UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA
MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

EVALUACIÓN DE HIPERPARÁMETROS EN REDES NEURONALES CONVOLUCIONALES

TESIS PARA OBTENER EL GRADO DE DOCTORA EN CIENCIAS QUE PRESENTA

DALILA BLANCA PÉREZ PÉREZ

DIRECTOR

DR. JUAN PABLO GARCÍA VÁZQUEZ

CO-DIRECTOR

DR. RICARDO SALOMÓN TORRES

Mexicali, Baja California, diciembre de 2023

TESIS DEFENDIDA POR

Dalila Blanca Pérez Pérez

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. Juan Pablo García Vázquez

Director de Tesis

Dr. Ricardo Salomón Torres

Co-Director

Dr. Ángel Gabriel Andrade Reátiga

Miembro del Comité

Dr. José Ángel González Fraga

Miembro del Comité

Dr. Juan Carlos García Gallegos

Miembro del Comité

Dra. Wendy Flores Fuentes

Coordinador de Posgrado de la FIM

Mexicali, Baja California, diciembre de 2023

RESUMEN de la Tesis de Dalila Blanca Pérez Pérez, presentada como requisito parcial para la obtención del grado de **DOCTOR EN CIENCIAS** en el campo del conocimiento de **COMPUTACIÓN**. Mexicali, Baja California, México. Diciembre de 2023.

Aprobado por:

Dr. Juan Pablo García Vázquez	**Dr. Ricardo Salomón Torres**
Director de Tesis	Co-Director

Esta tesis se centró en una evaluación exhaustiva de la precisión y el tiempo de procesamiento de ocho arquitecturas de redes neuronales convolucionales (CNNs), con un énfasis particular en la clasificación de la variedad de dátiles Medjool. Siete de estas arquitecturas fueron preentrenadas utilizando una extensa base de datos de imágenes diseñada para el reconocimiento de objetos (ImageNet), y recibieron transfer learning al reemplazar su última capa de clasificación. Estos modelos incluyeron VGG-16, VGG-19, Inception V3, ResNet-50, ResNet-101, ResNet-152 y AlexNet. Además, se incorporó un modelo que aprendía desde cero, es decir, sin obtener aprendizaje previo.

La evaluación de estas CNNs se llevó a cabo mediante la modificación de hiperparámetros críticos, como épocas, lotes, optimizadores y tasas de aprendizaje, ya que se ha informado que estos parámetros tienen efectos positivos en el rendimiento de las redes convolucionales. Los resultados destacaron que la arquitectura de la serie VGG, utilizando el optimizador Adam, demostró el mejor rendimiento en la clasificación de dátiles Medjool, siendo VGG-19 el modelo con mayor precisión, alcanzando un 99.32%. Por otro lado, las arquitecturas del grupo ResNet mostraron un rendimiento inferior, con ResNet-152 reportando la menor precisión, con un 64.17%, utilizando el mismo optimizador. El uso del optimizador SGD no tuvo un efecto significativo en la obtención de altas precisiones.

Palabras clave: Aprendizaje profundo, Redes Neuronales, Transferencia de Aprendizaje.

ABSTRACT of the thesis of **Dalila Blanca Pérez Pérez**, presented as a partial requirement for the degree of **DOCTOR OF SCIENCE** in the field of **COMPUTING**. Mexicali, Baja California, México. December 2023.

Approved by:

Dr. Juan Pablo García Vázquez	**Dr. Ricardo Salomón Torres**
Thesis Advisor	Co-Thesis Advisor

This thesis is focused on a comprehensive evaluation of the precision and processing time of eight Convolutional Neural Network (CNN) architectures, with a particular emphasis on the classification of the Medjool date variety. Seven of these architectures were pretrained using an extensive image database designed for object recognition (ImageNet), and they underwent transfer learning by replacing their last classification layer. These models included VGG-16, VGG-19, Inception V3, ResNet-50, ResNet-101, ResNet-152, and AlexNet. Additionally, a model that learns from scratch, i.e., without obtaining pretraining, was incorporated.

The evaluation of these CNNs was carried out by modifying critical hyperparameters such as epochs, batches, optimizers, and learning rates, as these parameters have been reported to have positive effects on the performance of convolutional networks. The results highlighted that the VGG series architecture, using the Adam optimizer, demonstrated the best performance in classifying Medjool dates, with VGG-19 achieving the highest accuracy at 99.32%. On the other hand, architectures from the ResNet group showed lower performance, with ResNet-152 reporting the lowest accuracy at 64.17%, using the same optimizer. The use of the SGD optimizer did not have a significant effect on achieving high accuracy.

Keywords: Deep Learning, Neural Networks, Transfer Learning.

Dedicatoria

A Dios por su infinita bondad y amor.

Agradecimientos

Quisiera agradecer a:

A Dios por todo lo que me ha brindado en la vida.

A mi esposo y a mi familia por su constante apoyo.

A mi director el Dr. Juan Pablo García Vázquez por su tiempo, consejos y retroalimentación.

A mi Co-Director el Dr. Ricardo Salomón Torres por ser parte de mi desarrollo como doctor y estar siempre de apoyo cuando lo necesité.

A los miembros de mi comité de tesis: al Dr. Ángel Gabriel Andrade Reátiga, Dr. Juan Carlos García Gallegos, y al Dr. José Ángel González Fraga por su apoyo y sugerencias.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada para la realización de mis estudios de maestría.

A la Universidad Autónoma de Baja California y a mis profesores del posgrado por brindarme siempre su apoyo y asesoría.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

En este capítulo se describe la motivación de la tesis y se introduce al tema de investigación objeto de ésta. Posteriormente, se plantean los objetivos de investigación, así como la metodología que se siguió durante esta tesis; y finalmente, se presenta la organización de este documento.

1.1 Contexto del problema

La evaluación de los hiperparámetros en redes convolucionales constituye una empresa esencial en el ámbito de la visión por computadora, con impacto crítico en la correcta clasificación de imágenes digitales. Este proceso se enfrenta a desafíos sustanciales, dada la complejidad inherente de las imágenes y la diversidad de contextos en los que estas redes se despliegan. La determinación de la combinación óptima de parámetros implica la consideración de factores diversos, que van desde las dimensiones de los filtros hasta la tasa de aprendizaje y la arquitectura global de la red.

La optimización de hiperparámetros resulta fundamental para facultar a la red para discernir patrones y características pertinentes en los datos de entrada, mejorando así su capacidad de clasificación. No obstante, este proceso no es trivial y demanda una comprensión profunda tanto de la naturaleza intrínseca de los datos como de la problemática específica que se aborda. La complejidad se ve exacerbada por la necesidad de considerar las interacciones entre diferentes hiperparámetros y su influencia en el rendimiento global del modelo.

Más aún, el ajuste fino de los hiperparámetros no es meramente un desafío teórico; también presenta consideraciones prácticas significativas. La búsqueda exhaustiva de la combinación óptima conlleva la necesidad de recursos computacionales sustanciales, lo que puede restringir la aplicabilidad de ciertos enfoques en entornos con limitaciones de recursos. Esta limitación práctica resalta la importancia de desarrollar métodos eficientes y escalables

para la optimización de hiperparámetros, particularmente en el contexto de las redes convolucionales.

En el meollo de esta problemática reside la tensión entre la complejidad del modelo y su capacidad para generalizar patrones relevantes en nuevos conjuntos de datos. La optimización de hiperparámetros, en consecuencia, se convierte en un ejercicio de equilibrio, en el que la selección de valores apropiados se fundamenta en un conocimiento profundo de la aplicación específica y una comprensión holística de cómo los hiperparámetros influyen en el rendimiento del modelo.

1.2 Enfoque tecnológico

En la actualidad, la tecnología desempeña un papel crucial al abordar desafíos y buscar mejoras en la vida diaria. A pesar de los notables avances tecnológicos, especialmente en el desarrollo de hardware especializado como las Unidades de Procesamiento de Redes Neuronales (NPU) y las Unidades de Procesamiento de Tensores (TPU), que agilizan el proceso de entrenamiento y la inferencia de las CNN, permitiendo aplicaciones en tiempo real y la gestión de cargas de trabajo intensivas en datos, persisten desafíos significativos, entre ellos, la optimización de los algoritmos.

El enfoque tecnológico adoptado en esta tesis se centra en la implementación de técnicas de entrenamiento más eficientes mediante la manipulación de hiperparámetros y la aplicación de transferencia de aprendizaje. Aunque se han logrado avances significativos, la mejora continua en la optimización de algoritmos sigue siendo esencial para aprovechar plenamente el potencial de las redes neuronales en diversos contextos.

1.3 Hipótesis

En esta tesis se plantean las siguientes hipótesis:

1. La variación de los hiperparámetros, incluyendo la tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador, afectará significativamente la precisión de las redes neuronales convolucionales con transferencia de aprendizaje en la clasificación del nivel de madurez de dátiles Medjool en imágenes digitales.

2. Existe una combinación específica de hiperparámetros (tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador) que maximizará el rendimiento de las redes neuronales convolucionales con transferencia de aprendizaje en la tarea de clasificar el nivel de madurez de dátiles Medjool en imágenes digitales.

Para comprobar las hipótesis anteriores, se plantean los siguientes objetivos.

1.4 Objetivos

1.4.1. Objetivo General

Comparar el rendimiento de redes neuronales convolucionales con transferencia de aprendizaje en la clasificación del nivel de madurez de dátiles Medjool en imágenes digitales, mediante la variación de sus hiperparámetros, como la tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador.

1.4.2. Objetivos Específicos

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

*Objetivo específico 1***.** Implementar redes neuronales convolucionales con transferencia de aprendizaje, variando los hiperparámetros, tales como la tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador, con el propósito específico de clasificar el nivel de madurez del dátil Medjool en imágenes digitales.

Objetivo específico 2. Evaluar el desempeño de las redes neuronales convolucionales con transferencia de aprendizaje al variar los hiperparámetros, como la tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador, en la clasificación del nivel de madurez del dátil Medjool en imágenes digitales.

1.5 Preguntas de investigación

Para alcanzar los objetivos antes planteados, se plantean las siguientes preguntas de investigación en esta tesis:

Pregunta de investigación 1.-¿Cómo afecta la variación de los hiperparámetros como la tasa de aprendizaje, número de épocas, profundidad de la red, tamaño de lote y optimizador, en la clasificación del nivel de madurez del dátil Medjool en imágenes digitales?

Pregunta de investigación 2.- ¿Cuáles son los impactos de la variación en los hiperparámetros, tales como la tasa de aprendizaje, número de épocas*, profundidad de la red, tamaño de lote y optimizador, en la clasificación del nivel de madurez del dátil Medjool en imágenes digitales?*

1.6 Metodología

Con el fin de lograr los objetivos planteados, se propuso la siguiente metodología (Figura 1):

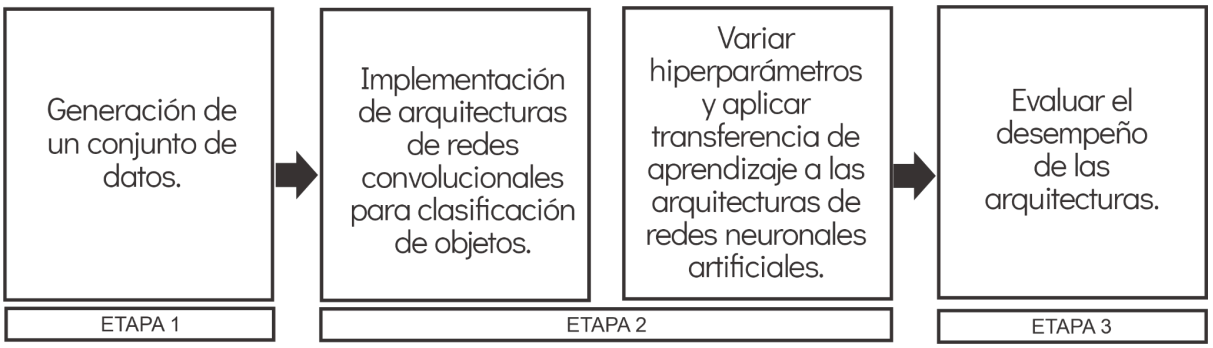

Figura 1. Metodología propuesta.

Etapa 1. En esta fase, se llevó a cabo la creación de un conjunto de datos destinado al entrenamiento de las redes neuronales convolucionales. Este conjunto de datos se compone de imágenes digitales que presentan dátiles de la variedad Medjool en dos estados de madurez: maduro y no maduro. Las imágenes abarcan diversas situaciones, como dátiles no maduros de manera individual con fondo claro, no maduros dispuestos en charolas, dátiles maduros de forma individual, y maduros en charolas. Además, se incluye la combinación de dátiles, tanto maduros como no maduros, tanto en charolas como con fondo claro.

Etapa 2. En esta fase, se procedió a la implementación de redes neuronales convolucionales, tanto con transferencia como sin ella. En cada caso, se llevó a cabo la variación de los hiperparámetros de estructura y entrenamiento. En lo que respecta a los hiperparámetros de estructura, se implementaron ocho arquitecturas distintas con variaciones en el número de capas, abarcando desde 16 hasta 152. En cuanto a los hiperparámetros de entrenamiento, se evaluaron dos optimizadores diferentes: Estimación Adaptativa de Momentos (Adam) y Estocástico de Gradiente Descendente (SGD). Además, se emplearon

dos valores diferentes de tasas de aprendizaje (0.001 y 0.01), dos tamaños de lote (64 y 128), y dos números de épocas (25 y 400).

Las implementaciones de las diversas arquitecturas de redes neuronales convolucionales que consideran la variación de la estructura incluyeron: VGG-16, VGG-19, ResNet-50, ResNet-101, ResNet-152, AlexNet, Inception V3 y CNN desde cero. La CNN desde cero consta de cuatro capas de convolución seguidas de cuatro capas de pooling, principalmente. En total, se implementaron 128 CNNs, lo que corresponde a 16 combinaciones de hiperparámetros (Optimizador Adam y SGD; Tasa de aprendizaje 0.001 y 0.01; Batch 64 y 128; Épocas 25 y 400) en las 8 arquitecturas con diferentes profundidades (desde 16 hasta 152 capas).

Etapa 3. Con el propósito de evaluar el rendimiento de las redes neuronales convolucionales implementadas, se procedió a entrenar y validar dichas redes utilizando un conjunto de datos compuesto por imágenes de dátiles de la variedad Medjool, clasificadas en dos niveles de maduración: maduro e inmaduro. En este contexto, se define como "inmaduro" a cualquier otro nivel de madurez en el cual el dátil no es apto para el consumo.

1.7 Estructura de la tesis

Esta tesis consta de seis capítulos. El primer capítulo aborda la motivación que impulsó la realización de este trabajo. En el siguiente capítulo, se exploran conceptos clave y trabajos relacionados, destacando las redes neuronales convolucionales (CNNs), así como los hiperparámetros de estructura y aprendizaje. Se aborda también el concepto de transferencia de aprendizaje aplicado a las CNNs.

En el tercer capítulo se detalla el conjunto de datos creado para el entrenamiento y la validación de las redes neuronales convolucionales, las cuales fueron generadas mediante la modificación de los hiperparámetros de estructura y entrenamiento. Además, se describe la implementación de estas redes convolucionales. Los resultados de la evaluación de estas implementaciones se exponen en el cuarto capítulo, seguidos por una discusión detallada en el quinto capítulo.

Para concluir, el capítulo 6 presenta las conclusiones derivadas de este estudio y propone posibles áreas de oportunidad para futuras investigaciones.

Capítulo 2. Marco Teórico

En este capítulo se ofrece una guía completa para comprender y aplicar redes convolucionales, desde sus componentes básicos hasta estrategias avanzadas como la transferencia de aprendizaje. Los lectores obtendrán una visión profunda de la arquitectura, los hiperparámetros clave y las estrategias prácticas para optimizar el rendimiento de estas redes en aplicaciones de visión por computadora.

2.1 Redes Neuronales Convolucionales

Las redes neuronales convolucionales, son utilizadas frecuentemente para tareas de clasificación de imágenes, detección y reconocimiento de objetos en imágenes digitales. Las CNNs logran esto a través de la extracción de características con operaciones matriciales para la identificación de patrones en una imagen. Una red neuronal convolucional es una arquitectura que contiene una capa de entrada, una o más capas ocultas y una capa de salida. En donde las capas intermedias comúnmente son de convolución y de agrupamiento (Gholamalinezhad and Khosravi, 2020) (Figura 2).

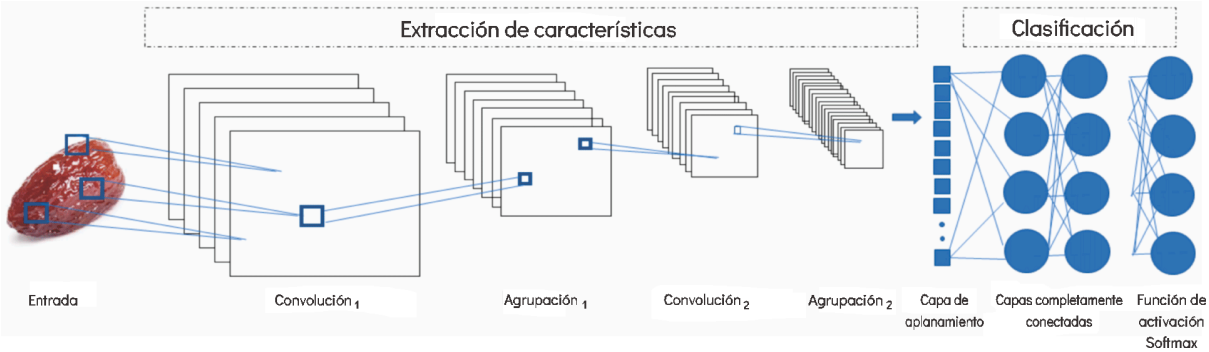

Figura 2. Estructura básica de una CNN.

Las redes neuronales convolucionales asumen explícitamente que las entradas son imágenes (Feng y Darrell, 2015), de esta manera ganan eficiencia y reducen el número de hiperparámetros de la red. Los hiperparámetros son variables de configuración externas al modelo y son especificados por el programador para ajustar el algoritmo (Torres, 2018). Así

pues, las CNN pueden modelar variaciones y comportamientos complejos proporcionando predicciones precisas de clasificación.

Al entrenar una red neuronal convolucional, se busca que un modelo pueda reconocer diversos objetos dentro de una imagen. El entrenamiento se realiza con una cantidad importante de imágenes ya que, de esta forma, las neuronas de la red extraen características únicas de cada objeto a detectar y a su vez generalizarlo.

2.2 Estructura de una Red Neuronal Convolucional

La arquitectura de una red neuronal convolucional consta de una serie de capas, cada una con un propósito específico en el proceso de aprendizaje y extracción de características, las cuales se describen a continuación.

2.2.1 Capa de entrada

Las redes neuronales artificiales asumen explícitamente que las entradas son imágenes (Feng y Darrell, 2015), no obstante, también puede procesar, señales de audio, video, texto y datos científicos (Torres, 2018). En la capa de entrada se toma la imagen o el conjunto de datos de entrada y las prepara para su procesamiento en la CNN.

2.2.2. Capa de convolución

La capa de convolución es una de las capas principales de una red neuronal convolucional. Su función principal es extraer características importantes de una imagen de entrada. Esta capa utiliza filtros, también llamados kernels, que se aplican a la imagen de entrada para extraer características específicas. Cada filtro es una matriz de números, por ejemplo: [1 0 1; 0 1 0; 1 0 1], [0 1 0; 1 -4 1; 0 1 0], [1 1 1; 1 1 1; 1 1 1], y cada filtro se desliza sobre la imagen de entrada en pequeñas ventanas, y se realiza un producto punto entre los elementos del filtro y los píxeles de la imagen que se encuentran debajo de cada ventana (Figura 3).

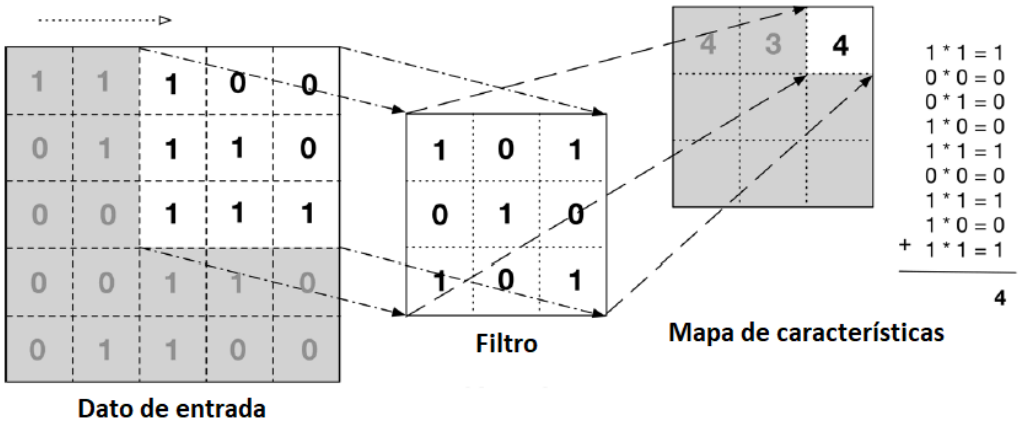

Figura 3. Operación de convolución. Fuente: Sotil, 2022.

Este proceso de convolución se repite en diferentes posiciones de la imagen de entrada, generando una matriz de salida llamada mapa de características (*feature map*) (Li *et al*., 2021). La capa de convolución también puede incluir otras operaciones, como la activación no lineal a través de una función de activación (como *ReLU*), la normalización de lotes (*batch normalization*) y la operación de agrupación (*pooling*), que reducen la dimensión del mapa de características. El número de filtros y la dimensión del kernel son parámetros de la capa de convolución que se ajustan durante el entrenamiento de la red para obtener un mejor desempeño en la tarea específica.

2.2.3 Capa de Agrupación

La capa de agrupamiento, también llamada capa de pooling, es otra capa importante en una red neuronal convolucional. Su función es reducir la dimensión del mapa de características generado por la capa de convolución, manteniendo las características más importantes.

La capa de agrupamiento utiliza ventanas de desplazamiento para dividir el mapa de características en regiones más pequeñas, y luego reduce el tamaño de cada región a través de una operación matemática, como la media (del inglés *average-pooling*) o el máximo (*Max-pooling*) (Torres, 2018).

En el caso del método de agrupación *max-pooling*, se aplica para reducir la muestra de las salidas de las operaciones convolucionales, lo que reduce la variabilidad. La agrupación máxima es una variante de submuestreo en la que se toma como salida el valor máximo de píxel de los píxeles que caen dentro del campo receptivo de una unidad dentro de

una capa de submuestreo. En la Figura 4, podemos ver un ejemplo de la operación de *max-pooling*.

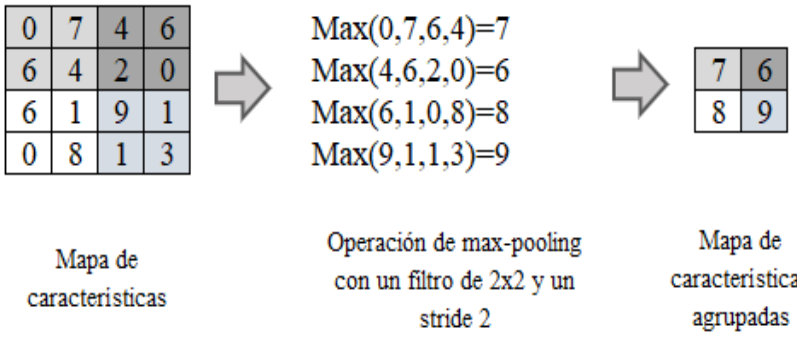

Figura 4. Ejemplo de operación max-pooling.

El método de agrupación promedio es una variante del submuestreo en la que se toma como resultado el promedio de píxeles que caen dentro del campo receptivo de una unidad, en el interior de una capa de submuestreo (Figura 5).

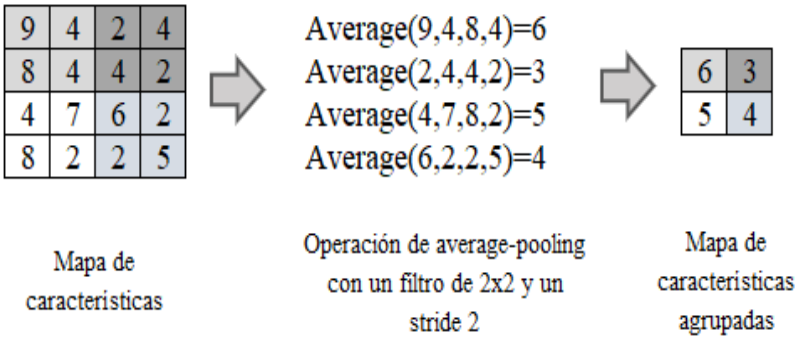

Figura 5. Ejemplo de operación average-pooling.

El agrupamiento puede realizarse en diferentes dimensiones, como en la altura y el ancho de las características, o en la profundidad (número de canales de características). La elección de la dimensión de agrupamiento y el tamaño de la ventana son hiperparámetros de la capa de agrupamiento, que se ajustan durante el entrenamiento de la red.

La capa de agrupamiento tiene varias ventajas (Gholamalinezhad y Khosravi, 2020; Zafar *et al.,* 2022), entre ellas:

- Reduce la cantidad de parámetros de la red, lo que disminuye el tiempo de entrenamiento y reduce la posibilidad de sobreajuste.

- Ayuda a hacer que la red sea invariante a pequeñas variaciones en la posición de los objetos en la imagen, ya que las características importantes se mantienen independientemente de su ubicación exacta.
- Reduce el efecto de ruido y la redundancia en los datos, mejorando la generalización del modelo.

2.2.4 Funciones de activación

La función de activación se utiliza para aumentar la capacidad de expresión del modelo de una red neuronal; se refiere a que las neuronas activas pueden retenerse y mapearse mediante una función no lineal, que puede usarse para resolver problemas no lineales. La función de activación es el núcleo de la estructura de una red neuronal profunda, las funciones de activación comunes incluyen: sigmoid, tanh, ReLU y softplus (Wang *et al*., 2020) (Figura 6).

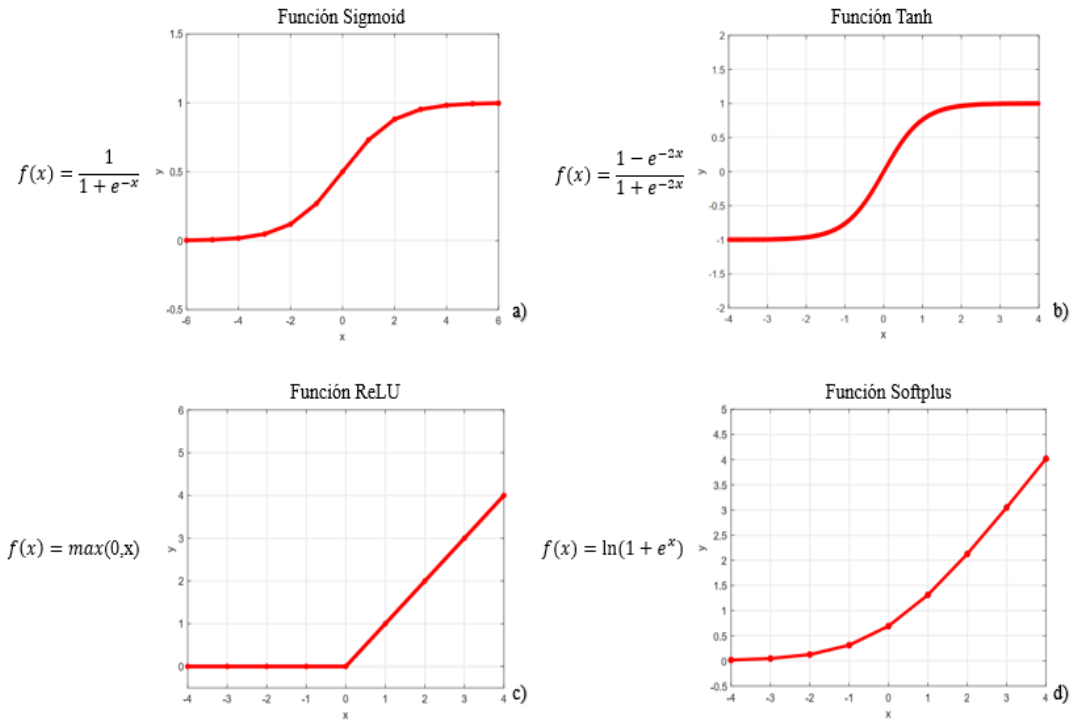

Figura 6. Curva de funciones de activación comúnmente usadas. Fuente: Wang et al. 2020.

2.2.4.1 Unidad Lineal Rectificada (ReLU)

ReLU (del inglés *Rectified Linear Unit*) es una función de activación utilizada en redes neuronales artificiales que se aplica a la salida de una capa de neuronas. La función *ReLU* se define matemáticamente como (He *et al*., 2016):

$$f(x) = max(0, x) \qquad (1)$$

Donde x es la entrada a la función y f(x) es la salida. La gráfica de la función *ReLU* se presenta en la Figura 7.

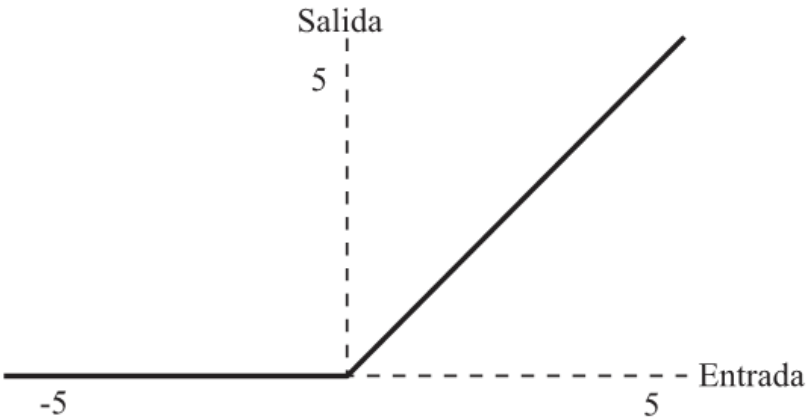

Figura 7. Unidad Lineal Rectificada (ReLU).

Como se puede visualizar en la Figura 7, la función *ReLU* devuelve el valor de entrada si es mayor que cero, y devuelve cero en caso contrario. Esto significa que la función *ReLU* activa (o "enciende") las neuronas que tienen una entrada positiva, mientras que desactiva (o "apaga") las neuronas que tienen una entrada negativa.

La función *ReLU* se ha vuelto muy popular en las redes neuronales debido a su simplicidad y eficiencia computacional, y a su capacidad para mejorar el rendimiento de la red (Zhao *et al*., 2018). Algunas de las ventajas de la función *ReLU* son:

- Es fácil de implementar y computacionalmente eficiente.
- Permite un aprendizaje más rápido de la red debido a su naturaleza no lineal.
- Ayuda a evitar el problema del desvanecimiento del gradiente, que es común en otras funciones de activación.

La función *ReLU* se utiliza comúnmente en la capa de activación de las redes neuronales profundas, especialmente en las redes convolucionales. También hay variantes de la función *ReLU* (Dubey *et al.*, 2022), como la *ReLU* truncada (que establece un valor

máximo para los valores de entrada negativos) y la *Leaky ReLU* (que evita la activación completa de las neuronas negativas estableciendo un valor pequeño para ellas).

2.2.5 Capa de aplanamiento

La capa de aplanamiento (también conocida como capa de aplanado o *Flatten layer*) es una capa comúnmente utilizada en las redes neuronales convolucionales que transforma un tensor multidimensional de características en un vector unidimensional para que pueda ser procesado por las capas completamente conectadas (o densas) de la red.

La capa de aplanamiento se sitúa al final de la capa de convolución y antes de las capas completamente conectadas. Su función es reorganizar el tensor de características generado por la capa de convolución en una matriz unidimensional o vector, que luego se conecta a una o varias capas densas para realizar la clasificación final.

Por ejemplo, después de aplicar varias capas de convolución y agrupamiento en una imagen de entrada, se genera un tensor multidimensional de características. Este tensor se pasa a la capa de aplanamiento, que lo transforma en un vector unidimensional. Este vector se pasa a las capas densas que realizan la clasificación final.

La capa de aplanamiento no tiene parámetros entrenables, su función es puramente mecánica, es decir, simplemente transforma el tensor de características en un vector unidimensional (Figura 8).

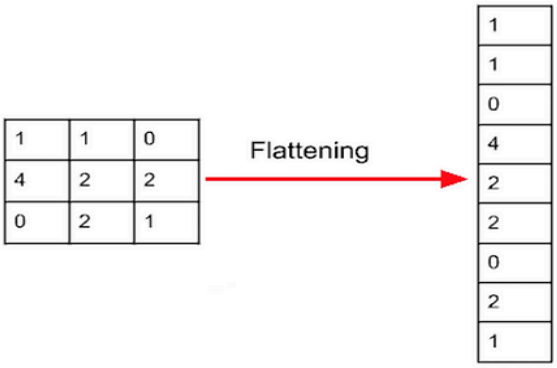

Figura 8. Transformación del tensor de características a un vector unidimensional. Fuente: Suriya et al. 2022.

2.2.6 Capas totalmente conectadas

Las capas completamente conectadas (o capas densas) en una red neuronal convolucional son las últimas capas de la red, que reciben la entrada de la capa de aplanamiento y realizan la clasificación final.

A diferencia de las capas de convolución, que procesan las características locales de la imagen, las capas completamente conectadas procesan todas las características de la imagen a la vez y utilizan estas características para realizar la clasificación.

Cada neurona en una capa densa está conectada a todas las neuronas de la capa anterior. Estas conexiones se representan mediante pesos y sesgos que se aprenden durante el entrenamiento de la red. Los pesos determinan la importancia relativa de cada neurona de entrada para la salida de cada neurona de salida, mientras que los sesgos se utilizan para ajustar la salida de cada neurona.

La salida de cada neurona en una capa densa se calcula utilizando una función de activación, como la función *ReLU* o la función *sigmoide*. La función de activación ayuda a introducir la no linealidad en la red y permite que la red aprenda representaciones más complejas de las características de entrada.

2.3 Arquitecturas de Redes Neuronales Convolucionales

Una Red Neuronal Convolucional es un modelo computacional basado en la estructura y funcionamiento de una neurona biológica. Está formada por capas interconectadas que procesan información. Se componen de una capa de entrada, una o más capas ocultas, con una capa de salida. Actualmente, las redes neuronales han cobrado gran interés por parte de la comunidad científica ya que son capaces de resolver problemas complejos de manera efectiva, son herramientas muy efectivas en el aprendizaje automático y la inteligencia artificial. Las redes neuronales están siendo ampliamente utilizadas en áreas de visión computacional, reconocimiento de voz, robótica y procesamiento del lenguaje natural, entre otras. Las arquitecturas utilizadas en este trabajo fueron:

2.3.1 VGG-16

La red VGG-16 es una red neuronal convolucional profunda (CNN) que fue desarrollada por el grupo de investigación Visual Geometry Group (VGG) en la Universidad

de Oxford en 2014 (Zaccone y Karim 2018; Simonyan y Zisserman, 2015). Esta red es una de las más populares y utilizadas para la clasificación de imágenes.

La arquitectura de la red VGG-16 consta de 16 capas de convolución y capas completamente conectadas (Simonyan y Zisserman, 2015). Las capas de convolución se encargan de extraer características de las imágenes de entrada y las capas completamente conectadas realizan la clasificación final.

La red VGG-16 tiene una estructura de bloques repetitivos, donde cada bloque consiste en múltiples capas convolucionales seguidas de una capa de Max Pooling, que reduce el tamaño de la imagen a la mitad. El primer bloque tiene dos capas convolucionales, mientras que los tres bloques restantes tienen tres capas convolucionales cada uno (Simonyan y Zisserman, 2015).

La red VGG-16 utiliza filtros de tamaño reducido (3x3) para las capas convolucionales, lo que permite que la red tenga una profundidad mayor con un número menor de parámetros. La capa de Max Pooling utiliza un filtro de 2x2.

La red VGG-16 tiene 13 capas convolucionales y 3 capas completamente conectadas. Las primeras 13 capas convolucionales forman el extracto de características de la red, mientras que las últimas tres capas completamente conectadas se encargan de la clasificación final. La red utiliza la función de activación ReLU (Rectified Linear Unit) en todas sus capas, excepto en la última capa completamente conectada, que utiliza la función Softmax para producir la distribución de probabilidad de las clases de salida (Simonyan y Zisserman, 2015). En la Figura 9, podemos observar un ejemplo de arquitectura VGG-16.

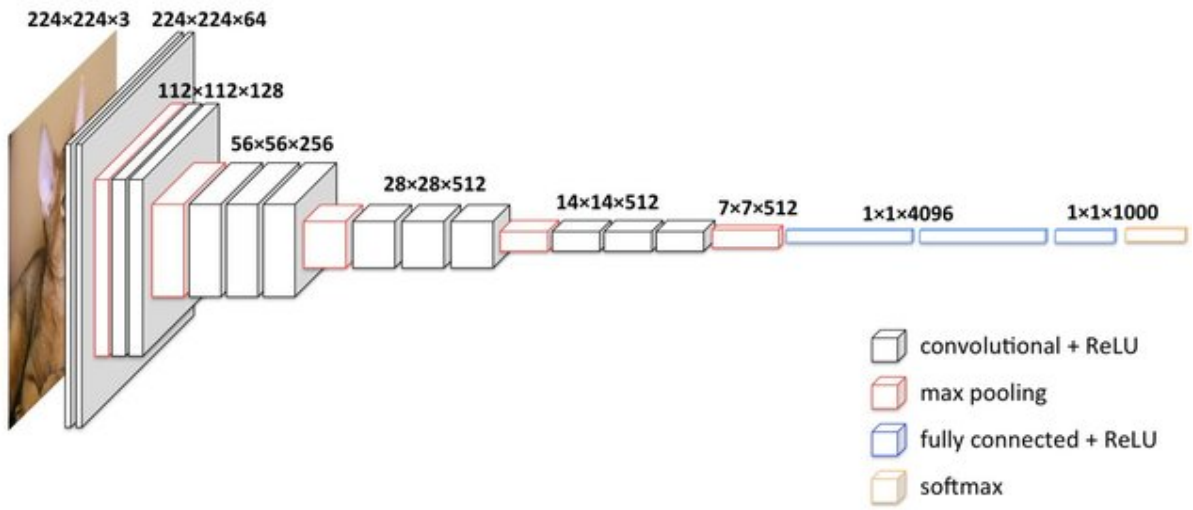

Figura 9. Ejemplo de arquitectura del modelo VGG-16 para representar diferentes niveles de profundidad, este ejemplo cuenta con cinco capas convolucionales antes de la agrupación máxima en cada grupo. Fuente: (Shi et al., 2018).

2.3.2 VGG-19

La red VGG-19 es una red neuronal convolucional profunda (CNN) la cual es una versión más grande y profunda de la red VGG-16. Fue desarrollada por el grupo de investigación Visual Geometry Group (VGG) en la Universidad de Oxford en 2014.

La arquitectura de la red VGG-19 es similar a la de la VGG-16, pero consta de 19 capas en lugar de 16. Al igual que la VGG-16, utiliza bloques repetitivos de capas convolucionales seguidas de una capa Max Pooling. Sin embargo, en la VGG-19, hay cuatro bloques de capas convolucionales en lugar de tres (Simonyan y Zisserman, 2015).

La red VGG-19 utiliza filtros de tamaño reducido (3x3) para las capas convolucionales, al igual que la VGG-16. También utiliza capas de Max Pooling con un filtro de 2x2 para reducir el tamaño de la imagen de entrada a la mitad después de cada bloque de capas convolucionales (Simonyan y Zisserman, 2015).

La red VGG-19 tiene un total de 19 capas, que consisten en 16 capas convolucionales y 3 capas completamente conectadas. Las primeras 16 capas convolucionales se utilizan para extraer características de la imagen, y las últimas tres capas completamente conectadas se utilizan para la clasificación final.

La VGG-19 utiliza la función de activación ReLU en todas sus capas convolucionales y completamente conectadas, excepto en la última capa completamente conectada, que utiliza la función Softmax para producir la distribución de probabilidad de las clases de salida (Simonyan y Zisserman, 2015).

2.3.3 Inception V3 (GoogLeNet V3)

Esta arquitectura nació con el nombre de GoogLeNet, pero las actualizaciones posteriores se han llamado Inception vN, donde N se refiere al número de versión publicado por Google (Zaccone y Karim, 2018).

La arquitectura de la red Inception v3 utiliza módulos de Inception que combinan convoluciones de diferentes tamaños de kernel en paralelo, lo que permite que la red capture características de diferentes escalas espaciales. Cada módulo de Inception se compone de varias ramas de convolución que se combinan a través de concatenación en la salida (Szegedy *et al.*, 2016).

Además, Inception v3 utiliza técnicas avanzadas de regularización y normalización, como la normalización por lote (Batch Normalization) y la regularización L2 para evitar el sobreajuste en los datos de entrenamiento.

La arquitectura Inception v3 tiene 48 capas convolucionales, con una gran cantidad de parámetros entrenables que la hacen más profunda y compleja que su predecesora, la Inception v2. También incluye una capa de Global Average Pooling al final para reducir el tamaño de la salida antes de la capa de clasificación (Szegedy *et al.*, 2016).

Inception v3 también utiliza una técnica llamada "Factorización espacial de convoluciones", que reduce la cantidad de parámetros de la red y mejora la eficiencia computacional sin comprometer la precisión de la clasificación (Szegedy *et al.*, 2016).

2.3.4 Arquitectura ResNet-50

ResNet-50 es una red neuronal convolucional profunda (CNN) desarrollada por Microsoft que se utiliza principalmente para la clasificación de imágenes (Zaccone y Karim, 2018). Fue lanzada en 2015 y es una mejora de la arquitectura ResNet-34 (He *et al*., 2016).

La arquitectura de ResNet-50 utiliza bloques residuales para permitir que la red aprenda representaciones más profundas y precisas de las imágenes de entrada. Los bloques

residuales contienen capas convolucionales que están conectadas por una "ruta de salto" que permite que la entrada original se agregue a la salida de las capas convolucionales (He *et al*., 2016).

ResNet-50 tiene 50 capas convolucionales, que incluyen bloques residuales y capas de agrupación máxima. También utiliza técnicas avanzadas de regularización y normalización, como la normalización por lotes y la regularización L2, para evitar el sobreajuste en los datos de entrenamiento (He *et al*., 2016).

En la capa de clasificación, ResNet-50 utiliza una capa de agrupación global promedio para reducir la dimensionalidad de la salida de la red antes de la clasificación. También utiliza una capa completamente conectada con una función de activación softmax para generar la probabilidad de cada clase de imagen (He *et al*., 2016).

2.3.5 Arquitectura ResNet-101

ResNet-101 es una variante de la arquitectura ResNet en (He *et al*., 2016). Al igual que ResNet-50, ResNet-101 utiliza bloques residuales para permitir el entrenamiento de redes neuronales profundas. La diferencia principal es que ResNet-101 tiene una arquitectura más profunda que ResNet-50, con un total de 101 capas.

ResNet-101 utiliza una combinación de bloques residuales simples y bloques residuales en botella. En los bloques residuales simples, se aplica una capa convolucional seguida de una función de activación ReLU y otra capa convolucional, y luego se agrega la entrada original a la salida de las capas convolucionales (He *et al*., 2016). En los bloques residuales en botella, se utiliza una estructura de tres capas: una capa convolucional con un tamaño de kernel de 1x1, una capa convolucional con un tamaño de kernel de 3x3 y otra capa convolucional con un tamaño de kernel de 1x1, seguida de la adición de la entrada original (He *et al*., 2016).

ResNet-101 también utiliza una técnica de agrupamiento por canal para reducir la dimensionalidad de las características en las capas convolucionales. Además, utiliza una técnica de normalización por lote para mejorar la generalización de la red (He *et al*., 2016).

2.3.6 Arquitectura ResNet-152

ResNet-152 es una variante de la arquitectura ResNet que consta de 152 capas (He *et al*., 2016). Al igual que ResNet-50 y ResNet-101, ResNet-152 utiliza bloques residuales para permitir el entrenamiento de redes neuronales profundas. Sin embargo, ResNet-152 es aún más profunda que ResNet-101, con una arquitectura de 152 capas (He *et al*., 2016).

ResNet-152 utiliza una combinación de bloques residuales simples y bloques residuales en botella, al igual que ResNet-101. También utiliza técnicas de agrupamiento por canal y normalización por lote para mejorar la generalización de la red (He *et al*., 2016).

Una diferencia importante entre ResNet-152 y ResNet-101 es que ResNet-152 utiliza bloques residuales en botella más profundos, con 50 capas convolucionales en lugar de 23 en ResNet-101. Esto permite a la red aprender características más complejas y facilita el entrenamiento de una red más profunda (He *et al*., 2016).

2.3.7 AlexNet

AlexNet es una arquitectura de red neuronal convolucional profunda (CNN) desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton en 2012, que fue diseñada para clasificar imágenes en el conjunto de datos ImageNet.

La arquitectura de AlexNet consta de ocho capas, cinco de las cuales son capas convolucionales y las tres últimas son capas completamente conectadas (ver Figura 10). Las capas convolucionales utilizan diferentes tamaños de kernel y funciones de activación no lineales, como la función ReLU (del inglés Rectified Linear Unit), para extraer características de las imágenes de entrada (Krizhevsky *et al.*, 2017).

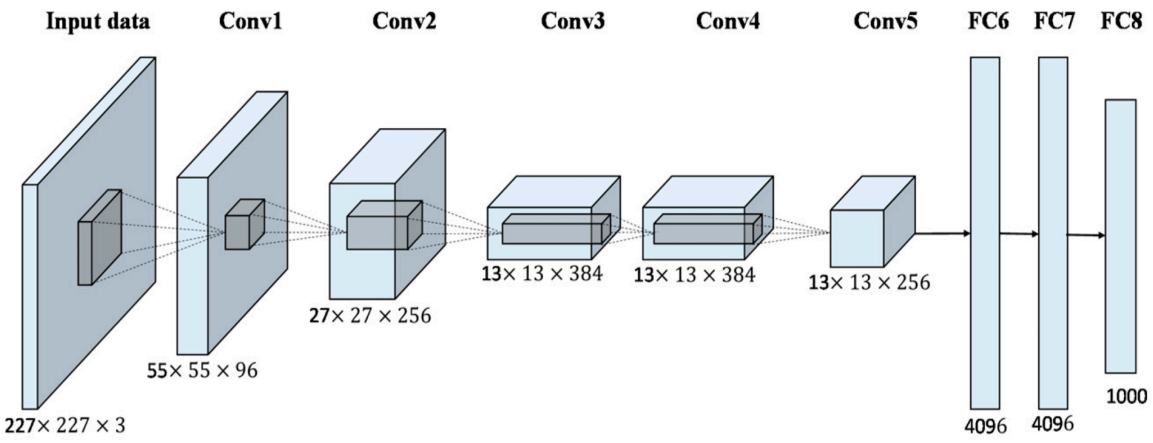

Figura 10. Arquitectura AlexNet, en esta figura, los autores muestran el uso de dos GPU's. Fuente: (Han et al.2017).

Además, AlexNet utiliza técnicas de regularización, como la regularización L2 y el Dropout, para evitar el sobreajuste y mejorar la generalización de la red. También utiliza la técnica de agrupamiento máximo para reducir la dimensionalidad de la salida de las capas convolucionales y acelerar el proceso de entrenamiento (Krizhevsky *et al.,* 2017).

La capa final de la red es una capa completamente conectada con una función de activación softmax que produce la probabilidad de cada clase de imagen. La red utiliza la función de pérdida de entropía cruzada para calcular el error de clasificación y actualizar los pesos de la red mediante el descenso del gradiente estocástico (Krizhevsky *et al.,* 2017).

2.4 Hiperpámetros

Al desarrollar un modelo de clasificación basado en redes convolucionales, implica ajustar su configuración, por ejemplo, elegir el número de capas o tamaño de las capas, el número de épocas o el optimizador a utilizar, a esto se le denomina hiperparámetros del modelo. Este ajuste se realiza principalmente en función del rendimiento del modelo. Dicho de otra manera, el ajuste de hiperparámetros es la configuración del algoritmo a utilizar. Los hiperparámetros son los valores numéricos de las variables que se establecen antes de la aplicación real a los datos, los cuales no se seleccionan directamente por el propio algoritmo de aprendizaje sino por el especialista, a diferencia de los parámetros (pesos) que se ajustan automáticamente por el algoritmo (Bengio, 2012). Pueden ser discretos (p. ej. la selección del algoritmo, el tamaño del kernel o el número de capas, número de épocas, entre otros) o continuos (p. ej. la tasa de aprendizaje) (Cui y Bai, 2019; Bengio, 2012). Cabe mencionar que éstos hiperparámetros se pueden ajustar a mano o mediante un algoritmo como Grid Search (Liashchynskyi y Liashchynskyi, 2019; Shekar and Dagnew, 2019), random search (Bergstra

y Bengio, 2012), Bayesian optimization y Nelder-Mead Methods (Ozaki *et al*., 2017), pero se debe seleccionar el valor de cada uno a utilizar. Podemos ver entonces que, para el correcto entrenamiento de una red neuronal convolucional, se necesita realizar el ajuste de hiperparámetros.

De acuerdo a Bengio (2012), los diferentes algoritmos de aprendizaje involucran diferentes conjuntos de hiperparámetros. Entre ellos se encuentran los hiperparámetros aproximados a la optimización, en los que podemos encontrar: la tasa de aprendizaje inicial, lote mínimo de datos *(mini-batch size*), número de iteraciones para entrenamiento, el momentum e hiperparámetros de optimización específicos de cada capa. Otro conjunto de hiperparámetros son los del modelo y criterios de entrenamiento. En este conjunto encontramos: el número de unidades ocultas (del inglés *Number of Hidden Units*), decaimiento de peso, el coeficiente de escala de inicialización de pesos y la aleatorización de semillas (*Random seeds*). Por otro lado, Torres (2018), define a los hiperparámetros como variables que definen la estructura de una red neuronal y permiten entrenarla. Los hiperparámetros que considera como parte de la estructura y tipología de la red neuronal son: el número de capas, número de neuronas y funciones de activación. En cuanto a los hiperparámetros de entrenamiento considera: la tasa de aprendizaje, el optimizador, el número de épocas y el tamaño de lote.

Con el ajuste de los hiperparámetros se busca que el modelo tenga el mejor rendimiento, siendo este ajuste automático, a través de un algoritmo, o bien manualmente ajustando cada hiperparámetro o varias combinaciones de ellos o bien, por familias de hiperparámetros. En la Tabla 1, se muestran algunos trabajos en los que han realizado el ajuste de hiperparámetros tanto de estructura como de entrenamiento. En el caso del ajuste manual, en el trabajo realizado por Kandel *et al*., (2020), se estudia el efecto del tamaño del lote en el rendimiento de redes neuronales convolucionales y el impacto de las tasas de aprendizaje para la clasificación de imágenes médicas. Para ello, los autores utilizan una red VGG16 con cinco valores de tamaño de lote (16, 32, 64, 128 y 256), dos valores de tasa de aprendizaje (0.0001 y 0.001) y los optimizadores SGD y Adam. Los autores señalan que un tamaño de lote más alto, generalmente no logra una alta precisión, y cuando el tamaño de lote es muy alto, puede hacer que la red tarde demasiado en lograr la convergencia sin tener mayor ganancia en su precisión. De la misma manera, la tasa de aprendizaje y el optimizador usado también tendrán un impacto significativo. Concluyen que al reducir la tasa de

aprendizaje y disminuir el tamaño de lote permiten que la red entrene mejor. En otro estudio realizado por Radiuk (2017), se exploró el efecto del tamaño del lote en una Red Neuronal Convolucional en la clasificación de imágenes utilizando dos bases de datos MNIST y CIFAR-10, así como el de una arquitectura LeNet con dos secuencias de valores del tamaño del lote, números elevados a dos (16, 32, 64, 128, 256, 512, 1024) y números múltiplos de diez (50, 100, 150). En total, se seleccionaron 12 valores de tamaño de lote, con un optimizador SGD. De acuerdo a sus resultados, el autor afirma que en cuanto mayor sea el valor del tamaño de lote, mayor será la precisión del reconocimiento de imágenes. Así mismo afirma que en cuanto mayor sea el valor del tamaño del lote, más tiempo se requiere para entrenar la red. En el trabajo realizado por Nguyen y Lee (2018), evaluaron hiperparámetros de entrenamiento como la tasa de aprendizaje (0.001 a 0.1), el optimizador (SGD y Momentum) y el tamaño del batch (32 y 64). Utilizaron cuatro bases de datos (CIFAR10, CIFAR 100, GTSRB Y DSDL-DB) para evaluar dos arquitecturas VGG-16 y la que proponen los autores. Los autores concluyen que la base de datos tiene una influencia significativa en la precisión de la red convolucional. En el trabajo realizado por Koutsoukas *et al.* (2017) exploraron diferentes configuraciones de hiperparámetros tanto seleccionadas arbitrariamente como con los métodos superficiales Naïve Bayes, K-vecinos más cercano, bosques aleatorios y máquinas de soporte vectorial, así como realizaron la comparación de sus rendimientos. Este estudio se centró en hiper parámetros de estructura como de entrenamiento, algunos hiperparámetros fueron: función de activación, unidad lineal rectificada (ReLU), Sigmoide (Sigm) y Tanh; tasa de aprendizaje; número de neuronas por capa; número de capas ocultas; regularización de dropout. Los autores concluyeron que la función de activación ReLu tuvo un desempeño mejor sobre Sigmoide y Tanh. Así como señalan que el aumento en el número de capas ocultas tiene rendimientos decrecientes. Finalmente señalan que los hiper parámetros del número de neuronas, la técnica de regularización de dropout y el número de neuronas por capa tiene impacto en el rendimiento de la red neuronal.

Tabla 1. Trabajo relacionado con ajuste de hiperparámetros.

Referencia	Hiperparámetros						
	Estructura			Entrenamiento			
	Número de capas	Número de Neuronas	Funciones de activación	Tasa de aprendizaje	Optimizador	Épocas	Batch size
Pérez-Pérez, B. D., Garcia Vazquez, J. P., & Salomón-Torres, R. (2021). Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting of ripe medjool dates. Agriculture, 11(2), 115.	X		X	X	X	X	X
Nguyen, H. N., & Lee, C. (2018). Effects of Hyper-parameters and Dataset on CNN Training. Journal of IEEE, 22(1), 14-20.				X	X		X
Koutsoukas, A., Monaghan, K. J., Li, X., & Huan, J. (2017). Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. Journal of cheminformatics, 9(1), 1-13.		X	X	X	X	X	X
Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT express, 6(4), 312-315.	X			X	X	X	X
Radiuk, P. M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Information Technology and Management Science, 20(1), 20-24.	X		X	X	X	X	X

2.4.1 Hiperparámetros de arquitectura

Los hiperparámetros de arquitectura en una red neuronal convolucional (CNN) son aquellos parámetros que definen la estructura y configuración de la red.

- **Número de capas.** Sirven para la extracción de características. Aplicando capas sucesivas de kernels la red puede aprender a extraer mapas de características de mayor nivel de abstracción, refinando secuencialmente los patrones observados en cada capa (Figura 11 y Figura 12).

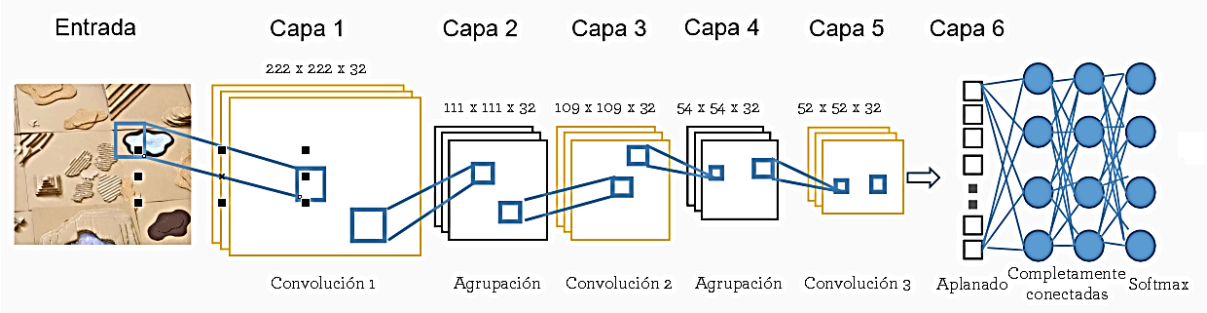

Figura 11. Ejemplo de una CNN de seis capas, tres capas convolucionales, dos capas de agrupación y una capa de aplanado.

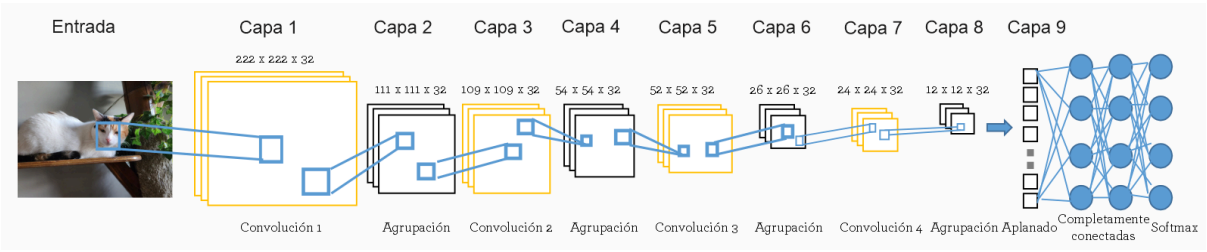

Figura 12. Ejemplo de una CNN de nueve capas, cuatro capas convolucionales, cuatro capas de agrupación y una capa de aplanado.

- **Tamaño de la capa oculta.** Este parámetro define el número de neuronas en la capa oculta de la red. Una capa oculta más grande puede capturar más características y aumentar la precisión del modelo, pero también puede aumentar el costo computacional y provocar sobreajuste.
- **Tamaño de la ventana de convolución (o kernel).** Este parámetro define el tamaño de la ventana de convolución que se desliza sobre la imagen de entrada para calcular las características. Un tamaño de ventana más grande puede detectar características más grandes en la imagen, pero también aumentará el costo computacional.
- **Tamaño del paso (stride)**. Este parámetro define el número de píxeles que se mueve la ventana de convolución en cada paso. Un tamaño de paso más grande reducirá el tamaño de salida y reducirá el costo computacional, pero también puede perder información importante en la imagen.
- **Tamaño del filtro de pooling.** Este parámetro define el tamaño de la ventana de pooling que se utiliza para reducir el tamaño de la salida de la capa de

convolución. Un tamaño de filtro de pooling más grande reducirá aún más el tamaño de salida, pero también puede perder información importante.

- **Número de filtros.** Este parámetro define el número de filtros de convolución que se aplican en cada capa. Cuanto mayor sea el número de filtros, más características diferentes puede detectar la capa, pero también aumentará el costo computacional.
- **Función de activación.** Este parámetro define la función matemática utilizada para introducir no linealidad en la red. La función de activación más común es ReLU (Rectified Linear Unit), pero también se utilizan otras funciones como tanh y sigmoide.
- **Dropout**. Este parámetro es una técnica de regularización que reduce el sobreajuste en la red. Dropout elimina aleatoriamente una fracción de las neuronas de la capa oculta en cada iteración de entrenamiento, lo que obliga a la red a aprender características más robustas.
- **Número de neuronas.** Cada neurona, realiza una operación sencilla y está conectada a las neuronas de la capa anterior y de la capa siguiente mediante pesos, cuya función es regular la información que se propaga de una neurona a otra.

2.4.2 Hiperparámetros de entrenamiento

Los hiperparámetros de entrenamiento son aquellos parámetros que se utilizan para controlar el proceso de entrenamiento de una red neuronal y pueden tener un gran impacto en la precisión del modelo resultante. A continuación, se describen algunos de los hiperparámetros de entrenamiento más comunes:

- **Tasa de aprendizaje (learning rate).** Este parámetro controla la cantidad de ajuste que se realiza en los pesos de la red durante el entrenamiento. Una tasa de aprendizaje alta puede provocar que la red se salte los mínimos locales y nunca converja, mientras que una tasa de aprendizaje baja puede hacer que el entrenamiento sea lento y lleve mucho tiempo.
- **Tamaño del lote (batch size)**. Este parámetro determina el número de ejemplos de entrenamiento que se utilizan en cada iteración de entrenamiento. Un tamaño de lote más grande puede acelerar el proceso de entrenamiento,

pero también puede requerir más memoria y reducir la capacidad de la red para generalizar.

- **Número de épocas (epochs).** Este parámetro determina el número de veces que se va a pasar por todo el conjunto de datos de entrenamiento. Un número de épocas demasiado bajo puede hacer que la red no aprenda lo suficiente, mientras que un número de épocas demasiado alto puede provocar sobreajuste.
- **Regularización**. Los métodos de regularización, como L1, L2 y Dropout, se utilizan para reducir el sobreajuste en la red. L1 y L2 penalizan los pesos grandes, mientras que Dropout elimina aleatoriamente una fracción de las neuronas en cada iteración de entrenamiento.
- **Momento (momentum).** Este parámetro controla la cantidad de impulso que se aplica a la actualización de los pesos en cada iteración de entrenamiento. El momentum puede ayudar a acelerar el proceso de entrenamiento y evitar mínimos locales, pero también puede provocar oscilaciones.
- **Función de pérdida (loss function).** Este parámetro define la función matemática que se utiliza para evaluar la precisión del modelo durante el entrenamiento. La función de pérdida más común es la entropía cruzada, pero también se utilizan otras funciones como el error cuadrático medio.
- **Optimizador**. Este parámetro define el algoritmo que se utiliza para actualizar los pesos de la red durante el entrenamiento. Los optimizadores más comunes son Adam, RMSprop y Stochastic Gradient Descent (SGD) (Li *et al.,* 2021).

2.5 Transferencia de Aprendizaje

La transferencia de aprendizaje es un concepto en el campo del aprendizaje automático y la inteligencia artificial que se refiere a la aplicación de conocimientos y habilidades adquiridos durante un proceso de aprendizaje previo para resolver un nuevo problema o tarea (Torrey *et al.,* 2010).

En otras palabras, la transferencia de aprendizaje se produce cuando se utiliza el conocimiento previo para facilitar la adquisición de nuevos conocimientos o habilidades (Pan y Yang, 2010; Lu *et al.*, 2015). Este proceso se basa en la idea de que las habilidades y conocimientos adquiridos durante una tarea o entrenamiento pueden ser aplicados en una tarea diferente pero relacionada, lo que resulta en una mejora en la velocidad y precisión del aprendizaje (Dehghani *et al*., 2021).

Por ejemplo, si un modelo de aprendizaje automático ha sido entrenado para clasificar imágenes de automóviles, podría utilizar los conocimientos adquiridos en este proceso para clasificar imágenes de camiones o motocicletas. En este caso, el modelo ha transferido su conocimiento de la clasificación de imágenes de automóviles a la clasificación de imágenes de otros tipos de vehículos.

La transferencia de aprendizaje es importante en el aprendizaje automático, ya que puede reducir la cantidad de datos necesarios para entrenar un modelo y mejorar la precisión de las predicciones (Dehghani *et al*., 2021). También puede reducir el tiempo necesario para entrenar un modelo, lo que es especialmente importante en entornos donde el tiempo de entrenamiento es crítico (Liu y Wang, 2021).

Existen diversas formas de aplicar el aprendizaje por transferencia (Niu *et al*., 2020):

Transferencia de aprendizaje basada en características. Este tipo de transferencia de aprendizaje implica utilizar las características aprendidas por una red neuronal pre-entrenada en una tarea de clasificación de imágenes para una nueva tarea de clasificación de imágenes similar. Se toman las características de las capas intermedias de la red pre-entrenada y se utilizan como entradas para una nueva red neuronal para la nueva tarea de clasificación de imágenes.

Transferencia de aprendizaje basada en modelos. Este tipo de transferencia de aprendizaje implica utilizar un modelo completo pre-entrenado para una tarea similar y reajustar o "sintonizar" el modelo para la nueva tarea. El modelo pre-entrenado se utiliza como punto de partida y se ajusta para la nueva tarea mediante la adición de capas adicionales y el reentrenamiento de la red para la nueva tarea.

Transferencia de aprendizaje de múltiples tareas. En este tipo de transferencia de aprendizaje, se utiliza una red neuronal pre-entrenada para múltiples tareas, y luego se utiliza el conocimiento adquirido en todas las tareas para mejorar la precisión en una tarea específica. La red neuronal pre-entrenada puede haber sido entrenada en tareas de clasificación de imágenes, detección de objetos, segmentación de imágenes, entre otros.

Transferencia de aprendizaje de dominio cruzado. En este tipo de transferencia de aprendizaje, se utiliza un conjunto de datos en un dominio fuente para entrenar una red neuronal, y luego se utiliza la red neuronal para una tarea de clasificación de imágenes en un

dominio objetivo. Por ejemplo, se puede utilizar una red neuronal pre-entrenada en imágenes de la naturaleza para clasificar imágenes de la ciudad.

En este trabajo de tesis se consideró utilizar Transferencia de aprendizaje basada en características.

Capítulo 3. Experimento

3.1 Conjunto de Datos

En la Figura 13, se describe el proceso que se siguió para la construcción del conjunto de datos utilizado en esta tesis. Como primer paso fue la captura de imágenes, seguido del filtrado y separación de imágenes por categorías, siendo maduros, no maduros. Posteriormente se renombraron y enlistan las imágenes de acuerdo con su categoría para finalmente realizar las anotaciones o etiquetado de cada fruto que aparecía en cada imagen.

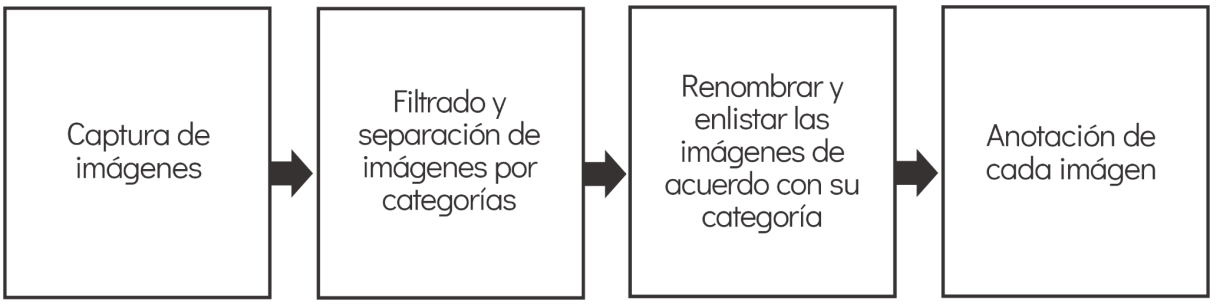

Figura 13. Diagrama de flujo del proceso en el desarrollo de un conjunto de datos.

El conjunto de datos consistió de imágenes de dátiles variedad Medjool en bandejas, las cuales fueron tomadas en el mes de septiembre de 2020, durante la primera ronda de cosecha de dátiles Medjool en la plantación ubicada en la Colonia La Herradura (32°36′56″ N, 115°15′36″ O) en el Valle de Mexicali, Baja California México (Figura 14).

Figura 14. Ubicación de la fuente de los datos, Rancho Palmeras RQ (32°36'56"N, 115°15'36"W).

3.1.1 Captura de imágenes

La adquisición de imágenes se realizó utilizando tres cámaras diferentes. Una cámara Canon EOS Rebel T6 y las cámaras de teléfonos inteligentes de la marca Samsung modelos SM-N950F (Note 8) y SM-N960F (Note 9). Las especificaciones de las cámaras utilizadas se presentan en la Tabla 2.

Tabla 2. Especificaciones de las cámaras.

Características	**Dispositivos**		
	Canon EOS Rebel T6	**Samsung Galaxy**	
		Note 8 modelo SM-N950F	**Note 9 modelo SM-N960F**
Resolución	18 megapíxeles	Cámara dual de 12 megapíxeles	Cámaras dobles de 12 megapíxeles
Unidad de píxel	Cuadrado de 4,30 µm	Cámara de 1,0 µm cuadrada/teleobjetivo	Cámara de 1,0 µm cuadrada/teleobjetivo
		Cámara cuadrada/gran angular de 1,4 µm	Cámara cuadrada/gran angular de 1,4 µm

Las imágenes fueron tomadas utilizando luz natural entre las 8:00 a.m. y las 2:00 p.m. Las imágenes incluyen dátiles variedad Medjool maduros e inmaduros. El esquema de

colores utilizado para las imágenes es RGB (del inglés *Red, Green, Blue*). El tamaño de las imágenes es variable, hay imágenes con tamaño 5184 × 3456, 4449 × 3071 y 4376 × 3375 píxeles. Todas las imágenes se encuentran en formato de archivo de imagen fotográfica llamado JPG (del inglés *Joint Photographic Experts Group*).

La organización del conjunto de datos es descrita en la Tabla 3.

Tabla 3. Estructura del conjunto de datos de Medjool.

Nombre de las carpetas		**Imágenes**	**Datos**	**Total**
Pascal Voc	Maduro	501	501	1002
	Maduro_inmaduro	313	313	626
	Inmaduro	501	501	1002
yolo	Maduro	501	501	1002
	Maduro_inmaduro	159	159	318
	Inmaduro	501	501	1002
Prueba	Maduro_inmaduro	100	100	300
Total		**2576**	**2576**	**5152**

Se recolectaron un total de 2,576 imágenes las cuales fueron anotadas usando dos tipos formatos para el etiquetado, PascalVoc (XML) y YOLO (TXT). Se eligieron estos formatos debido a que son los más utilizados en las tareas de reconocimiento de objetos en imágenes, así como para evaluar y comparar el desempeño de diferentes algoritmos (Everingham *et al.,* 2010; Li *et al.,* 2020). El conjunto de datos se encuentra disponible en Mendeley Data, V1, http://dx.doi.org/10.17632/872xk9npmz.1.

En la Figura 15, se observan los diferentes niveles de madurez de los dátiles que aparecen en las imágenes.

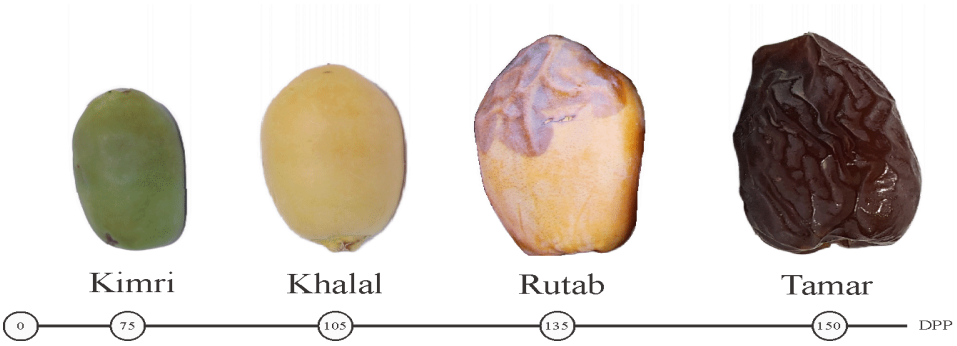

Figura 15. Niveles de madurez del dátil Medjool. DPP: Días después de la polinización.

3.1.2 Conjunto de datos de imágenes

El conjunto de datos de imágenes contenía 1002 imágenes en formato JPG, las cuales eran de diferentes tamaños (5184 × 3456, 4449 × 3071 y 4376 × 3375 píxeles). Las arquitecturas de red fueron entrenadas con imágenes JPG debido a que se alimentan de imágenes de baja calidad en escenarios reales. Se hace referencia a imágenes de baja calidad caracterizadas por desenfoque, ruido, contraste o compresión. Se considera que, si la arquitectura es entrenada con este tipo de imágenes, el sistema podrá clasificar dátiles Medjool en imágenes con estas características. Además, un estudio muestra que las redes neuronales convolucionales se ven mínimamente afectadas en su rendimiento al utilizar el formato JPG (Dodge *et al*, 2016).

El conjunto de datos de imágenes se distribuyó de la siguiente manera: 501 imágenes de dátiles maduros y 501 imágenes de dátiles inmaduros en bandejas (Figura 16). Los dátiles en bandejas fueron clasificados previamente como maduros o inmaduros por personas expertas.

Figura 16. Ejemplo de imágenes de conjuntos de datos. (A - C) Bandejas con dátiles maduros

3.1.3 Discusión

El conjunto de datos MedjoolDates se considera un recurso útil para los investigadores que trabajan en las áreas de visión artificial, procesamiento de imágenes, aprendizaje automático y aprendizaje profundo. Esto debido a que se puede utilizar para probar, comparar o crear algoritmos para localizar, reconocer, clasificar o contar frutos de dátiles Medjool, considerando sus características visuales como forma, color y textura, que son muy particulares para esta variedad de dátiles. Actualmente, existen conjuntos de datos con imágenes de dátiles o racimos de dátiles (Altaheri *et al.,* 2019). Sin embargo, estos dátiles no pertenecen a la variedad Medjool. Además, no se presentan en bandejas. El conjunto de datos puede ser de utilidad para crear aplicaciones de visión artificial para empresas empacadoras de este tipo de frutos.

Además, se considera que el conjunto de datos con imágenes etiquetadas es una base que se puede usar directamente para desarrollar nuevos algoritmos o mejorar los modelos desarrollados. Dado que el conjunto de datos se puede utilizar como parte del aprendizaje por transferencia, en el que, en lugar de comenzar el proceso de aprendizaje desde cero, se puede hacer uso de patrones o modelos pre-entrenados que se han aprendido al resolver un problema diferente (Siddiqi, 2019).

El conjunto de datos de MedjoolDates se puede utilizar para desarrollar un sistema de visión artificial que permita a los agricultores la selección, clasificación y conteo visual automáticos de los dátiles Medjool en bandejas o transportadores, teniendo en cuenta sus características visuales. Con ello asegurar la calidad del fruto considerando su madurez, tamaño y la cantidad a envasar.

3.2 Configuración de Software y Hardware

Para implementar y evaluar las arquitecturas CNN presentadas en la Sección 3.3, se utilizó el servicio en la nube Google Colab basado en Jupyter's Notebooks, el cual permite el uso gratuito de GPU o TPU de Google, con las bibliotecas Scikit-learn, PyTorch, TensorFlow, Keras y OpenCV. (Bisong, 2019). Las especificaciones de hardware utilizadas en este experimento fueron GPU: Nvidia-Tesla-T4; CPU: CPU Intel (R) Xeon ® con 2.20 GHz; RAM: ~ 12.78 GB disponibles; Disco duro: ~ 32.20 GB de espacio disponible. Las especificaciones de software fueron y Sistema Operativo: 18.04.5 LTS (Bionic Beaver) con las bibliotecas Keras 1.0.8 y Tensorflow 1.15 como back-end.

3.3 Configuraciones de las redes neuronales

3.3.1 Hiperparámetros

Un parámetro del modelo es una variable de configuración interna y cuyo valor puede ser estimado a partir de los datos. Por otro lado, los hiperparámetros, son variables de configuración externa al modelo en sí mismo, y cuyo valor no puede ser estimado a partir de los datos, y son especificados por el programador para ajustar los algoritmos de aprendizaje (Torres, 2018).

Los hiperparámetros son variables que determinan la estructura y topología de la red (Han *et al.,* 2020) (p. ej. número de capas ocultas, número de neuronas, sus funciones de

activación, entre otros) y las variables que determinan cómo esta red debe ser entrenada (p. ej. tasa de aprendizaje, momentum, épocas).

El optimizador específica la forma exacta cómo se utilizará la función de pérdida para actualizar los parámetros de las neuronas de una red neuronal y están basados en la idea del descenso del gradiente que consiste en disminuir iterativamente la función de pérdida siguiendo el gradiente (Kingma y Ba, 2017). Dos optimizadores comúnmente encontrados en la literatura son Adaptive Moment Estimation (Adam) y Stochastic Gradient Descent (SGD) (Li *et al*. 2021).

Adaptive Moment Estimation (Adam)

Su nombre se deriva de la estimación del momento adaptativo y es un método de optimización estocástica que solo requiere gradientes de primer orden calculando tasas de aprendizaje adaptativas individuales para diferentes parámetros, a partir de estimaciones del primer y segundo momento de los gradientes (Kingma y Ba, 2017).

Stochastic Gradient Descent (SGD)

En el método Gradient Descent (GD), hay tres tipos de métodos de descenso de gradiente utilizados para el entrenamiento de modelos, batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-BGD (MBGD) (Li *et al*., 2021). Nos centraremos en SGD, que es uno de los algoritmos básicos del método GD, SGD aprende rápidamente la dirección del descenso más pronunciado usando un solo ejemplo del conjunto de entrenamiento en cada paso de tiempo (Bisong, 2019). De acuerdo a la literatura, SGD es un método rápido (Li *et al*. 2021; Bisong, 2019), que mejora al reducir gradualmente la tasa de aprendizaje (Li *et al.* 2021), no obstante Zhang (2018), menciona que a menudo es difícil ajustar la tasa de aprendizaje de SGD, ya que las magnitudes de los parámetros varían ampliamente y se requiere un ajuste a lo largo del proceso de entrenamiento.

3.3.2 Implementación de las redes neuronales convolucionales.

Como se mencionó anteriormente, los hiperparámetros son variables que el programador ajusta en una red neuronal convolucional previo a su entrenamiento y que definen su arquitectura con el fin de mejorar su rendimiento.

Para esta tesis se implementaron redes neuronales convolucionales con y sin transferencia. En cada caso se realizó la variación de hiperparámetros de estructura y de entrenamiento. Para considerar los hiperparámetros de estructura, se implementaron ocho arquitecturas con diferente número de capas que van desde 16 hasta 152. En el caso de los hiperparámetros de entrenamiento, se evaluaron dos optimizadores diferentes: Adaptive Moment Estimation (Adam) y Stochastic Gradient Descent (SGD); así mismo, se utilizaron dos valores diferentes de tasas de aprendizaje, 0.001 y 0.01; además de dos tamaños de lote, 64 y 128 en dos números de épocas, 25 y 400. Las configuraciones de hiperparámetros se muestran en la Tabla 4. La implementación de arquitecturas de redes neuronales convolucionales en las que se consideró su variación de estructura fue: VGG-16, VGG-19, ResNet-50, ResNet-101, ResNet-152, AlexNet, Inception V3 y CNN con características desde cero. En total se implementaron 128 CNNs, las cuales se muestran en la Tabla 5, correspondiendo a 16 combinaciones de hiperparámetros.

Tabla 4. Variación de hiperparámetros.

Optimizador	Adaptive Moment Estimation (Adam)
	Stochastic Gradient Descent (SGD)
Tasa de aprendizaje	0.0001
	0.01
Tamaño del lote	64
	128
Épocas	25
	400
Número de capas	16 - 152

En la Tabla 5, se puede observar que se utilizaron dos optimizadores, Adaptive Moment Estimation (Adam) y Stochastic Gradient Descent (SGD) los cuales son los más reportados en la literatura por su buen rendimiento. En cuanto a la tasa de aprendizaje, el tamaño de lote y el número de épocas, se utilizaron dos valores extremos en cada caso, esto de acuerdo a que autores como Kandel *et al.,* (2020), Radiuk (2017), quienes afirman que, en el caso del tamaño de lote, los valores más altos son los que obtienen mejores rendimientos. El número de capas lo podemos observar en las ocho arquitecturas que cuentan con diferentes profundidades que van desde 16 hasta 152 capas.

Tabla 5. Redes Neuronales Convolucionales implementadas con sus respectivas configuraciones.

CNN from Scratch				VGG-16			
Optimizador	**Tasa de aprendizaje**	**Batch**	**Épocas**	**Optimizador**	**Tasa de aprendizaje**	**Batch**	**Épocas**
Adam	0.01	64	25	Adam	0.01	64	25
Adam	0.01	64	400	Adam	0.01	64	400
Adam	0.01	128	25	Adam	0.01	128	25
Adam	0.01	128	400	Adam	0.01	128	400
Adam	0.001	64	25	Adam	0.001	64	25
Adam	0.001	64	400	Adam	0.001	64	400
Adam	0.001	128	25	Adam	0.001	128	25
Adam	0.001	128	400	Adam	0.001	128	400
SGD	0.01	64	25	SGD	0.01	64	25
SGD	0.01	64	400	SGD	0.01	64	400
SGD	0.01	128	25	SGD	0.01	128	25
SGD	0.01	128	400	SGD	0.01	128	400
SGD	0.001	64	25	SGD	0.001	64	25
SGD	0.001	64	400	SGD	0.001	64	400
SGD	0.001	128	25	SGD	0.001	128	25
SGD	0.001	128	400	SGD	0.001	128	400
VGG-19				**ResNet-50**			
Optimizador	**Tasa de aprendizaje**	**Batch**	**Épocas**	**Optimizador**	**Tasa de aprendizaje**	**Batch**	**Épocas**
Adam	0.01	64	25	Adam	0.01	64	25
Adam	0.01	64	400	Adam	0.01	64	400
Adam	0.01	128	25	Adam	0.01	128	25
Adam	0.01	128	400	Adam	0.01	128	400
Adam	0.001	64	25	Adam	0.001	64	25
Adam	0.001	64	400	Adam	0.001	64	400
Adam	0.001	128	25	Adam	0.001	128	25
Adam	0.001	128	400	Adam	0.001	128	400
SGD	0.01	64	25	SGD	0.01	64	25
SGD	0.01	64	400	SGD	0.01	64	400
SGD	0.01	128	25	SGD	0.01	128	25
SGD	0.01	128	400	SGD	0.01	128	400
SGD	0.001	64	25	SGD	0.001	64	25
SGD	0.001	64	400	SGD	0.001	64	400
SGD	0.001	128	25	SGD	0.001	128	25
SGD	0.001	128	400	SGD	0.001	128	400
ResNet-101				**ResNet-152**			
Optimizador	**Tasa de aprendizaje**	**Batch**	**Épocas**	**Optimizador**	**Tasa de aprendizaje**	**Batch**	**Épocas**
Adam	0.01	64	25	Adam	0.01	64	25
Adam	0.01	64	400	Adam	0.01	64	400
Adam	0.01	128	25	Adam	0.01	128	25
Adam	0.01	128	400	Adam	0.01	128	400
Adam	0.001	64	25	Adam	0.001	64	25

Adam	0.001	64	400	Adam	0.001	64	400
Adam	0.001	128	25	Adam	0.001	128	25
Adam	0.001	128	400	Adam	0.001	128	400
SGD	0.01	64	25	SGD	0.01	64	25
SGD	0.01	64	400	SGD	0.01	64	400
SGD	0.01	128	25	SGD	0.01	128	25
SGD	0.01	128	400	SGD	0.01	128	400
SGD	0.001	64	25	SGD	0.001	64	25
SGD	0.001	64	400	SGD	0.001	64	400
SGD	0.001	128	25	SGD	0.001	128	25
SGD	0.001	128	400	SGD	0.001	128	400

AlexNet				**Inception V3**			
Optimizador	**Tasa de aprendizaje**	**Batch**	**Épocas**	**Optimizador**	**Tasa de aprendizaje**	**Batch**	**Épocas**
Adam	0.01	64	25	Adam	0.01	64	25
Adam	0.01	64	400	Adam	0.01	64	400
Adam	0.01	128	25	Adam	0.01	128	25
Adam	0.01	128	400	Adam	0.01	128	400
Adam	0.001	64	25	Adam	0.001	64	25
Adam	0.001	64	400	Adam	0.001	64	400
Adam	0.001	128	25	Adam	0.001	128	25
Adam	0.001	128	400	Adam	0.001	128	400
SGD	0.01	64	25	SGD	0.01	64	25
SGD	0.01	64	400	SGD	0.01	64	400
SGD	0.01	128	25	SGD	0.01	128	25
SGD	0.01	128	400	SGD	0.01	128	400
SGD	0.001	64	25	SGD	0.001	64	25
SGD	0.001	64	400	SGD	0.001	64	400
SGD	0.001	128	25	SGD	0.001	128	25
SGD	0.001	128	400	SGD	0.001	128	400

3.3.3 Transferencia de Aprendizaje

En este estudio se utilizó la transferencia de aprendizaje para detectar y clasificar la madurez de frutos de dátil utilizando imágenes de dátiles maduros y no maduros. En esta tesis la aplicación del aprendizaje por transferencia fue el modelo previo a la formación. Se utilizaron redes neuronales previamente entrenadas con ImageNet. En cuanto a su estructura, fue eliminada la capa de clasificación final, es decir, la capa de neurona softmax al final, que corresponde a ImageNet, quedando en su lugar la nueva capa de softmax para nuestro conjunto de datos de imágenes.

3.4 Métricas utilizadas para la detección

Para medir el rendimiento de un algoritmo en tareas como clasificación, es importante contar con métricas que permitan comparar el rendimiento de cada algoritmo.

En el caso de la tarea de clasificación, se emplean generalmente las métricas de: mAP (mean Average Precision), Precision (Precisión), Recall (Exhaustividad o Recuerdo), F1-score (Valor-F), Accuracy (Exactitud), Confusion Matrix (Matriz de Confusión). A continuación, se explicará cada una de ellas.

3.4.1 Precisión (Precision)

Precisión: Es la proporción de elementos clasificados como positivos que son realmente verdaderos positivos (Padilla *et al.,* 2020).

$$Precisión = \frac{vp}{vp + fp} \quad (2)$$

Donde vp son los verdaderos positivos y fp los falsos positivos (Padilla *et al.,* 2020).

3.4.2 Recuerdo (*Recall*)

Recuerdo (Recall): Es la proporción de elementos verdaderos positivos que son correctamente clasificados como positivos (Padilla *et al.,* 2020).

$$Recuerdo = \frac{vp}{vp + fn} \quad (3)$$

Donde *fn* son los falsos negativos.

3.4.3 Exactitud (*Accuracy*)

Exactitud (Accuracy): Es una medida de eficiencia generalizada para evaluar el desempeño de un clasificador, y se refiere a las instancias correctamente clasificadas (Padilla *et al.,* 2020).

$$Exactitud = \frac{vp + fn}{vp + vn + fp + fn} \quad (4)$$

Donde *vn* corresponde a los verdaderos negativos.

3.4.4 Matriz de Confusión (*Confusion Matrix*)

La matriz de confusión es una herramienta que permite la evaluación del desempeño de un modelo de clasificación. En esta métrica se muestra el número de predicciones correctas e incorrectas realizadas por el modelo en cada una de las clases en las que está clasificado. Como se puede observar en la Tabla 6. La matriz de confusión presenta filas de clases verdaderas y en las columnas se presentan las clases predichas por el modelo, por lo tanto, la diagonal principal muestra las predicciones correctas.

Tabla 6. Matriz de Confusión.

		Predicción	
		Positivos	**Negativos**
Realidad	**Positivos**	Verdaderos positivos (True Positive)	Falsos Positivos (False Positives)
	Negativos	Falsos negativos (False Negatives)	Verdaderos negativos (True Negative)

Positivo (Positive) o Negativo (Negative): se refiere a la predicción. Si el modelo predice 1 entonces será positivo, y si se predice y se predice 0 será negativo.

Verdadero (True) o Falso (False): se refiere si la predicción es correcta o no.

Capítulo 4. Resultados

En este capítulo, exploramos la variación de hiperparámetros en arquitecturas convolucionales específicamente diseñadas para determinar los parámetros óptimos que permiten identificar con precisión el nivel de madurez en los dátiles de la variedad Medjool. A lo largo de este análisis, examinaremos cómo la modificación de ciertos hiperparámetros afecta el rendimiento del modelo, centrándonos en la capacidad de la red convolucional para discernir niveles de madurez en estos frutos tan particulares.

4.1 Variando el Optimizador

Utilizando el optimizador Adam, observamos en la Tabla 7 que, durante la evaluación con 25 épocas, VGG-16 alcanzó el mejor rendimiento con un 96.63% y 95.27% de precisión para tasas de aprendizaje de 0.001 y 0.01, respectivamente. Por otro lado, AlexNet y ResNet-152 mostraron el rendimiento más bajo con un 64.19% para una tasa de aprendizaje de 0.001, y CNN desde cero tuvo el rendimiento más bajo con un 46.62% y 53.38% para tasas de aprendizaje de 0.01.

Tabla 7. Evaluación de la precisión de ocho arquitecturas CNN, cambiando los valores de los hiperparámetros de un lote, tasa de aprendizaje y épocas, utilizando el optimizador parámetro Adaptive Moment Estimation (Adam) como optimizador.

Parámetros	**CNN desde cero**	**VGG-16**	**VGG-19**	**ResNet-50**	**ResNet-101**	**ResNet-152**	**AlexNet**	**Inception V3**
Épocas	25 400	25 400	25 400	25 400	25 400	25 400	25 400	25 400
Lote = 64, Optimizador = Adam, Tasa de aprendizaje = 0.001								
Precisión (%)	93.24 94.59	96.63 96.62	90.54 95.95	68.92 81.08	71.62 80.41	74.32 80.41	64.19 88.51	96.62 98.65
Tiempo (min)	9 16	24 40	14 43	11 33	14 41	25 61	11 19	12 131
Lote= 128, Optimizador = Adam, Tasa de aprendizaje= 0.001								
Precisión (%)	85.13 93.92	95.27 97.29	87.84 98.65	70.95 83.11	70.27 75.67	64.17 81.08	75.00 85.81	93.24 95.27
Tiempo (min)	11 12	12 34	5 46	13 45	7 16	9 54	11 19	13 48
Lote = 64, Optimizador = Adam, Tasa de aprendizaje = 0.01								

Parámetros	CNN desde cero	VGG-16	VGG-19	ResNet-50	ResNet-101	ResNet-152	AlexNet	Inception V3
Precisión (%)	46.62 95.27	85.81 95.95	93.92 96.62	83.78 86.49	65.54 79.05	75.68 84.46	85.81 67.57	95.95 98.65
Tiempo (min)	10 14	12 43	12 45	10 34	11 45	16 65	16 18	11 47
Lote= 128, Optimizador = Adam, Tasa de aprendizaje = 0.01								
Precisión (%)	53.38 43.24	97.29 96.62	97.30 99.32	84.46 84.46	76.35 79.73	66.89 81.76	89.19 87.16	93.92 95.95
Tiempo (min)	11 16	12 38	13 43	12 33	11 46	15 60	11 18	13 44

Para 400 épocas, Inception V3 y VGG-19 lideraron con un rendimiento del 98.65% y 98.75% para una tasa de aprendizaje de 0.001, y VGG-19 y Inception V3 con 99.32% y 98.65%, respectivamente, para una tasa de aprendizaje de 0.01. Las peores actuaciones fueron de ResNet-101 y ResNet-152, ambas con un 80.41% para una tasa de aprendizaje de 0.001, y AlexNet y CNN desde cero con 67.57% y 43.24%, respectivamente, para una tasa de aprendizaje de 0.01.

En términos de tiempo de procesamiento, CNN desde cero mostró los valores más bajos, aunque algunos fueron superiores a ResNet-50, ResNet-101, ResNet-152 y AlexNet. En 25 épocas, los tiempos más altos fueron para ResNet-152 (25 min) e Inception V3 (13 min) con una tasa de aprendizaje de 0.001, y para ResNet-152 y AlexNet (16 min) con ResNet-152 (15 min) para una tasa de aprendizaje de 0.01. Para 400 épocas, los tiempos más altos fueron para Inception V3 (131 min) y ResNet-152 (54 min) con una tasa de aprendizaje de 0.001, y ResNet-152 (65 y 60 min) con una tasa de aprendizaje de 0.01. ResNet-152 fue la arquitectura que requirió más tiempo de procesamiento en la mayoría de los casos, independientemente de la precisión obtenida.

En la Tabla 8, al utilizar el Descenso de Gradiente Estocástico (SGD) como optimizador, notamos que para una evaluación con 25 épocas, VGG-19 y VGG-16 lideraron con un rendimiento del 87.16% para una tasa de aprendizaje de 0.001. Además, Inception V3 demostró un rendimiento sólido con 92.56% y 91.89% para tasas de aprendizaje de 0.01. En contraste, AlexNet y CNN desde cero mostraron el rendimiento más bajo con un 52.70% y 51.35%, respectivamente, utilizando una tasa de aprendizaje de 0.001, mientras que ResNet-50 y ResNet-152 obtuvieron un 45.94% con una tasa de aprendizaje de 0.01.

Tabla 8. Evaluación de la precisión de las arquitecturas de ocho CNN, cambiando los valores de los hiperparámetros del lote, la tasa de aprendizaje y las épocas, utilizando el optimizador parámetro de descenso de gradiente estocástico (SGD) como optimizador.

Parámetros	CNN desde cero	VGG-16	VGG-19	ResNet-50	ResNet-101	ResNet-152	AlexNet	Inception V3
Épocas	25 400	25 400	25 400	25 400	25 400	25 400	25 400	25 400
Lote= 64, Optimizador = SGD, Tasa de aprendizaje = 0.001								
Precisión (%)	54.05 93.24	86.49 93.24	87.16 91.21	66.89 75.68	53.38 75.00	54.05 64.19	52.70 56.08	86.50 95.94
Tiempo (min)	12 16	14 41	13 46	11 36	12 47	13 58	11 19	13 42
Lote= 128, Optimizador = SGD, Tasa de aprendizaje = 0.001								
Precisión (%)	51.35 94.59	87.16 90.54	85.81 92.57	68.92 71.62	53.37 74.32	53.37 64.87	53.38 60.81	79.05 93.24
Tiempo (min)	10 28	23 50	13 48	12 32	12 44	13 115	11 18	14 43
Lote= 64, Optimizador= SGD, Tasa de aprendizaje = 0.01								
Precisión (%)	83.11 89.86	88.51 92.57	77.10 94.59	45.94 50.00	46.62 66.89	45.94 65.54	53.38 83.78	92.56 93.92
Tiempo (min)	10 15	12 45	12 45	11 32	8 44	14 58	11 31	12 44
Lote= 128, Optimizador= SGD, Tasa de aprendizaje = 0.01								
Precisión (%)	79.72 91.26	78.37 90.54	79.73 88.51	45.94 52.03	46.62 56.08	53.37 64.19	54.05 80.41	91.89 95.27
Tiempo (min)	11 16	11 41	12 44	11 53	69 44	13 54	34 21	12 42

Para 400 épocas, Inception V3 y CNN desde cero destacaron con un rendimiento máximo del 95.94% y 94.59%, respectivamente, utilizando una tasa de aprendizaje de 0.001. VGG-19 e Inception V3 también demostraron un rendimiento sólido con 94.59% y 95.27%, respectivamente, con una tasa de aprendizaje de 0.01. En contraste, AlexNet mostró el rendimiento más bajo con un 56.08% y 60.81% para una tasa de aprendizaje de 0.001, y ResNet-50 obtuvo un 50% y 52.03% con una tasa de aprendizaje de 0.01.

Además, destacamos que las dos arquitecturas de CNN más efectivas fueron CNN desde cero (94.59%) e Inception V3 (95.27%) para un lote de 128, seguidas de Inception V3 (95.94%) y VGG-19 (94.59%) para un lote de 64.

La Tabla 8 nos proporciona información sobre el tiempo de procesamiento y la precisión de diferentes arquitecturas. No se identificó un patrón claro para determinar qué arquitectura tenía consistentemente el menor tiempo de procesamiento en todos los

hiperparámetros. Los valores más bajos generalmente se observaron en CNN desde cero, pero la cifra más destacada fue para ResNet-101, que registró solo 8 minutos en 25 épocas, con un lote de 64 y una tasa de aprendizaje de 0.01.

Además, la precisión de CNN desde cero superó a las arquitecturas ResNet-50, ResNet-101, ResNet-152 y AlexNet. En cuanto a los tiempos más altos, en 25 épocas, VGG-16 tuvo valores de 14 y 23 minutos con una tasa de aprendizaje de 0.001, y ResNet-152 registró 14 minutos, mientras que ResNet-101 alcanzó 69 minutos con una tasa de aprendizaje de 0.01. Para 400 épocas, ResNet-152 lideró con 58 y 115 minutos a una tasa de aprendizaje de 0.001, y 58 y 54 minutos con una tasa de aprendizaje de 0.01.

En resumen, ResNet-101 tuvo el menor tiempo en ciertos escenarios, mientras que ResNet-152 requirió el mayor tiempo en la mayoría de los casos. Es interesante notar que los tiempos más altos no necesariamente se correlacionaron con una precisión más alta o más baja. Además, se destaca la influencia de la complejidad de la base de datos en el tiempo de procesamiento y la precisión de la red, como señaló Radiuk en 2017.

Capítulo 5. Discusión

Las redes neuronales convolucionales (CNN) se han utilizado para el desarrollo de aplicaciones en el área agrícola, las cuales tienen por objetivo la detección de enfermedades en hojas y plantas, la clasificación de la cobertura del suelo, la clasificación del tipo de cultivo, el reconocimiento de plantas, la segmentación de la raíz y el suelo, la estimación del rendimiento de los cultivos, el recuento de frutas, la detección de obstáculos en los cultivos en hileras y corte de césped e identificación de malezas, por mencionar algunas (Kamilaris y Prenafeta-Boldú, 2018a; Kamilaris y Prenafeta-Boldú, 2018b). Por ejemplo, en Mohanty *et al.* (2016), presentaron el entrenamiento de las arquitecturas convolucionales AlexNet y Google Net con un conjunto de datos de imágenes de PlanVillage para detectar 26 tipos de enfermedades en 14 tipos de cultivos. Sus resultados mostraron una precisión del 99,35% para identificar plantas sanas y enfermas. Mientras tanto, Rahnemonfar y Sheppard (2017), propusieron utilizar las arquitecturas CNN y redes residuales (ResNet) para estimar el rendimiento de una planta de tomate utilizando imágenes sintéticas. Sus resultados indicaron que, con un 91% de precisión, pueden evaluar el rendimiento.

Otro ejemplo se presentó en Ashraf *et al.* (2019), donde los autores propusieron entrenar varias redes convolucionales para identificar cuatro frutas (mango, naranja, manzana y plátano). Se clasificaron en dos categorías: frescas y podridas. Los modelos con mejor rendimiento fueron Inception versión 3 y el Grupo Geométrico Visual de arquitecturas de 16 capas (VGG-16), que recibieron la transferencia de aprendizaje. Sus resultados mostraron porcentajes de identificación y clasificación de 90% de precisión. En Zeng (2017), se presentó un estudio similar, donde se propuso el uso de una red VGG-16 para clasificar verduras y frutas. Se clasificaron un total de 26 categorías: calabaza, apio, coliflor, piña, granada, pomelo, plátano, pepino, brócoli, cebolla, zanahoria, etc. Los autores afirmaron tener un 95,6% de precisión en la clasificación de estas frutas y verduras. En cuanto a dátiles, identificamos trabajos de investigación que propusieron utilizar CNN para clasificar entre dátiles o detectar entre sus diferentes niveles de madurez (Altaheri *et al.,* 2019b; Nasiri *et al.,* 2019; Haidar *et al.,* 2012).

Actualmente, para determinar la etapa de madurez en el dátil Medjool, se han utilizado métodos tradicionales para el procesamiento de imágenes y aprendizaje automático para su clasificación. Sin embargo, es una tarea compleja. Esto se debe a que estos métodos están entrenados para extraer características en varios cultivares, como su apariencia, color (asociado con las etapas de madurez), forma y textura (Altaheri *et al.,* 2019; Haidar *et al.,* 2012). Sin embargo, no hay estudios en los que tengamos conocimiento de una extracción de características o un modelo predictivo para clasificar dátiles Medjool. Además, los modelos recientes no pueden determinar la clasificación del cultivar Medjool porque esta variedad se cosecha, clasifica, empaqueta y consume en su etapa Tamar.

Para contribuir con un modelo que puede ser útil para clasificar dátiles Medjool a través de imágenes, comparamos el rendimiento de ocho arquitecturas de CNN en este estudio. Adicionalmente, se modificaron los valores de algunos hiperparámetros y se utilizó el aprendizaje por transferencia para identificar y proponer el uso de CNN con la mayor precisión. Los efectos de cada hiperparámetro fueron analizados en términos de la exactitud de clasificación. En esta tesis se busca investigar el efecto de los hiperparámetros en los resultados de entrenamiento.

Como se muestra en la Tabla 7, nuestros hallazgos indicaron que cuando usamos un optimizador Adam, las arquitecturas VGG muestran la mejor precisión, con el modelo VGG-19 que alcanzó el mayor porcentaje de precisión con 99.32%. Asimismo, las arquitecturas ResNet y CNN desde cero mostraron los porcentajes de desempeño más bajos; el modelo de CNN desde cero logró la precisión más insuficiente, con un 43,24%. El porcentaje promedio más alto generado entre las ocho arquitecturas fue 89.53%, usando la combinación de lote (64), tasa de aprendizaje (0.01) y épocas (400), con un tiempo promedio de 48.71 min, mientras que el más bajo fue 75%, combinando un lote (64), tasa de aprendizaje (0,001) y épocas (25), con un tiempo medio de 12,25 min.

Asimismo, la Tabla 8, indica que ninguna arquitectura mostró la mejor precisión cuando usamos un hiperparámetro SGD como optimizador. Sin embargo, la arquitectura ResNet-50 mostró los porcentajes de rendimiento más bajos, con lotes (64 y 128) y tasa de aprendizaje (0,001). El porcentaje más alto generado entre las ocho arquitecturas fue 80.57%, utilizando la combinación de lote (64), tasa de aprendizaje (0.001) y épocas (400), con un tiempo promedio de 38.13 min, mientras que el más bajo fue 66.21%, combinando un lote (128), tasa de aprendizaje (0,01) y épocas (25), con un tiempo medio de 21,63 min.

Se notó que, si se aumentaba el número de épocas para todos los modelos, también aumentaba el porcentaje de precisión y el tiempo de procesamiento requerido. Asimismo, observamos que los tiempos de procesamiento más altos correspondieron a la arquitectura ResNet-152, lo que podría estar asociado con el hecho de que esta arquitectura tuviera el mayor número de capas. Sin embargo, ninguna de sus precisiones superó el rendimiento del 85%.

El optimizador puede ayudarnos a minimizar la función de error que nos permite ajustarnos a los ejemplos del conjunto de entrenamiento. En este estudio, la precisión fue mayor para Adam que para SGD.

Varios estudios se han centrado en identificar la CNN que ofrece la mayor precisión para seleccionar dátiles de cultivares en sus distintas etapas de madurez (Altaheri *et al.,* 2019b; Nasiri *et al.,* 2019; Faisal *et al.,* 2020). Sin embargo, actualmente no hay estudios reportados que usen CNN para clasificar el cultivar de dátiles Medjool.

La Tabla 9, compara estudios similares al nuestro, donde se han reportado las arquitecturas de CNN con el mejor desempeño. Nasiri *et al*., (2019), solo trabajaron en VGG-16 con dos hiperparámetros, obteniendo la máxima precisión del 96,98%. Asimismo, Altaheri *et al.*, (2019b), trabajaron en dos CNN con aprendizaje por transferencia y ajuste fino, modificando tres hiperparámetros dos veces, obteniendo el mayor porcentaje para VGG-16 con una precisión del 97,25%. Faisal *et al.,* 2020, comparó el desempeño de cuatro CNN, evaluando cuatro hiperparámetros, lo que resultó en ResNet como el mejor modelo, con una precisión del 99.01%. Finalmente, nuestro estudio evaluó el desempeño de ocho CNN, utilizando el aprendizaje por transferencia y modificando cuatro hiperparámetros dos veces, lo que resultó en el modelo VGG-19 con el rendimiento más alto, con una precisión del 99,32%.

Tabla 9. Comparación de estudios que reportan arquitecturas de CNN en la detección de varios estados de madurez en el fruto de la palmera datilera.

Referencia	Cultivar de palma datilera	Estado de maduración	Número de imágenes (Set de datos)	Arquitectura CNN	Hiperparámetros	Mejor precisión
Nasiri *et al*., 2019.	Shahani	Khalal, Rutab, Tamar, y dátiles defectuosos	+1300 imágenes	VGG-16	Épocas= 15 Lote= 32	VGG-16 96.98%

Altaheri *et al*., 2019a.	Barhi, Khalas, Meneifi, Naboot Saif y Sullaj	Immature-1, Immature-2, pre-Khalal, Khalal, Khalal-with-Rutab, pre-Tamar, and Tamar.	8072 imágenes	AlexNet, VGG-16, Transfer learning and Fine-Tuning	Épocas = 50 and 200 Lote = 32 and 128 Tasa de aprendizaje = 0.0001, 0.0002	VGG-16 97.25%
Faisal *et al*., 2020.	Barhi, Khalas, Meneifi, Naboot Saif, y Sullaj	Immature, Khalal, Khalal with Rutab, Pre-Tamar, and Tamar	8079 imágenes	ResNet, VGG-19, Inception V3, NASNet and support vector machine (SVM) (regression and linear)	Épocas = 50 Lote = 16 Optimizador = Adam Tasa de aprendizaje = 0.0001	ResNet 99.01%
Este trabajo	Medjool	Maduro y no maduro	1002 imágenes	VGG-16, VGG-19, Inception V3, ResNet-50, ResNet-101, ResNet-152, AlexNet, CNN desde cero	Épocas = 25 and 400 Lote = 64 and 128 Optimizador = Adam, Stochastic Gradient Descent Tasa de aprendizaje = 0.001, 0.01	VGG-19 99.32%

Un aspecto a considerar en esta comparación es que el dátil Medjool solo se consume en su etapa Tamar. Por lo tanto, este estudio solo utilizó dos etapas para su clasificación. El número de imágenes fue menor en comparación con el resto de estudios. Sin embargo, en nuestro trabajo, el porcentaje de precisión fue mayor debido a la aplicación del aprendizaje por transferencia y la modificación en varios hiperparámetros, que influyen en el rendimiento de las arquitecturas (Kandel y Castelli, 2020; Motta *et al.,* 2020).

En nuestro estudio, como resultado de elegir los hiperparámetros épocas (400), el tamaño de lote (128), el optimizador (Adam) y una tasa de aprendizaje (0.01), identificamos que la arquitectura VGG-19 tenía el mejor rendimiento. Asimismo, esta arquitectura podría incluirse como parte del software que controle un mecanismo robótico para apoyar a productores de dátiles en un sistema automatizado de clasificación de frutos maduros.

En un estudio realizado por Nguyen y Lee (2018) para evaluar los efectos de los hiperparámetros y de diferentes dataset en el entrenamiento de dos CNN, VGG-16 y en una CNN propuesta por ellos. Los hiperparámetros evaluados fueron la tasa de aprendizaje (0.001 a 0.1), el tamaño de batch (32 y 64) y el optimizador (SGD y Momentum). Estas dos redes se alimentaron con tres bases de datos diferentes Cifar10, Cifar100 y DSDL-DB, con las cuales

contenía diferentes números de capas. De lo anterior, observan que la exactitud aumenta cuando la tasa de aprendizaje decrementa, por ejemplo, en la base de datos Cifar10 se observa que para una tasa de aprendizaje de 0.1 la exactitud fue de 54.6, cuando la tasa de aprendizaje fue de 0.001 la exactitud fue de 85.3 mayor que el primer caso. Otro hallazgo que mencionan los autores es que la normalización mejora la precisión entre 13.9 y un 31.4% en comparación con el entrenamiento con datos sin procesar, además que el efecto de la transformación de los datos en la exactitud es mayor que el de la normalización (p.ej. de 78.2 a 79.7, respectivamente para el caso de la base de datos CIFAR10 o 34.3 a 38.1 en CIFAR100). Sin embargo, la combinación de la normalización y la transformación de los datos son mejores que el entrenamiento con un solo procesamiento para todos sus conjuntos de datos. Finalmente, en cuanto al tiempo de entrenamiento encuentran que cuando el tamaño de lote es menor, el tiempo de entrenamiento es mayor.

En otro estudio por Nazir *et al.* (2020), mencionan que cada selección de los hiperpármetros para el ajuste también afecta los resultados, ya que cada hiperparámetro no tiene la misma importancia para el entrenamiento o la precisión del modelo. Ellos mencionan tres tipos de ajuste de hiperparámetros: ajuste manual, ajuste automático y ajuste aleatorio. Mencionan que el ajuste manual puede tener resultados prometedores en términos de tiempo e hiperparámetros seleccionados porque a diferencia del ajuste automático, un ser humano puede descartar hiperparámetros subóptimos fácilmente. De la misma manera, comentan que la persona tiene conocimiento y comprensión de la importancia relativa de los hiperparámetros para el modelo dado, así como, el experto podría detectar fallas y terminar la capacitación en una etapa temprana (Ozaki *et al*., 2017) y finalmente, que la optimización manual no se ve obstaculizada por ninguna sobrecarga técnica (Bergstra y Bengio, 2012).

Koutsoukas *et al.* (2017), exploró diferentes configuraciones de hiperparámetros en redes neuronales profundas, así como en métodos comunes como Naive Bayes, vecinos cercanos, bosques aleatorios y máquinas de soporte vectorial, para posteriormente comparar sus rendimientos. Mencionan que los hiperparámetros que juegan un papel importante son la función de activación, la regularización del droput, el número de capas profundas y el número de neuronas.

Capítulo 6. Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones de este trabajo de tesis con respecto a la evaluación de los hiperparametros en arquitecturas de redes convolucionales con transferencia de aprendizaje. Además, se listan las principales contribuciones y se presenta el posible trabajo futuro.

6.1 Conclusiones

La clasificación automatizada de dátiles Medjool ha sido objeto de un estudio exhaustivo que evaluó el rendimiento de ocho arquitecturas de redes neuronales convolucionales (CNN). Entre estas arquitecturas, se destacó la eficacia de VGG-19, que, mediante el uso de transferencia de aprendizaje y ajuste de hiperparámetros, logró una precisión excepcional del 99.32% en la identificación de la variedad de dátiles en su etapa Tamar. La transferencia de aprendizaje, que implica aprovechar conocimientos previos de modelos preentrenados con la base de datos ImageNet, se reveló como una estrategia crucial para adaptar las características aprendidas a la tarea específica de clasificación de dátiles Medjool, incluso con un conjunto de datos relativamente pequeño. Lo anterior aborda la hipótesis dos, donde se postula que precisión de una red neuronal convolucional con transferencia de aprendizaje tiene un impacto significativamente mayor al variar sus hiperparámetros de entrenamiento en comparación con la variación de los hiperparámetros de estructura.

Los hiperparámetros, como el número de épocas, el tamaño del lote, el optimizador y la tasa de aprendizaje, se exploraron sistemáticamente, evidenciando su impacto significativo en el rendimiento de las CNN. Se observó que un mayor número de épocas mejoraba la precisión, pero al mismo tiempo, incrementaba el tiempo de procesamiento. Este hallazgo destaca la importancia de encontrar un equilibrio entre la precisión y la eficiencia computacional al configurar estos hiperparámetros. Esto abordaría la hipótesis uno en la que se plantea que el ajuste de los hiperparámetros de entrenamiento de una red neuronal convolucional tiene un impacto significativamente mayor en su precisión para la clasificación de objetos y transferencia de aprendizaje en imágenes digitales.

En esta tesis se identificaron varios retos asociados con la clasificación automatizada de dátiles Medjool, especialmente debido a la necesidad de identificar con precisión la etapa Tamar, en la cual estos dátiles son consumidos. Se plantea que la implementación de modelos CNN en la industria agrícola podría ofrecer soluciones prácticas para optimizar los procesos de cosecha y clasificación, reduciendo la dependencia de la clasificación manual y mejorando la precisión y productividad en la cadena de producción de dátiles.

6.2. Contribuciones

Las contribuciones obtenidas como resultado de esta investigación, se presentan en las siguientes subsecciones:

6.2.1 Publicaciones de la Tesis

Con los resultados de la tesis se obtuvieron los siguientes artículos:

Pérez-Pérez, D. B., Salomón-Torres, R., & García-Vázquez, J. P. (2021). **Dataset for localization and classification of Medjool dates in digital images**. Data in Brief, 36, 107116. **(Q3, Factor de Impacto de 0.28).**

Se ha desarrollado el conjunto de imágenes "MedjoolDates", disponible en el enlace [MedjoolDates Dataset](https://data.mendeley.com/datasets/872xk9npmz/1). Este conjunto, recopilado durante la primera cosecha de dátiles Medjool en el Rancho Palmeras RQ en Mexicali, México, abarca 5,052 imágenes con diferentes tamaños y niveles de madurez de los frutos. Permitiendo el desarrollo de modelos para reconocimiento, clasificación, localización y recuento visual de dátiles en bandejas, destaca por sus anotaciones en formatos YOLO y PascalVOC. Incluye imágenes de dátiles maduros e inmaduros, con anotaciones que consideran características visuales como forma, color, tamaño y textura. Este recurso resulta fundamental para la investigación en algoritmos de análisis automatizado de dátiles Medjool, ofreciendo diversidad y detalle visual para robustos desarrollos en el campo.

Pérez-Pérez, B. D., Garcia Vazquez, J. P., & Salomón-Torres, R. (2021). **Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting** of ripe medjool dates. Agriculture, 11(2), 115. **(Q1, Factor de Impacto 3.4)**

Se realizó una evaluación del rendimiento de ocho arquitecturas de redes CNNs, considerando aprendizaje por transferencia para su entrenamiento, así como cinco hiperparámetros. Esta evaluación se describe en detalle en (Kamilaris *et al.,* 2028a). Las arquitecturas CNN evaluadas fueron VGG-16, VGG-19, ResNet-50, ResNet-101, ResNet-152, AlexNet, Inception V3 y CNN from cero. Asimismo, los hiperparámetros analizados fueron el número de capas, el número de épocas, el tamaño del lote, el optimizador y la tasa de aprendizaje. La precisión y el tiempo de procesamiento se consideraron para determinar el rendimiento de las arquitecturas CNN, en la clasificación del cultivar de dátiles maduros Medjool. El modelo obtenido de la arquitectura VGG-19 con un lote de 128 y el optimizador Adam con una tasa de aprendizaje de 0,01 presentó el mejor rendimiento con una precisión del 99,32%. Concluimos que el modelo VGG-19 puede utilizarse para construir sistemas de visión por ordenador que ayuden a los productores a mejorar su proceso de clasificación para detectar la fase Tamar de un dátil Medjool.

Vazquez, J. P. G., Torres, R. S., Perez, D. B. P., Demarigny, Y., Soldat, V., Gemelas, L., ... & Bersimis, F. G. (2021). **Scientometric analysis of the application of artificial intelligence in agriculture.** Journal of Scientometric Research, 10(1), 55-62.

Se presenta un análisis cienciométrico sobre la aplicación de la inteligencia artificial en agricultura, se revisaron 5,048 artículos desde 1939 hasta 2020 en Scopus. Se empleó la metodología Michán y Muñoz-Velasco (2013), abordando etapas de recuperación, migración, análisis, visualización e interpretación. Destacan el crecimiento de publicaciones desde 2002, con el 63.43% en los últimos cinco años. China lidera en publicaciones (17.71%), seguida por EE.UU. (15.16%) e India (14.74%). Temas clave incluyen agricultura de precisión, IoT, sistemas de apoyo a decisiones, aprendizaje profundo y procesamiento de imágenes. La agricultura sostenible inteligente se vislumbra como un concepto unificador. El análisis revela la importancia de IA en áreas emergentes y transversales, enfocándose en la mejora de la producción alimentaria y la necesidad de un tratamiento cuidadoso de datos en estudios cienciométricos.

Vazquez, J. P. G., Torres, R. S., Perez, D. B. P., Zatarain, A. H., Soto, V. J. E., García, C. E. R., (2023). **Inteligencia Artificial al Servicio de los Pequeños Productores de Dátil**

en México. TIES, Revista de Tecnología e Innovación en Educación Superior, no. 8, noviembre, 2023. [En línea]. Disponible en: https://ties.unam.mx/

En el presente artículo se presenta la problemática de la clasificación de dátiles y se describe la participación en la convocatoria Alianza Huawei-UNAM en la que se otorgaban recursos físicos (hardware) específicamente acceso a horas de procesamiento para entrenamiento y evaluación de modelos en servidores Huawei con procesadores Ascend. En esta convocatoria los equipos disponibles eran Atlas 800 (Modelo 9000), Huawei Atlas 800 (Modelo 3010) o Huawei Thaishan (Modelo 2280). Además, ofrecían recursos lógicos (software) y capacitación en temas relacionados con la inteligencia artificial, aprendizaje automático y profundo, certificaciones de nube Huawei, tales como Huawei Certified ICT Associate (HCIA), Huawei Certified ICT Professional (HCIP). Huawei Certified ICT Expert (HCIE) y talleres de innovación centrada en las personas (Design Thinking). La principal contribución del proyecto fue el desarrollo de aplicaciones de inteligencia artificial centradas en las necesidades y problemáticas de los pequeños productores de dátil en México. Estas tienen el objetivo de ayudar a los productores a reducir costos, optimizar procesos, incrementar productividad y sustentabilidad.

6.3 Trabajo Futuro

6.3.1 Mejoras en la Arquitectura del Modelo

En el camino hacia la excelencia en la clasificación de dátiles Medjool, es esencial explorar arquitecturas de modelos más avanzados, por ejemplo, tensores (Chen *et al.*, 2022). La investigación y comparación de redes neuronales convolucionales (CNN) de última generación o la consideración de modelos preentrenados específicos para tareas de clasificación de imágenes podrían potenciar la capacidad del sistema. Además, la transferencia de aprendizaje desde modelos entrenados en grandes conjuntos de datos de imágenes podría acelerar el proceso de entrenamiento y mejorar la generalización del modelo.

6.3.2 Manejo de Datos Desbalanceados

Dada la naturaleza desbalanceada de los conjuntos de datos, técnicas específicas son necesarias para abordar este desafío. La implementación y evaluación de técnicas de aumento

de datos (Shorten *et al.*, 2019), así como enfoques de remuestreo como la submuestreo y sobremuestreo (Mohamed *et al*., 2020), podrían contrarrestar el desbalance de clases y mejorar la capacidad del modelo para clasificar con precisión diferentes categorías.

6.3.3 Interpretabilidad del Modelo

La interpretación de modelos es crucial para ganar la confianza de los usuarios y entender el proceso de toma de decisiones. La aplicación de técnicas como saliency maps (Gómez *et al.*, 2022) y métodos basados en atención (Niu *et al.,* 2021) puede proporcionar información valiosa sobre las características que el modelo utiliza para realizar predicciones. Además, la visualización de activaciones en las capas intermedias del modelo puede ofrecer una comprensión más profunda de cómo se extraen y procesan las características.

6.3.4 Optimización para Recursos Limitados

Con miras a implementaciones prácticas en entornos agrícolas, es esencial optimizar los modelos para recursos limitados. La adaptación del modelo para su ejecución eficiente en dispositivos embebidos y la evaluación del rendimiento en tiempo real son aspectos críticos. Esto garantizará que el modelo pueda desplegarse en sistemas con restricciones de recursos, lo que es especialmente importante en entornos agrícolas.

6.3.5 Robustez en Condiciones Variables

La evaluación del modelo en condiciones adversas es esencial para garantizar su robustez y confiabilidad en diversas circunstancias. La prueba del modelo en condiciones climáticas variables y la investigación de métodos de adaptación continua son pasos esenciales para asegurar que el modelo pueda ajustarse a cambios en el entorno y en las condiciones de los datos a lo largo del tiempo.

Una posible solución en la clasificación de frutas en un supermercado, sería la implementación de un sistema de reconocimiento de frutas y verduras. Actualmente, un cajero de un supermercado, no solo tiene que reconocer el tipo de producto (fruta o verdura) del que se trata (tomate, peras, manzanas, entre otros), sino, además, reconocer su variedad (saladette, bola, uva, etc.).

Una posible solución para la identificación de enfermedades fúngicas en dátiles. Estas enfermedades presentan afectaciones iniciales en pericarpio, mesocarpio, endocarpio, así

como en la semilla de dátil, daños que podrían ser reconocidos con un sistema de reconocimiento y clasificación como el propuesto en este trabajo de investigación.

Referencias

Altaheri, H., Alsulaiman, M., Muhammad, G., Amin, S. U., Bencherif, M., & Mekhtiche, M. (2019a). Date fruit dataset for intelligent harvesting. Data in brief, 26, 104514. https://www.sciencedirect.com/science/article/pii/S2352340919308698

Altaheri, H., Alsulaiman, M., & Muhammad, G. (2019b). Date fruit classification for robotic harvesting in a natural environment using deep learning. IEEE Access, 7, 117115-117133.

Ashraf, S., Kadery, I., Chowdhury, M. A. A., Mahbub, T. Z., & Rahman, R. M. (2019). Fruit Image Classification Using Convolutional Neural Networks. International Journal of Software Innovation (IJSI), 7(4), 51-70.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In Neural networks: Tricks of the trade (pp. 437-478). Springer, Berlin, Heidelberg.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

Bisong, E. (2019). Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners. Apress.

Chen, C., Batselier, K., & Wong, N. (2022). Tensor network algorithms for image classification. In Tensors for Data Processing (pp. 249-291). Academic Press.

Cui, H., and Bai, J. (2019). A new hyperparameters optimization method for convolutional neural networks. Pattern Recognition Letters, 125, 828-834. https://doi.org/10.1016/j.patrec.2019.02.009

Dehghani, M., Mobaien, A., & Boostani, R. (2021). A deep neural network-based transfer learning to enhance the performance and learning speed of BCI systems. Brain-Computer Interfaces, 8(1-2), 14–25. doi:10.1080/2326263x.2021.1943955

Dodge, S., & Karam, L. (2016, June). Understanding how image quality affects deep neural networks. In 2016 eighth international conference on quality of multimedia experience (QoMEX) (pp. 1-6). IEEE. DOI: 10.1109/QoMEX.2016.7498955.

Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. Neurocomputing.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338. https://link.springer.com/article/10.1007/s11263-009-0275-4

Faisal, M., Albogamy, F., Elgibreen, H., Algabri, M., & Alqershi, F. A. (2020). Deep learning and computer vision for estimating date fruits type, maturity level, and weight. IEEE Access, 8, 206770-206782.

Feng, J., & Darrell, T. (2015). Learning the structure of deep convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 2749-2757). https://openaccess.thecvf.com/content_iccv_2015/html/Feng_Learning_The_Structure_ICCV_2015_paper.html

García-Vázquez, J. P., Torres, R. S., and Pérez, D. B. P. (2021). Scientometric Analysis of the Application of Artificial Intelligence in Agriculture. *Journal of Scientometric Research*, *10*(1), 55-62. DOI: 10.5530/jscires.10.1.7

García-Vázquez, J. P. R., Torres, S., Pérez, D. B., Zatarain, H., Soto, J. E. y García, E. R. Inteligencia Artificial al Servicio de los Pequeños Productores de Dátil en México," TIES, Revista de Tecnología e Innovación en Educación Superior, n.o. 8, noviembre, 2023. [En línea]. Disponible en: https://ties.unam.mx/ [Consultado 21/12/2023].

Gholamalinezhad, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. arXiv preprint arXiv:2009.07485.

Gomez, T., Fréour, T., & Mouchère, H. (2022, June). Metrics for saliency map evaluation of deep learning explanation methods. In International Conference on Pattern Recognition and Artificial Intelligence (pp. 84-95). Cham: Springer International Publishing.

Haidar, A., Dong, H., & Mavridis, N. (2012, October). Image-based date fruit classification. In 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems (pp. 357-363). IEEE.

Han, J. H., Choi, D. J., Park, S. U., & Hong, S. K. (2020). Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network. Journal of Electrical Engineering & Technology, 15(2), 721-726.

Han, X., Zhong, Y., Cao, L., & Zhang, L. (2017). Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification. Remote Sensing, 9(8), 848.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Kamilaris, A., & Prenafeta-Boldú, F. (2018a). A review of the use of convolutional neural networks in agriculture. The Journal of Agricultural Science, 156(3), 312-322. doi:10.1017/S0021859618000436

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018b). Deep learning in agriculture: A survey. Computers and electronics in agriculture, 147, 70-90.

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT express, 6(4), 312-315.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90.

Koutsoukas, A., Monaghan, K. J., Li, X., & Huan, J. (2017). Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. Journal of cheminformatics, 9(1), 1-13.

Liashchynskyi, P., and Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059.

Li, Y., Zhao, Z., Luo, Y., & Qiu, Z. (2020). Real-time pattern-recognition of GPR images with YOLO V3 implemented by TensorFlow. Sensors, 20(22), 6476. https://www.mdpi.com/1424-8220/20/22/6476

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems.

Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. Plant Methods, 17(1), 1-18.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., & Zhang, G. (2015). Transfer learning using computational intelligence: A survey. Knowledge-Based Systems, 80, 14-23.

Michán, L., & Muñoz-Velasco, I. (2013). Cienciometría para ciencias médicas: definiciones, aplicaciones y perspectivas. *Investigación en educación médica*, *2*(6), 100-106. ISSN 2007-5057.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 1419.

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020, April). Machine learning with oversampling and undersampling techniques: overview study and experimental results. In 2020 11th international conference on information and communication systems (ICICS) (pp. 243-248). IEEE.

Motta, D., Santos, A. Á. B., Machado, B. A. S., Ribeiro-Filho, O. G. V., Camargo, L. O. A., Valdenegro-Toro, M. A., ... & Badaró, R. (2020). Optimization of convolutional

neural network hyperparameters for automatic classification of adult mosquitoes. Plos one, 15(7), e0234959.

Nasiri, A., Taheri-Garavand, A., & Zhang, Y. D. (2019). Image-based deep learning automated sorting of date fruit. Postharvest biology and technology, 153, 133-141.

Nazir, S., Patel, S., & Patel, D. (2020). Assessing hyper parameter optimization and speedup for convolutional neural networks. International Journal of Artificial Intelligence and Machine Learning (IJAIML), 10(2), 1-17.

Niu, S., Liu, Y., Wang, J., & Song, H. (2020). A Decade Survey of Transfer Learning (2010–2020). IEEE Transactions on Artificial Intelligence, 1(2), 151–166. doi:10.1109/tai.2021.3054609

Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of deep learning. Neurocomputing, 452, 48-62.

Nguyen, H. N., & Lee, C. (2018). Effects of Hyper-parameters and Dataset on CNN Training. Journal of IKEEE, 22(1), 14-20.

Ozaki, Y., Yano, M., & Onishi, M. (2017). Effective hyperparameter optimization using Nelder-Mead method in deep learning. IPSJ Transactions on Computer Vision and Applications, 9(1), 1-12.

Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

Pérez-Pérez, D. B., Salomón-Torres, R., & García-Vázquez, J. P. (2021). Dataset for localization and classification of Medjool dates in digital images. *Data in Brief*, *36*, 107116.

Pérez-Pérez, B. D., García Vázquez, J. P., & Salomón-Torres, R. (2021). Evaluation of convolutional neural networks' hyperparameters with transfer learning to determine sorting of ripe medjool dates. *Agriculture*, *11*(2), 115.

Radiuk, P. M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Information Technology and Management Science, 20(1), 20-24.

Rahnemoonfar, M., & Sheppard, C. (2017). Deep count: fruit counting based on deep simulated learning. Sensors, 17(4), 905.

Shekar, B. H., & Dagnew, G. (2019, February). Grid search-based hyperparameter tuning and classification of microarray cancer data. In 2019 second international conference on advanced computational and communication paradigms (ICACCP) (pp. 1-8). IEEE.

Shi, B., Hou, R., Mazurowski, M. A., Grimm, L. J., Ren, Y., Marks, J. R., ... & Lo, J. Y. (2018, February). Learning better deep features for the prediction of occult invasive disease in ductal carcinoma in situ through transfer learning. In *Medical imaging 2018: computer-aided diagnosis* (Vol. 10575, pp. 620-625). SPIE.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.

Siddiqi, R. (2019, July). Effectiveness of transfer learning and fine tuning in automated fruit image classification. In Proceedings of the 2019 3rd International Conference on Deep Learning Technologies (pp. 91-100).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556.

Soil-Ccama D.A. (2022). Redes Neuronales Convolucionales (CNN). Disponible en línea en: https://rstudio-pubs-static.s3.amazonaws.com/896010_68e6c7c0995e4d90b75f02961a5d67fb.html (accedido, 21/12/2023)

Suriya, M., Chandran, V., & Sumithra, M. G. (2022). Enhanced deep convolutional neural network for malarial parasite classification. *International Journal of Computers and Applications*, 44(12), 1113-1122.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

Torres, J. (2018). DEEP LEARNING Introducción práctica con Keras. WATCH THIS SPACE, Tercera Edición, Barcelona, 207 páginas, ISBN: 9780244078959

Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques (pp. 242-264). IGI global.

Wang, Y., Li, Y., Song, Y., & Rong, X. (2020). The influence of the activation function in a convolution neural network model of facial expression recognition. Applied Sciences, 10(5), 1897.

Zaccone, G., & Karim, M. R. (2018). Deep learning with tensorFlow: Explore neural networks and build intelligent systems with python. Packt Publishing Ltd.

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., ... & Almotairi, S. (2022). A Comparison of Pooling Methods for Convolutional Neural Networks. Applied Sciences, 12(17), 8643.

Zeng, G. (2017, October). Fruit and vegetables classification system using image saliency and convolutional neural network. In 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC) (pp. 613-617). IEEE

Zhang, Z. (2018, June). Improved adam optimizer for deep neural networks. In 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS) (pp. 1-2). Ieee.

Zhao, G., Zhang, Z., Guan, H., Tang, P., & Wang, J. (2018, August). Rethinking ReLU to train better CNNs. In 2018 24th International Conference on Pattern Recognition (ICPR) (pp. 603-608). IEEE.