación de imágenes.

5.1 Datasets

Para este trabajo se utilizó dos repositorios, el primer conjunto de imágenes con nomenclatura ISBI, ISIC_UDA-2_1 y ISIC_MSK-1_1 para el análisis de cáncer de piel son del repositorio ISIC Arhcive con un total de 2650 imágenes [7], el segundo conjunto de imágenes con nomenclatura IMD se obtuvieron del repositorio PH2Dataset con un total de 200 imágenes, estas imágenes son validadas por el hospital Pedro Hispano [8]. El conjunto de datos está disponible públicamente; todas las imágenes se etiquetan como benignas o malignas e incluyen para etiquetar la lesión dentro de la imagen. Para el conjunto de imágenes se distribuyó de la siguiente manera:

- Segmentación de imágenes. El conjunto de imágenes se divide en 60% de imágenes de entrenamiento, 20% de validación y 20% de prueba, esto se realizó por los dos conjuntos de imágenes por separado y juntos.
- Clasificación de imágenes. Excluir las imágenes con lesiones alteradas o borrosas.

El conjunto de datos tiene pocas muestras de la clase maligno a comparación de la benigno, los subgrupos de prueba, validación y entrenamiento.

5.1.1 ISIC Arhcive

La International Skin Imaging Collaboration (ISIC) es un esfuerzo internacional para mejorar el diagnóstico de melanoma, patrocinado por la Sociedad Internacional de Imágenes Digitales de la Piel (ISDIS). El archivo ISIC contiene la mayor colección disponible públicamente de imágenes dermatoscópicas de calidad controlada de lesiones cutáneas.

Todas las imágenes entrantes al Archivo ISIC se examinan para garantizar la privacidad y la calidad. La mayoría de las imágenes tienen metadatos clínicos asociados, que han sido verificados por reconocidos expertos en melanoma. Un subconjunto de las imágenes ha sido sometido a anotaciones y marcas por reconocidos expertos en cáncer de piel. Estas marcas incluyen características dermatoscópicas (es decir, elementos morfológicos globales y focales en la imagen que se sabe que discriminan entre los tipos de lesiones cutáneas) [43].

5.1.2 PH2 Dataset

La base de datos PH2 se creó a través de una colaboración de investigación conjunta entre la Universidade do Porto, el Instituto Superior Técnico Lisboa y el Servicio de Dermatología del Hospital Pedro Hispano en Matosinhos, Portugal.

Este trabajo de investigación fue apoyado por la Fundación Portuguesa para la Ciencia y la Tecnología (FCT) a través de los proyectos PTDC / SAU-BEB / 103471/2008 y PEst-OE / EEI / LA0009 / 2011, con fondos de FEDER a través del Programa Operativo Factores de Competitividad (COMPETE), por fondos portugueses a través del Centro de Investigación y Desarrollo en Matemáticas y Aplicaciones y elFCT , dentro del proyecto PEst-C / MAT / UI4106 / 2011 con el número COMPETE FCOMP-01-0124-FEDER-022690 [8].

La base de datos PH2 puede ser utilizada independientemente por el usuario o con la herramienta de navegador PH2. El software del navegador PH2 es una herramienta personalizada de la base de datos de imágenes PH2 [8].

5.2 Framework de Desarrollo

Para el desarrollo de este proyecto, se utilizó el lenguaje de programación Python, es de código abierto y nos permite crear aplicaciones e involucrarnos en su desarrollo, es multiplataforma ya que permite ejecutarse en muchas plataformas como Windows, Mac y Linux. Python ha crecido bastante debido a que su base se asienta en lenguajes matemáticos como R y con librerías y framework como *NumPy* o *PyBrain*. para ejecutar las librerías de desarrollo en *Deep Learning*. Mencionado en capítulo 2.

Python cuenta con el framework *Keras*, encargado de realizar el aprendizaje profundo dentro de las redes neuronales, *Keras* nos proporcionas una capa de abstracción sobre *TensorFlow*, que se utiliza como framework principal de la red neuronal. *Keras* nos va permitir:

- Modularidad para que las redes neuronales definidas sigan una secuencia de pila lineal de capas.

- Minimizar las funciones incluidas en las librerías permitiendo crear y modificar capas de red fácilmente.

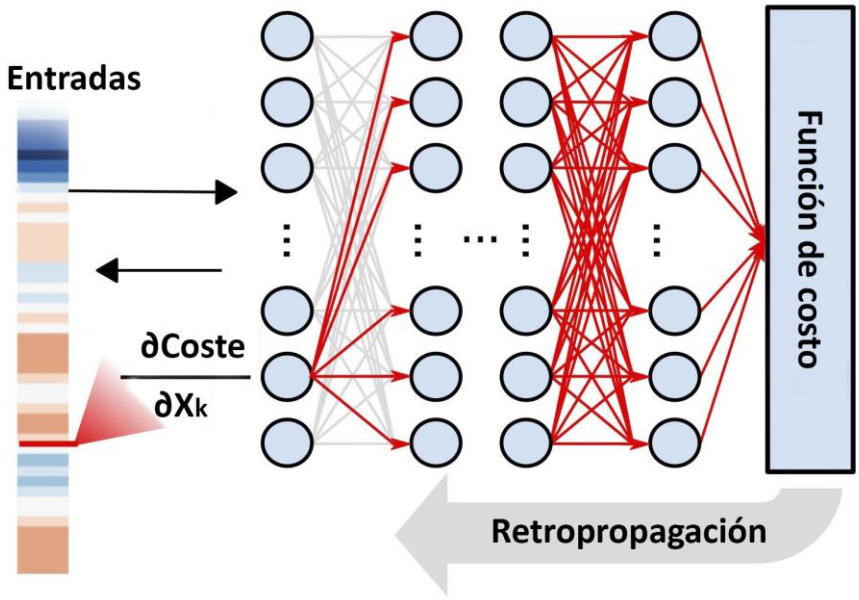
Es por estas razones que se utilizó *Keras* para implementar la arquitectura de las redes neuronales en este proyecto.

5.3 Algoritmo de la red neuronal convolucional

Backpropagation es un algoritmo para que redes neuronales aprendan a través de patrones de entrada y clases correspondientes, utilizando más niveles de neuronas que el Perceptrón. El algoritmo retropropagación del error o *Backpropagation*, que es un tipo de red con aprendizaje, el cual emplea un ciclo de propagación y adaptación de fases.

El algoritmo de *Backpropagation* nos va indicar el error que se comete en cada neurona calculando las derivadas parciales de la función de costo o perdida con respecto a las variables, de tal forma que se calcula el error en cada neurona empezando desde la última capa y lo ira propagando hacia atrás hasta actualizar los pesos correctos de cada neurona. A continuación, se muestra la Figura 34, como se realiza el algoritmo de *Backpropagation* en el proyecto.

Figura 34. Backpropagation en el proyecto.



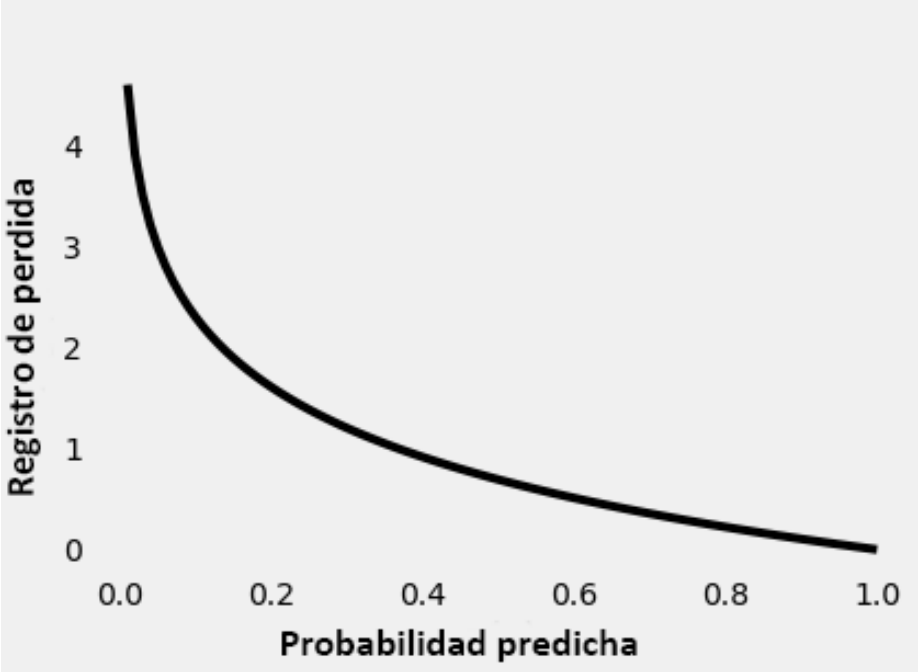
Fuente: Autoría propia.

5.4 Función de costo o perdida

Ya que la propuesta a realizar en este proyecto es de reconocer el cáncer en maligno o benigno, podemos utilizar un clasificar binario, ya que tomaremos como 1 a los valores positivos como maligno y cero a los negativos que son los benigno. Esto significa que nuestra función de costo o perdida debe realizarse con la función Binary_Cross_Entropy, ya que va evaluar las probabilidades predichas según nuestras probabilidades predichas en 0 y 1.

En nuestro proyecto va calcular una perdida, donde va penalizar las malas predicciones, si la probabilidad asociada con la clase verdadera es 1.0, necesitamos que perdida sea cero, por el contrario, si esa probabilidad es baja, digamos 0.01 necesitamos que su perdida sea enorme.

Figura 35. Función de costo o error



Fuente: Autoría propia.

Esto dará la probabilidad predicha de la clase verdadera que se acerca a cero, la perdida aumenta exponencialmente. Finalmente se calcula la media de todas las perdidas.

5.5 Algoritmo de optimización

Un algoritmo de optimización nos ayuda en el proceso de minimizar o maximizar la función de coste o error. Dentro de nuestra propuesta tendremos como parámetro a la tasa de aprendizaje que se utiliza para calcular los valores objetivo que le llamaremos *Y* del conjunto de predictores *X* en nuestro modelo. Entonces tendremos los valores de peso que le llamaremos *W* y el *Bias* como *B* de nuestra red neuronal como sus parámetros de aprendizaje que se utilizan para calcular los valores de salida y estos

se irán actualizando en la dirección de la solución óptima, es decir, deberá ir minimizando la perdida mediante proceso de capacitación de la red.

Existen diferentes tipos de algoritmos de optimización que utilizan en redes neuronales y las nombraremos para seleccionar la mejor para nuestra red neuronal propuesta:

- Descenso de gradiente, calcula el gradiente de todo el conjunto de datos, pero lo realizara solo una actualización, por lo tanto, puede ser muy lento y difícil de controlar para conjunto de datos que son muy grandes y no tenga mucha capacidad de memoria.
- Adagrad, utiliza una tasa de aprendizaje para cada parámetro en un tiempo basado en los gradientes anteriores que calcularon para ese parámetro. Su principal debilidad es que su tasa de aprendizaje siempre está disminuyendo y decayendo, porque sabemos que a medida que la tasa de aprendizaje se hace cada vez más pequeña, la capacidad del modelo para aprender disminuye rápidamente y eso proporciona una convergencia muy lenta y toma mucho tiempo entrenar y aprender, es decir, la velocidad de aprendizaje sufre y disminuye
- Adadelta, es una extensión de Adagrad que tiende a eliminar el problema de la tasa de aprendizaje en descomposición. En lugar de acumular todos los gradientes cuadrados anteriores, Adadelta limita la ventana de gradientes pasados acumulados a un tamaño fijo. Una de sus debilidades encontradas es de porque no calcular los cambios de *momentum* para cada parámetro y almacenarlos por separado.
- Adam, representa la estimación adaptativa del momentum, es otro método que calcula las tasas de aprendizaje adaptativo para cada parámetro y además de

almacenar un promedio exponencial decreciente de gradientes cuadrados pasados como Adadelta.

Entonces al realizar las diferentes comparaciones entre los algoritmos de optimización, podemos escoger a Adam ya que es un algoritmo adaptativo, convergen muy rápido y encuentran rápidamente la dirección correcta en la que deberían ocurrir las actualizaciones de parámetros.

5.6 Arquitectura de la red neuronal

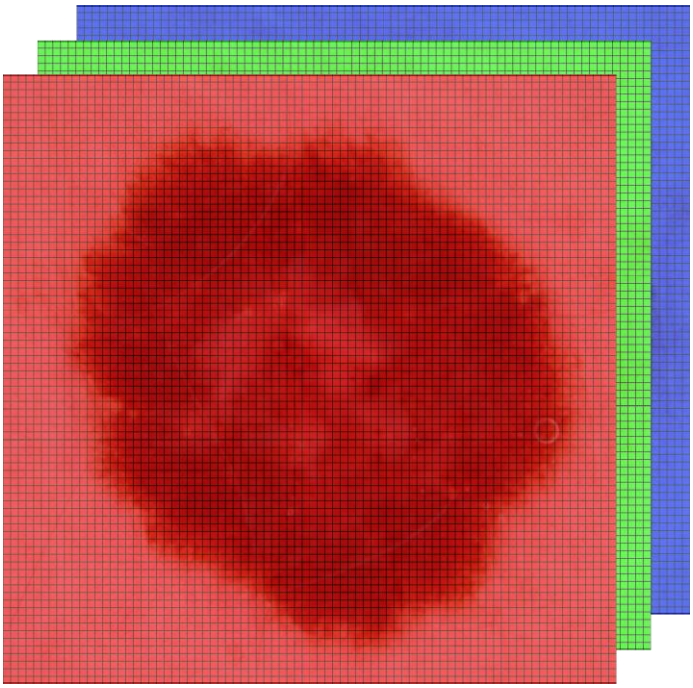
Hoy en día, existen varias arquitecturas de redes neuronales convolucionales como el de reconocimiento visual ImageNet Large Scale, el cual se hizo muy popular dentro del aprendizaje profundo. Existen diversas arquitecturas de clasificación de imágenes como VGG-16, AlexNet y GoogleNet, nuestro propósito es proponer una arquitectura para el reconocimiento de cáncer de piel.

5.6.1 Red neuronal propuesta

Las clasificaciones de imágenes de CNN toman una imagen de entrada, la procesan y la clasifican bajo el criterio de su categoría. Las computadoras ven una imagen como una matriz de pixeles y depende de la resolución de la imagen. Según la resolución de la imagen, se verá como *H x W x D*, donde H es la altura, *W* es el ancho y *D* es la dimensión.

Para el proyecto nuestras imágenes estarán basado de tipo RGB, donde las imágenes de entrada serán 150 x 150 x 3, el valor 3 se refieren a los valores RGB. Entonces los valores se reflejan de la siguiente manera:

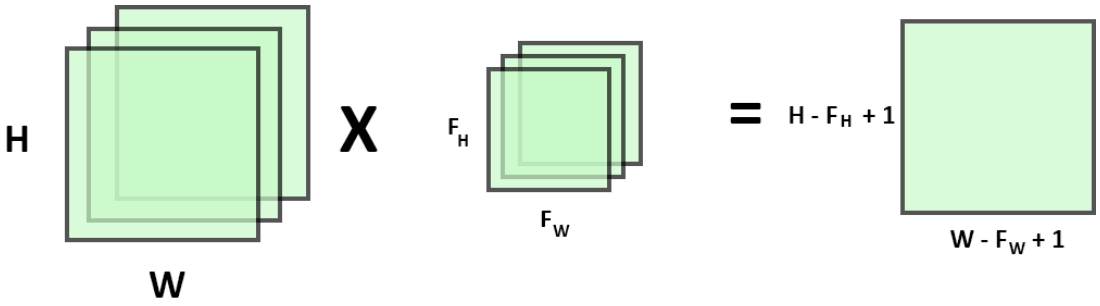
Figura 36. Matriz RGB 150 x 150 x 3.



Fuente: Autoría propia.

Después de definir las dimensiones de la imagen de entrada, los siguiente es definir la convolución, la cual es la primera capa que extrae características de una imagen de entrada. La convolución que realizaremos preserva la relación entre pixeles al aprender las características de la imagen utilizando pequeños cuadros de datos de entrada. Es una operación matemática que toma dos entradas, como una matriz de imágenes y un filtro.

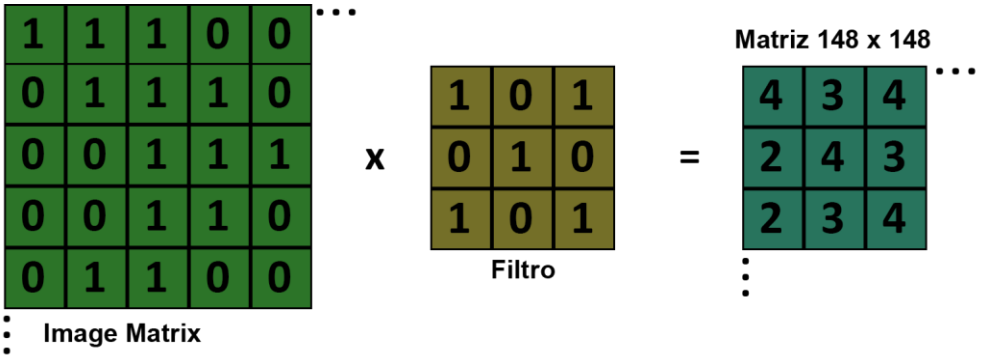
Figura 37. Matriz RGB multiplica a la matriz filtro.



Fuente: Autoría propia.

Entonces considerando la convolución, reemplazaremos en los valores de la imagen de entrada de *150 x 150* cuyos valores de pixeles de imagen son 0 y 1; y una matriz de filtro *3 x 3* como se muestra a continuación:

Figura 38. Matriz de salida 148 x 148.



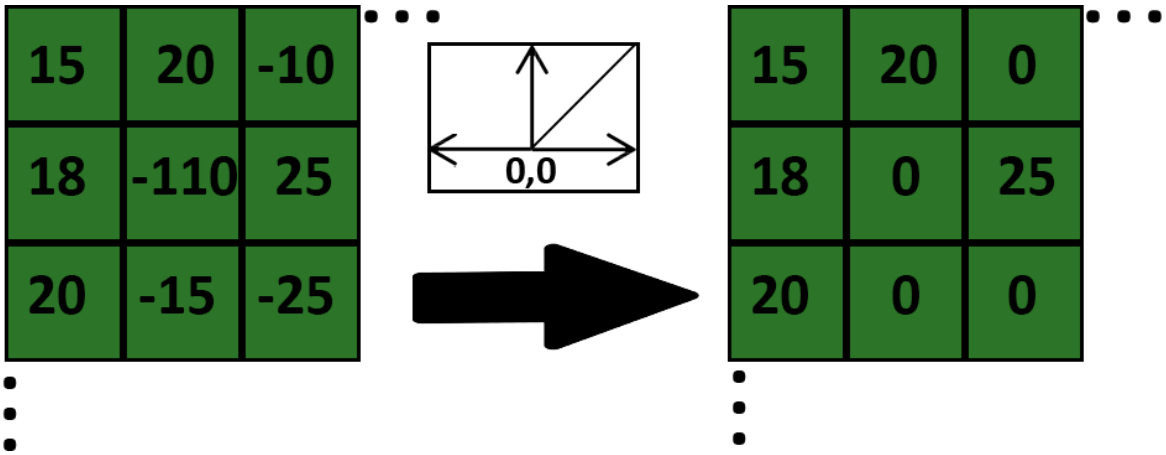
Fuente: Autoría propia.

La convolución de las imágenes al realizar los diferentes filtros puede encontrar detección de bordes, desenfoque y nitidez. El stride o desplazamiento es de 1 es decir que avanzara el filtro de 1 en 1 por vez.

Seguido tendremos una matriz 148 x 148 con un filtro de profundidad de 32, al cual aplicaremos una función de activación en este caso usaremos la función

ReLu, ya que garantiza un mejor rendimiento sobre tanh o sigmoid en clasificación de imágenes. El propósito de ReLu es introducir la no linealidad en nuestra red neuronal convolucional.

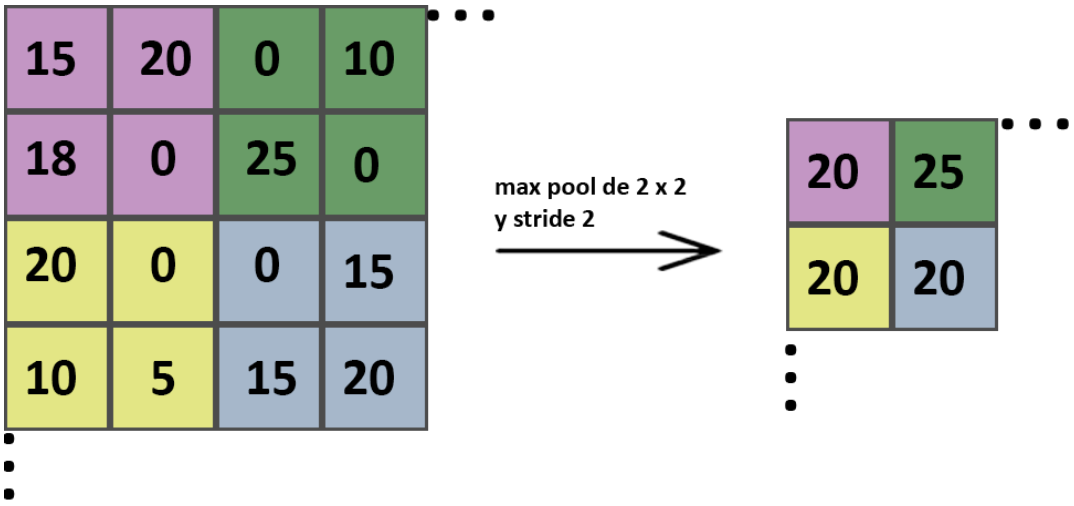
Figura 39. Operación ReLu a la CNN.



Fuente: Autoría propia.

Ahora, lo siguiente será reducir el número de parámetros, para esto utilizaremos la agrupación Max Pooling, donde toma el elemento más grande o la característica más importante. Entonces posteriormente quedaría nuestra matriz de la siguiente manera aplicando un Max Pool de 2 x 2 y un stride de 2 por defecto:

Figura 40. Agrupamiento max pooling 2x2.

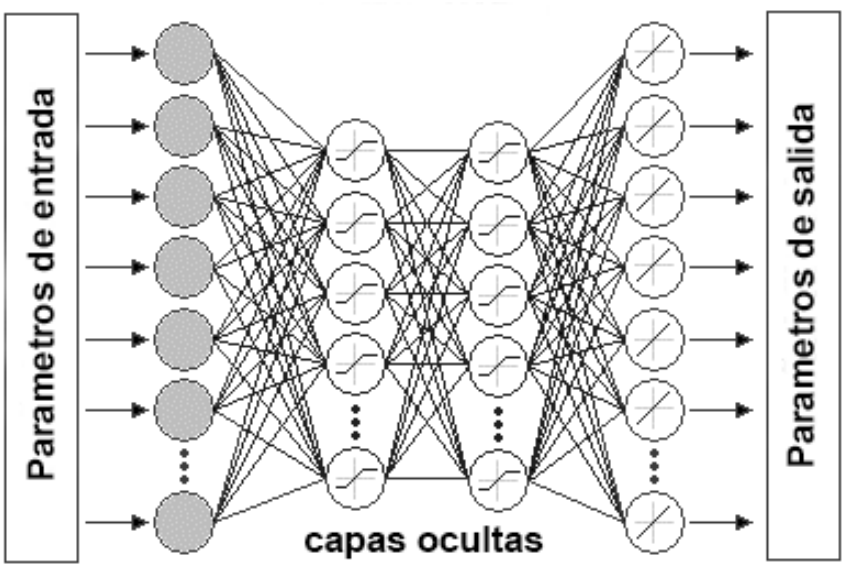


Fuente: Autoría propia.

Aquí concluimos con la convolución, para eso iremos añadiendo capas convolucionales y capas pooling, hasta que la red neuronal obtenga las mejores características y dejar de añadir más convoluciones hasta que comience a obtener otras características.

Una vez concluido este proceso, aplanaremos nuestra matriz en un vector y la alimentamos en una capa completamente conectada que será nuestra red neuronal utilizando el algoritmo *Backpropation*, esto se representaría de la siguiente manera.

Figura 41. Capas totalmente conectas.

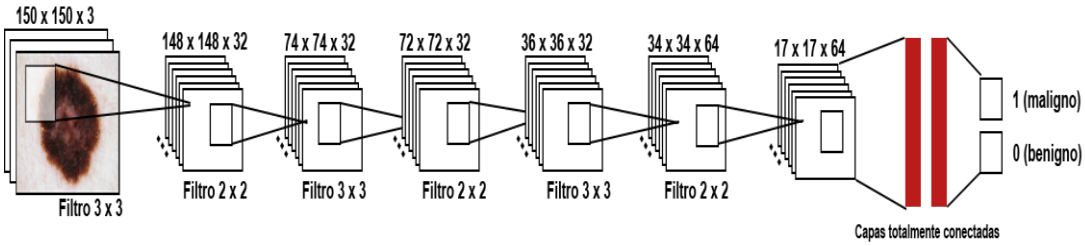


Fuente: Autoría propia.

En la imagen anterior, nos muestra que entrarán 64 neuronas como parámetros y estos estarán en un vector y finalmente nos dará otros parámetros de salida. Definimos como 64 debido a que al aplanar nos da esta cantidad de parámetros. Este proceso lo realizaremos una vez más, pero le diremos que lo haga a una sola neurona, esto nos sirve para obtener solo 1 respuesta.

Finalmente, tenemos una función de activación como softmax o sigmoid, que nuestro caso será sigmoid para clasificar la imagen como cáncer maligno o benigno.

Figura 42. Arquitectura propuesta.



Fuente: Autoría propia.

Esta red neuronal propuesta se realizó para lograr un rendimiento excelente sobre el conjunto de imágenes encontrados, se trabajó en base un modelo secuencial la cual la capa de entrada está basada en tipo RGB, donde estas imágenes pasan a través de tres bloques convolucionales, se utilizan pequeños filtros de 3 por 3 y cada bloque incluye una operación de convolución 2D, el cual cambia entre bloques. Todas las capas ocultas están equipadas con una *ReLu* como la capa de función de activación, que significa operación de no linealidad e incluyen la agrupación espacial mediante el uso de una capa de agrupación máxima. La red concluye con un bloque clasificador que consta de una capa. La capa de salida final completamente conectada realiza una clasificación binaria y la función de activación es *Sigmoidal*.

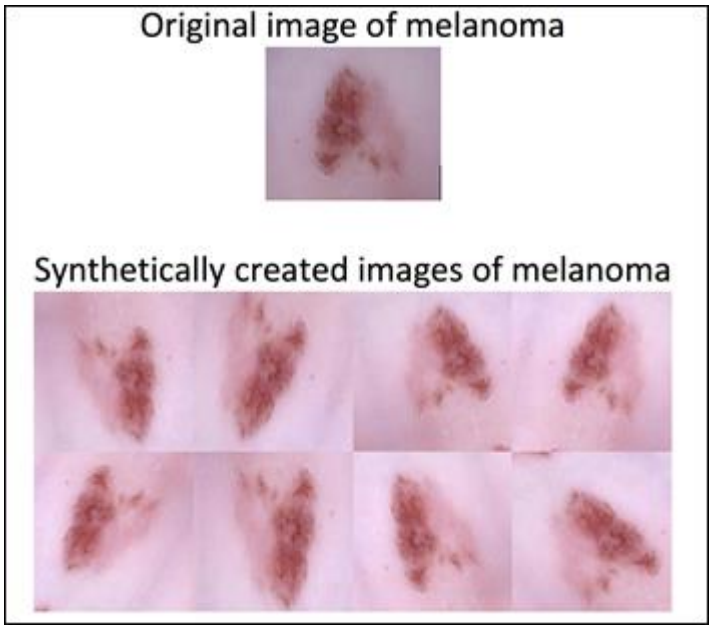
5.7 Data augmentation

Con el fin de aprovecha al máximo nuestros pocos ejemplos de capacitación y aumentar la precisión del modelo, se aumentó a través de varias transformaciones aleatorias. Las técnicas de aumento de datos seleccionadas fueron: re escalado de tamaño, rotación de 255 grados, desplazamiento horizontal y zoom de imagen.

Además, se espera que el aumento de datos también ayude a prevenir el ajuste excesivo, ya que generalmente es un problema común en el aprendizaje automático relacionado con el conjunto de imágenes pequeños, cuando el modelo, expuesto a muy pocos ejemplos, aprende patrones que no se generalizan a nuevos datos y por esta razón, mejora la capacidad del modelo para generalizar.

También nos ayuda equilibrar los dataset para generar la misma cantidad de imágenes y poder comparar los resultados.

Figura 43. Data augmented imagen maligna



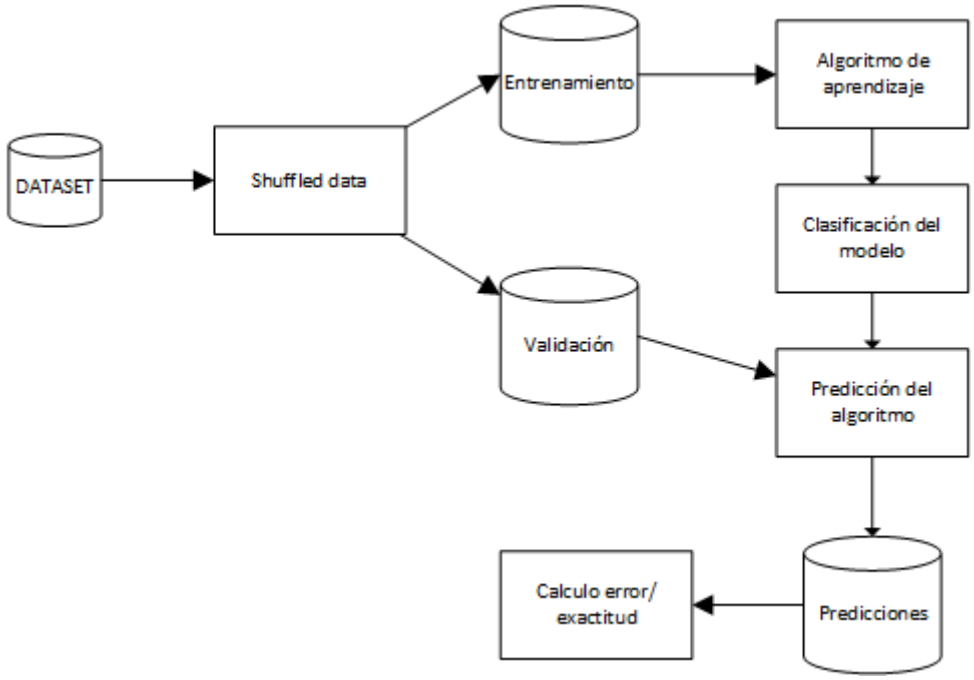
Fuente: *[34].*

5.8 Método de entrenamiento

Se propone la metodología de capacitación para solucionar los problemas de este proyecto. Siguiendo el diagrama de la siguiente figura, los datos de entrenamiento se entrenan a través del algoritmo de aprendizaje definido por cada modelo, que aplica el descenso del gradiente estocástico. Una vez que el modelo ha aprendido los pesos, un algoritmo de predicción clasifica los datos de validación de acuerdo con el

entrenamiento. Se realiza una evaluación final del modelo comparando las predicciones con los datos de la verdad básica.

Figura 44. Metodología de entrenamiento.



Fuente: Autoría propia.

5.9 Definición de parámetros

Durante el entrenamiento, algunos parámetros deben considerarse para obtener el mejor rendimiento de la red propuesta en relación con el problema. Los paramentos son los siguientes:

- **Tamaño batch:** El tamaño se atribuye a la cantidad de imágenes de entrenamiento en *forward* o *backward*. Es importante destacar en cuanto mayor sea el tamaño, más memoria se necesitará.
- **Iteraciones:** El número de iteraciones es el número de *forward* o *backward*, cada paso usa un numero de imágenes del tamaño *batch*.

- **Épocas:** El número de épocas mide cuantas veces se ha visto cada imagen durante el entrenamiento; es decir una época significa que cada imagen se ha visto una vez. También se puede entender como un *forward* y *backward* de todos los ejemplos de entrenamiento. Se computa numéricamente como:

$$epocas = \frac{tamaño\ batch * iteraciones}{imagenes\ entrenadas}$$

- **Función de perdida:** Evalúa la penalización entre la predicción y la etiquetada de la verdad fundamental en cada *batch*.
- **Tasa de aprendizaje:** El parámetro de aprendizaje define el tamaño del paso para el cual se actualizan los pesos de un modelo con respecto al descenso del gradiente estocástico.
- **Optimizador:** El framework Keras proporciona optimizadores para encontrar el conjunto óptimo de parámetros para el modelo. Algunos ejemplos de optimizadores son el *SGD*, *RMSprop* y *ADAM*.

5.10 Clasificación

El principal problema de la tarea de clasificación es evitar el ajuste excesivo causado por el pequeño número de imágenes de lesión de la piel en la mayoría de los *datasets* de dermatología. Para resolver este problema, el objetivo entrenar nuestro modelo utilizando el primer *dataset*, posteriormente entrenar con el segundo dataset y por último combinar los dos *dataset*. El supuesto subyacente detrás del aprendizaje por transferencia es que el modelo pre entrenado ya ha aprendido características que podrían ser útiles para la tarea de clasificación en cuestión. *Transfer Learning*, en la práctica, a usar las capas convolucionales previamente entrenada como un extractor de características fija, que se puede lograr congelando todos los bloques

convolucionales y entrenando solo las capas completamente conectadas con el nuevo conjunto de datos. Otra técnica común de *Fine Tuning* consiste no solo en volver a capacitar el clasificador en la parte superior de la red con el nuevo conjunto de datos, sino también en aplicar un ajuste fino de la red al capacitar solo a la parte de mayor nivel de las capas convolucionales y continuar con el *backpropagation*. Y por último la técnica que se utilizará será un aprendizaje desde cero, ya que se va a proponer una nueva arquitectura para el aprendizaje de nuestra red neuronal.

CAPÍTULO 6

PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

6.1 Resultados

Se realizaron los siguientes experimentos:

- Primero se utilizó el *dataset* ISIC realizando el entrenamiento y la validación realizadas por el modelo propuesto.
- Segundo se utilizó el *dataset* PH2 realizando el entrenamiento y la validación realizadas por el modelo propuesto.
- Tercero se unieron los dos dataset realizando el entrenamiento y la validación realizadas por el modelo propuesto.

Según los parámetros elegidos para el proceso de clasificación las cuales fueron 100 épocas elegidas en base al examen de comportamiento de los gráficos de precisión/perdidas vs número de épocas, donde cada época tomo 300 segundos de la GPU. El proceso de ajuste se realizó con una tasa de aprendizaje muy pequeña y el optimizador *Adam* ayudo a minimizar la función de perdida.

6.2 Métricas

Las principales métricas utilizadas para la evaluación de este trabajo son la siguientes:

- Exactitud: Calcula el número de predicciones correctas dividido por el número total de muestras.

$$Exactitud = \frac{número\ de\ predicciones\ correctas}{número\ de\ ejemplos}$$

- Sensibilidad: Se calcula a la proporción de las veces en que un resultado muestra verdaderos positivos. Mientras la sensibilidad se acerca al 100%, es más probable que un paciendo padece una enfermedad.

$$Sensibilidad = \frac{número\ de\ verdaderos\ positivos}{número\ de\ verdaderos\ positivos + número\ de\ falsos\ negativos}$$

- Precisión: Se calcula como la fracción de instancias recuperadas que son relevantes.

$$Precisión = \frac{número\ de\ verdaderos\ positvos}{numero\ de\ verdaderos\ positivos + numero\ de\ falsos\ positivos}$$

- Especificidad: Se calcula a la proporción del tiempo en la que un resultado da negativos verdaderos. Mientras la especificidad se acerca al 100% es más probable que un resultado signifique que el paciente no tenga una enfermedad.

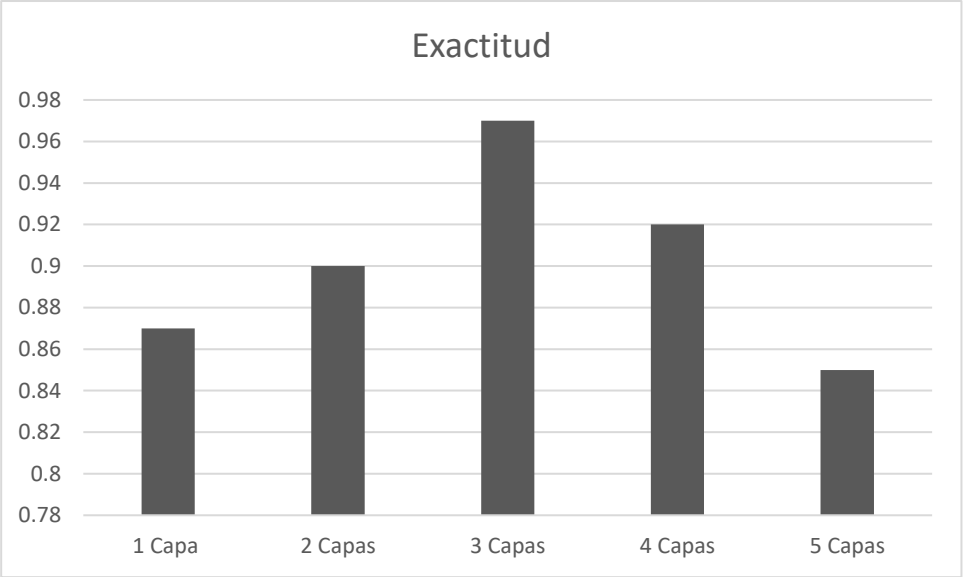
$$Especificidad = \frac{número\ de\ negativos\ verdaderos}{número\ de\ negativos\ verdaderos + número\ de\ falsos\ positivos}$$

6.3 Entrenamiento

Para el entrenamiento, primero se tuvo que ir agregando una capa convolucional, para encontrar los mejores resultados, sobre la capa con mayor exactitud. En la figura 34 que se muestra a continuación se detallan los valores de exactitud en base al *dataset*

PH2, ya que contiene la menor cantidad de imágenes y el entrenamiento será rápido, para ver la mejor arquitectura.

Figura 45. Resultados con diferentes capas convolucionales



Fuente: Autoría propia.

El resumen de cada una de las arquitecturas se encuentra en los anexos, donde podemos ver la representación resumida de salida de cada modelo, a partir de esto vamos obtener las métricas de precisión y recuperación.

Tabla 3.Métrica de precisión y recuperación por capas

Capas	Precisión	Recall	F1-Score
1	0.836576	0.853175	0.844793
2	0.893216	0.903114	0.898137
3	0.973301	0.970012	0.971653
4	0.921554	0.910789	0.916139
5	0.859927	0.811364	0.834939

Fuente: Autoría propia.

La métrica F1-Score busca un equilibrio entre la precisión y recuperación, entonces en la tabla podemos ver que la precisión en la capa 3, F1-Score nos indica que tenemos una buena precisión y recuperación en esta capa. A partir, podemos realizar los entrenamientos utilizando la arquitectura con capa 3, las arquitecturas se pueden ver en el anexo B.

Los resultados del entrenamiento se basan en un cálculo binario creado automáticamente. Vamos a obtener el promedio de las probabilidades que obtuvo nuestra red neuronal en el entrenamiento:

- *Dataset* ISIC – 530 Imágenes
  - Verdadero positivo (TP) = 258
  - Falso positivo (FP) = 7
  - Verdadero negativo (TN) = 213
  - Falso negativo (FN) = 52
- *Dataset* PH2 – 40 Imágenes
  - Verdadero positivo (TP) = 19
  - Falso positivo (FP) = 1
  - Verdadero negativo (TN) = 18
  - Falso negativo (FN) = 2
- *Dataset* junto – 570 Imágenes
  - Verdadero positivo (TP) = 270
  - Falso positivo (FP) = 15
  - Verdadero negativo (TN) = 242
  - Falso negativo (FN) = 53

A continuación, se muestra los siguientes resultados por dataset según las métricas que se obtuvieron de acuerdo a las pruebas realizadas la capa 3:

Tabla 4. Comparación de las métricas de entrenamiento por dataset

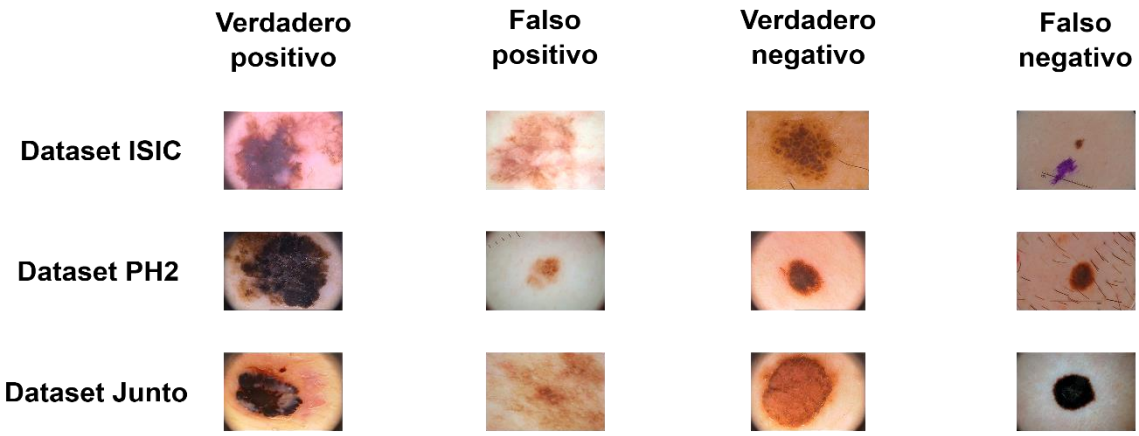
Dataset	Exactitud	Sensibilidad	Especificidad	Precisión
DATASET ISIC	0.8887	0.8322	0.9681	0.9735
DATASET PH2	0.9250	0.9047	0.9473	0.9500
DATASET JUNTO	0.8982	0.8626	0.9747	0.9473

Fuente: Autoría propia.

Se comparó los resultados con los dos conjuntos de datos por separado y juntos como se puede ver en la Tabla 4, lo que se quiere lograr con estos resultados, es de ver el rendimiento por separado de cada conjunto de datos, y no de ver cuál es el mejor *dataset*, debido a la poca cantidad de imágenes del *dataset* PH2.

En la Figura 46, son las imágenes del entrenamiento para diagnosticar cáncer maligno o benigno, podemos observar que el error, con la red neuronal puede afectarse a una gran similitud o que tan limpia y clara sea la imagen. Es un ejemplo de cómo ingresan a la red neuronal convolucional ya entrenada y esta finalmente nos dará el resultado de benigna o maligna, no hay un detalle interno de cómo se realiza el proceso, debido a que en redes neuronales los parámetros y procesamiento se hace internamente.

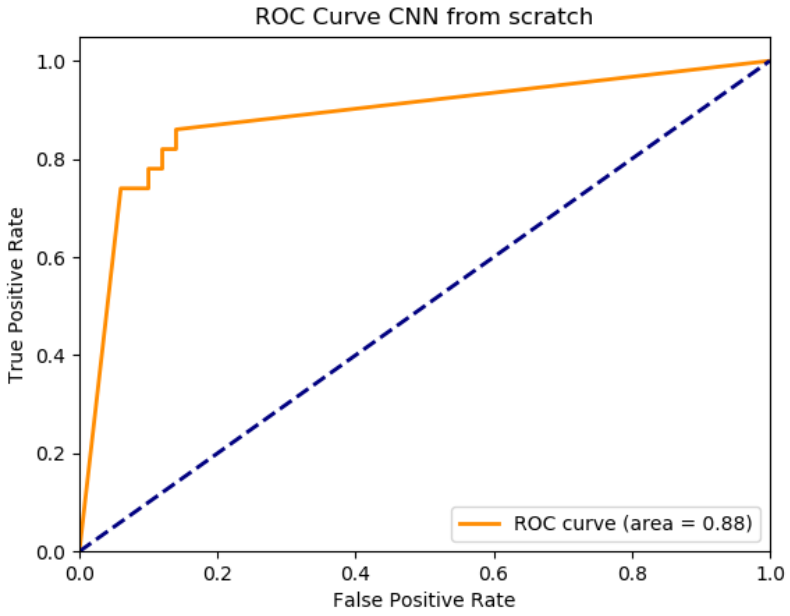
Figura 46. Imagen benigna



Fuente: Autoría propia.

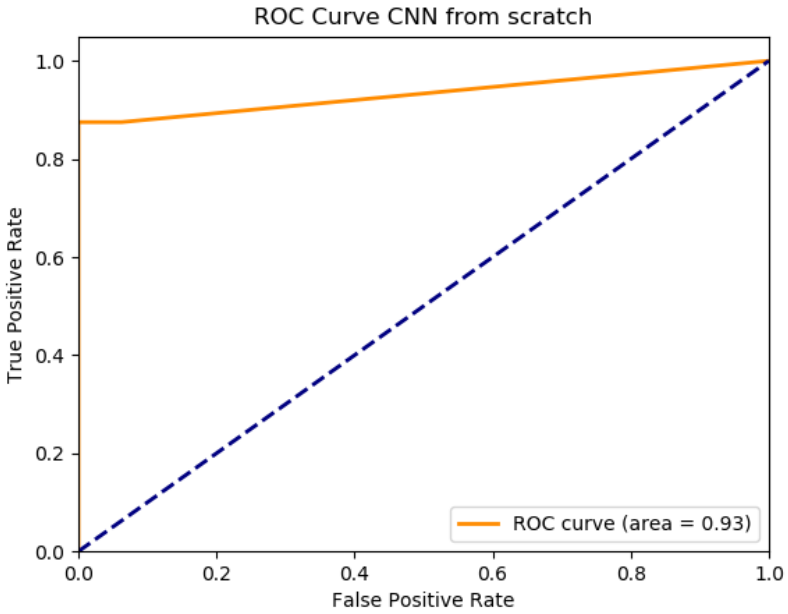
Finalmente, al final de esta sección se adjunta la curva ROC para evaluar el desempeño de la clasificación binaria.

Figura 47.Curva ROC para dataset ISIC



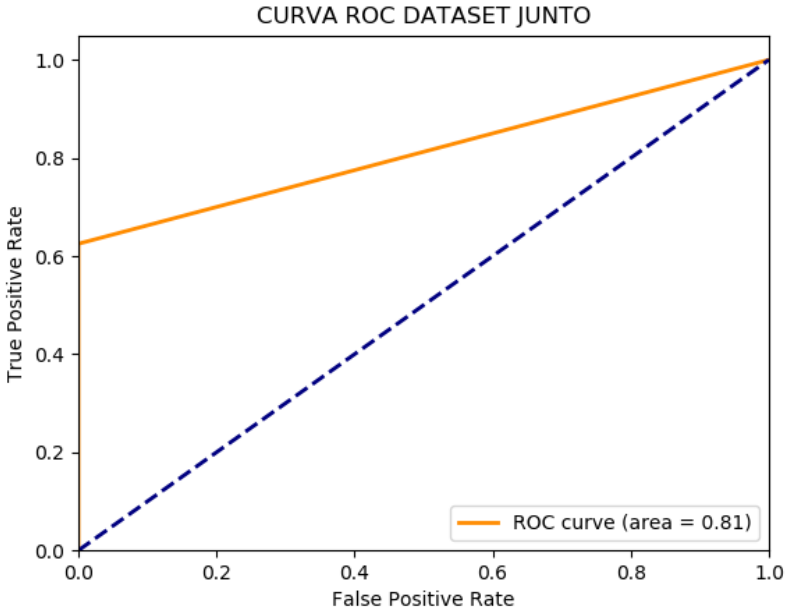
Fuente: Autoría propia.

Figura 48. Curva ROC para dataset PH2



Fuente: Autoría propia.

Figura 49. Curva ROC para dataset junto.



Fuente: Autoría propia.

Podemos visualizar que en nuestros tres casos la curva ROC representa la sensibilidad frente a la especificad para clasificar en cáncer maligno y benigno, viendo que nuestra red encontró una mayor cantidad de verdaderos positivos a falsos positivos, esto quiere decir que el umbral de discriminación puede clasificar las imágenes dermatoscópicas.

6.4 Validación

Los resultados de este modelo demuestran que se puede obtener un buen rendimiento mediante la propuesta de la arquitectura originalmente entrenado para la clasificación de imágenes del dataset. Esto implica que el modelo propuesto puede generalizar bien a una amplia variedad de problemas de clasificación, incluso con imágenes que no se encuentren los *dataset*.

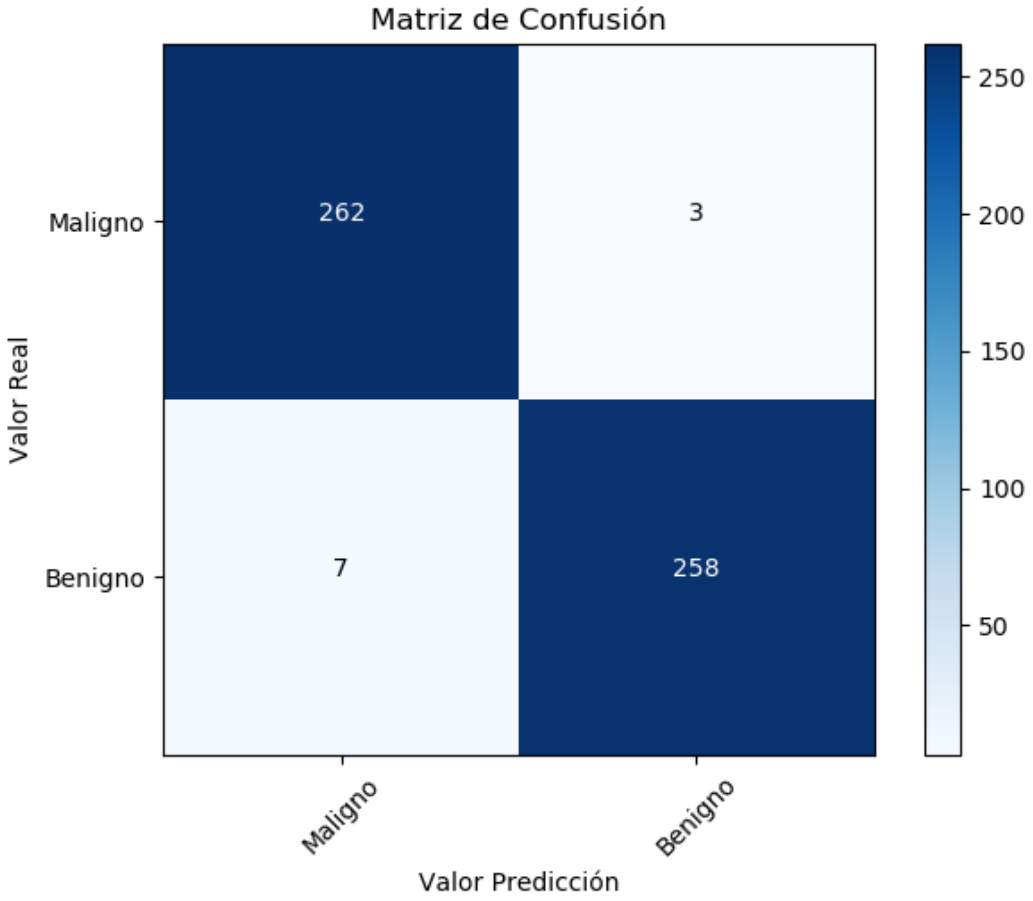
La matriz de confusión, son representadas en las siguientes tablas, donde muestran el rendimiento de clasificación del modelo en validación de los dataset. Se debe considerar que la evaluación del modelo se relación con el número de melanomas verdaderos clasificados como casos benignos. Tenga en cuenta que este es el escenario de caso incorrecto, donde el modelo no detecta un caso real de melanoma que podría terminar en una situación peligrosa y potencialmente mortal para el paciente.

Tabla 5. Matriz de confusión ISIC

	Maligno	Benigno
Maligno	Verdaderos Positivos(262)	Falsos Positivos(3)
Positivo	Falsos Negativos(7)	Verdaderos Negativos(258)

Fuente: Autoría propia.

Figura 50.Representación de la matriz de confusión ISIC



Fuente: Autoría propia

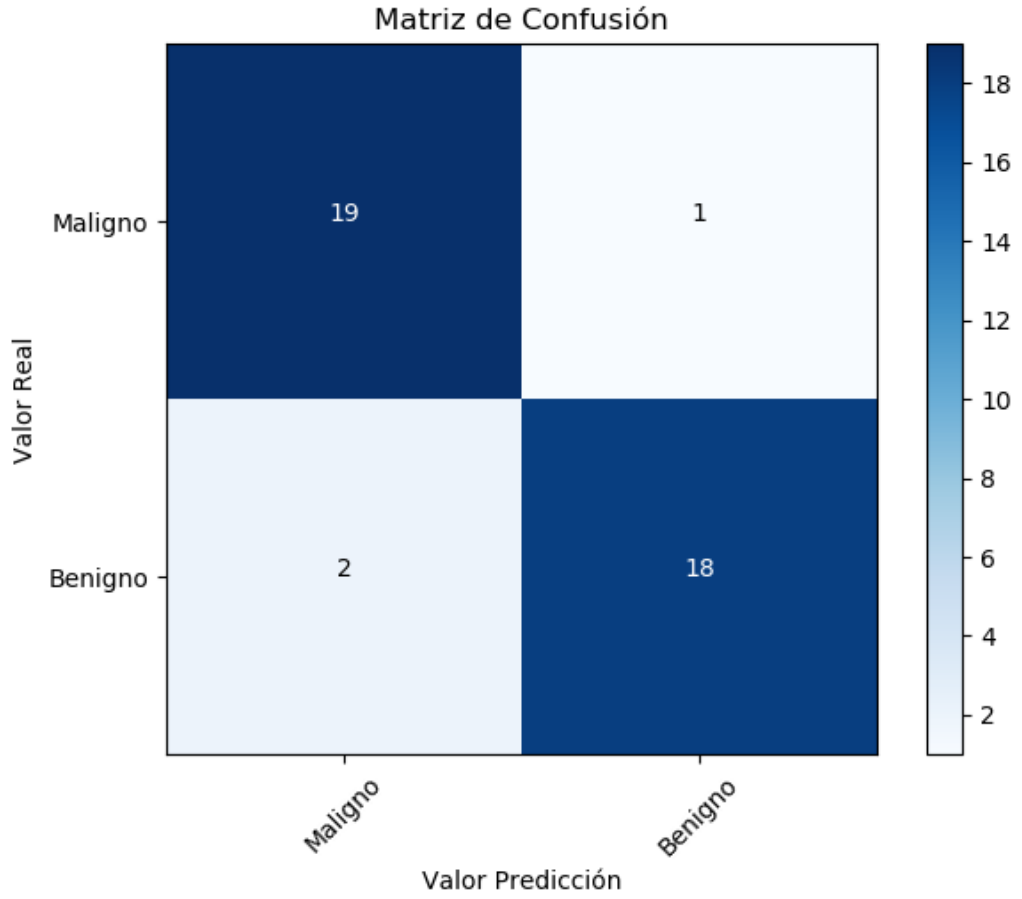
Se puede observar en la matriz de confusión que, se ingresaron 265 imágenes tipo maligno y 265 tipo benigno. Donde 262 imágenes fueron predichas correctamente y 3 fueron clasificadas en benignas, y para el caso de las benignas 7 fueron clasificadas en maligno y 258 en benigno.

Tabla 6. Matriz de confusión PH2

	Maligno	Benigno
Maligno	Verdaderos Positivos(19)	Falsos Positivos(1)
Benigno	Falsos Negativos(2)	Verdaderos Negativos(18)

Fuente: Autoría propia.

Figura 51.Representación de la matriz de confusión PH2



Fuente: Autoría propia.

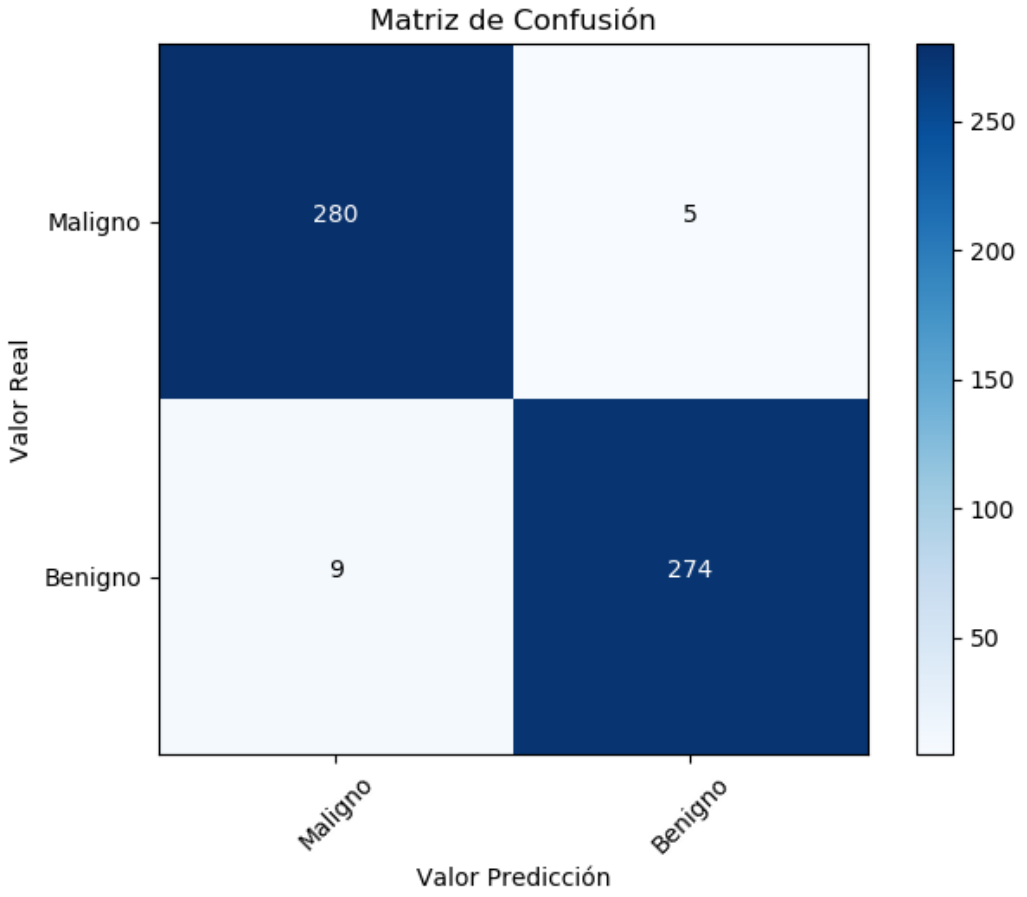
Se puede observar en la matriz de confusión que, se ingresaron 20 imágenes tipo maligno y 20 tipo benigno. Donde 19 imágenes fueron predichas correctamente y 1 fueron clasificadas en benignas, y para el caso de las benignas 2 fueron clasificadas en maligno y 18 en benigno.

Tabla 7. Matriz de confusión JUNTO

	Maligno	Benigno
Maligno	Verdaderos Positivos(280)	Falsos Positivos(5)
Benigno	Falsos Negativos(9)	Verdaderos Negativos(274)

Fuente: Autoría propia.

Figura 52.Representación de la matriz de confusión JUNTO



Fuente: Autoría propia.

Se puede observar en la matriz de confusión que, se ingresaron 285 imágenes tipo maligno y 285 tipo benigno. Donde 280 imágenes fueron predichas correctamente y 5 fueron clasificadas en benignas, y para el caso de las benignas 9 fueron clasificadas en maligno y 274 en benigno.

En la medicina la sensibilidad como se dijo anteriormente es el porcentaje de verdaderos positivos que vienen a ser las lesiones malignas que se identifican correctamente, mientras que la especificidad mide cuantas muestras predichas como verdaderas negativas son realmente así. Los resultados que se muestran en la tabla

9, son los mejores ya que explican la clasificación de la mejora de resultados de exactitud, sensibilidad y precisión. Estos resultados fueron comparados con el *dataset* ISIC de nuestro proyecto y con el proyecto de *Skin lesion detection from dermoscopic images using convolutional neural networks* [42], debido a que este proyecto utiliza el mismo dataset, con la arquitectura VGG-16.

Tabla 8.Comparación de las métricas de validación con otros proyectos

Proyecto	Exactitud	Perdida	Sensibilidad	Precisión
Proyecto Dataset ISIC	0.8887	0.1910	0.8322	0. 9735
Skin Lesion Detection CNN	0.8390	0.4723	0.8243	0.9523

Fuente: Autoría propia.

CONCLUSIONES

1. La red neuronal Deep Learning propuesta va a ayudar a los médicos y las personas para el diagnóstico de las lesiones de la piel. La herramienta ayuda a detectar en cáncer maligno o benigno, utilizando la red neuronal convolucional para clasificación temprana del melanoma.
2. Se analizó las arquitecturas para la clasificación de imágenes, que se utilizan en *Deep Learning,* como son *AlexNet, ZFNet, GoogleNet y VGGNet*. Posteriormente se comparó estas arquitecturas y las características que tiene cada una por separado.
3. Se realizó la propuesta, donde se indica las herramientas para el diseño y la implementación, donde usamos repositorios de internet para construir nuestro *dataset* y poder usar como datos de entrada en nuestra red neuronal.
4. Se analizó cada uno de los *dataset* por separado, para obtener el rendimiento de la red neuronal, cabe destacar que la comparación se realizó con el proyecto *Skin lesion detection from dermoscopic images using convolutional neural networks*, se utilizó el mismo *dataset* para la comparación y ver la mejora de nuestra red neuronal convolucional sobre este proyecto.

TRABAJOS FUTUROS

Según la herramienta *Keras*, se puede obtener mejores resultados ya no solo binarios, sino que también se puede clasificar en varios tipos de cáncer ya una vez detectado el melanoma. Utilizando el tipo *categorical* y *no binary* para que pueda cargar diferentes tipos de imágenes.

Para obtener una mejor clasificación se podría usar una segmentación antes del entrenamiento de imágenes para que la exactitud mejore al momento de validar las imágenes, para esto tendría que implementar una arquitectura *U-Net*.

RECOMENDACIONES

Se recomienda, que el uso de esta red neuronal convolucional esta aplicada a imágenes dermatoscópicas, puede variar el rendimiento de la red neuronal debido a imágenes que no sea de este tipo.

Se recomienda utilizar la versión de *Python* 3.5 para poder replicar el proyecto y utilizar las características de las redes neuronales propuesta en base a *Keras* y *TensorFlow*.

GLOSARIO DE TÉRMINOS

ALGORITMO: Se define como un grupo de un número finito de operaciones secuenciales de tal manera que ordenadas de manera lógica den solución a un problema determinado [52].

APRENDIZAJE PROFUNDO: Técnica de aprendizaje automático mediante el cual los ordenadores aprenderán mediante ejemplos de la misma forma en que lo hacen las personas, generalmente se realiza mediante redes neuronales [53].

CNN: Red neuronal organizada en capas que constan de una serie de nodos interconectados, las cuales pueden tener decenas o cientos de capas ocultas. La CNN funciona mediante la extracción de características directamente de las imágenes, en donde las características relevantes no se entrenan previamente; se aprenden mientras la red se entrena con una colección de imágenes [53].

CONVOLUCIÓN: Capa de neuronas en la red neuronal convolucional encargada del filtrado de imágenes a través de máscaras de red [56]

ÉPOCA: En redes neuronales es el periodo de tiempo en el cual se cumplen un determinado número de iteraciones, en el proceso de entrenamiento de este [57].

ITERACIÓN: Proceso de actualización de los pesos de la red de manera continua [59].

PADDING: Corresponde al relleno que se realiza en la imagen para que el filtro trabaje sin pérdida de información [61].

POOLING: Se refiere a la extracción de características en alguna región específica de la imagen, provocando una reducción de la misma, se puede asimilar su funcionamiento al de un filtro [62].

RED NEURONAL ARTIFICIAL: Modelo computacional basado en el funcionamiento de las redes neuronales naturales con el fin de emular el proceso de aprendizaje de los seres

vivos por parte de los ordenadores de tal manera que pueda procesar dicha información [57].

ReLu: Función de activación en redes neuronales convolucionales [56].

TASA DE APRENDIZAJE: Correspondo al factor que indica a la velocidad a la cual se realiza el proceso de entrenamiento en la red, un valor muy bajo producirá un proceso de aprendizaje que tomará más tiempo, un valor muy alto mostrará resultados del entrenamiento sub-óptimos [60].

VALIDACIÓN DE DATOS: Corresponde a los datos que se utilizan para la validación del entrenamiento [60].

DIAGNOSTICO: Recoger y analizar datos para evaluar problemas de diversa naturaleza [63].

MELANOMA: Tumor de las células pigmentarias que contienen melanina [64].

DERMATOSCOPIA: Se trata de una técnica sencilla, no invasiva, que mejora el diagnóstico clínico de las lesiones cutáneas, especialmente las pigmentadas, y se ha convertido en una técnica diagnóstica imprescindible en la consulta del dermatólogo. El instrumento empleado en esta técnica se denomina dermatoscopio [65].

DATA AUGMENTATION: Es una Técnica Data que, como indica su nombre, nos permite aumentar nuestro dataset de dos formas: Introduciendo perturbaciones en los datos originales o utilizando distintas distribuciones [66].

DISPENSARIZACIÓN: Proceso organizado, continuo y dinámico de evaluación e intervención planificada e integral, con un enfoque clínico, epidemiológico y social, del estado de salud de los individuos y familias [67].

NEVOS: Crecimiento benigno (no canceroso) en la piel formado por un racimo de melanocitos (células que elaboran melanina, que es la sustancia que le da color a la piel y

los ojos). Por lo general, un nevo es oscuro y puede sobresalir de la piel. También se llama lunar [68].

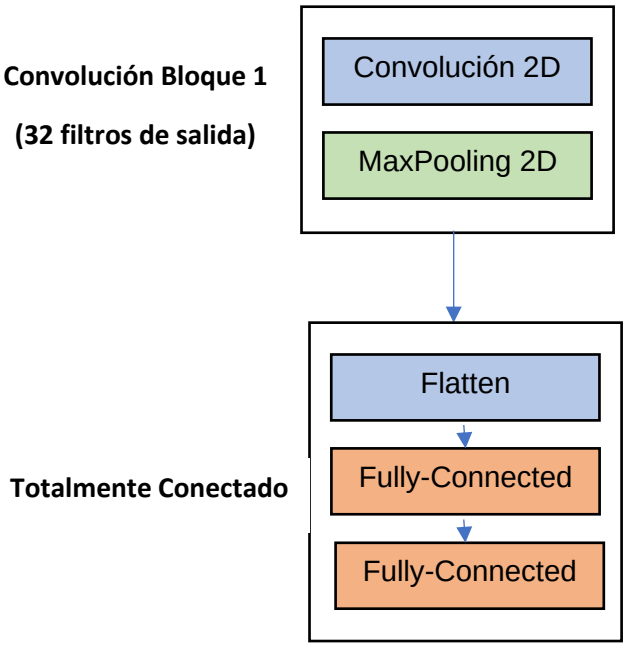
SIGMOIDAL: Función de activación en redes neuronales convolucionales [69].

FLATTEN: Es una capa que convierte la imagen de tres dimensiones a una sola [70].

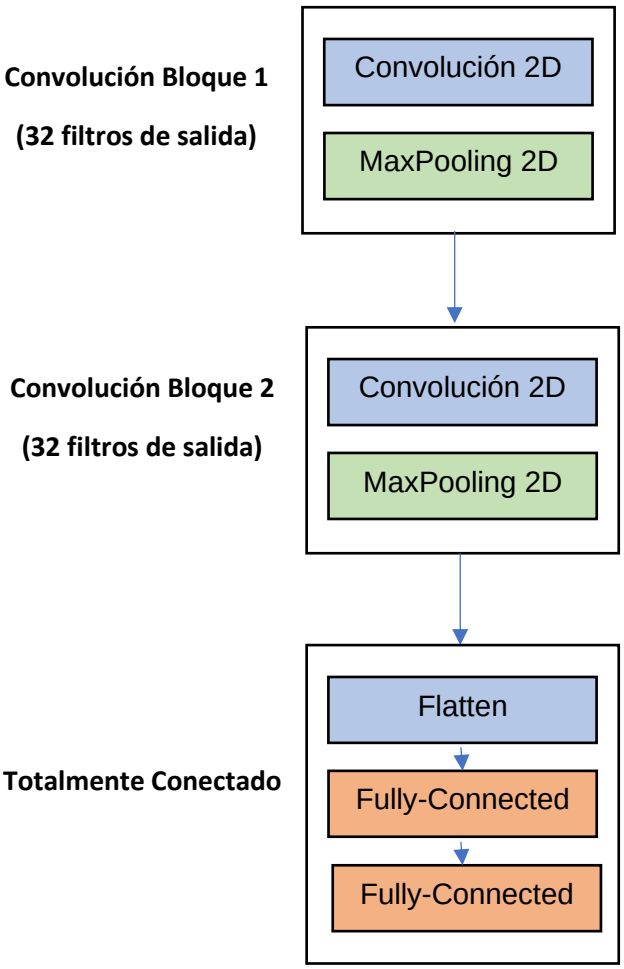
ANEXOS

Anexo A

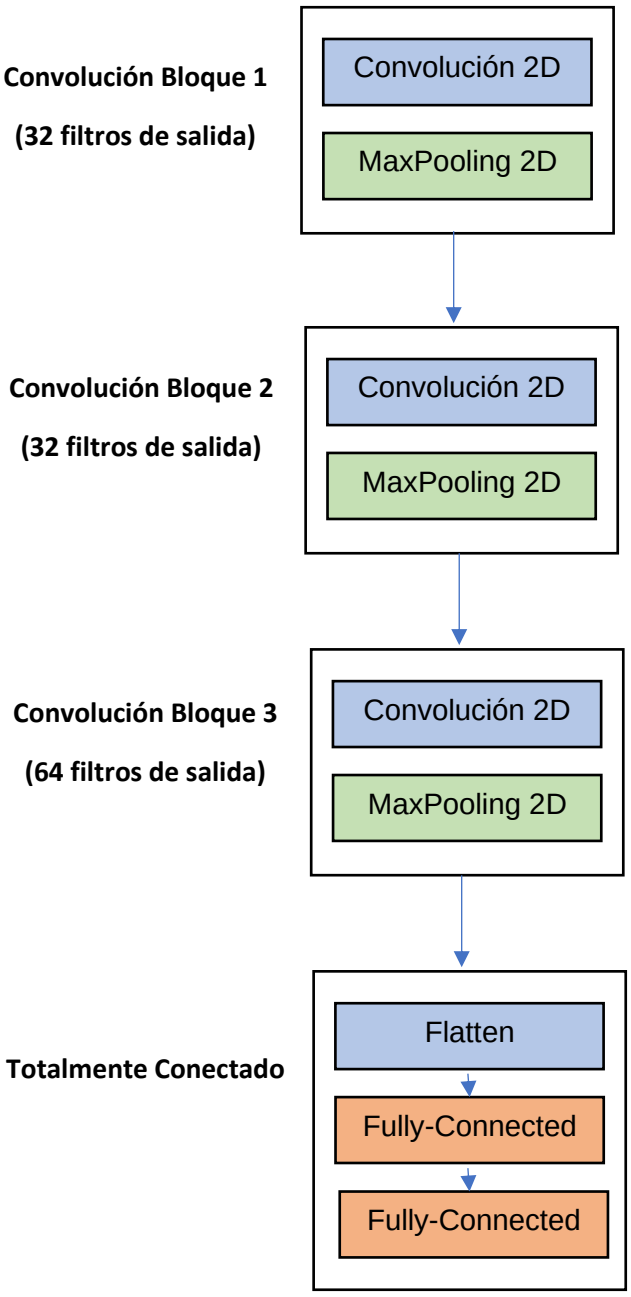
Arquitectura convolucional 1 capa



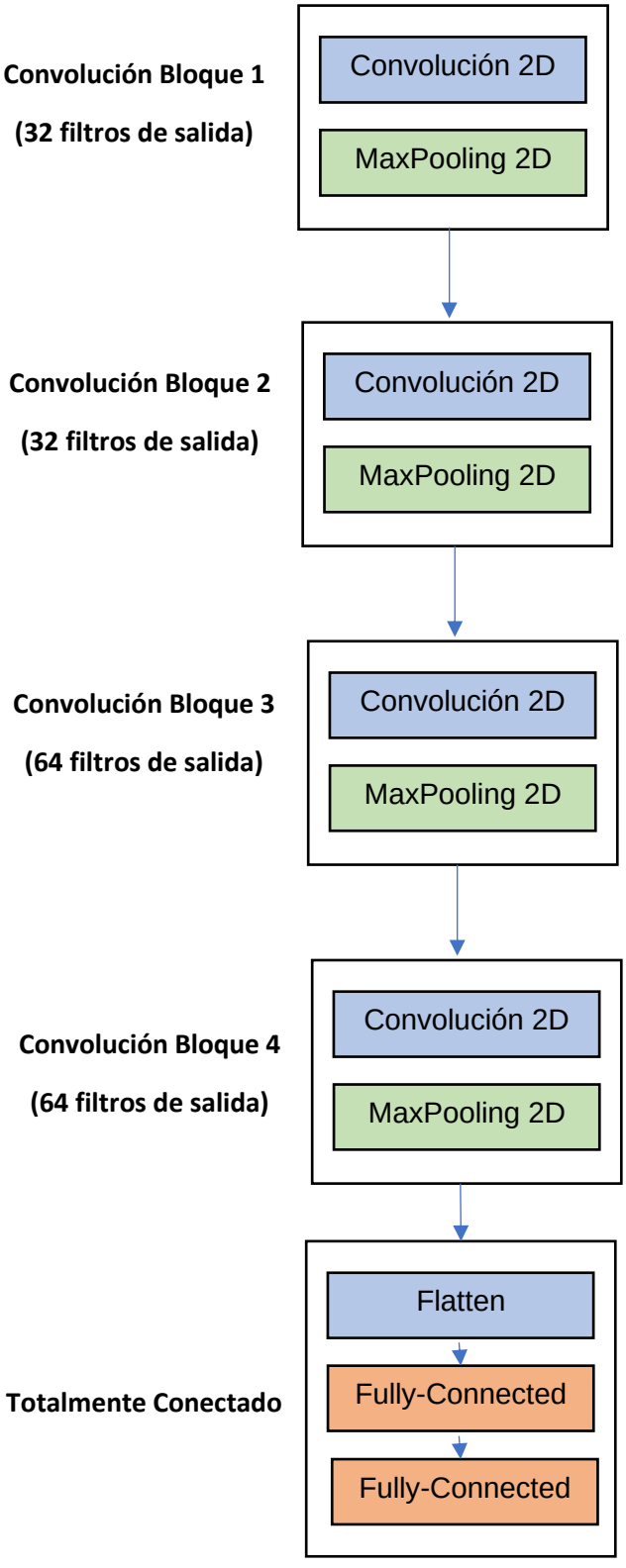
Arquitectura convolucional 2 capas



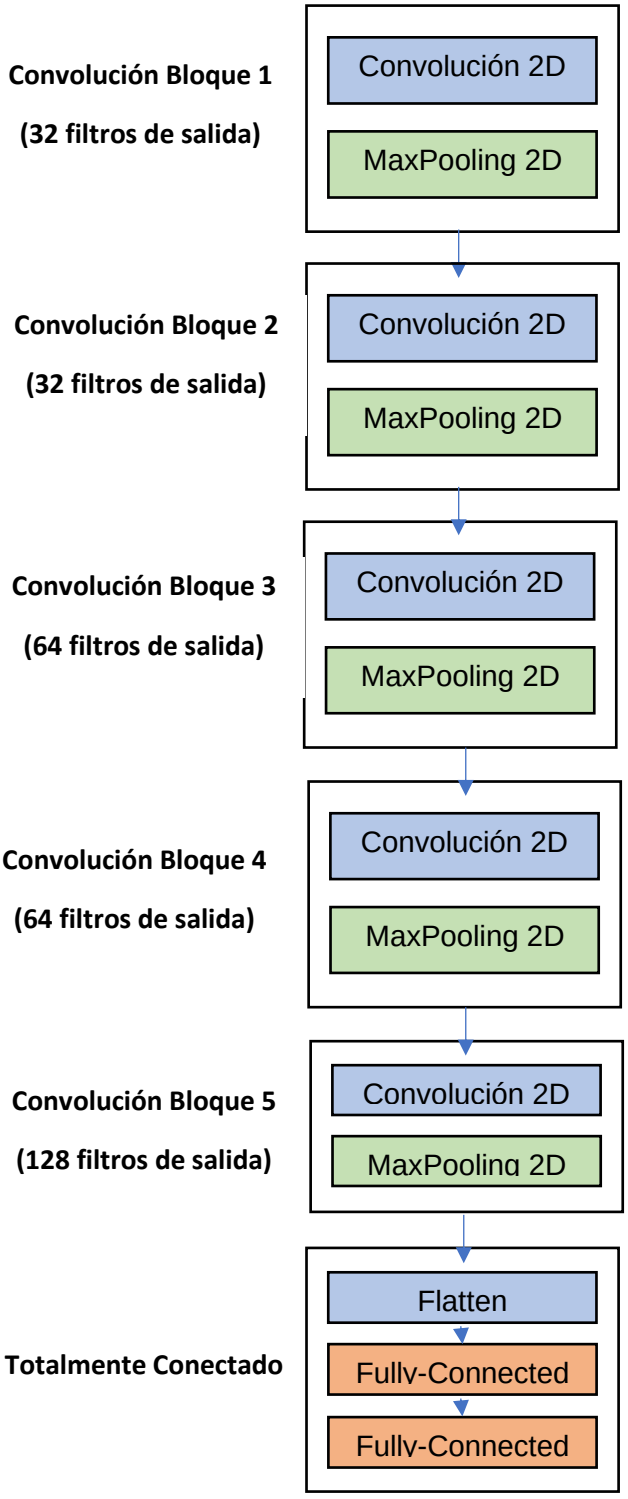
Arquitectura convolucional 3 capas



Arquitectura convolucional 4 capas

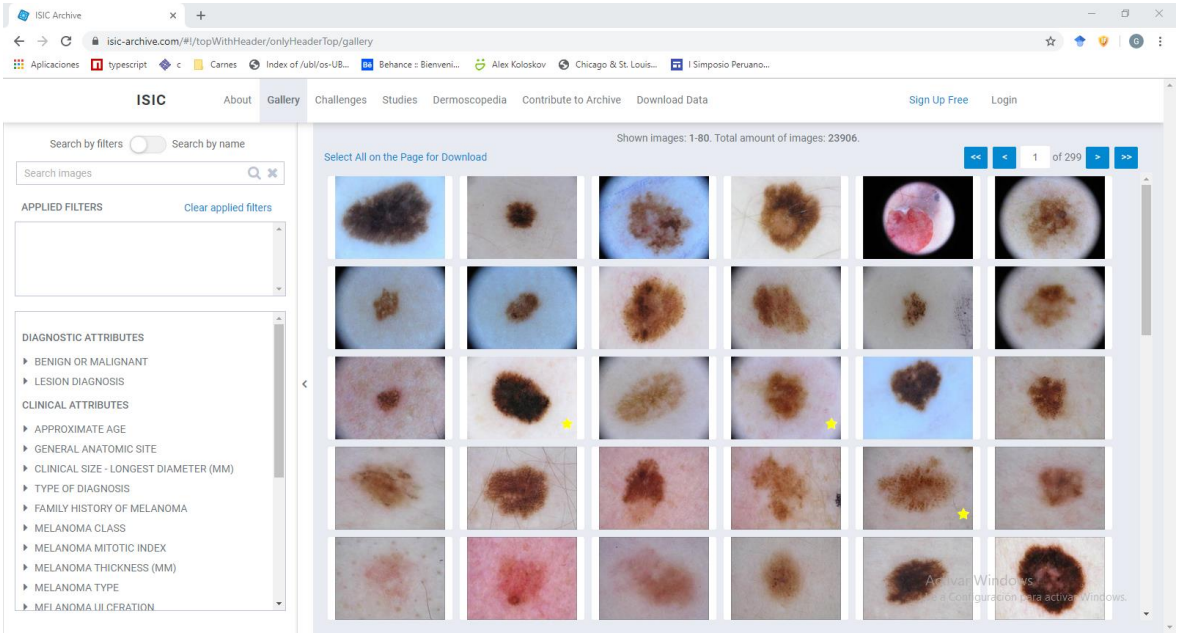


Arquitectura convolucional 5 capas

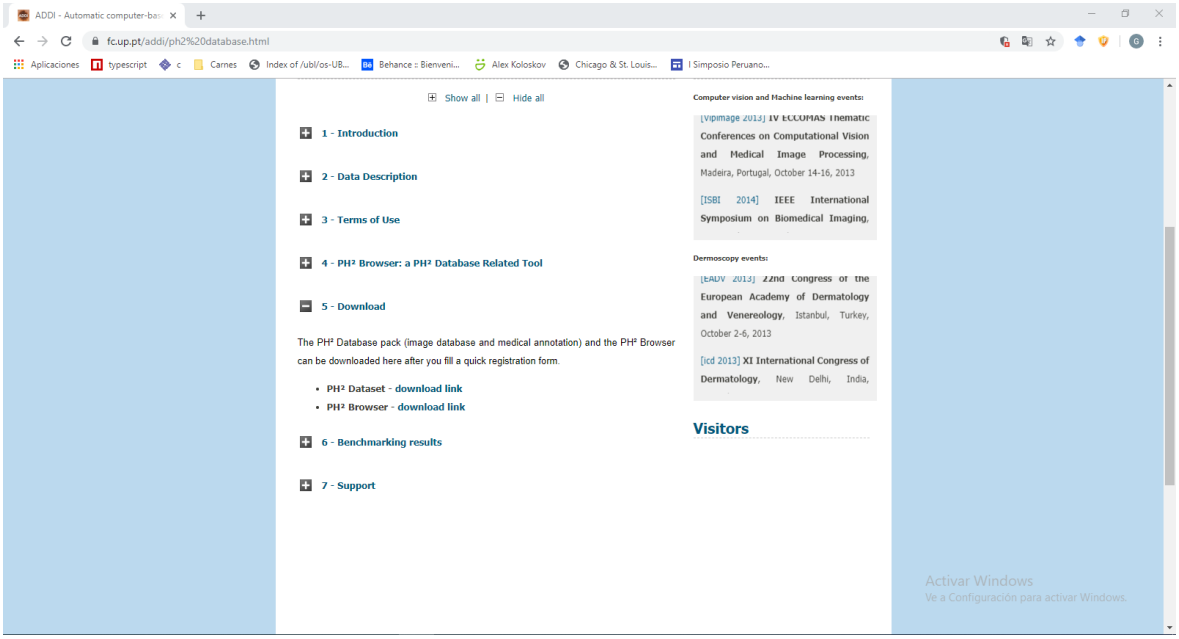


Anexo B

Captura de pantalla del repositorio ISIC



Captura de pantalla del repositorio PH2



Anexo C

Código fuente para el entrenamiento de la red neuronal convolucional propuesta

cancer.py - C:\Users\giorz\Documents\cancer\cancer.py (3.5.2)

File Edit Format Run Options Window Help

```
#modelo de cancer entrenado
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
from sklearn.metrics import classification_report
from skimage import io
from skimage.transform import resize
from keras.preprocessing.image import img_to_array, load_img
import cv2
import matplotlib.pyplot as plt
import glob

# Declaraci de metricas personalizadas
def specify(y_true, y_pred):
     y_pred_pos = K.round(K.clip(y_pred, 0, 1))
     y_pred_neg = 1 - y_pred_pos

     y_pos = K.round(K.clip(y_true, 0, 1))
     y_neg = 1 - y_pos

     tp = K.sum(y_pos * y_pred_pos)
     tn = K.sum(y_neg * y_pred_neg)
     fp = K.sum(y_neg * y_pred_pos)
     fn = K.sum(y_pos * y_pred_neg)

     specificity = fn/(fn+fp)
     return specificity

def sensitivity(y_true, y_pred):
     y_pred_pos = K.round(K.clip(y_pred, 0, 1))
     y_pred_neg = 1 - y_pred_pos

     y_pos = K.round(K.clip(y_true, 0, 1))
     y_neg = 1 - y_pos

     tp = K.sum(y_pos * y_pred_pos)
     tn = K.sum(y_neg * y_pred_neg)
     fp = K.sum(y_neg * y_pred_pos)
     fn = K.sum(y_pos * y_pred_neg)

     specificity = tp/(tp+fn)
     return specificity
```

```
def precision(y_true, y_pred):
    # Calculates the precision
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision
def recall(y_true, y_pred):
    # Calculates the recall
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall
# Declaración de la curva ROC
def plot_roc(y_test, y_score, title='ROC Curve'):
    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(title)
    plt.legend(loc="lower right")
    plt.savefig(title + '.png')
    plt.show()
# Declaración para convertir las imagenes de validación para curva roc
def load_test_images(dir_b, dir_m):
    X = []
    image_list_b = glob.glob(dir_b + '/*.png')
    n_images_b = len(image_list_b)
    print('Loading {} images of class benign:'.format(n_images_b))
    for i, imgfile in enumerate(image_list_b):
        print('Loading image {} of {}'.format(i+1, n_images_b))
        img = load_img(imgfile)
        X.append(cv2.resize(img_to_array(img), (150,150)))
    image_list_m = glob.glob(dir_m + '/*.png')
    n_images_m = len(image_list_m)
    print('Loading {} images of class benign:'.format(n_images_m))
    for i, imgfile in enumerate(image_list_m):
        print('Loading image {} of {}'.format(i+1, n_images_m))
        img = load_img(imgfile)
        X.append(cv2.resize(img_to_array(img), (150,150)))
    y = np.hstack((np.zeros(n_images_b), np.ones(n_images_m)))
    return np.array(X), y.reshape(len(y),1)
```

```
# dimensiones de nuestras imagenes.
img_width, img_height = 150, 150


train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
nb_train_samples = 1600
nb_validation_samples = 400
epochs = 50
batch_size = 16

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

model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

top_model_weights_path = 'cancer_model.h5'
#model.load_weights(top_model_weights_path)

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy',
                          recall,precision, specify, sensitivity])

# Configuración para el data augmentation para el entrenamiento
train_datagen = ImageDataGenerator( rescale=1. / 255, shear_range=0.2, zoom_rang

# Configuración para el data augmentation para el validación
## solo reescalando
test_datagen = ImageDataGenerator(rescale=1. / 255)
```

```
# Generador de imagenes de prueba en un directorio virtual
train_generator = train_datagen.flow_from_directory( train_data_dir, target_size

# Generador de imagenes de validación en un directorio virtual
validation_generator = test_datagen.flow_from_directory( validation_data_dir, ta

# Iniciar el entrenamiento y validación
model.fit_generator( train_generator, steps_per_epoch=nb_train_samples // batch_
    validation_steps=nb_validation_samples // batch_size)

# Guardar el modelo entrenado en formato json y h5
model.save('cancer_model.h5')
model_json = model.to_json()
with open("cancer.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights("pesosCancer.h5")

# Plotear la curva roc
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba >0.5)*1
print(classification_report(y_test,y_pred))
plot_roc(y_test, y_pred_proba, title='ROC Curve CNN from scratch')
```

Ln: 100 Col: 24

Captura de pantalla ejecución de la red neuronal convolucional en entrenamiento

Código fuente para la matriz ROC de las imágenes de validación

confusion_matriz.py - E:\CANCER PROYECTO\confusion_matriz.py (3.5.2)

File Edit Format Run Options Window Help

```
#modelo de cancer para prueba

import numpy as np
from matplotlib import pyplot
from keras.preprocessing.image import ImageDataGenerator
from keras.models import load_model,model_from_json
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import roc_curve, auc
from skimage import io
from skimage.transform import resize
from keras.preprocessing.image import img_to_array, load_img
import cv2
import matplotlib.pyplot as plt
import glob
import itertools
# Declarar la matriz de confusi
title = 'Matriz de Confusión'
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title=title,
                          cmap=pyplot.cm.Blues):
    pyplot.imshow(cm, interpolation='nearest', cmap=cmap)
    pyplot.title(title)
    pyplot.colorbar()
    tick_marks = np.arange(len(classes))
    pyplot.xticks(tick_marks, classes, rotation=45)
    pyplot.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        pyplot.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    pyplot.tight_layout()
    pyplot.ylabel('Valor Real')
    pyplot.xlabel('Valor Predicción')

    pyplot.figure()
```

```
def load_test_images(dir_b, dir_m):
    X = []
    image_list_b = glob.glob(dir_b + '/*.png')
    n_images_b = len(image_list_b)
    print('Loading {} images of class benign:'.format(n_images_b))
    for i, imgfile in enumerate(image_list_b):
        print('Loading image {} of {}'.format(i+1, n_images_b))
        img = load_img(imgfile)
        #img = io.imread(imgfile)
        print(cv2.resize(img_to_array(img), (150,150)))
        X.append(cv2.resize(img_to_array(img), (150,150)))
    image_list_m = glob.glob(dir_m + '/*.png')
    n_images_m = len(image_list_m)
    print('Loading {} images of class benign:'.format(n_images_m))
    for i, imgfile in enumerate(image_list_m):
        print('Loading image {} of {}'.format(i+1, n_images_m))
        img = load_img(imgfile)
        X.append(cv2.resize(img_to_array(img), (150,150)))
    y = np.hstack((np.zeros(n_images_b), np.ones(n_images_m)))
    print(np.array(X))
    return np.array(X), y.reshape(len(y),1)

X_test, y_test = load_test_images('data/test/benigno',
                                  'data/test/maligno')
# Cargar json and crear modelo
json_file = open('cancer.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# Cargar pesos dentro nuevo modelo
loaded_model.load_weights("pesosCancer.h5")
print("Loaded model from disk")

# Evaluar el modelo cargado en las imagenes de prueba
loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
score = loaded_model.evaluate(X_test, y_test, verbose=0)

# plot metricas
y_pred_proba = loaded_model.predict(X_test)
y_pred = (y_pred_proba >0.5)*1

class_names = ['Maligno', 'Benigno']
cnf_matrix = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cnf_matrix, classes=class_names,
                      title=title)

pyplot.show()
```

Ln: 47 Col: 4

BIBLIOGRAFÍA

[1] M. A. Arasi, El-Dahshan, A. El-Sayed, El-Horbaty, M. El-Sayed, Salem y M. Abdel-Badeeh, «Malignant Melanoma Detection Based on Machine Learning Techniques: A Survey,» *Egyptian Computer Science Journal,* nº 3, p. 40, 2016.

[2] X. Κατρίνη y C. Katrini, «Skin lesion classification using deep learning neural networks,» 2017.

[3] H. Chang, «Skin cancer reorganization and classification with deep neural network,» *Yale University School of Medicine,* p. 6, 2017.

[4] C. Sordo y C. Gutierrez, «Cáncer de piel y radiación solar: experiencia peruana en la prevención y detección temprana del cáncer de piel y melanoma,» scielosp, 2013. [En línea]. Available: www.scielosp.org. [Último acceso: 16 01 2020].

[5] A. Noori Hoshyar, Artist, *Automatic skin cancer detection system.* [Art]. UNIVERSITY OF TECHNOLOGY, SYDNEY.

[6] A. R. Lopez, X. Giro-i-Nieto, J. Burdick y O. Marques, Skin lesion classification from dermoscopic images using deep learning techniques, Innsbruck: IEEE, 2017.

[7] U. D. Center, «ISIC,» ISDIS Treasurer, 1979. [En línea]. Available: https://www.isic-archive.com. [Último acceso: 5 11 2019].

[8] P. FERREIRA, «ADDI Project,» 2012. [En línea]. Available: https://www.fc.up.pt/addi/ph2%20database.html. [Último acceso: 05 05 2019].

[9] J. Malvehy Guilera, S. Puig, C. Carrera, P. Aguilera, M. Gamboa y M. A. Jesús Silva, Comprender el melanoma y otros cánceres de piel, Barcelona: AMAT, 2015.

[10] L. Dzubow, J. C. Bystryn, D. Reintgen y D. Rigel, Cancer de Piel, España: Elsevier España, 2006.

[11] R. J. Larrondo Muguercia, L. Hernández García y R. Larrondo Lamadrid, «Consideraciones sobre la prevención del cáncer de piel,» studylib, 1996. [En línea]. Available: studylib.es. [Último acceso: 16 01 2020].

[12] R. Flórez López y R. Fernández Fernández, Las Redes Neuronales Artificiales, La Coruña: Netbiblo, 2008.

[13] I. SUÁREZ BÁRCENA, Utilidad diagnóstica en el glaucoma del análisis de las fibras retinianas mediante polarimetría láser asociado a la autoperimetría y a sistemas de inteligencia artificial (redes neuronales), Salamanca: Ediciones Universidad de Salamanca, 2011.

[14] C. J. P. Viana, Red Neuronal Artificial Aplicada Al Estudio de la Corrupción, EAE, 2012.

[15] J. Burdick, O. Marques, A. Romero-Lopez, G. Nieto, Xavier y J. Weinthal, The impact of segmentation on the accuracy and sensitivity of a melanoma classifier based on skin lesion images, Pittsburgh: scientific program SIIM, 2017.

[16] V. L. Chamorro Alvarado, Clasificación de tweets mediante modelos de aprendizaje supervisado, Madrid: Universidad Complutense de Madrid, 2018.

[17] S. Morales García, Identificación de actividad humana usando aprendizaje no supervisado en sistemas multimodales, Pereira: Universidad Tecnológica de Pereira, 2016.

[18] B. E. Baracaldo Doikova, Cooperación de múltiples agentes mediante aprendizaje por refuerzo y modelado de reglas sociales, Ecuador: Uniandes, 2015.

[19] I. Goodfellow, Y. Bengio y A. Courville, Deep Learning, London: MIT Press, 2016.

[20] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu y M. Pietikäinen, «Deep learning for generic object detection: A survey,» *arXiv preprint arXiv,* nº 1809.02165, p. 30, 2018.

[21] K. O'Shea y R. Nash, «An introduction to convolutional neural networks,» *arXiv preprint arXiv,* 2015.

[22] J. Torres, DEEP LEARNING Introducción práctica con Keras, Barcelona: Lulu.com, 2018.

[23] M. Abadi, TensorFlow: learning functions at scale, Acm Sigplan Notices, 2016.

[24] L. Lu, Y. Zheng, G. Carneiro y L. Yang, Deep Learning and Convolutional Neural Networks for Medical Image Computing: Precision Medicine, High Performance and Large-Scale Datasets, Princeton: Springer International Publishing, 2017.

[25] P. Loncomilla, «Deep learning: Redes convolucionales,» 2016. [En línea]. Available: https://ccc.inaoep.mx/~pgomez/deep/presentations/2016Loncomilla.pdf. [Último acceso: 1 Mayo 2019].

[26] W. Condori Quispe, Artist, *Introduction to Convolutional.* [Art]. Universidad Nacional de San Agustin, 2019.

[27] T. Doan, Convolutional Neural Network in classifying scanned documents, Alemania: GRIN Publishing, 2016.

[28] L. Cruz Beltrán y M. Acevedo Mosqueda, «Reconocimiento de Voz usando Redes Neuronales Artificiales Backpropagation y Coeficientes LPC,» Lindavista, México. D. F, 2008.

[29] M. Summerfield, Python 3, Anaya Multimedia, 2009.

[30] Á. Arias, Aprende a Programar Python, IT Campus Academy, 2014.

[31] G. Barrero , Artist, *Qsource en la calidad del software desarrollado en IBM RPG.* [Art]. Universidad Cesar Vallejo, 2018.

[32] J. Tang, Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence apps using TensorFlow Mobile and Lite for iOS, Android, and Raspberry Pi, Packt Publishing Ltd, 2018.

[33] R. Garreta y G. Moncecchi, Learning scikit-learn: Machine Learning in Python, Packt Publishing Ltd, 2013.

[34] L. V. Fausett, Fundamentals of Neural Networks: Architectures, Algorithms, and Applications, USA: Prentice-Hall, 1994.

[35] R. Atienza, Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more, Philippines: Packt Publishing Ltd, 2018.

[36] C. Antona Cortés, «HERRAMIENTAS MODERNAS EN REDES NEURONALES: LA LIBRERIA KERAS,» Madrid, 2017.

[37] A. Williams, Deep Learning with Keras: Introduction to Deep Learning with Keras, CreateSpace Independent Publishing Platform, 2017.

[38] A. Bhardwaj y J. Wei, Deep Learning Essential: Your hands-on guide to the fundamentals of deep learning and neural network modeling, Packt Publishing Ltd, 2018 .

[39] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, "O'Reilly Media, 2017.

[40] A. Moustafa, «Skin cancer Detection byTemperature VariationAnalysis,» *KTH Tecnology and Health,* p. 56, 2012.

[41] E. Singhal y S. Tiwari, «Skin Cancer Detection using Arificial Neural Network,» *International Journal of Advanced Research in Computer Science,* 2015.

[42] A. Romero Lopez, Artist, *Skin lesion detection from dermoscopic images using convolutional neural networks.* [Art]. Universitat Politécnica de Catalunya, 2017.

[43] K. Dana y E. Celebi, «ISIC 2018: Skin Lesion Analysis Towards Melanoma Detection,» Canfield Scientific, 2018. [En línea]. Available: https://challenge2018.isic-archive.com/. [Último acceso: 5 Diciembre 2019].

[44] R. Pino Diez, A. Gómez Gómez y N. Abajo Martínez, Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva, Oviedo: Editorial Servicios de Publicaciones Universidad de Oviedo, 2001.

[45] Y. Li y L. Shen, «Skin lesion analysis towards melanoma detection using deep learning network,» *Sensors,* p. 556, 2018.

[46] J. Esqueda Elizondo y L. Palafox Maestre, Fundamentos para el procesamiento de imágenes, California: Uabc, 2015.

[47] R. F. De Mello y M. A. Ponti, Machine Learning: A Practical Approach on the Statistical Learning Theory, Sao Paulo: Springer International Publishing, 2018.

[48] d. y. c. d. p. Aplicaciones de las redes de neuronas en supervisión, Fuente Aparicio, M J; Cano, T C, Caracas: Equinoccio, 1999.

[49] L. Mazare , «tensorflow-ocaml: OCaml bindings for TensorFlow,» 2018.

[50] L. Lanzarini, «Redes Neuronales Artificiales. Un Enfoque Practico,» *Journal of Computer Science & Technology,* vol. 4, nº 2, p. 122, 2004.

[51] A. Martelli y D. Ascher, Python Cookbook, "O'Reilly Media, 2013.