TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

TITULACIÓN INTEGRAL
TESIS

TEMA:
CLASIFICACIÓN DE LOS NIVELES DE DENSIDAD MAMOGRÁFICA USANDO DIFERENTES MODELOS DE REDES NEURONALES CONVOLUCIONALES

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO INFORMÁTICO

PRESENTA:
HUGO ADRIÁN OSORIO ORTIZ

ASESORA:
DRA. MARÍA GUADALUPE SÁNCHEZ CERVANTES

CD. GUZMÁN JALISCO, MÉXICO, MARZO DE 2018

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MEXICO

Instituto Tecnológico de Cd. Guzmán

Cd. Guzmán, Municipio de Zapotlán el Grande, Jal. 16/05/2018

ASUNTO: Liberación de Proyecto para Titulación integral

M.C. FAVIO REY LÚA MADRIGAL
JEFE DE LA DIVISION DE ESTUDIOS PROFESIONALES
Presente

Por este medio informo que ha sido liberado el siguiente proyecto para la Titulación integral:

Nombre del Egresado	**Hugo Adrián Osorio Ortiz**
Carrera:	Ingeniería Informática
No. De Control	**11290710**
Nombre del proyecto:	"Clasificación de los niveles de densidad mamográfica usando diferentes modelos de redes neuronales convolucionales"
Producto	TESIS

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

Atentamente,

M.C. Rubén Zepeda García
Jefe del Depto. Sistemas y Computación

SEP TecNM
I. T. DE CD. GUZMÁN
SISTEMAS Y COMPUTACIÓN

DRA. MARÍA GUADALUPE SÁNCHEZ CERVANTES	DR. DANIEL FAJARDO DELGADO	M.C. RAQUEL OCHOA ORNELAS
Nombre y Firma del Asesor	Nombre y Firma del Revisor	Nombre y Firma del Revisor

C.p. Archivo

DLAS/JMTS/RZG/*tps**

Dedicatoria

Para todas aquellas personas que traen luz a este mundo.

AGRADECIMIENTOS

A mi padre, mi madre y mi hermana, quiero agradecer su apoyo, atención, ánimo, orientación, consejo, cariño y demás cosas maravillosas que me han ayudado a continuar adelante.

A mis tíos por sus palabras de aliento, cariño y respaldo en momentos difíciles.

A mis maestros por su paciencia y sus enseñanzas. En especial a la Dra. María Guadalupe Sánchez Cervantes por considerarme para este proyecto y guiarme en el trayecto.

Al comité de tesis, la Dra. Raquel Ochoa Ornelas y al Dr. Daniel Fajardo Delgado por sus aportaciones y apoyo durante el proceso de revisión de la tesis.

Al Cuerpo Académico "Cómputo Paralelo/Distribuido y Control" con número de registro ITCGUZ-CA-7 del Programa para el Desarrollo Profesional Docente para el tipo superior (PRODEP), quien me apoyó económicamente para la realización del presente trabajo.

RESUMEN

El cáncer de mama es el segundo tipo más común de cáncer en todo el mundo y las estadísticas recientes muestran que una de cada diez mujeres lo desarrollará en algún momento de sus vidas. Es importante mencionar que cuando se detecta en una etapa temprana, el pronóstico es bueno, abriendo la puerta a Sistemas de Diagnóstico Asistido por Computadora que se enfocan en la prevención de esta enfermedad. La investigación médica para la prevención del cáncer de mama ha demostrado que la densidad mamaria es un fuerte indicador del riesgo de cáncer. La clasificación de densidad mamaria es un problema en el dominio de imágenes médicas que tiene como objetivo asignar la categoría BIRADS (I-IV) del Colegio Americano de Radiología a una mamografía, como indicación de la densidad de tejido.

En este trabajo se investigó la exactitud de la clasificación de la densidad mamaria de acuerdo a las categorías de BIRADS utilizando las redes neuronales convolucionales. Debido a la complejidad y consumo de tiempo que se experimentó al crear una red neuronal, se optó por utilizar la API TensorFlow.

Se probaron varios modelos de redes neuronales, Inception V3, MobileNet V0.50 y MobileNet V1. El modelo con el que se obtuvieron mejores resultados en exactitud al momento de clasificar la densidad mamaria de una imagen mamográfica fue el MobileNet. Éste modelo funciona con equipos de cómputo con recursos limitados, además está pre-entrenado con la base de datos ILSVRC-2012-CLS de ImageNet[1].

La base de datos con los que se realizaron los experimentos fue CBIS-DDSM (*Curated Breast Imaging Subset for DDSM*). Esta base de datos contiene imágenes con ROI (*Region of Interest*) en formato DICOM. Previo al entrenamiento de la red neuronal, se llevó a cabo una etapa de pre-procesamiento de la imagen con la finalidad de resaltar las zonas del tejido mamario para una mejor clasificación.

Este trabajo forma parte de las actividades que se contemplan en un proyecto de investigación financiado por PRODEP para el fortalecimiento del cuerpo académico "Cómputo Paralelo/Distribuido y control", cuyo registro es ITCGUZ-CA-7.

[1]El proyecto ImageNet es una gran base de datos visual diseñada para su uso en la investigación de software de reconocimiento de objetos visuales. Más de 14 millones de URL de imágenes han sido anotadas a mano por ImageNet para indicar qué objetos se representan. ImageNet contiene más de 20 mil categorías ambiguas.

TABLA DE CONTENIDO

PAGINA

Lista de Tablas

Tabla **Página**

Lista de Figuras

Figura **Página**

CAPÍTULO 1

INTRODUCCIÓN

1.1 Motivación

El cáncer de mama es uno de los cánceres más comunes en mujeres después del cáncer cervicouterino, y su tendencia en México va en incremento [1]. Existen diversos factores de riesgo que permiten la detección de estadios iniciales del cáncer de mama; entre los principales se encuentran las masas, calcificaciones y densidades mamarias visibles desde una imagen mamográfica [1].

La densidad mamaria se asocia con un mayor riesgo de cáncer de mama y dificulta la detección del cáncer mediante mamografía, pero se desconoce la influencia de la densidad sobre el riesgo según el método de detección de cáncer.

Las variaciones en la densidad mamaria se conocen como el patrón parenquimatoso de la mama. La grasa es radiológicamente transparente o translúcida (apariencia más oscura), y los tejidos conectivos y epiteliales son radiológicamente densos (apariencia más clara). Los estudios previos en general han respaldado una asociación entre los patrones parenquimatosos y el riesgo de cáncer de mama (mayor riesgo con densidades crecientes), pero ha habido una considerable heterogeneidad en las estimaciones de riesgo informadas.

En [2] determinan el nivel de riesgo de cáncer de mama asociado con la variación de las densidades mamarias mediante la clasificación cuantitativa de la densidad mamaria con métodos radiológicos convencionales y nuevos métodos asistidos por computadora. Los mamogramas se clasificaron en seis categorías de densidad, ya sea por radiólogos o

por mediciones asistidas por computadoras. Se encontraron aumentos estadísticamente significativos en el riesgo de cáncer de mama asociado con el aumento de la densidad mamaria tanto por radiólogos como por métodos asistidos por computadora.

Algunos estudios relacionan la densidad mamaria con los años del paciente, ya que son predicciones importantes de la exactitud de detección mamográfica.

El Colegio Americano de Radiología (ACR por sus siglas en inglés) presente un sistema para la clasificación de la densidad mamaria. Las categorías que presenta el ACR son las siguientes:

- Las mamas son casi enteramente grasas (Clasificación I).
- Se encuentran áreas dispersas de densidad fibroglandular (Clasificación II).
- Las mamas son heterogéneamente densas, lo que puede oscurecer las masas pequeñas (Clasificación III).
- Las mamas son extremadamente densas, lo que reduce la sensibilidad de la mamografía (Clasificación IV).

La aplicación de redes neuronales para el procesamiento de imágenes mamográficas es importante debido a que se puede entrenar el modelo de la red y llevar a cabo dicha clasificación automáticamente. Por otra parte, los nivel de densidad 2 y 3 (densidad dispersa y densidad heterogénea) son difíciles de diferenciar por los especialistas [3], de tal manera que ésta clasificación a través de las redes neuronales apoyará en realizar un diagnóstico fiable al especialista.

La presente tesis consiste en entrenar modelos de redes neuronales para realizar la clasificación del nivel de densidad mamaria y posteriormente realizar una comparativa de exactitud entre dichos modelos, para determinar cuál de ellos tiene mejor desempeño en equipos de cómputo donde la memoria RAM, la potencia máxima de cálculo y el consumo de energía son limitados.

Este trabajo forma parte de las actividades que se contemplan en un proyecto de investigación financiado por PRODEP para el fortalecimiento del cuerpo académico "Cómputo Paralelo/Distribuido y control", cuyo registro es ITCGUZ-CA-7.

1.2 Objetivos

1.2.1 Objetivo general

Realizar una comparativa de exactitud entre diferentes modelos de redes neuronales para la clasificación de los niveles de densidad mamaria.

1.2.2 Objetivos específicos

Los objetivos específicos planteados para este trabajo son los siguientes:

- Investigar los modelos de redes neuronales para llevar a cabo la clasificación de los niveles de densidad mamaria.
- Investigar la clasificación de los niveles de densidad mamaria.
- Realizar el pre-procesamiento de las imágenes a utilizar.
- Realizar el entrenamiento de la red neuronal.
- Realizar las pruebas experimentales.
- Realizar una comparativa de exactitud entre los modelos de redes neuronales.

1.3 Hipótesis

La hipótesis planteada para este trabajo es: el uso de la red neural convolucional MobileNets para la clasificación de los niveles de densidad mamaria, proporciona buena exactitud en la clasificación en comparación con otros tipos de redes convolucionales como Inception V3.

1.4 Estructura del trabajo

Esta tesis esta organizada de la siguiente manera:

- Capítulo 2 ESTADO DE ARTE: en este capítulo se presentan algunas investigaciones similares a esta tesis, donde se exponen diferentes enfoques para la clasificación de los niveles de densidad mamaria. Unas investigaciones abordan el análisis de la textura subyacente contenida dentro del tejido mamario mediante

técnicas de procesamiento de imágenes en pipeline, segmentación y extracción de características. Por otro lado, otras investigaciones abordan este tema utilizando el aprendizaje profundo mediante el uso de redes neuronales convolucionales, en especial la red AlexNet.

- Capítulo 3 MARCO TEÓRICO: en este capítulo se describe el cáncer de mama y el impacto psicosocial que conlleva el mismo. Además, se aborda el método de imagen por rayos X denominado mamografías como una forma de detección de este tipo de cáncer en etapas tempranas. Aunado a lo anterior, se presentan los conceptos de clasificación de la densidad mamaria y de redes neuronales, específicamente las convolucionales, como un aporte para la detección de este tipo de cáncer, lo que implica un procesamiento digital de las imágenes mamográficas.

- Capítulo 4 METODOLOGÍA: en este apartado se describe el procedimiento empleado para realizar la comparativa de exactitud entre los diferentes modelos de redes neuronales para la clasificación de los niveles de densidad mamaria. Se describe las fases del proceso, la población de datos con las que se llevó a cabo la experimentación, las variables involucradas y el análisis de los datos.

- Capítulo 5 RESULTADOS: en este capítulo se muestran los resultados obtenidos de exactitud en la clasificación de la densidad mamaria a través de los modelos de redes neuronales Inception V3 y MobileNet en sus versiones 0.50 y 1. Los resultados muestran que el modelo MobileNet proporciona muy buenos resultados en exactitud.

- Capítulo 6 CONCLUSIONES: finalmente en este capítulo, se presentan las conclusiones de los resultados obtenidos al implementar los modelos Inception y MobileNet para la clasificación de la densidad mamaria. Con estos resultados se comprueba la hipótesis planteada en la presente tesis, en el que la red convolucional MobileNets proporciona buena exactitud en la clasificación, en comparación con la red neuronal Inception V3.

CAPÍTULO 2

ESTADO DE ARTE

La densidad mamaria es una medida usada para describir la porción fibroglandular del tejido mamario de la mujer en una mamografía. Las mujeres con mamas extremadamente densas están en riesgo de 4 a 6 veces más que mujeres con mamas grasas. Comparando mujeres con mamas heterogéneamente densas contra mujeres con una densidad mamaria promedio, el riesgo de desarrollar cáncer aumenta 1.2 veces; asimismo, el riesgo es aproximadamente 2.1 veces mayor cuando se compara a una mujer con mamas extremadamente densas con mujeres con densidad media [3].

El Diagnóstico Asistido por Computadora (CAD) en imágenes médicas ha florecido en las pasadas décadas. Los nuevos avances en software y hardware de computadora y la mejora de la calidad de las imágenes tomadas por escáneres han permitido este progreso. Las principales motivaciones para CAD han sido: reducir los errores, permitir una medición más eficiente e interpretación de imágenes. Las dificultades con la recopilación de datos han obstaculizado seriamente la investigación CAD [4].

La mejora de la precisión y la coherencia de la evaluación de la densidad mamaria es una necesidad clínica no satisfecha, mientras que está surgiendo una evaluación automatizada de la densidad cuantitativa y está disponible en instalaciones limitadas [3].

La precisión de la mayoría de los algoritmos de clasificación convencionales, por ejemplo, Máquinas de Vectores Soporte (MVS), se basa en una sólida ingeniería de características, que requiere un conocimiento experto previo de los datos y un duro proceso de elaboración manual para crear características descriptivas. Por el contrario,

el aprendizaje profundo puede extraer características de manera automática y directa de los datos originales. El aprendizaje profundo se ha probado para el diagnóstico de lesiones en varios escenarios, como la diferenciación entre masas benignas y malignas; discriminación de masas, microcalcificaciones y su combinación; entre tumor / masa y tejido normal; o entre tres clases de tejido benigno, maligno y normal [3].

En la evaluación clínica de la densidad mamaria, las dos categorías más difíciles de distinguir son "densidad dispersa" y "heterogéneamente densa" [3]. De las cuatro categorías de densidad mamaria BI-RADS, es bastante fácil distinguir las categorías de "casi completamente graso" (I) y "extremadamente denso" (IV) por evaluación visual, donde los radiólogos son realistas y cómodos para tomar decisiones sin necesidad de asistencia. Sin embargo, es un desafío para los radiólogos en la asignación de una clasificación consistente de "densidad dispersa" (II) frente a "heterogéneamente densa" (III) [3].

En [5] se toma un enfoque para la clasificación de densidad mamaria que consiste en realizar una discriminación implementada de la densidad que se basa en analizar la textura subyacente contenida dentro del tejido mamario evidente en un mamograma digital mediante técnicas procesamiento de imágenes en pipeline, segmentación y extracción de características adoptando una variación en la agregación de bootstrap ("bagging"), logrando una tasa de reconocimiento de 71-4% para BI-RADS.

El planteamiento en [6] se basa en acuerdo inter-intraobservador también aplicado para una clasificación según BI-RADS, donde con el acuerdo interobservador se logra un coeficiente de 0.71 y en el acuerdo intraobservador un 0.54.

En [7] los autores experimentan con dos enfoques de clasificación automática, la red Krizhevsky (AlexNet) entrenada utilizando la base de datos de imágenes ImageNet, y la arquitectura HT-L3. Los autores concluyen que el enfoque de aprendizaje profundo y su aplicación mediante redes neuronales convolucionales proveen resultados satisfactorios para la clasificación de densidad mamaria.

CAPÍTULO 3

MARCO TEÓRICO

En este capítulo se presentan algunas consideraciones teóricas del cáncer de mamá y su clasificación, las redes neuronales y el procesamiento digital de imágenes, teoría que sustenta el desarrollo de la tesis.

El cáncer de mama es uno de los cánceres mas presentes en mujeres tanto en países desarrollados como en países en vías de desarrollo, este último grupo demuestra un incremento en la incidencia de casos de cáncer de mama debido a factores como el incremento en la expectativa de vida, la urbanización y la adopción de estilos de vida occidental [8]. También se muestra que los bajos niveles de conciencia dentro de la población respecto al riesgo de la enfermedad, una infraestructura médica pobre y la inadecuada experiencia del personal médico conllevan a una detección del cáncer en etapa avanzada [9] donde se vuelve difícil tratar la enfermedad. Por lo tanto, la detección en etapas tempranas es la herramienta más importante para mejorar el control y el tratamiento del cáncer de mama.

Según la Organización Mundial de la Salud el proceso de cribado para la detección de cáncer es una tarea compleja que resulta solo rentable cuando se aplica mediante programas de alto nivel dirigidos a toda la población en riesgo, ofreciéndole a todo el nivel de servicios de cribado, diagnóstico y tratamiento. Programas como éstos son los que ofrecen mastografías (o mamografías).

La mamografía es el método más efectivo probado para identificar la posible presencia de cáncer, pero son programas solo rentables en países con una buena infraestructura en salud y que pueden pagar a largo plazo este tipo de programas [8]. Por lo tanto,

para países en vías de desarrollo la aplicación de este tipo de programas resulta muy complicado o es de poca calidad, debido a las carencias que se tiene en la infraestructura de equipo médico, siendo la auto-exploración la medida más empleada por la limitación de recursos.

En México la gran diversidad de ingreso entre los habitantes, donde mas de la mitad de la población se encuentra bajo la línea de pobreza provocando que el acceso a servicios médicos se base en el nivel económico y puede variar, haciendo evidente que marcan una diferencia en la incidencia y mortalidad debido al cáncer [9].

Los primeros programas para la detección de cáncer mediante la toma de mamografías con intención de reducir la etapa de diagnóstico aparecieron en 1990. En 2009 se contaban con aproximadamente 509 unidades que realizaban el proceso de detección mediante mamografía al servicio de 12 millones de mujeres de edad de 40 o más, con un estimado de un total de 3,352,000 estudios realizados anualmente, y promedio de 30 mamografías tomadas diariamente [9].

El diagnosis del cáncer de mama en etapas es aún un problema con más de la mitad de la población sin acceso a servicios médicos privados siendo del 5% la gente que puede acceder a este tipo de servicios. Los servicios en hospitales públicos son pobres debido a que deben lidiar con enfermedades avanzadas, presupuestos limitados, tratamientos costosos y bajas probabilidades de supervivencia, además de contar con un limitado número de especialistas hábiles y entrenados [9].

La mamografía es un procedimiento hecho con una máquina que toma rayos X a la mama, como se muestra en la Figura 3.1.

Las mamografías se pueden usar para buscar el cáncer de mama en mujeres que no presentan signos o síntomas de la enfermedad. Este tipo de mamografía se llama mamografía selectiva de detección. Por lo general, las mamografías de detección requieren dos o más radiografías o imágenes de cada mama. Con frecuencia, las radiografías hacen posible que se detecten tumores que no se pueden palpar. Las mamografías pueden usarse también para buscar cáncer de mama después de haberse encontrado un abultamiento u otro signo o síntoma de la enfermedad. Este tipo de mamografía se llama mamografía de diagnóstico [10].

Los resultados positivos falsos ocurren cuando los radiólogos ven una anomalía (que posiblemente es “positiva”) en una mamografía, pero en realidad no hay cáncer presente. Los resultados positivos falsos de mamografía pueden conducir a ansiedad y a otras formas de angustia psicológica en las mujeres afectadas.

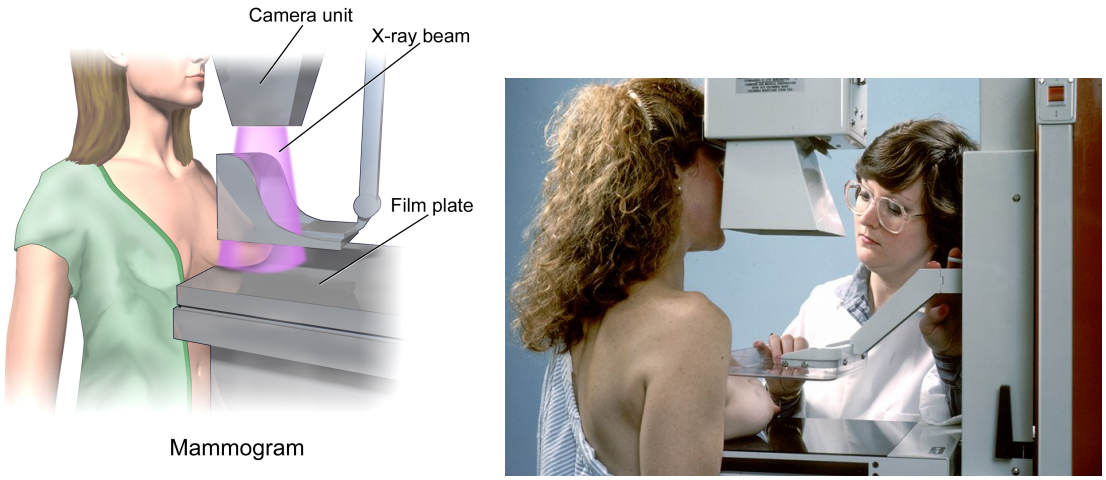


Figura 3.1: toma de mamografía.

Fuente: BruceBlaus (2014). Recuperado de: https://bit.ly/2vVa2Mf

Las pruebas adicionales que se requieren para descartar la presencia de cáncer pueden también ser costosas, requerir tiempo y causar molestias físicas. Los resultados positivos falsos son más comunes en mujeres más jóvenes, en mujeres con mamas densas, en mujeres que han tenido biopsias anteriores de mama, en mujeres con antecedentes familiares de cáncer de mama y en mujeres que toman estrógeno.

La posibilidad de tener un resultado positivo falso aumenta con el número de mamografías que haya tenido la mujer. Más de 50% de las mujeres que se hacen exámenes de detección anualmente en 10 años en los Estados Unidos experimentarán un resultado positivo falso, y como consecuencia muchas de estas mujeres tendrán una biopsia [10].

Las mamografías selectivas de detección pueden encontrar cánceres y casos de Carcinoma Ductal In Situ (CDIS) un tumor no invasivo en el que células anormales que pueden hacerse cancerosas se acumulan en el revestimiento de los conductos de la mama, que necesitan tratarse. Sin embargo, estas mamografías pueden encontrar también casos de CDIS y cánceres pequeños que nunca causarían síntomas ni pondrían en peligro la vida de la mujer. Este fenómeno se llama "sobrediagnóstico" [10].

En los exámenes de detección de cáncer, un resultado negativo significa que no hay presente una anomalía. Los resultados negativos falsos ocurren cuando las mamografías parecen normales aun cuando el cáncer de mama está presente. En general, las mamografías de detección fallan en cerca de 20% de los cánceres de mama que están presentes cuando se hace el examen. Los resultados negativos falsos pueden conducir a retrasos en el tratamiento y a un sentido falso de seguridad en las mujeres afectadas [10].

Una causa de los resultados negativos falsos es la alta densidad mamaria. Las mamas están formadas tanto por tejido denso (es decir, tejido glandular y tejido conjuntivo, lo cual en conjunto se conoce como tejido fibroglandular) como por tejido adiposo (graso). El tejido adiposo aparece de color oscuro en una mamografía, mientras que el tejido fibroglandular aparece como zonas blancas. Ya que el tejido fibroglandular y los tumores tienen una densidad semejante, puede ser más difícil detectar los tumores en mujeres con mamas más densas [10].

3.1 Clasificación de la densidad mamaria

Dos clasificaciones para la densidad mamaria se utilizan más frecuentemente: el patrón de parénquima de Wolfe y el porcentaje de densidad mamaria. Éste último, ha sido reportada como la más válida y es el enfoque mas comúnmente usado para determinar la densidad mamaria [1]. El sistema de categorización más extensamente conocido para la densidad del tejido mamario es el Sistema de Reporte para Datos e Imágenes de mama (BI-RADS por sus siglas en inglés) del Colegio Americano de Radiología (ACR por sus siglas en inglés), una medida de cuatro categorías de densidad que es componente de un léxico mamográfico estandarizado con la intención de promover una forma de criterio universal para informes en mamografía [6]. Las categorías de BI-RADS en su cuarta edición son:

- Clasificación I = 0-25%.
- Clasificación II = 26-50%.
- Clasificación III = 51-75%.
- Clasificación IV = 76-100%.

3.2 Redes Neuronales

La inteligencia artificial nació en los años 50 como un esfuerzo para automatizar inteligentemente tareas cotidianas realizadas por los humanos. Sin embargo, se limitaba a problemas bien definidos, por ejemplo: problemas lógicos y cómo jugar al ajedrez, siendo incapaz de resolver problemas cuyos parámetros pueden variar. Debido a estas limitantes se creó el aprendizaje de máquina para responder a la pregunta de ¿Como le decimos

a la computadora que hacer para realizar las cosas por ella misma?. Los algoritmos de aprendizaje de máquina se basan en sacar reglas y requiere de tres puntos:

- Datos de entrada
- Ejemplos de lo que se espera de la salida
- Una manera de medir si el algoritmo está realizando su tarea correctamente

El aprendizaje de máquina transforma sus datos de entrada en salidas de valor, un proceso que es "aprendido" del estar expuesto de entradas y salidas a ejemplos. El aprendizaje profundo es un subcampo del aprendizaje de máquina: una nueva forma de aprender representaciones de los datos que pone énfasis en aprender de capas sucesivas de representaciones cada vez más significativas. En la Figura 3.2 se muestra una representación de los tres conceptos mencionados anteriormente.

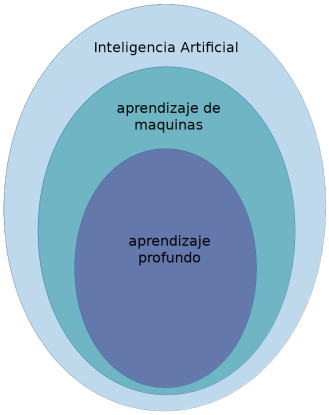


Figura 3.2: inteligencia artificial, aprendizaje de máquina, aprendizaje profundo.

La Figura 3.3 muestra un enfoque típico de aprendizaje de máquina basado en [3].

Los algoritmos de aprendizaje de máquina ayudan a recopilar datos de diversas fuentes, transformar conjuntos de datos enriquecidos y ayudar a tomar medidas inteligentes basadas en los resultados proporcionados.

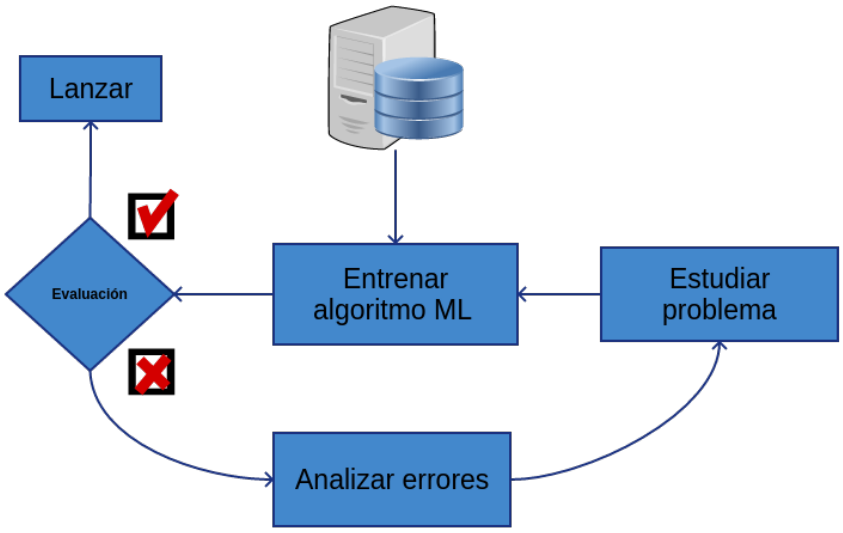


Figura 3.3: Enfoque típico de aprendizaje de máquina.

Los algoritmos de aprendizaje de máquina están diseñados para ser eficientes y precisos y para proporcionar un aprendizaje general para hacer lo siguiente [11]:

- Tratar problemas a gran escala.
- Realizar predicciones precisas.
- Manejar una variedad de problemas de aprendizaje.
- Aprendizaje que puede derivar y las condiciones bajo las cuales se pueden aprender.

Las redes neuronales representan una metáfora cerebral para el procesamiento de la información. Estos modelos tienen una inspiración biológica en lugar de una réplica exacta de cómo funciona realmente el cerebro. Se ha demostrado que las redes neuronales son sistemas muy prometedores en muchas aplicaciones de pronóstico y aplicaciones de clasificación empresarial debido a su capacidad para aprender de los datos [11].

En la década de 1940, el neurofisiólogo Warren McCulloch y el matemático Walter Pitts diseñaron la primera implementación matemática de una neurona artificial que combina los fundamentos de la neurociencia con operaciones matemáticas. En ese momento, el cerebro humano se estaba estudiando en gran medida para comprender sus comportamientos ocultos y misteriosos, pero dentro del campo de la neurociencia. Se

sabía que la estructura neuronal natural tenía un núcleo, las dendritas recibían señales entrantes de otras neuronas y un axón activaba una señal a otras neuronas, como se muestra en la Figura 3.4.

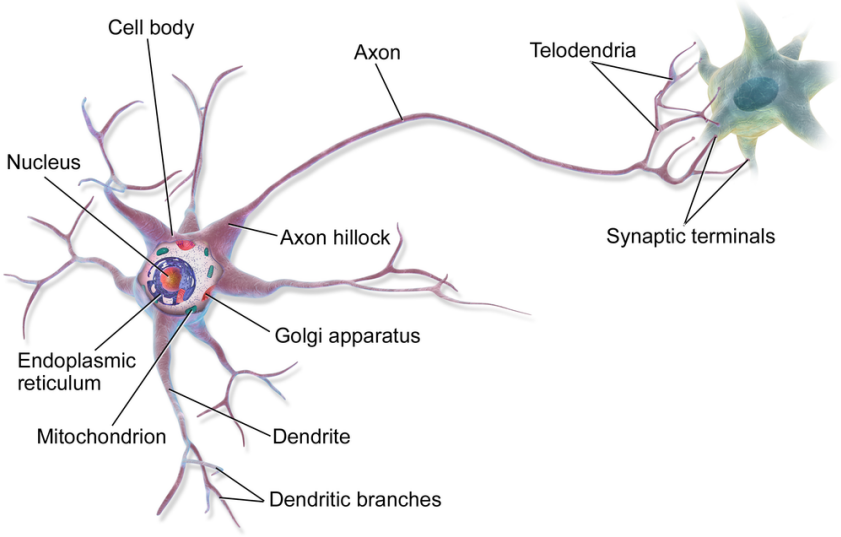


Figura 3.4: neurona biológica.

Fuente: Blausen (2013). Recuperado de: https://en.wikipedia.org/wiki/Neuron#/media/File:Blausen_0657_MultipolarNeuron.png

Una neurona se conecta a una serie de otras neuronas. Éstas neuronas a su vez se conectan a otras series de neuronas y así sucesivamente, por lo que es una estructura altamente interconectada [12].

La novedad de McCulloch y Pitts fue el componente matemático incluido en el modelo de neuronas, suponiendo que una neurona es un procesador simple que suma todas las señales entrantes y activa una nueva señal para otras neuronas, ver Figura 3.5 [12].

Las redes neuronales es una forma de lograr el aprendizaje por capas. Existen diferentes tipos de redes neuronales, entre los que se encuentran:

- Perceptrón de capas múltiples.
- Autoencoder.
- Probabilista.
- Tiempo retardado.
- Convolucional.

- Jerarquica.
- Estocástica.
- Asociativa.
- Neocognitron.
- Neuro-fuzzy.

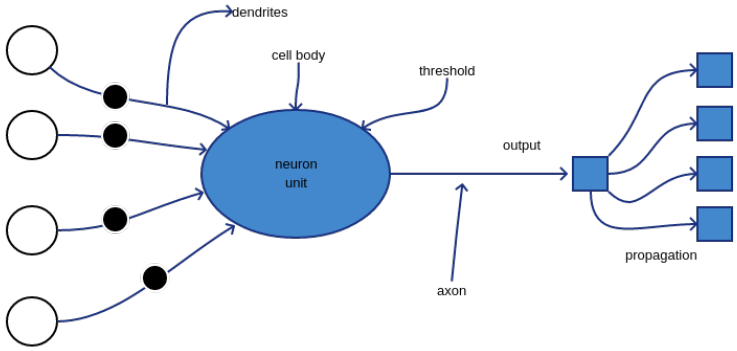


Figura 3.5: neurona artificial.

Fuente: medium. Recuperado de: https://bit.ly/2I0e2Qg

El Perceptrón es una de las arquitecturas de Redes Neuronales Artificiales (ANN por sus siglas en inglés) más simples, inventada en 1957 por Frank Rosenblatt. Se basa en una neurona artificial ligeramente diferente, las entradas y salidas ahora son números (en lugar de valores binarios on/off) y cada conexión de entrada está asociada con un peso, ver Figura 3.6.

Las Redes Neuronales Artificiales (ver Figura 3.7) son el núcleo del aprendizaje profundo. Son versátiles, potentes y escalables, lo que los hace ideales para abordar grandes y complejas tareas de aprendizaje de máquinas, como clasificar miles de millones de imágenes (por ejemplo, Google Images), potenciar los servicios de reconocimiento de voz (por ejemplo, Siri de Apple), recomendar los mejores videos para ver a cientos de millones de usuarios todos los días (por ejemplo, YouTube) [13].

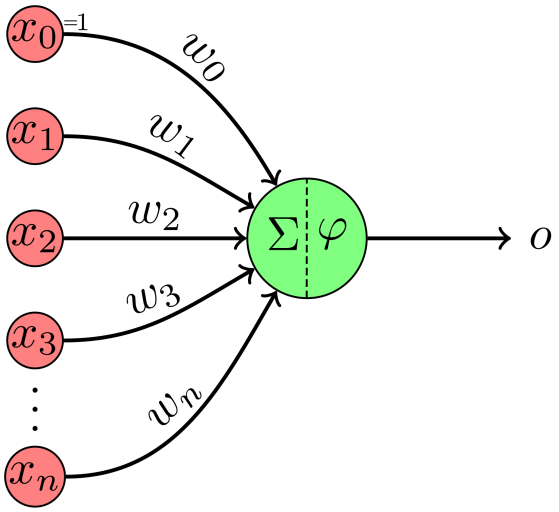


Figura 3.6: perceptrón.

Fuente: Towards Data Science (2017). Recuperado de https://bit.ly/2rahcpZ

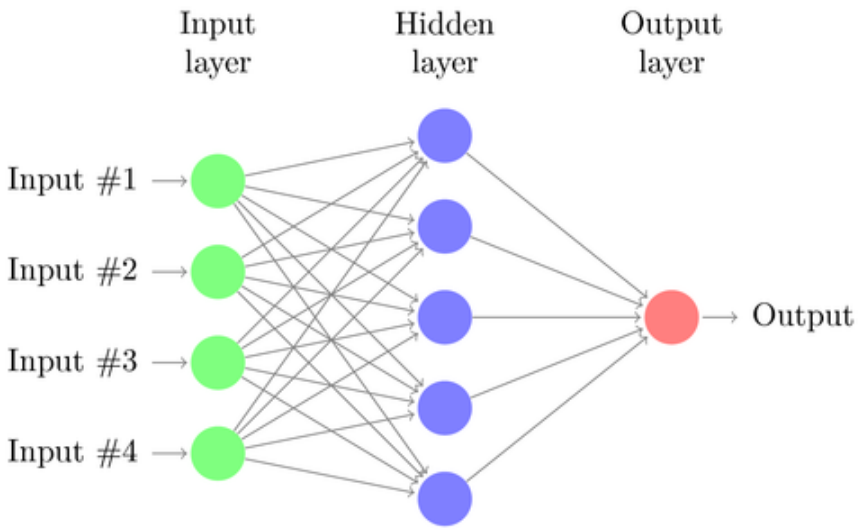


Figura 3.7: red neuronal.

Fuente: The Bleeding Edge Machine (2015). Recuperado de: https://bit.ly/2Funbeq

Hoy en día es casi imposible hablar sobre redes neuronales sin mencionar el aprendizaje profundo, porque la investigación reciente sobre extracción de características, representación de datos y transformación ha encontrado que muchas capas de información de procesamiento pueden abstraer y producir mejores representaciones de datos

para el aprendizaje. En este sentido, una red profunda tendría muchas capas que podrían actuar como unidades de procesamiento de datos para transformarlos y proporcionarlos a la siguiente capa para el posterior procesamiento [12].

Red Neuronal Convolucional

Las redes neuronales convolucionales usan una serie de operaciones bien definidas pero parametrizables que obtienen un representación de la imagen y luego realizar la clasificación [7].

La arquitectura de la Red Neuronal Convolucional (CNN por sus siglas en inglés) las capas pueden tener una organización multidimensional (Figura 3.8). Inspirada en la corteza visual de los animales, la dimensionalidad típica aplicada a las capas es tridimensional. En este tipo de redes, parte de las señales de una capa anterior y se alimenta a otra parte de las neuronas en la siguiente capa [12].

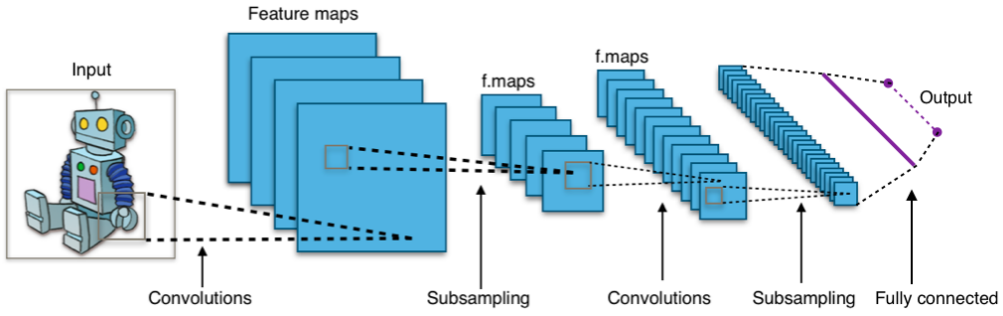


Figura 3.8: Red neuronal convolucional.

El componente más importante de una CNN es la capa convolucional, en la primera capa convolucional las neuronas no están conectadas a cada píxel de la imagen de entrada sino solo a píxeles en sus campos receptivos. Cada neurona en la segunda capa convolucional está conectada solo a las neuronas ubicadas dentro de un pequeño rectángulo en la primera capa. Esta arquitectura permite que la red se concentre en funciones de bajo nivel en la primera capa oculta, luego las ensambla en funciones de nivel superior en la siguiente capa oculta, y así sucesivamente. Esta estructura jerárquica es común en las imágenes del mundo real, que es una de las razones por las que las CNN funcionan tan bien para el reconocimiento de imágenes [13].

Las arquitecturas CNN apilan capas convolucionales, las cuales se mencionan a continuación:

LeNet-5

La arquitectura LeNet-5 es quizás la arquitectura CNN más conocida. Fue creada por Yann LeCun en 1998 y ampliamente utilizada para el reconocimiento de dígitos escritos a mano (MNIST), ver Tabla 3.1.

Capa	Tipo	Mapas	Tamaño	Tamaño de Kernel
1	6	87837	787	789
2	7	78	5415	x
3	545	778	7507	x
4	545	18744	7560	x
5	88	788	6344	x

Tabla 3.1: LeNet-5.

AlexNet

La arquitectura AlexNet CNN ganó el desafío ImageNet ILSVRC 2012 logrando una tasa de error del 17 % top-5, mientras que el segundo mejoró solo el 26%. Fue desarrollado por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton.

AlexNet es bastante similar a LeNet-5, solo que es mucho más grande y profundo, y fue el primero en apilar capas convolucionales directamente una encima de la otra, en lugar de apilar una capa de agrupamiento en la parte superior de cada capa convolucional, ver Tabla 3.2.

Capa	Tipo	Mapas	Tamaño	Tamaño de Kernel
1	6	87837	787	789
2	7	78	5415	x
3	545	778	7507	x
4	545	18744	7560	x
5	88	788	6344	x

Tabla 3.2: AlexNet.

Inception

Inception es factible de usar en escenarios de datos grandes (Big Data). En la Figura 3.9 se puede apreciar el diseño del modelo. En [14] se describen algunas decisiones y trucos que llevaron al éxito de este modelo. Por ejemplo, se muestra cómo cualquier convolución cuyo núcleo es mayor que 3x3 se puede expresar de manera más eficiente con una serie de convoluciones más pequeñas inclusive sugieren reemplazar filtros grandes de 7x7 con un par de capas convolucionales de 1x7 y 7x1. Luego, el documento pasa por varias iteraciones de la red Inception v2 que adoptan por ejemplo, factorización de las convoluciones y mejoras en la normalización. Aplicando lo anterior en la misma red, finalmente se obtiene Inception v3, superando con facilidad a su antecesor GoogLeNet en el benchmark de ImageNet.

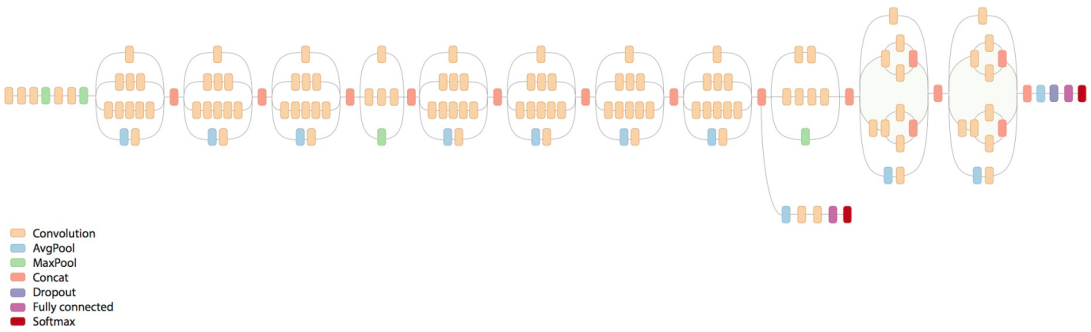


Figura 3.9: modelo Inception V3.

MobileNets

Diseñado para maximizar de manera efectiva la precisión sin perder de vista los recursos restringidos para una aplicación incorporada o en el dispositivo. MobileNets es un modelo de baja latencia y baja potencia parametrizados para cumplir con las limitaciones de recursos de una variedad de casos de uso. Se pueden construir para la clasificación, detección, incrustaciones y segmentación similar a otros modelos populares, por ejemplo, Inception v3.

Las MobileNets se basan en una arquitectura optimizada que utiliza convoluciones separables en profundidad para construir redes neuronales profundas livianas [15].

La tendencia general ha sido crear redes más profundas y complicadas para lograr una mayor precisión. Sin embargo, estos avances para mejorar la precisión no necesariamente hacen que las redes sean más eficientes con respecto al tamaño y la velocidad [15].

Las MobileNets se centran principalmente en la optimización de la latencia, pero también en redes pequeñas. Las MobileNets se construyen principalmente a partir de convoluciones separables en profundidad para reducir el cálculo en las primeras capas. Las capas en que se basa MobileNet, son filtros separables en profundidad [15].

El modelo MobileNet se basa en convoluciones profundas separables. Éstas son una forma de convoluciones factorizadas, las cuales separan:

- Una convolución estándar en una convolución profunda (ejemplo Figura 3.10).
- Una convolución 1 × 1 llamada convolución puntual (ejemplo Figura 3.11).

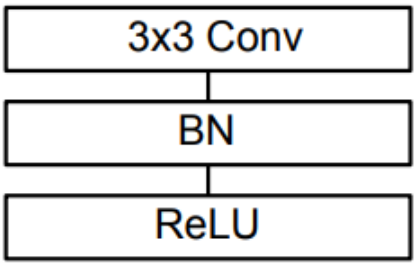


Figura 3.10: capa convolucional estándar.

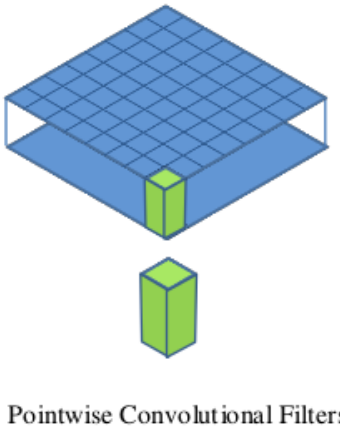


Figura 3.11: filtro de convolucional puntual.

Para MobileNets, la convolución en profundidad aplica un solo filtro (Figura 3.12) a cada canal de entrada. La convolución puntual aplica entonces una convolución de 1×1 para combinar las salidas de la convolución profunda.

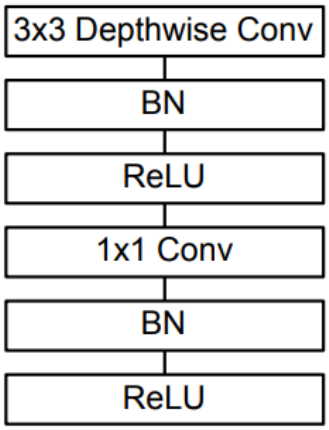


Figura 3.12: convoluciones separables en profundidad con capas depthwise y pointwise seguidos por batchnorm y ReLU.

La convolución separable de profundidad divide esto en dos capas, una capa separada para el filtrado y una capa separada para la combinación (Figura 3.13). Ésta factorización tiene el efecto de reducir drásticamente el cálculo y el tamaño del modelo [15].

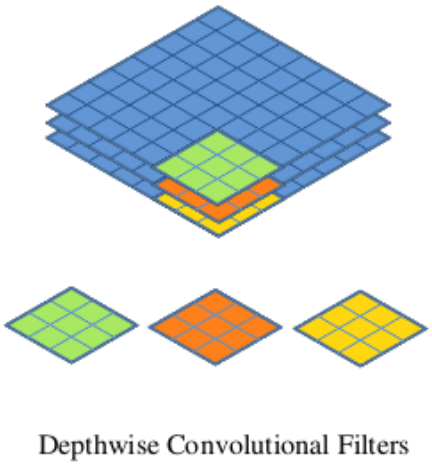


Figura 3.13: filtros convolucionales profundos.

3.3 Procesamiento digital de imágenes

Es un campo de la computación que se encarga de aplicar técnicas a imágenes tales como, rotación, segmentación, extracción de características, realzado de contraste, etc., con la finalidad de preparar las imágenes para un posterior procesamiento. Con las nuevas tecnologías y técnicas desarrolladas en los últimos tiempos, el procesamiento digital de imágenes tiene aplicación en muchos campos entre los que se pueden encontrar: la medicina, visión artificial, detección de patrones y clasificación de imágenes, entre otros.

3.3.1 Pre-procesamiento de imágenes

Hoy en día gracias al avance de las tecnologías computacionales, las imágenes funcionan como un elemento del cual se puede obtener información. Técnicas de pre-procesamiento y procesamiento de imágenes digitales permiten obtener y re-saltar las características contenidas en las mamografías. Algunas de las técnicas que se aplican a las imágenes con el fin de mejorar la calidad son: eliminación de ruido, segmentación y ecualizado del histograma.

3.3.2 Ecualizado del histograma

El histograma es la base de numerosas técnicas de pre-procesamiento de la imagen en el dominio espacial, son las distribuciones que describen la frecuencia con la que se presentan los valores de intensidad (píxeles) de la imagen.

Una imagen en niveles de grises, el rango de tonos de gris es de 0 a 255 (256). El histograma de la imagen consiste en una gráfica donde se muestra el número de píxeles, $n\ k$, de cada nivel de gris, que aparecen en la imagen. Un histograma ideal es aquel que se extiende ocupando casi todo el rango de tonos. La ecualización del histograma mejora el contraste de la imagen. Reparte de forma más o menos uniforme los valores del histograma. La idea es obtener una distribución de probabilidades “uniforme” de los niveles de gris en la imagen.

CAPÍTULO 4

METODOLOGÍA

En este capítulo se describen las fases que se han empleado para la realizar la comparativa de exactitud entre los diferentes modelos de redes neuronales convolucionales para la clasificación de los niveles de densidad mamaria.

A continuación se exponen las etapas de la metodología a seguir:

1. Investigación de redes neuronales convolucionales. Debido a la naturaleza del proyecto, primeramente se examinaron los conceptos básicos que dieron lugar a las redes neuronales convolucionales y los modelos o arquitecturas que existen.

   Para entender la estructura de la red neuronal convolucional se programó en el lenguaje java una neurona, para conocer cómo se programa, el funcionamiento de las capas y la interconexión de ellas. La programación de la red se puede visualizar en el Apéndice A.

   Después, se buscó una API que abstrajera todo el proceso de una red neuronal convolucional, lo que condujo a utilizar la API de *TensorFlow*, pues varios investigadores tienen reportados buenos resultados utilizando ésta API.

   Posteriormente, se indagó en modelos de redes neuronales convolucionales que han sido pre-entrenados en un conjunto de datos de gran tamaño con la finalidad de asegurar que se obtenga una mejor clasificación. Esto condujo a la aplicación de los modelos Inception y MovileNet en el proyecto.

2. Búsqueda de las mamografías. Para llevar a cabo el entrenamiento de los modelos Inception y MovileNet, se realizó una búsqueda exhaustiva de un conjunto de

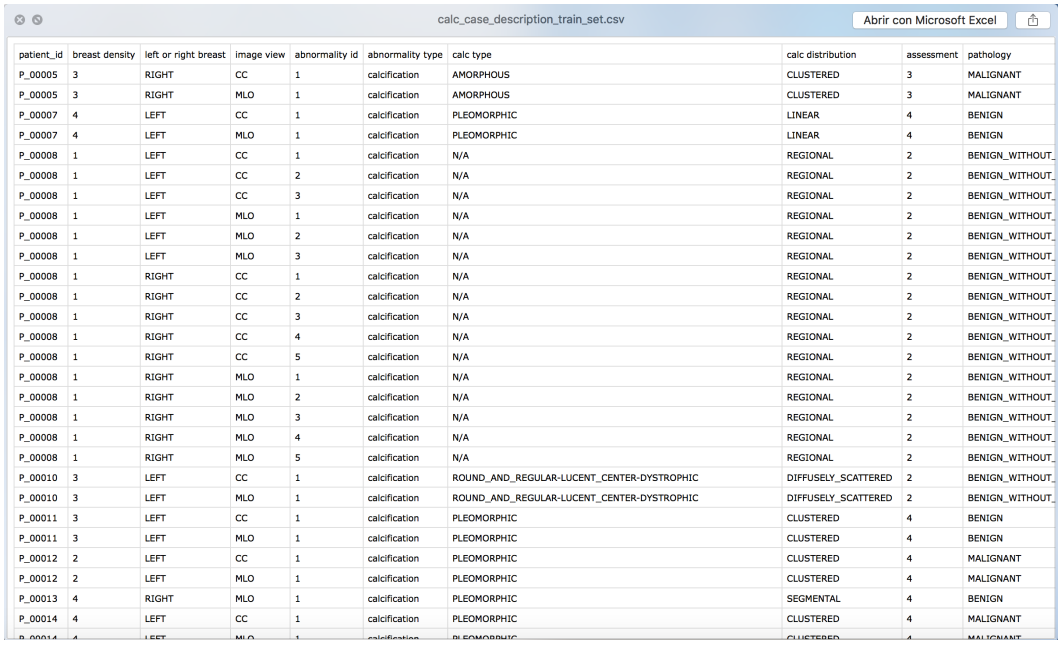
imágenes mamográficas que proporcionaran información útil a través del ROI (Región de Interés por sus siglas en inglés), donde se señalara la zona que contenía la categoría BI-RADS de la densidad. El conjunto de imágenes serían aquellas que contuvieran las siguientes características:

- Licencia libre.
- Clasificado por radiólogos expertos.
- Clasificado usando BI-RADS.
- Que contenga imágenes recortadas en ROI.

Se encontró un servicio que recopila un conjunto de imágenes médicas de cáncer y es accesibles para su descarga pública *The Cancer Imaging Archive* (TCIA)[16]. Los datos están organizados como "colecciones", por pacientes relacionados por una enfermedad común (por ejemplo, cáncer de pulmón), modalidad de imagen (MRI, CT y MG) o por enfoque de investigación. El formato de archivo utilizado por TCIA para el almacenamiento de las imágenes es DICOM. Además, se pueden obtener otros datos de apoyo relacionados con las imágenes, tales como: resultados del paciente, detalles del tratamiento, genómica, patología y los análisis expertos.

El servicio TCIA hospeda la colección completa CBIS-DDSM (*Curated Breast Imaging Subset of DDSM*) del tipo de cáncer: *Breast Cáncer* (cáncer de mama), cuya modalidad son mamografías, y se la colección se actualizó el 27 de septiembre de 2017. Es una base de datos de 2,620 estudios de mamografías que contienen casos normales, benignos y malignos con información patológica verificada. Se incluye la segmentación ROI y cuadros delimitados, y el diagnóstico patológico para los datos de entrenamiento. Con esta información, se puede concluir que este conjunto de datos cumple con las características que se requieren para trabajar en la presente investigación.

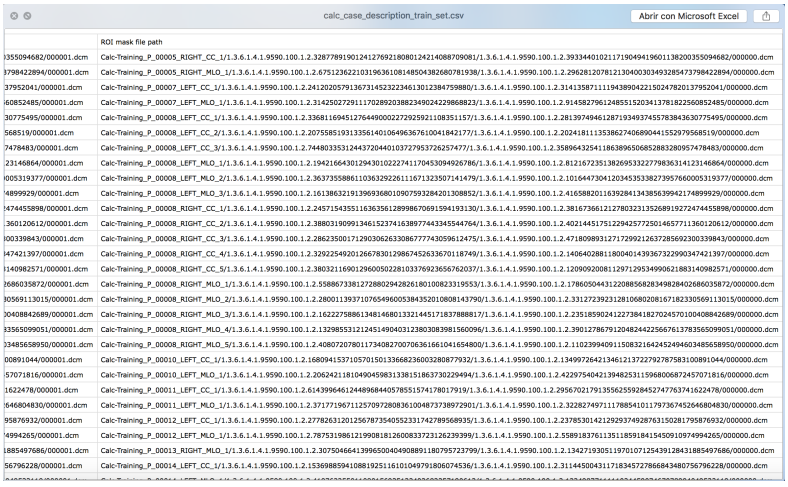
3. Selección de imágenes. Se organizaron, separaron y acomodaron las imágenes con las que se iban a trabajar, obtenidas de la colección CBIS-DDSM, en una estructura para facilitar su lectura al momento de leerlas tanto en Matlab como en Python. En la Figura 4.1 se muestra un archivo de la colección de datos donde se especifican algunas de las características de las imágenes mamográficas. En la segunda columna se especifica la clasificación de la densidad mamaria.



patient_id	breast density	left or right breast	image view	abnormality id	abnormality type	calc type	calc distribution	assessment	pathology
P_00005	3	RIGHT	CC	1	calcification	AMORPHOUS	CLUSTERED	3	MALIGNANT
P_00005	3	RIGHT	MLO	1	calcification	AMORPHOUS	CLUSTERED	3	MALIGNANT
P_00007	4	LEFT	CC	1	calcification	PLEOMORPHIC	LINEAR	4	BENIGN
P_00007	4	LEFT	MLO	1	calcification	PLEOMORPHIC	LINEAR	4	BENIGN
P_00008	1	LEFT	CC	1	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	LEFT	CC	2	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	LEFT	CC	3	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	LEFT	MLO	1	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	LEFT	MLO	2	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	LEFT	MLO	3	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	CC	1	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	CC	2	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	CC	3	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	CC	4	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	CC	5	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	MLO	1	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	MLO	2	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	MLO	3	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	MLO	4	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00008	1	RIGHT	MLO	5	calcification	N/A	REGIONAL	2	BENIGN_WITHOUT_
P_00010	3	LEFT	CC	1	calcification	ROUND_AND_REGULAR-LUCENT_CENTER-DYSTROPHIC	DIFFUSELY_SCATTERED	2	BENIGN_WITHOUT_
P_00010	3	LEFT	MLO	1	calcification	ROUND_AND_REGULAR-LUCENT_CENTER-DYSTROPHIC	DIFFUSELY_SCATTERED	2	BENIGN_WITHOUT_
P_00011	3	LEFT	CC	1	calcification	PLEOMORPHIC	CLUSTERED	4	BENIGN
P_00011	3	LEFT	MLO	1	calcification	PLEOMORPHIC	CLUSTERED	4	BENIGN
P_00012	2	LEFT	CC	1	calcification	PLEOMORPHIC	CLUSTERED	4	MALIGNANT
P_00012	2	LEFT	MLO	1	calcification	PLEOMORPHIC	CLUSTERED	4	MALIGNANT
P_00013	4	RIGHT	MLO	1	calcification	PLEOMORPHIC	SEGMENTAL	4	BENIGN
P_00014	4	LEFT	CC	1	calcification	PLEOMORPHIC	CLUSTERED	4	MALIGNANT
P_00014	4	LEFT	MLO	1	calcification	PLEOMORPHIC	CLUSTERED	4	MALIGNANT

Figura 4.1: características del conjunto de imágenes CBIS-DDSM.

Se eligieron al azar algunas imágenes del conjunto de imágenes, considerando el ROI, tanto de la mama derecha como de la izquierda. En la Figura 4.2 se muestra la ruta de las imágenes como se encuentran en el archivo de descripción de las imágenes.



	ROI mask file path
0355094682/000001.dcm	Calc-Training_P_00005_RIGHT_CC_1/1.3.6.1.4.1.9590.100.1.2.328778919012412769218080124214088709081/1.3.6.1.4.1.9590.100.1.2.393344010211719049419601138200355094682/000000.dcm
1798422894/000001.dcm	Calc-Training_P_00005_RIGHT_MLO_1/1.3.6.1.4.1.9590.100.1.2.675123622103196361081485043826807819 38/1.3.6.1.4.1.9590.100.1.2.296281207812130400303493285473798422894/000000.dcm
37952041/000001.dcm	Calc-Training_P_00007_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.241202057913673145232234613012384759880/1.3.6.1.4.1.9590.100.1.2.314135871119438904221502478201379520 41/000000.dcm
60852485/000001.dcm	Calc-Training_P_00007_LEFT_MLO_1/1.3.6.1.4.1.9590.100.1.2.314250272911170289203882349024229868823/1.3.6.1.4.1.9590.100.1.2.914582796124855152034137818225508524 85/000000.dcm
30775495/000001.dcm	Calc-Training_P_00008_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.336811694512764490002272925921108351157/1.3.6.1.4.1.9590.100.1.2.281397494612871934937455783843630775495/000000.dcm
568519/000001.dcm	Calc-Training_P_00008_LEFT_CC_2/1.3.6.1.4.1.9590.100.1.2.207558519313356140106496367610041842177/1.3.6.1.4.1.9590.100.1.2.202418111353862740689044155297956851 9/000000.dcm
7478483/000001.dcm	Calc-Training_P_00008_LEFT_CC_3/1.3.6.1.4.1.9590.100.1.2.744803353124437204401037279537262574 77/1.3.6.1.4.1.9590.100.1.2.358964325411863896506852883280957478483/000000.dcm
23146864/000001.dcm	Calc-Training_P_00008_LEFT_MLO_1/1.3.6.1.4.1.9590.100.1.2.194216643012943010222741170453094926786/1.3.6.1.4.1.9590.100.1.2.812167235138269533227798363141231 46864/000000.dcm
005319377/000001.dcm	Calc-Training_P_00008_LEFT_MLO_2/1.3.6.1.4.1.9590.100.1.2.363735588611036329226111671323507141479/1.3.6.1.4.1.9590.100.1.2.101644730412034553538273957660005319377/000000.dcm
4899929/000001.dcm	Calc-Training_P_00008_LEFT_MLO_3/1.3.6.1.4.1.9590.100.1.2.161386321913969368010907593284201308852/1.3.6.1.4.1.9590.100.1.2.416588201163928413438563994217489929/000000.dcm
474455898/000001.dcm	Calc-Training_P_00008_RIGHT_CC_1/1.3.6.1.4.1.9590.100.1.2.245715435511636356128998670691594193130/1.3.6.1.4.1.9590.100.1.2.381673661212780323135268919272474455898/000000.dcm
360120612/000001.dcm	Calc-Training_P_00008_RIGHT_CC_2/1.3.6.1.4.1.9590.100.1.2.388031909913461523741638977443345544764/1.3.6.1.4.1.9590.100.1.2.402144517512294257725014657711360120612/000000.dcm
00339843/000001.dcm	Calc-Training_P_00008_RIGHT_CC_3/1.3.6.1.4.1.9590.100.1.2.286235001712903062633086777743059612475/1.3.6.1.4.1.9590.100.1.2.471809893127172992126372856923003398 43/000000.dcm
47421397/000001.dcm	Calc-Training_P_00008_RIGHT_CC_4/1.3.6.1.4.1.9590.100.1.2.329225492012667830129867452633670118749/1.3.6.1.4.1.9590.100.1.2.140640288118004014393673229903474 21397/000000.dcm
14098257 1/000001.dcm	Calc-Training_P_00008_RIGHT_CC_5/1.3.6.1.4.1.9590.100.1.2.380321169012960050228103376923656762037/1.3.6.1.4.1.9590.100.1.2.120909208112971295349906218831409825 71/000000.dcm
686035872/000001.dcm	Calc-Training_P_00008_RIGHT_MLO_1/1.3.6.1.4.1.9590.100.1.2.558867338127288029428261801008233195 53/1.3.6.1.4.1.9590.100.1.2.178605044312208856828349828402686035872/000000.dcm
0569113015/000001.dcm	Calc-Training_P_00008_RIGHT_MLO_2/1.3.6.1.4.1.9590.100.1.2.280011393710765496005384352010808143790/1.3.6.1.4.1.9590.100.1.2.331272392312810680208167182330569113015/000000.dcm
0408842689/000001.dcm	Calc-Training_P_00008_RIGHT_MLO_3/1.3.6.1.4.1.9590.100.1.2.162227588613481468013321445171837888817/1.3.6.1.4.1.9590.100.1.2.235185902412273841827024570100408842689/000000.dcm
3356099051/000001.dcm	Calc-Training_P_00008_RIGHT_MLO_4/1.3.6.1.4.1.9590.100.1.2.132985531212451490403123803083981560096/1.3.6.1.4.1.9590.100.1.2.390127867912048244225667613783565099051/000000.dcm
3485658950/000001.dcm	Calc-Training_P_00008_RIGHT_MLO_5/1.3.6.1.4.1.9590.100.1.2.408072078011734082700706361661041654800/1.3.6.1.4.1.9590.100.1.2.110239940911508321642454946034856589 50/000000.dcm
00891044/000001.dcm	Calc-Training_P_00010_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.168094153710570150133668236003280877932/1.3.6.1.4.1.9590.100.1.2.134997264213461213722792787583100891044/000000.dcm
57071816/000001.dcm	Calc-Training_P_00010_LEFT_MLO_1/1.3.6.1.4.1.9590.100.1.2.206242118104904598313381518637302294 94/1.3.6.1.4.1.9590.100.1.2.422975404213948253115968006872457071816/000000.dcm
1622478/000001.dcm	Calc-Training_P_00011_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.614399646124489684405785515741780179 19/1.3.6.1.4.1.9590.100.1.2.295670217913556255928455274775374162247 8/000000.dcm
6646804830/000001.dcm	Calc-Training_P_00011_LEFT_MLO_1/1.3.6.1.4.1.9590.100.1.2.371771967112570972808361004873738972901/1.3.6.1.4.1.9590.100.1.2.322827497111788541011797367452646804830/000000.dcm
95876932/000001.dcm	Calc-Training_P_00012_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.277826312012567873540552331742789568935/1.3.6.1.4.1.9590.100.1.2.237853014212929374928763150281795876932/000000.dcm
4994265/000001.dcm	Calc-Training_P_00012_LEFT_MLO_1/1.3.6.1.4.1.9590.100.1.2.787531986121990818126008337231262393 99/1.3.6.1.4.1.9590.100.1.2.558918376113511859184154509109749426 5/000000.dcm
1885497686/000001.dcm	Calc-Training_P_00013_RIGHT_MLO_1/1.3.6.1.4.1.9590.100.1.2.307504664139965004049088911807957237 99/1.3.6.1.4.1.9590.100.1.2.134271930511970107125439128431885497686/000000.dcm
56796228/000001.dcm	Calc-Training_P_00014_LEFT_CC_1/1.3.6.1.4.1.9590.100.1.2.153698859410881925116101049791860745 36/1.3.6.1.4.1.9590.100.1.2.311445043117183457278668434807567962 28/000000.dcm

Figura 4.2: ruta dentro de la colección de imágenes donde se encuentra el ROI de cada una de ellas.

En la Figura 4.3 se muestra una de las imágenes utilizadas en los experimentos.

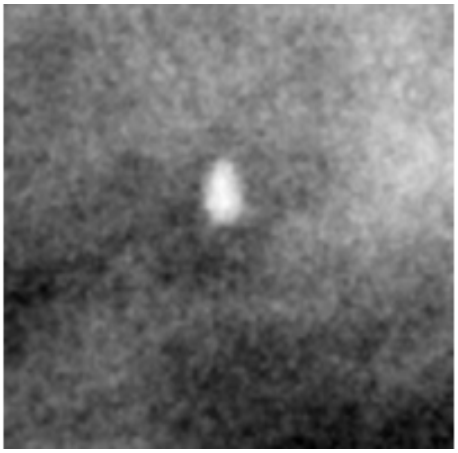


Figura 4.3: ROI de imagen usada en los experimentos.

4. Pre-procesamiento de la imágenes. Todas las imágenes que se utilizaron, fueron pre-procesadas en Matlab utilizando las funciones de media e histograma ecualizado. La Figura 4.4 muestra una pantalla donde se visualiza el proceso para una sola imagen.

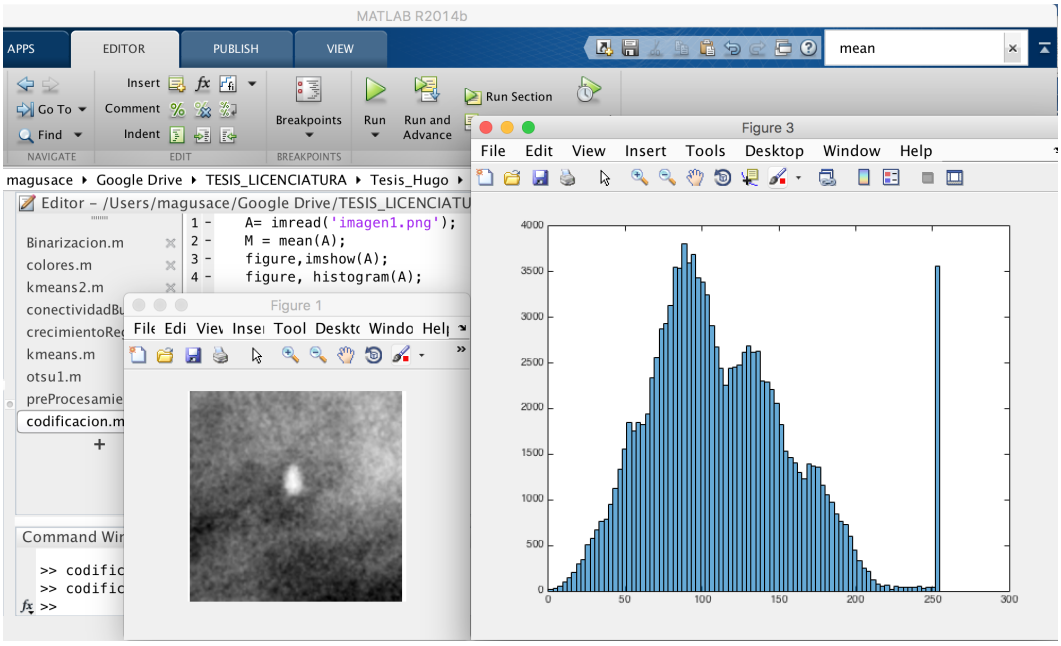


Figura 4.4: pre-procesamiento de las imágenes.

5. Conversión de imágenes. Las imágenes que proporciona la colección CBIS-DDSM vienen en formato DICOM, pero debido a que *Tensorflow* no soporta ese formato, se convirtieron a un formato con el que se pudiera trabajar. Para llevar acabo esta conversión se utilizó la herramienta *mogrify*. La sintaxis de *mogrigy* es la siguiente:

   ```
   magick mogrify -format jpg *.dcm
   ```

6. Entrenamiento del modelo. El entrenamiento consistió en etiquetar las imágenes a la categoría perteneciente, enseguida se descarga el modelo a utilizar y se ejecuta el script que se encarga de realizar el entrenamiento.

7. Clasificación de imágenes. Para la clasificación de las imágenes, se cargan las imágenes en una lista, se realiza un ciclo donde cada una de las imágenes se clasifican, al final de cada iteración se guarda en un lista la relación de la imagen y los porcentajes que corresponden a cada categoría:

   ```
   imagen -> [categoria1-0.02 | categoria2-0.51 | categoria3-0.82 | categoria4-0.15]
   ```

8. Análisis estadístico. Obtenidos los resultados de la clasificación, se determinó la exactitud con la que el modelo logró clasificar utilizando *pandas* y *matplotlib* para la creación de las gráficas. Pandas es una librería de Python destinada al análisis de datos, que proporciona unas estructuras de datos flexibles y que permiten trabajar con ellos de forma muy eficiente. Se debe instalar pandas y numpy al mismo tiempo

   ```
   $ pip install pandas $ pip install numpy
   ```

9. GPU con Inception y MobileNet. Se intentó experimentar el desempeño de los modelos Inception y MobileNet usando GPU (*Graphics Processing Unit*). Se preparó el sistema y se descargaron las librerías, pero la GPU del equipo de cómputo donde se realizaron los experimentos, no cumple con la capacidad de cálculo, y *Tensorflow* lo notifica a través de un mensaje, como se visualiza en la Figura 4.5.

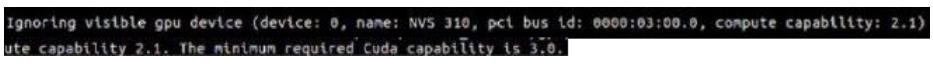


Figura 4.5: mensaje de error al usar gpu

CAPÍTULO 5

RESULTADOS

En ese capítulo se muestran los resultados obtenidos de exactitud por los modelos Inception V3 y MobileNet, al momento de realizar la clasificación de las imágenes con densidad mamaria.

Para llevar a cabo la experimentación, se utilizó una computadora Dell Precision Tower 5810 con las siguientes características:

- Dell Precision Tower 5810.
- Procesador: Intel® Xeon® E5-1600 v4 Series.
- Chipset: Intel® C612.
- Memoria: Quad channel memory up to 256GB 2133MHz DDR4 RDIMM ECC.
- Tarjeta de Video: NVIDIA® Quadro® graphics.

El sistema operativo en el que se desarrolló el proyecto fue Ubuntu 16.04 LTS, en el cual se instalaron las siguientes aplicaciones, herramientas de software y lenguajes de programción:

- Python 3.5.
  - NumPy.
  - Jupyter.

  - Pandas.
  - Tensorflow.
  - Matplotlib.

- Matlab.
- mogrify.
- Aeskulap.

Para el entrenamiento de la red neuronal, se ejecutó un comando cuyos parámetros varían según el modelo que se quisiera utilizar y las iteraciones que se desean llevar a cabo. Este comando se observa en la Figura 5.1 donde los parámetros mas importantes son: *how_many_training_steps* y *architecture*.

```
python3 -m scripts.retrain \
  --bottleneck_dir=tf_files/bottlenecks \
  --how_many_training_steps=500 \
  --model_dir=tf_files/models/ \
  --summaries_dir=tf_files/training_summaries/"${ARCHITECTURE}" \
  --output_graph=tf_files/retrained_graph.pb \
  --output_labels=tf_files/retrained_labels.txt \
  --architecture="${ARCHITECTURE}" \
  --image_dir=tf_files/mammograms
```

Figura 5.1: comando para el entrenamiento del modelo

El primer parámetro (*how_many_training_steps*) indica la cantidad de iteraciones que se desea para el entrenamiento y el segundo parámetro (*architecture*) indica el modelo que se va a utilizar.

Las iteraciones de entrenamiento se refieren al número de repeticiones que se realizan sobre el conjunto de imágenes que se seleccionan para entrenar el modelo. En cada iteración se eligen imágenes al azar que se introducen en la capa final para obtener predicciones. Las predicciones se comparan con las etiquetas reales para actualizar los pesos de la capa final a través de un proceso de backpropagation.

Para realizar la clasificación de las imágenes y determinar la exactitud con la que el modelo logró clasificar se utilizó la herramienta Jupyter. Jupiter es una aplicación Web de código abierto que permite crear y compartir documentos que contienen "live code", ecuaciones, visualizaciones y texto narrativo.

Posteriormente, una vez entrenado el modelo se realizaron una serie de pasos con la finalidad de clasificar y determinar la exactitud del mismo.

- Paso 1. En la Figura 5.2 se muestra el código para cargar el script que apoya en la clasificación. Se crea una lista donde se guardan todas las mamografías que se desean clasificar.

```
import scripts.label_image as label
imgs = []
labels = None
imgs.append(glob.glob("/mammograms/one/*.jpg"))
imgs.append(glob.glob("/mammograms/two/*.jpg"))
imgs.append(glob.glob("/mammograms/three/*.jpg"))
imgs.append(glob.glob("/mammograms/four/*.jpg"))
```

Figura 5.2: código para cargar el script para la clasificación.

- Paso 2. Se crea una lista donde se van a guardar los resultados que se obtengan del clasificador (Ver Figura 5.3). Para obtener el resultado de cada imagen se itera a través de la lista creada en el primer paso.

```
data = []
for i in range(len(imgs)):
    for j in imgs[i]:
        labels, results = label.get_results(j)
        data.append(results)
```

Figura 5.3: código para la clasificación de las imágenes.

- Paso 3. Se crea un diccionario con cada tipo de densidad que se almacenará en una lista los registros que se tienen del conjunto de datos, una vez clasificados correctamente por radiólogos profesionales, con el fin de poder comparar la exactitud que se obtiene del modelo implementado (Figura 5.4).

```
pacients = {'one': [], 'two': [], 'three': [], 'four': []}
with open('../mass_case_description_train_set.csv') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
        if row['image view'] == 'CC':
            if row['breast_density'] == '1':
                pacients['one'].append(row['patient_id'])
            elif row['breast_density'] == '2':
                pacients['two'].append(row['patient_id'])
            elif row['breast_density'] == '3':
                pacients['three'].append(row['patient_id'])
            else:
                pacients['four'].append(row['patient_id'])
```

Figura 5.4: código para la preparación de los datos.

- Paso 4. En la Figura 5.5 se muestra la comprobación de la correspondencia de la imagen a la categoría correcta.

```
accerts = []
for i, row in tbl.iterrows():
    if i.rfind("one") is not -1:
        if row.argmax() == "one":
            accerts.append([i, True])
        else:
            accerts.append([i, False])
    elif i.rfind('two') is not -1:
        if row.argmax() == "two":
            accerts.append([i, True])
        else:
            accerts.append([i, False])
    elif i.rfind('three') is not -1:
        if row.argmax() == "three":
            accerts.append([i, True])
        else:
            accerts.append([i, False])
    else:
        if row.argmax() == "four":
            accerts.append([i, True])
        else:
            accerts.append([i, False])
```

Figura 5.5: código para la comprobación del clasificador de las imágenes en la categoría correcta.

- Paso 5. Se contabiliza la cantidad de aciertos y errores obtenidos con la finalidad de verificar el desempeño del modelo, ver Figura 5.6.

```
trues = 0
falses = 0
for i in accerts:
    if True in i:
        trues+=1
    else:
        falses+=1
```

Figura 5.6: código para la contabilización de aciertos y errores.

- Paso 6. Se obtienen las gráficas, ver Figura 5.7.

```
import matplotlib.pyplot as plt
labels = 'Accert', 'Missed'
sizes = [trues, falses]
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.savefig('../ACCERTS_500_ITERATIONS.png', dpi=600)
plt.show()
```

Figura 5.7: código para graficar los aciertos y errores en la clasificación.

La Figura 5.8, muestra la cantidad de aciertos y errores obtenidos al realizar la clasificación utilizando el modelo Inception V3, con 500, 4000 y 8000 iteraciones de entrenamiento.

La Figura 5.9, muestra la cantidad de aciertos y errores obtenidos al realizar la clasificación utilizando el modelo MobileNet V0.50, con 500, 4000 y 8000 iteraciones de entrenamiento.

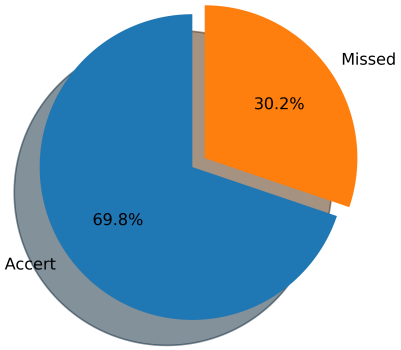
La Figura 5.10, muestra la cantidad de aciertos y errores obtenidos al realizar la clasificación utilizando el modelo MobileNet V1, con 500, 4000 y 8000 iteraciones de entrenamiento.

En la Tabla 5.1 se presenta una comparativa de aciertos que lograron cada uno de los modelos probados para clasificar la densidad mamaria en sus cuatro categorías.

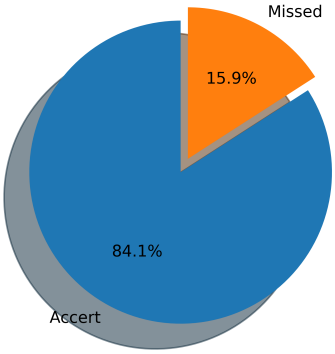
Iteraciones	Inception V3	MobileNet V0.50	MobileNet V1
500	69.8%	65.6%	54.4%
4000	84.1%	**85.4%**	84.4%
8000	84.7%	**85.4%**	84.7%

Tabla 5.1: comparativa de exactitud entre los modelos Inception V3, MobileNet V0.50 y MobileNet V1.

(a)



(b)



(c)

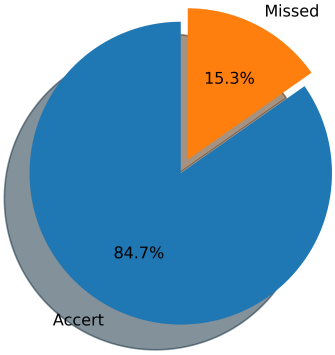
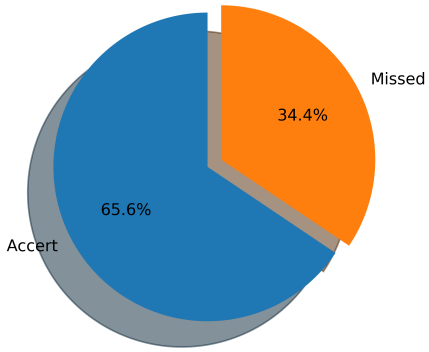
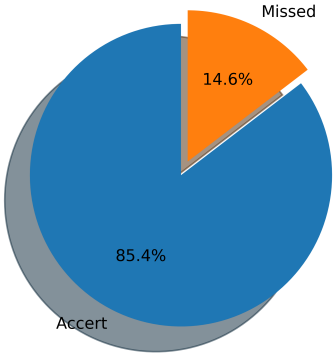


Figura 5.8: resultados de Inception V3: (a) 500, (b) 4000 y (c) 8000 iteraciones.

(a)



(b)



(c)

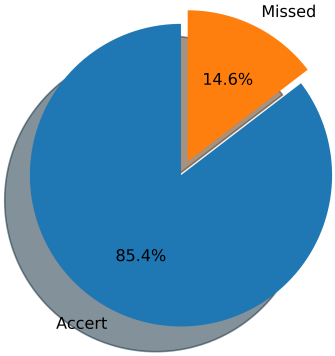
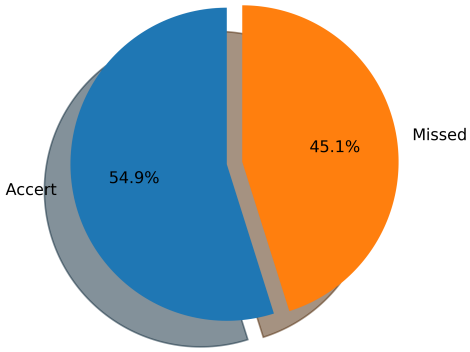
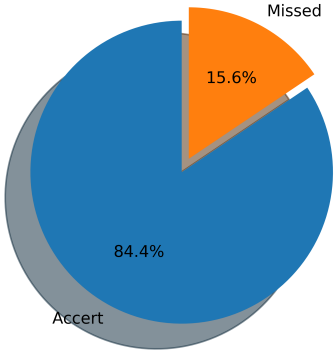


Figura 5.9: resultados de MobileNet V0.50: (a) 500, (b) 4000 y (c) 8000 iteraciones.

(a)



(b)



(c)

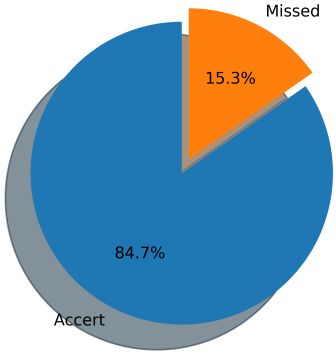


Figura 5.10: resultados de MobileNet V1: (a) 500, (b) 4000 y (c) 8000 iteraciones.

CAPÍTULO 6

CONCLUSIONES

El presente proyecto de investigación cuyos resultados fueron plasmados en ésta tesis, tiene como principal objetivo la clasificación de la densidad mamaria utilizando redes neuronales convolucionales, específicamente se utilizaron los modelos Inception y MobileNet.

Al término de esta tesis, se concluye que el modelo con el que obtuvieron mejores resultados para la clasificasión de los cuatro modelos de densidad mamaria según BI-RADS es MobileNet V0.50, no solo por su porcentaje de puntuación, sino también por la ligereza del modelo. El uso de redes neuronales como MobileNets abre la posibilidad de llevar asistencia médica por computadora a lugares donde no se tiene acceso a un equipo de cómputo costoso. El modelo MobileNet puede arrojar resultados muy competentes, en comparación con otros, como Inception.

El uso de redes neuronales es una tendencia que ha ido creciendo en estos años y es una tecnología que se irá afinando, simplificando, haciéndose mas accesible y barata en años venideros. Debido a lo anterior, es importante voltear a ver todas las investigaciones y aplicaciones utilizando redes neuronales en diferentes áreas.

De manera personal, me ha dejado mucho aprendizaje en varios aspectos. Uno de ellos es que durante la carrera no había tenido la oportunidad de realizar investigación, y ahora trabajé desde las residencias en éste proyecto y por otro lado, los resultados más significativos fueron plasmados en esta tesis. Además, puedo decir que tuve que aprender el lenguaje Python y la herramienta Matlab para poder obtener los resultados.

GLOSARIO

aprendizaje de máquinas: es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender, de forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos.

aprendizaje profundo: es un conjunto de algoritmos de clase aprendizaje de máquinas que intenta modelar abstracciones de alto nivel en datos usando arquitecturas compuestas de transformaciones múltiples no lineales.

axón: prolongación de las neuronas especializadas en conducir el impulso nervioso desde el cuerpo celular o soma (el cuerpo de una célula excluyendo sus prolongaciones) hacia otra célula.

benchmark: resultado de la ejecución de un programa informático o un conjunto de programas en una máquina, con el objetivo de estimar el rendimiento de un elemento concreto y, poder comparar los resultados con máquinas similares.

bootstrap: también llamado "bagging", es un meta-algoritmo de un conjunto de aprendizaje de máquinas diseñado para mejorar la estabilidad y la precisión de los algoritmos de aprendizaje automático utilizados en la clasificación estadística y la regresión.

capa depthwise: capa que aplica una convolución depthwise.

capa pointwise: capa de una convolución de 1 x 1.

convulación: es un operador matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g.

cribado: en medicina es una estrategia aplicada sobre una población para detectar una enfermedad en individuos sin signos o síntomas de esa enfermedad.

dentritas: prolongaciones protoplásmicas ramificadas, bastante cortas de la neurona, dedicadas principalmente a la recepción de estímulos y, secundariamente, a la alimentación celular.

dicom: DICOM (Digital Imaging and Communication in Medicine) es el estándar reconocido mundialmente para el intercambio de imágenes médicas, pensado para su manejo, visualización, almacenamiento, impresión y transmisión.

interobservador: es el grado de acuerdo entre los calificadores, da una puntuación de cuánta homogeneidad, o consenso, hay en las calificaciones otorgadas.

intraobservador: es el grado de acuerdo entre repetidas administraciones de una prueba de diagnóstico realizada por un único evaluador.

latencia: es un intervalo de tiempo entre la estimulación y la respuesta, o, desde un punto de vista más general, un retraso de tiempo entre la causa y el efecto de algún cambio físico en el sistema observado.

parenquimatoso: perteneciente o relativo al parénquima (tejido de los órganos glandulares).

ReLU: Una función de activación con las siguientes reglas:

- Si la entrada es negativa o cero, la salida es 0.
- Si la entrada es positiva, la salida es igual a la entrada.

tejido adiposo: es el tejido de origen mesenquimal (un tipo de tejido conjuntivo) conformado por la asociación de células que acumulan lípidos en su citoplasma: los adipocitos, cumple funciones mecánicas: una de ellas es servir como amortiguador, también protegiendo y manteniendo en su lugar los órganos internos así como a otras estructuras más externas del cuerpo, y también tiene funciones metabólicas y es el encargado de generar grasas para el organismo.

tejido conjuntivo: es también llamado tejido conectivo. Es un conjunto heterogéneo de tejidos orgánicos que comparten un origen común a partir del mesénquima embrionario originado a partir del mesodermo.

tejido fibroglandular: es el conjunto del tejido glandular y tejido conjuntivo.

tejido glandular: tejido constituido por células que presentan, como actividad característica, la producción de secreciones.

APÉNDICE. NEURONA EN JAVA

Neuron.java

```
public class Neuron {

    protected ArrayList<Double> weight;

    private ArrayList<Double> input;

    private Double output;

    private Double outputBeforeActivation;

    private int numberOfInputs = 0;

    protected Double bias = 1.0;

    private IActivationFunction activationFunction;

    private NeuralLayer neuralLayer;

    private Double firstDerivative;

    public Neuron(){
```

```
}

public Neuron(int numberofinputs){
   numberOfInputs=numberofinputs;
   weight=new ArrayList<>(numberofinputs+1);
   input=new ArrayList<>(numberofinputs);
}

public Neuron(int numberofinputs,IActivationFunction iaf){
   numberOfInputs=numberofinputs;
   weight=new ArrayList<>(numberofinputs+1);
   input=new ArrayList<>(numberofinputs);
   activationFunction=iaf;
}

public void setNeuralLayer(NeuralLayer _neuralLayer){
   if(this.neuralLayer==null){
      this.neuralLayer=_neuralLayer;
   }
}

public void init(){
   init(new UniformInitialization(0.0,1.0));
}

public void init(WeightInitialization weightInit){
   if(numberOfInputs>0){
      for(int i=0;i<=numberOfInputs;i++){
         double newWeight = weightInit.Generate();
         try{
            this.weight.set(i, newWeight);
         }
         catch(IndexOutOfBoundsException iobe){
            this.weight.add(newWeight);
         }
      }
```

```
    }
}

public void setInputs(double [] values){
    if(values.length==numberOfInputs){
        for(int i=0;i<numberOfInputs;i++){
            try{
                input.set(i, values[i]);
            }
            catch(IndexOutOfBoundsException iobe){
                input.add(values[i]);
            }
        }
    }
}

public void setInputs(ArrayList<Double> values){
    if(values.size()==numberOfInputs){
        input=values;
    }
}

public ArrayList<Double> getArrayInputs(){
    return input;
}

public double[] getInputs(){
    double[] inputs = new double[numberOfInputs];
    for (int i=0;i<numberOfInputs;i++){
        inputs[i]=this.input.get(i);
    }
    return inputs;
}

public void setInput(int i,double value){
    if(i>=0 && i<numberOfInputs){
        try{
```

```
            input.set(i, value);
        }
        catch(IndexOutOfBoundsException iobe){
            input.add(value);
        }
    }
}

public double getInput(int i){
    return input.get(i);
}

public double[] getWeights(){
    double[] weights = new double[numberOfInputs+1];
    for(int i=0;i<=numberOfInputs;i++){
        weights[i]=weight.get(i);
    }
    return weights;
}

public Double getWeight(int i){
    return weight.get(i);
}

public Double getBias(){
    return weight.get(numberOfInputs);
}

public ArrayList<Double> getArrayWeights(){
    return weight;
}

public void updateWeight(int i, double value){
    if(i>=0 && i<=numberOfInputs){
        weight.set(i, value);
    }
}
```

```
public int getNumberOfInputs(){
   return this.numberOfInputs;
}

public void setWeight(int i,double value) throws NeuralException{
   if(i>=0 && i<numberOfInputs){
      this.weight.set(i, value);
   }
   else{
      throw new NeuralException("Invalid weight index");
   }
}

public double getOutput(){
   return output;
}

public void calc(){
   outputBeforeActivation=0.0;
   if(numberOfInputs>0){
      if(input!=null && weight!=null){
         for(int i=0;i<=numberOfInputs;i++){
            outputBeforeActivation+=(i==numberOfInputs?bias:input.get(i))*weigh
         }
      }
   }
   output=activationFunction.calc(outputBeforeActivation);
   if(neuralLayer.getNeuralMode()==NeuralNet.NeuralNetMode.TRAINING){
      firstDerivative=activationFunction.derivative(outputBeforeActivation);
   }
}

public Double calc(ArrayList<Double> _input){
   Double _outputBeforeActivation=0.0;
   if(numberOfInputs>0){
      if(weight!=null){
```

```
            for(int i=0;i<=numberOfInputs;i++){
                _outputBeforeActivation+=(i==numberOfInputs?bias:_input.get(i)
            }
        }
    }
    return activationFunction.calc(_outputBeforeActivation);
}

public Double calc(Double[] _input){
    Double _outputBeforeActivation=0.0;
    if(numberOfInputs>0){
        if(weight!=null){
            for(int i=0;i<=numberOfInputs;i++){
                _outputBeforeActivation+=(i==numberOfInputs?bias:_input[i])*we
            }
        }
    }
    return activationFunction.calc(_outputBeforeActivation);
}

public Double derivative(double[] _input){
    Double _outputBeforeActivation=0.0;
    if(numberOfInputs>0){
        if(weight!=null){
            for(int i=0;i<=numberOfInputs;i++){
                _outputBeforeActivation+=(i==numberOfInputs?bias:_input[i])*we
            }
        }
    }
    return activationFunction.derivative(_outputBeforeActivation);
}

public ArrayList<Double> calcBatch(ArrayList<ArrayList<Double>>
    _input){
    ArrayList<Double> result = new ArrayList<>();
    for(int i=0;i<_input.size();i++){
        result.add(0.0);
```

```
        Double _outputBeforeActivation=0.0;
        for(int j=0;j<numberOfInputs;j++){
            _outputBeforeActivation+=(j==numberOfInputs?bias:_input.get(i).get(j))
        }
        result.set(i,activationFunction.calc(_outputBeforeActivation));
    }
    return result;
}

public ArrayList<Double>
    derivativeBatch(ArrayList<ArrayList<Double>> _input){
    ArrayList<Double> result = new ArrayList<>();
    for(int i=0;i<_input.size();i++){
        result.add(0.0);
        Double _outputBeforeActivation=0.0;
        for(int j=0;j<numberOfInputs;j++){
            _outputBeforeActivation+=(j==numberOfInputs?bias:_input.get(i).get(j))
        }
        result.set(i,activationFunction.derivative(_outputBeforeActivation));
    }
    return result;
}

public void setActivationFunction(IActivationFunction iaf){
    this.activationFunction=iaf;
}

public double getOutputBeforeActivation(){
    return outputBeforeActivation;
}

public void deactivateBias(){
    this.bias=0.0;
}

public void activateBias(){
    this.bias=1.0;
```

```
    }

    public double getFirstDerivative(){
        return firstDerivative;
    }

    public double getBiasSource(){
        return this.bias;
    }

}
```

NeuralLayer.java

```
public abstract class NeuralLayer {

    protected int numberOfNeuronsInLayer;
    protected ArrayList<Neuron> neuron;
    protected IActivationFunction activationFnc;
    protected NeuralLayer previousLayer;
    protected NeuralLayer nextLayer;
    protected ArrayList<Double> input;
    protected ArrayList<Double> output;
    protected int numberOfInputs;

    protected NeuralNet neuralNet;

    public NeuralLayer(NeuralNet _neuralNet,int numberofneurons){
        this.neuralNet=_neuralNet;
        this.numberOfNeuronsInLayer=numberofneurons;
        neuron = new ArrayList<>(numberofneurons);
        output = new ArrayList<>(numberofneurons);
    }

    public NeuralLayer(NeuralNet _neuralNet,int numberofneurons
          ,IActivationFunction iaf){
```

```
    this.neuralNet=_neuralNet;
    this.numberOfNeuronsInLayer=numberofneurons;
    this.activationFnc=iaf;
    neuron = new ArrayList<>(numberofneurons);
    output = new ArrayList<>(numberofneurons);
}

public int getNumberOfNeuronsInLayer(){
    return numberOfNeuronsInLayer;
}

public ArrayList<Neuron> getListOfNeurons(){
    return neuron;
}

public NeuralLayer getPreviousLayer(){
    return previousLayer;
}

public NeuralLayer getNextLayer(){
    return nextLayer;
}

protected void setPreviousLayer(NeuralLayer layer){
    previousLayer=layer;
}

protected void setNextLayer(NeuralLayer layer){
    nextLayer=layer;
}

protected void init(WeightInitialization weightInitialization){
    if(numberOfNeuronsInLayer>=0){
        for(int i=0;i<numberOfNeuronsInLayer;i++){
            try{
                neuron.get(i).setActivationFunction(activationFnc);
                neuron.get(i).setNeuralLayer(this);
```

```
                neuron.get(i).init(weightInitialization);
            }
            catch(IndexOutOfBoundsException iobe){
                neuron.add(new Neuron(numberOfInputs,activationFnc));
                neuron.get(i).setNeuralLayer(this);
                neuron.get(i).init(weightInitialization);
            }
        }
    }
}

protected void setInputs(ArrayList<Double> inputs){
    this.numberOfInputs=inputs.size();
    this.input=inputs;
}

protected void calc(){
    if(input!=null && neuron!=null){
        for(int i=0;i<numberOfNeuronsInLayer;i++){
            neuron.get(i).setInputs(this.input);
            neuron.get(i).calc();
            try{
                output.set(i,neuron.get(i).getOutput());
            }
            catch(IndexOutOfBoundsException iobe){
                output.add(neuron.get(i).getOutput());
            }
        }
    }
}

protected ArrayList<Double> getOutputs(){
    return output;
}

public Neuron getNeuron(int i){
    return neuron.get(i);
```

```
}

protected void setNeuron(int i, Neuron _neuron){
   try{
      this.neuron.set(i, _neuron);
   }
   catch(IndexOutOfBoundsException iobe){
      this.neuron.add(_neuron);
   }
}

public Double getWeight(int i,int j){
   return this.neuron.get(j).getWeight(i);
}

public ArrayList<Double> getArrayListInputs(){
   return input;
}

public double[] getInputs(){
   double[] result = new double[numberOfInputs];
   for(int i=0;i<numberOfInputs;i++){
      result[i]=input.get(i);
   }
   return result;
}

public NeuralNet.NeuralNetMode getNeuralMode(){
   return this.neuralNet.getNeuralNetMode();
}

public void deactivateBias(){
   for(Neuron n:this.neuron){
      n.deactivateBias();
   }
}
```

```
    public void activateBias(){
        for(Neuron n:this.neuron){
            n.activateBias();
        }
    }

    public boolean isBiasActive(){
        if(neuron.get(0).bias==1.0)
            return true;
        else
            return false;
    }

}
```

NeuralNet.java

```
public class NeuralNet {

    protected InputLayer inputLayer;
    protected ArrayList<HiddenLayer> hiddenLayer;
    protected OutputLayer outputLayer;
    protected int numberOfHiddenLayers;
    protected int numberOfInputs;
    protected int numberOfOutputs;
    protected ArrayList<Double> input;
    protected ArrayList<Double> output;

    protected boolean activeBias=true;


    protected WeightInitialization weightInitialization
          =new UniformInitialization(0.0,1.0);


    protected int[] neuronsInHiddenLayers;
```

```
protected int[] indexesWeightPerLayer;


public enum NeuralNetMode { BUILD, TRAINING, RUN };

protected NeuralNetMode neuralNetMode = NeuralNetMode.BUILD;

public NeuralNet(int numberofinputs,int numberofoutputs,
      int [] numberofhiddenneurons,IActivationFunction[]
         hiddenAcFnc,
      IActivationFunction outputAcFnc,
      WeightInitialization _weightInitialization){
   weightInitialization=_weightInitialization;
   numberOfHiddenLayers=numberofhiddenneurons.length;
   neuronsInHiddenLayers = new int[numberOfHiddenLayers+1];
   indexesWeightPerLayer = new int[numberOfHiddenLayers+2];
   for(int i=0;i<=numberOfHiddenLayers;i++){
      if(i==numberOfHiddenLayers){
         neuronsInHiddenLayers[i]=numberofoutputs;
      }
      else{
         neuronsInHiddenLayers[i]=numberofhiddenneurons[i];
      }
      if(i==0){
         indexesWeightPerLayer[i]=0;
      }
      else{
         indexesWeightPerLayer[i]=indexesWeightPerLayer[i-1]
               + (neuronsInHiddenLayers[i-1]*
                  ((i==1?numberofinputs:neuronsInHiddenLayers[i-2])
                  +1));
      }
   }
   if(numberOfHiddenLayers>0){
      indexesWeightPerLayer[numberOfHiddenLayers+1]=
```

```
            indexesWeightPerLayer[numberOfHiddenLayers]
            + neuronsInHiddenLayers[numberOfHiddenLayers]
               *(neuronsInHiddenLayers[numberOfHiddenLayers-1]+1);
      }
      else{
         indexesWeightPerLayer[numberOfHiddenLayers+1]=
            indexesWeightPerLayer[numberOfHiddenLayers]
               + neuronsInHiddenLayers[numberOfHiddenLayers]
                  *(numberOfInputs+1);
      }
      numberOfInputs=numberofinputs;
      numberOfOutputs=numberofoutputs;
      if(numberOfHiddenLayers==hiddenAcFnc.length){
         input=new ArrayList<>(numberofinputs);
         inputLayer=new InputLayer(this,numberofinputs);
         if(numberOfHiddenLayers>0){
            hiddenLayer=new ArrayList<>(numberOfHiddenLayers);
         }
         for(int i=0;i<numberOfHiddenLayers;i++){
            if(i==0){
               try{
                  hiddenLayer.set(i,new
                     HiddenLayer(this,numberofhiddenneurons[i],
                     hiddenAcFnc[i],
                     inputLayer.getNumberOfNeuronsInLayer()));
               }
               catch(IndexOutOfBoundsException iobe){
                  hiddenLayer.add(new
                     HiddenLayer(this,numberofhiddenneurons[i],
                     hiddenAcFnc[i],
                     inputLayer.getNumberOfNeuronsInLayer()));
               }
               inputLayer.setNextLayer(hiddenLayer.get(i));
            }
            else{
               try{
                  hiddenLayer.set(i, new
```

```
                    HiddenLayer(this,numberofhiddenneurons[i],
                     hiddenAcFnc[i],hiddenLayer.get(i-1)
                    .getNumberOfNeuronsInLayer()
                    ));
                }
                catch(IndexOutOfBoundsException iobe){
                   hiddenLayer.add(new
                      HiddenLayer(this,numberofhiddenneurons[i],
                       hiddenAcFnc[i],hiddenLayer.get(i-1)
                      .getNumberOfNeuronsInLayer()
                      ));
                }
                hiddenLayer.get(i-1).setNextLayer(hiddenLayer.get(i));
             }
          }
          if(numberOfHiddenLayers>0){
             outputLayer=new
                OutputLayer(this,numberofoutputs,outputAcFnc,
                   hiddenLayer.get(numberOfHiddenLayers-1)
                   .getNumberOfNeuronsInLayer()
                   );
             hiddenLayer.get(numberOfHiddenLayers-1).setNextLayer(outputLayer);
          }
          else{
             outputLayer=new OutputLayer(this,numberofoutputs,
                outputAcFnc,
                   numberofinputs);
             inputLayer.setNextLayer(outputLayer);
          }
       }
       setNeuralNetMode(NeuralNetMode.RUN);
    }

    public NeuralNet(int numberofinputs,int numberofoutputs,
          int [] numberofhiddenneurons,IActivationFunction[]
             hiddenAcFnc,
          IActivationFunction outputAcFnc){
```

```
    this(numberofinputs,numberofoutputs,numberofhiddenneurons,hiddenAcFnc
        ,outputAcFnc,new UniformInitialization(0.0,1.0));
}


public NeuralNet(int numberofinputs,int numberofoutputs,
    IActivationFunction outputAcFnc){
    this(numberofinputs,numberofoutputs,new int[0],new
    IActivationFunction[0],outputAcFnc);
}

protected NeuralNet(){

}

public void setInputs(ArrayList<Double> inputs){
    if(inputs.size()==numberOfInputs){
        this.input=inputs;
    }
}

public void setInputs(double[] inputs){
    if(inputs.length==numberOfInputs){
        for(int i=0;i<numberOfInputs;i++){
            try{
                input.set(i, inputs[i]);
            }
            catch(IndexOutOfBoundsException iobe){
                input.add(inputs[i]);
            }
        }
    }
}

public ArrayList<Double> getArrayInputs(){
    return input;
}
```

```
public Double getInput(int i){
    return input.get(i);
}

public double[] getInputs(){
    double[] result=new double[numberOfInputs];
    for(int i=0;i<numberOfInputs;i++){
        result[i]=input.get(i);
    }
    return result;
}

public void calc(){
    inputLayer.setInputs(input);
    inputLayer.calc();
    if(numberOfHiddenLayers>0){
        for(int i=0;i<numberOfHiddenLayers;i++){
            HiddenLayer hl = hiddenLayer.get(i);
            hl.setInputs(hl.getPreviousLayer().getOutputs());
            hl.calc();
        }
    }
    outputLayer.setInputs(outputLayer.getPreviousLayer().getOutputs());
    outputLayer.calc();
    this.output=outputLayer.getOutputs();
}

public ArrayList<Double> getArrayOutputs(){
    return output;
}

public double[] getOutputs(){
    double[] _outputs = new double[numberOfOutputs];
    for(int i=0;i<numberOfOutputs;i++){
        _outputs[i]=output.get(i);
    }
```

```
        return _outputs;
    }

    public double getOutput(int i){
        return output.get(i);
    }

    public void print(){
        System.out.println("Neural Network: "+this.toString());
        System.out.println("\tInputs:"+String.valueOf(this.numberOfInputs));
        System.out.println("\tOutputs:"+String.valueOf(this.numberOfOutputs));
        System.out.println("\tHidden Layers:
            "+String.valueOf(numberOfHiddenLayers));
        for(int i=0;i<numberOfHiddenLayers;i++){
            System.out.println("\t\tHidden Layer "+
                    String.valueOf(i)+": "+
                    String.valueOf(this.hiddenLayer.get(i)
                            .numberOfNeuronsInLayer)+" Neurons");
        }

    }

    public void setNeuralDataSet(NeuralDataSet _neuralDataSet){
        _neuralDataSet.neuralNet=this;
    }

    public int getNumberOfHiddenLayers(){
        return numberOfHiddenLayers;
    }

    public int getNumberOfInputs(){
        return numberOfInputs;
    }

    public int getNumberOfOutputs(){
        return numberOfOutputs;
    }
```

```
public InputLayer getInputLayer(){
   return inputLayer;
}

public HiddenLayer getHiddenLayer(int i){
   return hiddenLayer.get(i);
}

public ArrayList<HiddenLayer> getHiddenLayers(){
   return hiddenLayer;
}

public OutputLayer getOutputLayer(){
   return outputLayer;
}

public void deactivateBias(){
   if(numberOfHiddenLayers>0){
      for(HiddenLayer hl:hiddenLayer){
         for(Neuron n:hl.getListOfNeurons()){
            n.deactivateBias();
         }
      }
   }
   for(Neuron n:outputLayer.getListOfNeurons()){
      n.deactivateBias();
   }
}

public void activateBias(){
   for(HiddenLayer hl:hiddenLayer){
      for(Neuron n:hl.getListOfNeurons()){
         n.activateBias();
      }
   }
   for(Neuron n:outputLayer.getListOfNeurons()){
```

```
            n.activateBias();
        }
    }

    public boolean isBiasActive(){
        return activeBias;
    }

    public WeightInitialization getWeightInitialization(){
        return weightInitialization;
    }

    public double[] getAllWeights(){
        int
            numberOfWeights=indexesWeightPerLayer[numberOfHiddenLayers+1];
        double[] weights=new double[numberOfWeights];
        for(int l=0;l<=numberOfHiddenLayers;l++){
            int j=0;
            NeuralLayer nl;
            if(l==numberOfHiddenLayers) // outputlayer
                nl = outputLayer;
            else
                nl = hiddenLayer.get(l);

            for(Neuron n:nl.getListOfNeurons()){
                for(int i=0;i<=n.getNumberOfInputs();i++){
                    weights[indexesWeightPerLayer[l]
                          +j*(neuronsInHiddenLayers[l]+1)
                          +i]=n.getWeight(i);
                }
                j++;
            }
        }
        return weights;
    }

    public double getWeight(int layer,int neuron,int input){
```

```
        if(layer==numberOfHiddenLayers){
            return outputLayer.getWeight(input, neuron);
        }
        else{
            return hiddenLayer.get(layer).getWeight(input, neuron);
        }
    }

    public int getTotalNumberOfWeights(){
        int result=0;
        for(HiddenLayer hl:this.hiddenLayer){
            result+=hl.numberOfNeuronsInLayer*(hl.numberOfInputs+1);
        }
        result+=outputLayer.numberOfNeuronsInLayer
              *(outputLayer.numberOfInputs+1);
        return result;
    }

    public void setNeuralNetMode(NeuralNet.NeuralNetMode
        _neuralNetMode){
        this.neuralNetMode=_neuralNetMode;
    }

    public NeuralNetMode getNeuralNetMode(){
        return this.neuralNetMode;
    }

}
```

HiddenLayer.java

```
public class HiddenLayer extends NeuralLayer {

    public HiddenLayer(NeuralNet _neuralNet, int numberofneurons
          ,IActivationFunction iaf,int numberofinputs){
        super(_neuralNet,numberofneurons,iaf);
```

```
      numberOfInputs=numberofinputs;
      this.init(_neuralNet.getWeightInitialization());
   }

   @Override
   public void setPreviousLayer(NeuralLayer previous){
      this.previousLayer=previous;
      if(previous.nextLayer!=this)
         previous.setNextLayer(this);
   }

   @Override
   public void setNextLayer(NeuralLayer next){
      nextLayer=next;
      if(next.previousLayer!=this)
         next.setPreviousLayer(this);
   }

}
```

InputLayer.java

```
public class InputLayer extends NeuralLayer {

   public InputLayer(NeuralNet _neuralNet,int numberofinputs){
      super(_neuralNet,numberofinputs,new Linear(1));
      previousLayer=null;
      numberOfInputs=numberofinputs;
      this.init(_neuralNet.getWeightInitialization());
   }

   @Override
   public void setNextLayer(NeuralLayer layer){
      nextLayer=layer;
      if(layer.previousLayer!=this)
         layer.setPreviousLayer(this);
```

```
}

@Override
public void setPreviousLayer(NeuralLayer layer){
   previousLayer=null;
}

@Override
public void init(WeightInitialization weightInitialization){
   for(int i=0;i<numberOfInputs;i++){
      this.setNeuron(i,new InputNeuron());
      this.getNeuron(i).setNeuralLayer(this);
      this.getNeuron(i).init();
   }
}

@Override
public void setInputs(ArrayList<Double> inputs){
   if(inputs.size()==numberOfInputs){
      input=inputs;
   }
}

@Override
public void calc(){
   if(input!=null && getListOfNeurons()!=null){
      for(int i=0;i<numberOfNeuronsInLayer;i++){
         double[] firstInput = {this.input.get(i)};
         getNeuron(i).setInputs(firstInput);
         getNeuron(i).calc();
         try{
            output.set(i,getNeuron(i).getOutput());
         }
         catch(IndexOutOfBoundsException iobe){
            output.add(getNeuron(i).getOutput());
         }
      }
```

```
        }
    }
}
```

OutputLayer.java

```
public class OutputLayer extends NeuralLayer {

    public OutputLayer(NeuralNet _neuralNet,int numberofneurons
            ,IActivationFunction iaf,int numberofinputs){
        super(_neuralNet,numberofneurons,iaf);
        numberOfInputs=numberofinputs;
        nextLayer=null;
        init(_neuralNet.getWeightInitialization());
    }

    public OutputLayer(NeuralNet _neuralNet,int numberofneurons
            ,IActivationFunction iaf,NeuralLayer _previousLayer){
        super(_neuralNet,numberofneurons,iaf);
        setPreviousLayer(_previousLayer);
        numberOfInputs=_previousLayer.getNumberOfNeuronsInLayer();
        init(_neuralNet.getWeightInitialization());
    }

    @Override
    public void setNextLayer(NeuralLayer layer){
        nextLayer=null;
    }

    @Override
    public void setPreviousLayer(NeuralLayer layer){
        previousLayer=layer;
        if(layer.nextLayer!=this)
            layer.setNextLayer(this);
    }
```

```
}
```

Código bajo licencia open-source, para mayor información: `https://github.com/PacktPublishing/Neural-Network-Programming-with-Java-SecondEdition`

BIBLIOGRAFÍA

[1] H. N. F., Guo, "Mammographic density and the risk and detection of breast cancer. new england journal of medicine," pp. 227–236, 2007.

[2] N. F. Boyd, J. W. Byng, R. A. Jong, E. K. Fishell, L. E. Little, A. B. Miller, G. A. Lockwood, D. L. Tritchler, and M. J. Yaffe, "Quantitative classification of mammographic densities and breast cancer risk: Results from the canadian national breast screening study," *JNCI: Journal of the National Cancer Institute*, vol. 87, no. 9, pp. 670–675, 1995.

[3] A. Abdelrahim, W. A. Berg, H. Peng, Y. Luo, R. Jankowitz, and S. Wu, "A deep learning method for classifying mammographic breast density categories," vol. 45, 11 2017.

[4] R. M. Summers, *Deep Learning and Computer-Aided Diagnosis for Medical Image Processing: A Personal Perspective*, pp. 3–10.
Cham: Springer International Publishing, 2017.

[5] K. Bovis and S. Singh, "Classification of mammographic breast density using a combined classifier paradigm," in *In 4th International Workshop on Digital Mammography*, 2002.

[6] S. Ciatto, N. Houssami, A. Apruzzese, E. Bassetti, B. Brancato, F. Carozzi, S. Catarzi, M. Lamberini, G. Marcelli, R. Pellizzoni, B. Pesce, G. Risso, F. Russo, and A. Scorsolini, "Categorizing breast mammographic density: intra- and interobserver reproducibility of bi-rads density categories," *The Breast*, vol. 14, no. 4, pp. 269 – 275, 2005.

[7] P. Fonseca, B. Castañeda, R. Valenzuela, and J. Wainer, "Breast density classification with convolutional neural networks," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (C. Beltrán-Castañón, I. Nyström, and F. Famili, eds.), (Cham), pp. 101–108, Springer International Publishing, 2017.

[8] W. H. Organization, "Breast cancer: prevention and control." `http://who.int/cancer/detection/breastcancer/en/`.

[9] G. Agarwal, P. Ramakant, E. Forgach, J. Carrasco Rendón, J. Manuel Chaparro, C. Sánchez Basurto, and M. Margaritoni, "Breast cancer care in developing countries," vol. 33, pp. 2069–76, 09 2009.

[10] I. N. del Cáncer, "Mamografías." https://www.cancer.gov/espanol/tipos/seno/hoja-informativa-mamografias.

[11] A. Tripathi, *Practical Machine Learning Cookbook*.
Packt Publishing, 2017.

[12] F. M. Soares and A. M. F. Souza, *Neural Network Programming with Java*.
Packt Publishing, 2017.

[13] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*.
O'Reilly, 2017.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[16] T. C. I. A. (TCIA). http://www.cancerimagingarchive.net.