ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

“DETECCIÓN DE FALLAS EN TERMOGRAMAS UTILIZANDO APRENDIZAJE PROFUNDO”

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR:

ANDRÉS SEBASTIÁN VALENCIA ZAVALA

Riobamba – Ecuador

2023

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

"DETECCIÓN DE FALLAS EN TERMOGRAMAS UTILIZANDO APRENDIZAJE PROFUNDO"

Trabajo de Integración Curricular

Tipo: Proyecto de Investigación

Presentado para optar al grado académico de:

INGENIERO EN MANTENIMIENTO INDUSTRIAL

AUTOR: ANDRÉS SEBASTIÁN VALENCIA ZAVALA

DIRECTOR: Ing. FÉLIX ANTONIO GARCÍA MORA

Riobamba – Ecuador

2023

Yo, Andrés Sebastián Valencia Zavala, declaro que el presente Trabajo de Integración Curricular es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Integración Curricular; el patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 11 de enero de 2023

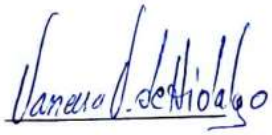
Andrés Sebastián Valencia Zavala
060495218-4

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE MECÁNICA

CARRERA MANTENIMIENTO INDUSTRIAL

El Tribunal del Trabajo de Integración Curricular certifica que: El Trabajo de Integración Curricular; Tipo: Proyecto de Investigación, **DETECCIÓN DE FALLAS EN TERMOGRAMAS UTIIZANDO APRENDIZAJE PROFUNDO**, realizado por el señor: **ANDRÉS SEBASTIÁN VALENCIA ZAVALA**, ha sido minuciosamente revisado por los Miembros del Tribunal del Trabajo de Integración Curricular, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

	FIRMA	FECHA
Ing. Edwin Ángel Jácome Domínguez **PRESIDENTE DEL TRIBUNAL**		2023-01-11
Ing. Félix Antonio García Mora **DIRECTOR DE TRABAJO DE TITULACIÓN**		2023-01-11
Ing. Vanessa Lorena Valverde Gonzáles **MIEMBRO DEL TRIBUNAL**		2023-01-11

DEDICATORIA

Quiero dedicar este Trabajo de Integración Curricular a Dios, a mis padres Santiago Valencia y a mi madre Rosario Zavala, de igual manera a todas las personas que me han apoyado incondicionalmente en esta trayectoria de mi vida.

Andrés Valencia

AGRADECIMIENTO

Quiero agradecer nuevamente a Dios, a mis padres y a los docentes por haber impartido sus conocimientos dentro de las aulas de estudio, especialmente al Ing. Félix García y a la Ing. Vanessa Valverde quienes me ha orientado en el desarrollo del Trabajo de Integración Curricular; además quiero agradecer a la Escuela Superior Politécnica de Chimborazo por permitirme desarrollar los estudios universitarios, principalmente a la Carrera de Mantenimiento Industrial por haberme acogido como uno de sus ejemplares estudiantes.

Andrés Valencia

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

RESUMEN

En el presente Trabajo de Integración Curricular se realizó la detección de fallas en termogramas utilizando Aprendizaje Profundo, este campo de la Inteligencia Artificial basa su funcionamiento en redes neuronales artificiales las cuales tratan de simular el comportamiento del cerebro humano, por lo que se utilizó información de artículos científicos, libros, tesis, revistas e información de páginas web y librerías de Inteligencia Artificial. Para desarrollar el algoritmo se necesitó una base de datos que contenga termogramas con fallo y termogramas sin fallo, dicha base fue obtenida de un estudio previo de termografía realizado en la Carrera de Mantenimiento Industrial de la ESPOCH. Para la programación del algoritmo se utilizó el software libre denominado Python. Inicialmente se realizó un preprocesamiento de las imágenes y clasificación de los termogramas para entrenamiento, validación y prueba; Debido a que la base de datos fue relativamente pequeña se utilizó una herramienta denominada ImageDataGenerator que sirve para aumentar la cantidad de imágenes, realizando ciertas modificaciones en los termogramas originales. Se generó 4 modelos de redes neuronales para observar cuál de ellos presentaba mejores resultados y posteriormente ser seleccionado como el mejor algoritmo de clasificación, en este caso el modelo 4 fue el que mejor resultados presentó, obteniendo una precisión de 97,29% y una exactitud de 97,94%, además se obtuvo resultados de otras métricas como el F1-score dando un resultado de 97,95%. Finalmente se realizaron predicciones con los termogramas de prueba y el algoritmo los clasificó de manera correcta corroborando así que es un modelo exitoso para detectar fallas en termogramas por lo cual se recomienda realizar más temas de investigación relacionados a termografía con inteligencia artificial dentro del mantenimiento industrial.

Palabras clave: <INTELIGENCIA ARTIFICIAL>, <DEEP LEARNING>, <REDES NEURONALES>, <REDES CONVOLUCIONALES >, <TERMOGRAMAS>

0235-DBRA-UPT-2023

SUMMARY

This research carried out the detection of failures in thermograms using Deep Learning. This field of Artificial Intelligence bases its operation on artificial neural networks which try to simulate the behavior of the human brain. For that reason, we got information from scientific articles, books, research, magazines, and news from Artificial Intelligence web pages and bookstores. A database containing thermograms with failure and thermograms without failure was needed to develop the algorithm. This database was obtained from a previous thermography study carried out in the Industrial Maintenance Career of ESPOCH. For the programming of the algorithm, the free software called Python was used. Initially, a pre-processing of the images and classification of the thermograms for training, validation, and testing were carried out. Because the database was relatively small, a tool called Image Data Generator was used. It serves to increase the number of images, making certain modifications to the original thermograms. Four models of neural networks were generated to observe which of them presented the best results and later selected as the best classification algorithm. Model 4 was the one that showed the best results, obtaining a precision of 97.29% and an accuracy of 97.94%. In addition, results of other metrics, such as the F1-score, were acquired, giving a mark of 97.95%. Finally, predictions were made with the test thermograms, and the algorithm classified them correctly, thus corroborating that it is a successful model for detecting failures in thermograms. For this reason, it is recommended to carry out more research on thermography with artificial intelligence within industrial maintenance.

Keywords: <ARTIFICIAL INTELLIGENCE>, <DEEP LEARNING>, <NEURAL NETWORKS>, <CONVOLUTIONAL NETWORKS>, <THERMOGRAMS>

Lic. Sandra Leticia Guijarro Paguay
C.I.: 0603366113

INTRODUCCIÓN

La inteligencia artificial es un amplio campo de estudio en la cual se incluye varias teorías, métodos y tecnologías que apuntan al mismo lugar en dispositivos en los cuales se les provee de softwares entrenados y precisos para que puedan conocer el entorno en el que se encuentran y así poder procesarlo y producir un resultado óptimo y consistente. (Fuentes, 2022, p.19)

Una de las técnicas empleadas en inteligencia artificial es el aprendizaje profundo y el aprendizaje por refuerzo profundo, el cual consigue extender las técnicas previas de aprendizaje por refuerzo incluyendo redes neuronales, logrando de esta manera aplicaciones muy exitosas tanto en dominios discretos como continuos. (Sánchez, 2021, p.3)

Los algoritmos de IA toman decisiones a partir de un entrenamiento previo basándose en datos, por lo tanto, si estos datos no son de buena calidad o no representan el dominio en el que se quiere trabajar, el sistema aprenderá erróneamente. Por ello es de vital importancia tener un buen conjunto de datos para lograr el éxito de la solución que se desee. (Cabrera, 2021, p.5)

La aplicación de técnicas de sistemas cognitivos artificiales es relevante para solucionar problemas debido a la alta capacidad de identificación de características y clasificación que poseen, además abre la puerta para el uso de teorías de transferencia de aprendizaje en donde se extrapolan las soluciones propuestas con datos experimentales a problemas similares presentes en escenarios industriales.(Arias, 2020, p.10)

Incluso se ha demostrado que la utilización de aprendizaje profundo en la odometría visual puede llegar a estimar valores cercanos a los valores reales de las secuencias de imágenes, por consiguiente, los resultados son competitivos con los métodos analíticos desarrollados para la estimación de posición y orientación.(Tayupanta, 2020, p.1.)

En el mantenimiento industrial una de las características más importantes es la detección de fallos, ya sea para mantenimiento preventivo o correctivo, para ello existen diferentes técnicas de detección, pero lo ideal es determinar la falla antes de que se produzca la avería. Dentro del mantenimiento basado en la condición existe el mantenimiento predictivo en el cual se utilizan diferentes métodos para poder detectar los síntomas que se presentan, uno de ellos es la termografía la cual permite determinar los puntos calientes en un elemento o sistema y así obtener una premisa para el posterior análisis del mismo.

Definición del problema

La detección de fallos mediante termografía es un método muy avanzado y útil para el mantenimiento, sin embargo, al momento de interpretar los termogramas se generan confusiones ya que las imágenes obtenidas suelen ser similares para diferentes fallos o a su vez puede no estar en fallo el elemento analizado, para ello se requiere de una especialización rigurosa y un amplio conocimiento dentro de dicha área, a pesar de ello se puede producir una gran tasa de fallo de error humano.

Justificación del problema

Actualmente la tecnología avanza de manera acelerada a nivel global, con la utilización de la inteligencia artificial aplicando la técnica de aprendizaje profundo se han creado tecnologías que obligan estar a la par en cuanto al mantenimiento, la inteligencia artificial es una disciplina en auge que ha redefinido muchos de los procesos que se realizan en la industria, presentando aplicaciones muy diversas, las cuales abarcan el reconocimiento y síntesis de voz, comprensión lectora, sistema de traducción, compresión del lenguaje, entre otros. (Pérez, 2021, p.72.)

Gracias al gran avance de las redes neuronales dentro del Deep learning, en lo que se refiere a redes convolucionales, el análisis de las imágenes es más preciso y exacto al momento de analizarlas. En el Ecuador aún no se aprecia mucho está técnica aplicada al mantenimiento industrial, sin embargo, ya se ha propuesto algunas técnicas basadas en inteligencia artificial, aplicada a diferentes sectores de la industria, incluso se ha utilizado para analizar el inmenso volumen de datos de consumo de energía eléctrica y así obtener información de la demanda energética en los sectores residenciales.

En la Escuela Superior Politécnica de Chimborazo se han desarrollado trabajos de investigación basados en inteligencia artificial y enfocadas precisamente al mantenimiento industrial gracias a las diferentes técnicas que propone la IA, se ha comprobado la gran ventaja de aplicarla en el mantenimiento ya que de esta manera se optimiza el tiempo y la detección de fallos son más rápidos y eficientes.

Hipótesis

Utilizando el método de aprendizaje profundo se detecta fallas en imágenes termográficas.

Variable dependiente

Detección de fallas

Variable independiente

Matriz de confusión.
Precisión.
Exactitud.

Objetivos

Objetivo general

Detectar fallos en termogramas utilizando aprendizaje profundo.

Objetivos específicos

Procesar la base de datos de termografía para la detección de fallos.

Dividir la base de datos para entrenamiento y prueba.

Encontrar las características de extracción que mejor correlacionan los termogramas.

Aplicar el algoritmo de clasificación para la detección de fallas utilizando termogramas.

Verificar la efectividad del método empleado para la detección de fallos utilizando termogramas.

CAPÍTULO I

1. MARCO TEÓRICO REFERENCIAL

1.1. Fallo

Cese en la capacidad de un elemento para desarrollar una función requerida.(UNE-EN 13306, 2018, p.9)

1.2. Termograma

El termograma es una representación gráfica obtenida de una cámara termográfica la cual basa su funcionamiento en la radiación infrarroja que emiten los cuerpos o materiales. Los colores del termograma varían según la paleta de colores que se seleccione, pero el punto más caliente es representado por el color blanco. Para poder determinar si existe o no un fallo, se deben tomar ciertos datos adicionales como la temperatura ambiente en la que se encuentra el elemento, velocidad del viento, humedad, entre otras, una vez determinados estos datos se realiza la diferencia entre la temperatura del objeto y la temperatura ambiente obteniendo un nuevo valor de temperatura, el cual es comparado con valores de tablas la cual nos indican el estado del elemento y el tiempo de acción para corregir el fallo en caso de existir.

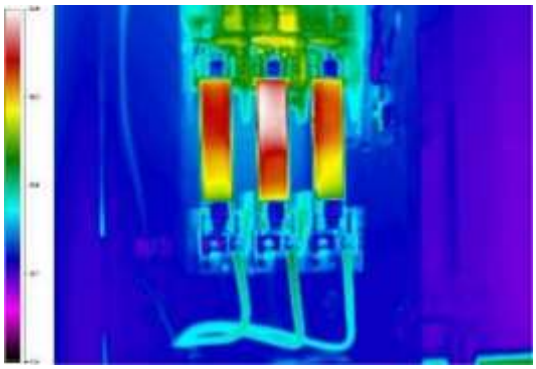


Figura 1-1: Termograma

Fuente: Pinterest, 2022

1.3. Algoritmo

Un algoritmo es un conjunto de pasos ordenados que permiten solucionar problemas.

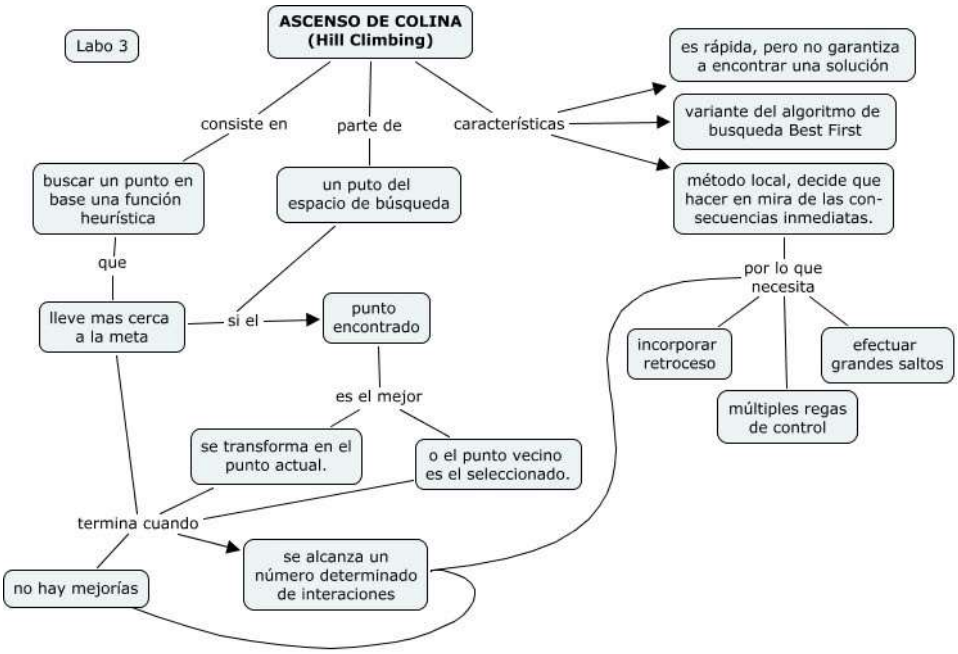


Gráfico 1-1: Algoritmo

Fuente: Sites Inteligencia Artificial, 2022

1.4. Inteligencia artificial

La inteligencia artificial es una disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, cómo el aprendizaje o el razonamiento lógico.

La IA intenta crear comportamientos inteligentes a partir de modelos matemáticos. Se suele oponer que el comportamiento humano tiene partes fundamentales que no son susceptibles de ser analizadas a través de un modelo matemático como son el sentido común, los sentimientos y las emociones. (Yague, 2021, p.208)

Figura 2-1: Inteligencia Artificial

Fuente: Lavanguardia, 2022

1.5. Procesamiento del lenguaje natural (*Natural languaje processing*)

El procesamiento del lenguaje natural o NLP por sus siglas en inglés, es un componente de la inteligencia artificial, aplicado a las máquinas para tener la capacidad de comprender el lenguaje humano tal como se lee y se escribe, mediante la aplicación de un programa.

El Natural languaje processing utiliza inteligencia artificial para tomar información del mundo real, procesarla y darle sentido de tal manera que la computadora pueda entender, todo esto independiente del idioma que sea hablado o escrito. Así como los humanos poseen diferentes sensores naturales como los ojos, oídos; las computadoras tienen programas para leer o micrófonos para recolectar audio, luego procesan la información adquirida y convierte la entrada en un código que pueda entender. (Atanassova, 2019, p.1)

1.6. Aprendizaje profundo

La IA abarca diferentes disciplinas cómo: el aprendizaje automático o machine learning, el aprendizaje profundo o Deep learning, minería de datos, entre otras; El aprendizaje profundo es un tipo de inteligencia artificial computarizada la cual es entrenada para que realice tareas que los humanos usualmente las realizamos, basándose en redes neuronales artificiales.(Schmetterer, 2021, p.7)

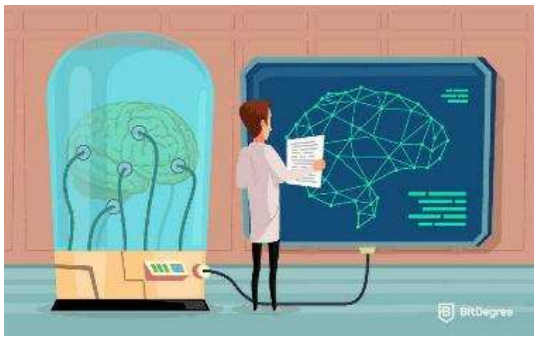


Figura 3-1: Aprendizaje profundo

Fuente: BitDegree, 2022

A diferencia de los algoritmos tradicionales de aprendizaje automático que tienen una capacidad de aprendizaje finito, los sistemas de aprendizaje profundo acceden a un mayor número de datos adquiriendo así mayor experiencia, una vez adquirida está experiencia las máquinas pueden resolver tareas complejas cómo conducir un auto, detectar enfermedades, identificar errores en máquinas, etc.

Para la extracción y transformación de datos se emplea una gran cantidad de múltiples capas de

unidad y procesamiento no lineales, tienes dos tipos de aprendizaje, supervisado y no supervisado; La era de la nube y los grandes datos han llevado al Deep learning a una mayor precisión y capacidad para entrenarse y ser más autónomo. En la entrada recibe los datos que se incluyen y pasa a las capas ocultas que realizan cálculos matemáticos que le permite tener una base de entradas y la capa de salida entrega un resultado final.

1.7. Perceptrón

El perceptrón, fue el primer modelo de red neuronal artificial desarrollado por Rosenblatt en 1958. Su estructura consta de una capa de entrada de varias neuronas que transmiten la información sin procesarla y una capa de salida formada por una única neurona que procesa las señales de entrada, permitiendo clasificar datos en dos categorías según los patrones de salida u outputs que ellos producen.(García, 2022, p.3)

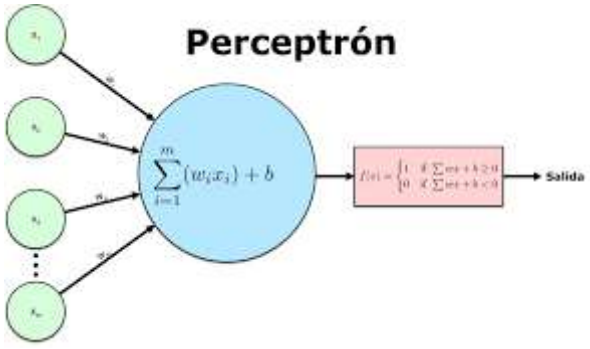


Figura 4-1: Perceptrón

Fuente: blog José Mariano Álvarez, 2022

Una neurona artificial está inspirada en el funcionamiento de una neurona biológica, esta última es una célula nerviosa con un cuerpo o soma, rodeado de una serie de ramificaciones o dendritas, unidas al cuerpo existe un filamento conocido como axón que se extiende y ramifica en terminales axónicos, cada neurona recibe señales a través de las dendritas, las señales recibidas son procesadas o integradas en el cuerpo, cuando la señal eléctrica resultante supera un potencial o voltaje umbral se dispara un potencial de acción a través del axón. El potencial de acción llega a los terminales axónicos y mediante un mecanismo químico en el que participan sustancias denominadas neurotransmisores se establecen conexiones o sinapsis con las dendritas de otra neurona generándose un potencial postsináptico excitatorio o inhibitorio cuya magnitud depende de la intensidad o peso sináptico con que se conecten ambas neuronas. Este proceso de ponderación de señales suma y activación se repite en las demás neuronas y se recupera para modelizar el funcionamiento de una neurona artificial. (García, 2022, p.4)

1.8. Redes neuronales

Una red neuronal artificial es una red computacional inspirada en las redes neuronales humanas, utilizadas para resolver tareas que son difíciles en la programación estructurada. La primera neurona artificial fue inventada en el año 1943 denominada la neurona de McCulloch-Pits, seguido del perceptrón estructurado por una capa de entrada, una capa de neuronas y una de salida. Las neuronas son la base de construcción de una red neuronal, cada una de estas unidades tiene un número k de conexiones de entrada y puede tomar diferentes números de estado. Por ejemplo, una red neuronal que reconozca números del 0 al 9 a partir de imágenes, entonces, las conexiones de entrada será el número de pixeles de la imagen y los posibles estados de la red son imágenes de número del 0 al 9. (Fernández, 2021, p.2)

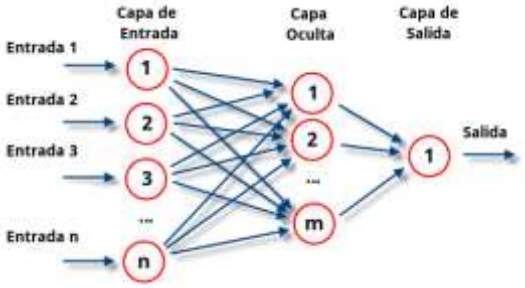


Figura 5-1: Red Neuronal

Fuente: Atriainnovation, 2022

1.9. Redes convolucionales

Las redes convolucionales son redes que se usan para procesar imágenes, estas pueden aprender relaciones de entrada-salida, donde la entrada es una imagen. La convolución se refiere a la obtención de una tercera señal a partir de 2 señales, a través de un proceso con características especiales, es decir en una imagen consiste en filtrarla usando una máscara, por lo tanto, si aplicamos diferentes mascaras obtendremos distintos resultados. (Gómez, 2016, p.8)

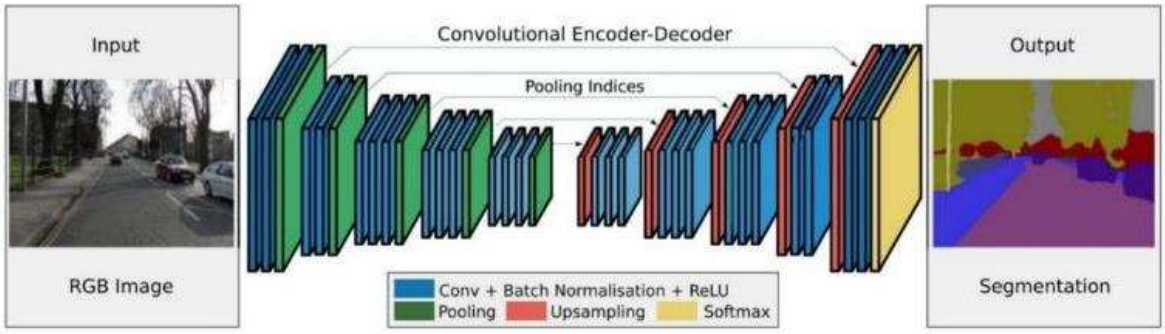


Figura 6-1: Red convolucional

Fuente: IArtificial, 2022

Las tareas más comunes que desarrollan son: detección de objetos, clasificación de escenas y clasificación de imágenes en general. En la convolución cada pixel de salida es una combinación lineal de los pixeles de entrada. Las redes se forman usando tres tipos de capas:

- Capas convolucionales
- Capas de pooling
- Capas totalmente conectadas

1.10. Operador sobel

El operador de sobel es un operador de detección de bordes más popular hasta el desarrollo de las técnicas de detección de bordes con una base teórica ya que proporcionaba un mejor rendimiento que otros operadores contemporáneos de detección de bordes, como el operador Prewit. (Falconí, 2021, p.7)

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figura 7-1: Operador Sobel

Fuente: programador clic, 2022

Figura 8-1: Resultado operador Sobel

Fuente: researchgate, 2022

1.11. Algoritmo de Canny

El algoritmo de Canny funciona de la siguiente manera, permite identificar los bordes, límites del objeto y detectar a través de cambios abruptos el nivel de gris con el fin de detectar el objeto. (Prieto, 2022, p.46)

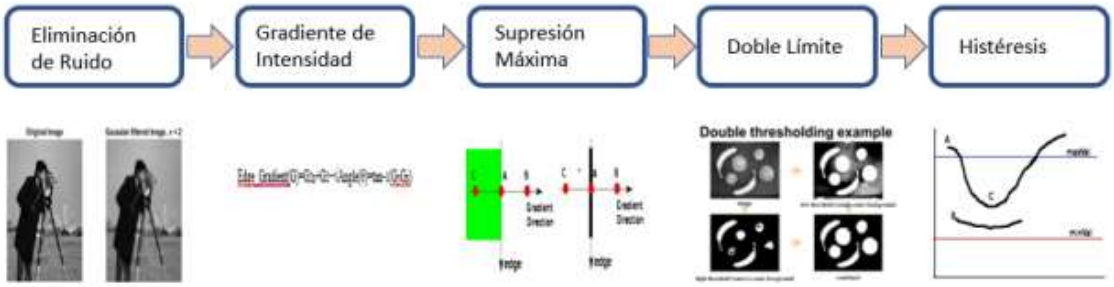


Figura 9-1: Algoritmo de Canny

Fuente: Prieto, 2022

1.12. Agrupación máxima

La agrupación máxima mejor conocida como maxpooling es una técnica de las redes neuronales convolucionales, la cual consiste en realizar un mapeo de la imagen y obtener como salida el pixel con mayor valor numérico de todo el conjunto de pixeles analizado, también existen otras técnicas que consisten en obtener un valor medio del kernel o núcleo que se le aplique o a su vez obtener el menor número de ellos.

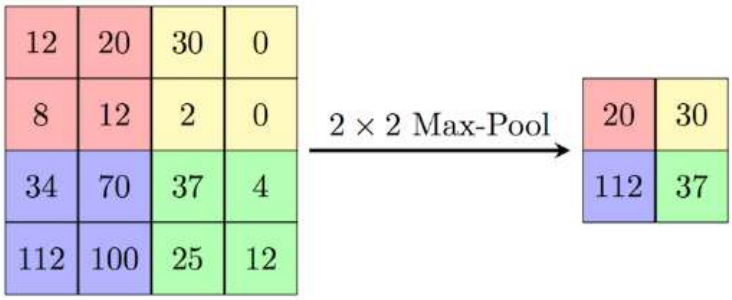


Figura 10-1: Maxpooling

Fuente: paperswithcode, 2022

1.13. Técnica dropout

La técnica del dropout utilizada en redes neuronales, consiste en desconectar de manera aleatoria cierto porcentaje de neuronas durante el entrenamiento, esto para evitar que la red neuronal se memorice los datos y exista un sobreajuste.

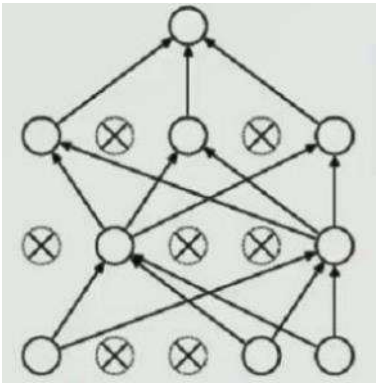


Figura 11-1: Dropout

Fuente: programador clic, 2022

1.14. Transfer learning

El transfer learning o aprendizaje por transferencia tiene como objetivo transferir el conocimiento del dominio de origen al dominio de destino. Es una herramienta importante para resolver el problema de los datos de entrenamiento ineficiente. (Xiao, 2022, p.3)

El aprendizaje por transferencia ajusta los modelos con datos a pequeña escala de la tarea y también mantiene la capacidad de aprendizaje con la diferencia individual. (Wan, 2021, p.2)

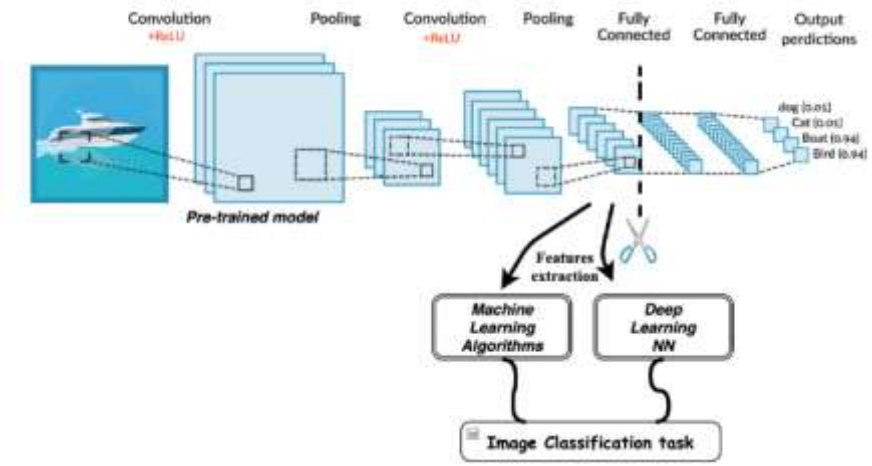


Figura 12-1: Transfer learning

Fuente: médium, 2022

1.15. Tipos de aprendizaje

Dentro del Machine learning existen diferentes tipos de aprendizaje tales como: Aprendizaje supervisado, aprendizaje semi supervisado, aprendizaje por refuerzo y aprendizaje no supervisado, este último es el que más se relaciona con inteligencia artificial, a continuación, se explicará cada uno de ellos.

1.15.1. Aprendizaje Supervisado

El aprendizaje supervisado se aplica cuando cada dato o conjunto de datos de entrada (muestra), tiene asociada una etiqueta. Por ejemplo, en el caso de termografía se dispone de termogramas con fallo y sin fallo, estas categorías vendrían a ser las etiquetas de cada termograma. Partiendo de este conjunto de datos se puede usar diferentes algoritmos de clasificación con el objetivo de entrenar un modelo y posteriormente predecir la etiqueta correspondiente a una imagen que nunca ha visto el algoritmo. Los nuevos algoritmos destinados a resolver diferentes tipos de aprendizaje supervisado hacen posible la obtención de resultados significativos como: la conducción automática, reconocimiento facial, sistemas de recomendación, etc.(Bobadilla, 2021, p.14)

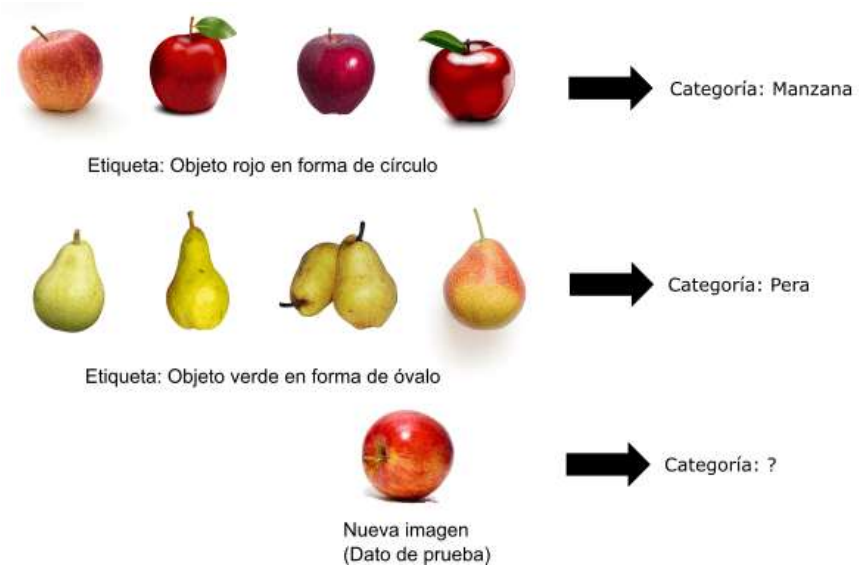


Figura 13-1: Aprendizaje Supervisado

Fuente: ceupe, 2022

1.15.2. Aprendizaje no Supervisado

El aprendizaje no supervisado o unsupervised transfer learning está estrechamente más relacionado con la inteligencia artificial, ya que da la idea de que una computadora pueda aprender a identificar patrones y procesos complejos sin un humano para proporcionar orientación en el transcurso, es por ello que tiene la capacidad de resolver problemas complejos utilizando solo los datos de entrada y los algoritmos lógicos sin tener datos de referencia. Este tipo de aprendizaje utiliza información no etiquetada, el algoritmo se encarga por sí solo de extraer características que relacionen un dato con el otro, una de las aplicaciones más conocida es la de clustering o agrupamiento, como su nombre lo indica se encarga de agrupar muestras, para posteriormente etiquetarlas o clasificarlas por sí solo. (Bobadilla, 2021, p.17)

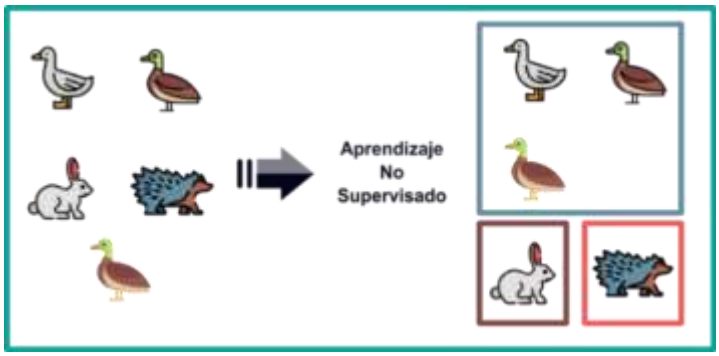


Figura 14-1: Aprendizaje no Supervisado

Fuente: Machine learning en español.com, 2022

1.15.3. Aprendizaje Semi supervisado

El aprendizaje semisupervisado trata con conjuntos de datos en los que una parte de los datos está etiquetada y el resto no lo está. Regularmente, el número de muestras etiquetadas es mucho más pequeña que las no etiquetadas, la mayoría de algoritmos con este tipo de aprendizaje son una mezcla de métodos supervisados y no supervisados. (Bobadilla, 2021, p.18)

Figura 15-1: Aprendizaje Semisupervisado

Fuente: stratebi, 2022

1.15.4. Aprendizaje por refuerzo

El aprendizaje por refuerzo es un área innovadora y con un gran futuro, ya que está inspirada en mecanismos naturales. En este caso, el algoritmo de aprendizaje recibe información de un entorno real o simulado. Cuando el sistema realiza una acción es recompensado o penalizado, tal y como pasa con los seres vivos. Estos algoritmos de aprendizaje se denominan agentes y pueden aprender siguiendo los principios de la evolución natural, los agentes aprenden estrategias, denominadas políticas, que maximizan las recompensas y minimizan las penalizaciones. La mayoría de los sistemas de inteligencia artificial actuales, que están especializados en juegos, están basados en

el enfoque de aprendizaje por refuerzo. (Bobadilla, 2021, p.18)

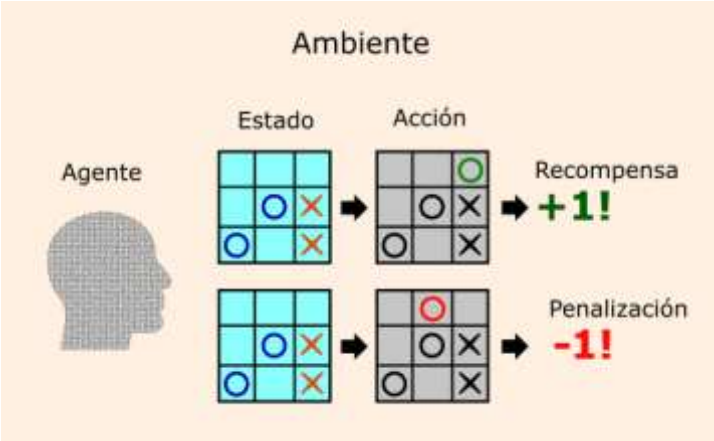


Figura 16-1: Aprendizaje por refuerzo

Fuente: ceupe, 2022

1.16. Técnica hold out

La técnica hold out consiste en realizar una división de los datos en tres subconjuntos: el 60% destinado a entrenamiento, el 20% a validación y el 20% restante a las pruebas del modelo. (Pérez-Aguilar, 2020, p.2)

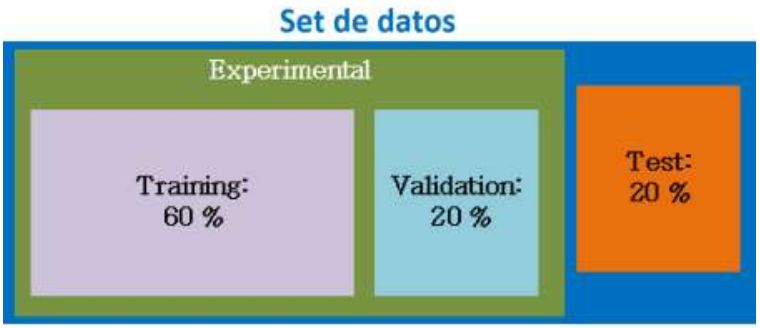


Figura 17-1: Técnica hold out

Fuente: INGENIUS, 2022

1.17. Tensor

Un tensor es una matriz, es decir una estructura de datos que almacena una colección de números a los que se puede acceder individualmente mediante un índice y que se puede indexar con múltiples índices. Desde el punto de vista del aprendizaje profundo los tensores pueden ser de forma escalar, de un vector de fila, de un vector de columna, de una matriz o de una matriz multidimensional.

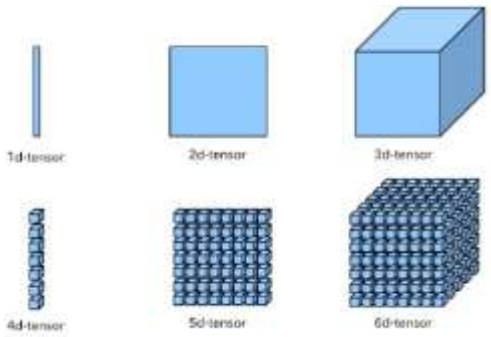


Figura 18-1: Tensores

Fuente: DataDriveInvestor, 2022

1.18. Funciones de activación

A la señal integrada de entrada a una determinada neurona, está le aplica una función que se denomina función de activación, dicha salida constituirá una de las entradas a las neuronas de la siguiente capa. Una característica necesaria que debe tener la función de activación es que sea continua y diferenciable, las funciones de activación pueden ser de tipo escalón, lineales o no lineales, siendo estás últimas las más utilizadas para entrenar redes neuronales artificiales que resuelven problemas complejos.(Valero, 2021, p.6). En cualquier lugar entre dos capas convolucionales y al final de la red se debe agregar una función de activación, para que esta pueda ayudar a tomar decisiones para continuar o para no seguir la secuencia. Las funciones de activación más utilizadas son las siguientes:

1.18.1. Función de activación sigmoide

La función de activación sigmoide se trata de un enfoque probabilístico hacia la toma de decisiones, caracterizándose por tener una gran eficiencia, esta función produce una salida entre 0 y 1, es decir cuando necesitamos tomar una decisión o predecir una salida podemos utilizar dicha función ya que el rango es el mínimo y de esta manera obtendremos una predicción más precisa. (Pelegero, 2022, p.22)

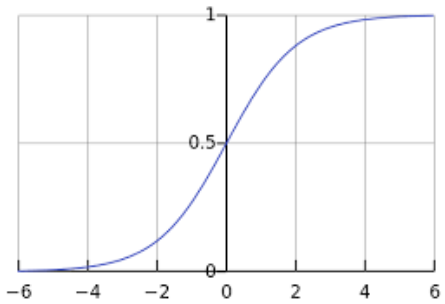


Gráfico 2-1: Función de activación Sigmoide

Fuente: Dpto. de ciencias de la computación e IA, 2022

$$\sigma(X) = \frac{1}{1+e^{-X}} \quad \textbf{(1.1)}$$

1.18.2. Función de activación tangente hiperbólica

Este de tipo de función de activación es un poco mejor que la función de activación sigmoide, pero mapea la entrada negativa solo en cantidad negativa y oscila entre -1 y 1.(Pelegero, 2022, p.23)

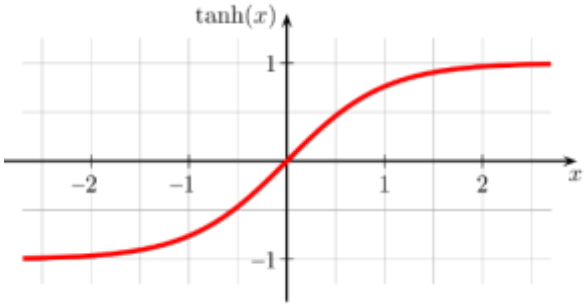


Gráfico 3-1: Función de activación tangente hiperbólica

Fuente: Wikipedia, 2022

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2\sigma(2x) - 1 \quad \textbf{(2.1)}$$

1.18.3. Función de activación Softmax

La función de activación softmax básicamente da valor a la variable de entrada de acuerdo a su peso, y la suma de estos pesos es 1, por ello es más usual utilizarla en la última capa, es decir en la capa de salida para la toma de decisiones. Para los problemas de clasificación de múltiples clases, la función softmax es la más utilizada, también produce una salida entre 0 y 1.(Pelegero, 2022, p.23)

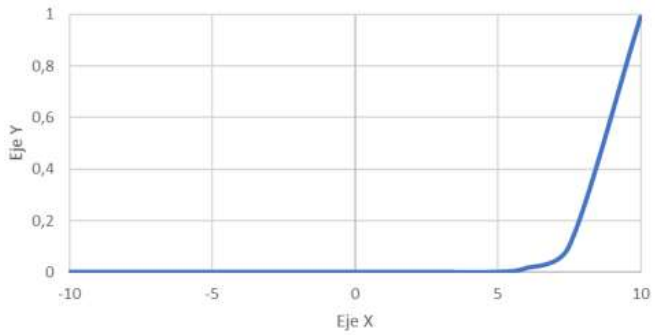


Gráfico 4-1: Función de activación softmax

Fuente: UOC, 2022

$$Zn = \frac{e^{yn}}{\sum_{k=1}^{n} e^{yk}} \quad \textbf{(3.1)}$$

1.18.4. Función de activación ReLU

Este tipo de función es la más utilizada, convierte todos los valores negativos en 0, por lo tanto, todos los valores van de 0 al infinito. Esta función se utiliza en las capas ocultas de las redes neuronales, no en las de salida.

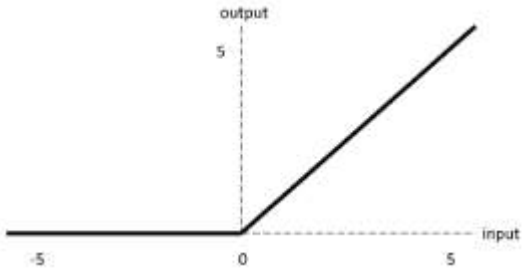


Gráfico 5-1: Función de activación ReLU

Fuente: ResearchGate, 2022

$$ReLU(x) = \max\{0, x\} \quad \textbf{(4.1)}$$

1.19. Flatten

La capa flatten es encargada de convertir en una sola dimensión la imagen de 3 dimensiones.

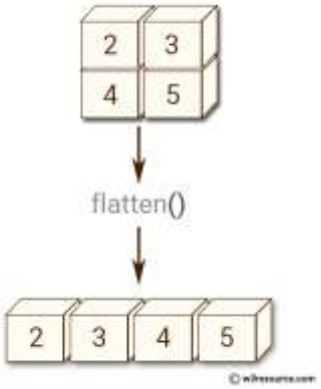


Figura 19-1: Flatten

Fuente: w3resource, 2022

1.20. Shuffle

Shuffle significa barajar, es decir consiste en seleccionar de manera aleatoria la cantidad de

objetos de algún dataset o base de datos.

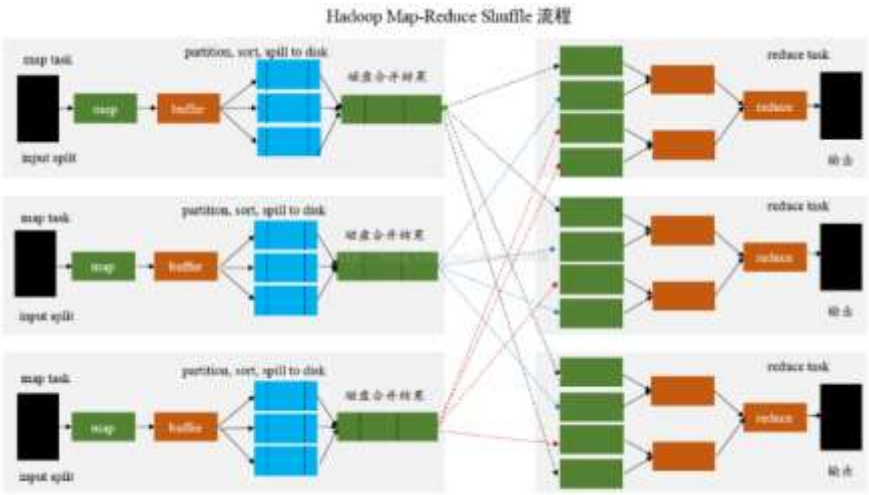


Figura 20-1: Shuffle

Fuente: programador clic, 2022

1.21. Optimización

La optimización es un proceso en el cual consiste asignar variables para que el entrenamiento de la red neuronal sea más eficiente.

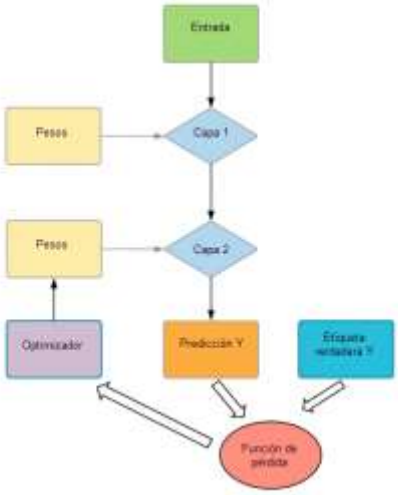


Figura 21-1: Optimizador

Fuente: Guru99, 2022

1.22. Python

Python es un lenguaje de programación de código abierto e interpretado, es decir que cada línea de código que escribamos es leída por un intérprete que las va ejecutando. Por lo tanto, es más rápido al momento de ejecutar programas ya que no tienen que compilarse previamente dichos

programas ahorrándose el tiempo de compilación. Además, su sintaxis es muy simple lo que hace que su curva de aprendizaje no sea tan pronunciada como otros programas. (Pertuz 2022, p.15)

1.22.1. Principales librerías de Python para IA

1.22.1.1. Pandas

Es un paquete de Python el cual dispone estructuras de datos que son versátiles para el usuario, el objetivo es que sea fácil e intuitivo al momento de trabajar con datos relacionales o etiquetados. (riptutorial, 2022, p.2)

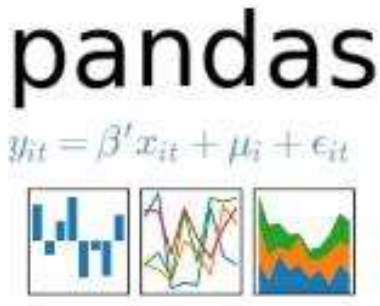


Figura 22-1: Librería Pandas

Fuente: InteractiveChaos, 2022

1.22.1.2. Numpy

Es una biblioteca de Python que sirve para trabajar con arreglos multidimensionales, constituyendo una biblioteca de funciones matemáticas de alto nivel, para operar con vectores y matrices, uno de los principales tipos de datos es el array. (Rivera, 2022, p.35)

Figura 23-1: Librería Numpy

Fuente: aprendeconalf, 2022

1.22.1.3. Matplotlib

Matplotlib es una biblioteca para visualizar diferentes tipos de gráficas o imágenes a partir de datos contenidos en listas o arrays en Python y su extensión matemática Numpy.

Figura 24-1: Librería de Matplotlib

Fuente: InteractiveChaos, 2022

1.22.1.4. Sklearn

Sklearn a veces conocido como scikit learn, es una librería de Python que incluye una variedad de algoritmos supervisados y no supervisados, se utiliza ampliamente para categorización, análisis predictivo y una variedad de otras aplicaciones de aprendizaje automático. Utiliza una interfaz de consistencia de Python para dar un conjunto de herramientas rápidas para el aprendizaje automático modelado y estadístico, como clasificación, regresión, agrupamiento y reducción de dimensionalidad. Numpy, Scypi y Matplotlib son los cimientos de esta librería.(Qadir, 2022, p.29)

Figura 25-1: Librería de Sklearn

Fuente: Wikipedia, 2022

1.22.1.5. Tensorflow

Tensorflow es una aplicación que proporciona a los usuarios una plataforma programable y modular que sirve para construir modelos de machine learning para una gama de aplicaciones, incluyendo voz, sonido, reconocimiento de imágenes, algoritmos de serie de tiempo y detección de video. Tensorflow ofrece conjunto de datos de entrenamiento preetiquetados, capas preentrenadas y modelos de machine learning, así como acceso a una larga lista de bibliotecas de código externo a las que se puede acceder a través de varias aplicaciones como la de Keras.

Figura 26-1: Librería de TensorFlow

Fuente: Wikipedia, 2022

1.22.1.6. Keras

Keras es un framework de alto nivel para el aprendizaje, escrito en Python y capaz de correr en otros frameworks como tensorflow, theano o CNTK. Fue desarrollado con el objetivo de facilitar el proceso de experimentación rápida. (Rivera, 2022, p.34)

Figura 27-1: Librería de Keras

Fuente: médium, 2022

1.23. Anaconda

Es un espacio de código abierto que contiene diferentes entornos, librerías y conceptos que son diseñados para el desarrollo de la Ciencia de datos con Python. (Rivera, 2022, p.34)

Figura 28-1: Anaconda

Fuente: el pythonista, 2022

1.24. Jupyter notebook

Es una aplicación web de código abierto, donde se pueden desarrollar códigos de Python. entre sus usos están: la limpieza y transformación de datos, la simulación numérica, el modelo estadístico, aprendizaje automático, etc.(Rivera, 2022, p.34)

Figura 29-1: Jupyter Notebook

Fuente: Wikipedia, 2022

1.25. Métricas de Evaluación

Para poder evaluar los algoritmos existen diferentes métricas estadísticas las cuales permiten determinar si el algoritmo es aceptable o no es confiable para ser utilizado como un algoritmo de clasificación, a continuación, se detallará cada una de ellas.

1.25.1. Matriz de confusión

Una matriz de confusión es un resumen de las predicciones correctas e incorrectas en forma de recuento sobre un problema de clasificación. En otras palabras, se puede decir que la matriz de confusión muestra el grado de confusión, de manera gráfica, de nuestro algoritmo de clasificación al momento de realizar las predicciones. El tamaño de la matriz de confusión depende del número de clases o etiquetas que se tenga en el clasificador, en las columnas se ubican los valores reales y en las filas se ubican los valores de la predicción.(Alverca, 2022, p.9)

Figura 30-1: Matriz de confusión
Fuente: HealthBIGDATA, 2022

1.25.1.1. Falso positivo

Los falsos positivos o false positives (FP) también denominados error de tipo I, son aquellos valores que realmente son falsos y el algoritmo los clasifico como verdaderos.

1.25.1.2. Falso negativo

Los falso negativos o negative false (NF) también denominados error de tipo II, son aquellos valores que son verdaderos y el algoritmo los clasifico como falsos.

1.25.1.3. Verdadero positivo

Los verdaderos positivos o true positives (TP), son valores que son verdaderos y el algoritmo los clasifico como verdaderos, en este caso sería el número de termogramas que fueron clasificados como “con fallo” y realmente eran “con fallo”.

1.25.1.4. Verdadero negativo

Los verdaderos negativos o negative True Negatives (TN), son valores que son falsos y el algoritmo los clasificó como falsos, en este caso sería el número de termogramas que fueron clasificados como “sin fallo” y realmente eran “sin fallo”.

1.25.2. Precisión

La precisión es una métrica que se utiliza para describir la cercanía de una medición al valor verdadero, si el conjunto de medidas presenta mucha dispersión el método es impreciso, mientras que si esta más concentradas en torno a su valor, es preciso. Actualmente es una de las más utilizadas para estimar el desempeño de los sistemas de aprendizaje en problemas de clasificación. (Alverca, 2022, p.8).

$$precisión = \frac{TP}{TP+FP} \quad \textbf{(5.1)}$$

1.25.3. Exactitud o accuracy

La exactitud se define como el grado de cercanía o proximidad del valor de una medición independiente con respecto al valor real, donde la exactitud tiene una medida mayor cuando más cerca se encuentra del valor real. Además, la precisión indica el grado de proximidad de los resultados de diferentes mediciones entre sí.(Peralta, 2022, p.4)

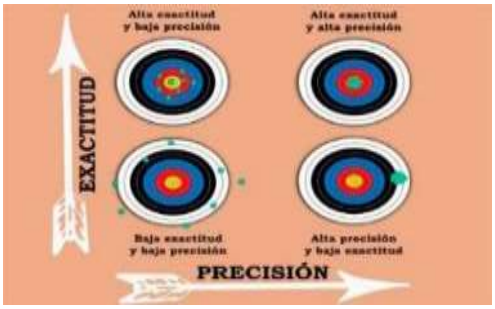


Figura 31-1: Precisión vs Exactitud

Fuente: Logicbus, 2022

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad \textbf{(6.1)}$$

1.25.4. Sensibilidad, exhaustividad o recall

Porcentaje de casos positivos detectados.(Vera, 2022, p.134)

$$recall = \frac{TP}{TP+FN} \quad \textbf{(7.1)}$$

1.25.5. Especificidad o specifity

Porcentaje de casos negativos detectados. (Vera, 2022, p.134)

$$specifity = \frac{TN}{TN+FP} \quad \textbf{(8.1)}$$

1.25.6. F1 score

Esta métrica es una medida armónica de la precisión y la exhaustividad, se puede diferenciar de manera más rápida si el modelo tiene una buena relación entre dichos parámetros, es muy utilizada ya que aglutina la información de ambas métricas, sin embargo, favorecerá más a aquellos modelos con valor similar de precisión y recall.(Vera, 2022, p.134)

$$F1\ score = 2 * (\frac{precisión * recall}{precisión + recall}) \quad \textbf{(9.1)}$$

1.25.7. Curva de características operativas del receptor (ROC)

La curva ROC es una gráfica en la cual se traza la sensibilidad o la tasa de verdaderos positivos en el eje "Y", y el eje "X" la especificidad o la tasa de falsos positivos. (Alverca, 2022, p.8). En la Figura 32-1 se puede observar como se puede clasificar el modelo del algoritmo entrenado según los resultados obtenidos y representados en la curva característica del receptor.

$$1 - especificidad = \frac{FP}{TN+FP} \quad \textbf{(10.1)}$$

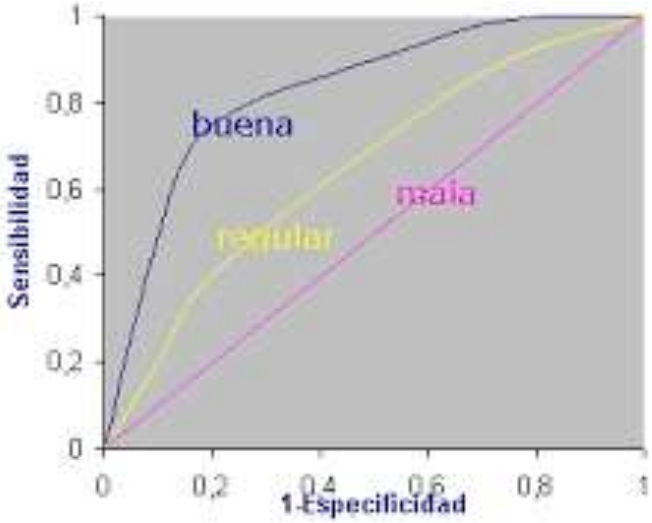


Figura 32-1: Curva de ROC

Fuente: Hospital Universitario Ramón y Cajal, 2022

CAPÍTULO II

2. MARCO METODOLÓGICO

Para la creación del algoritmo de clasificación de termogramas, se utilizó la base de datos de termogramas de un estudio realizado en la carrera de Mantenimiento Industrial de la ESPOCH, dicha base contiene imágenes termográficas con fallo y sin fallo las cuales han sido clasificadas previamente y deben ser descargadas para posteriormente procesarlas e introducirlas a una configuración de redes neuronales aplicando la herramienta de "ImageDataGenerator" para aumentar el número de termogramas en la base de datos y extraer el mayor número de características posibles y así obtener una alta exactitud y precisión, en este caso se aplicará el aprendizaje supervisado y las etiquetas serán "con fallo" y "sin fallo".

2.1. Preprocesamiento de termogramas

Antes de programar el algoritmo de clasificación, es necesario realizar unos pasos, como la descarga de los termogramas, la limpieza de los mismos, la creación de las carpetas de prueba y validación, entre otros, a continuación, se detallará cada uno de estos pasos.

2.1.1. Descarga y clasificación de termogramas

El enlace de los termogramas se encuentra en una base de datos de Excel, realizada en el trabajo de integración curricular de (Guzmán, 2022, p.35) en la carrera de Mantenimiento Industrial, que contiene alrededor de 1.000 termogramas, los cuales ya han sido analizados y clasificados en termogramas con fallo y termogramas sin fallo, por lo tanto, se deben descargar uno por uno y ser ubicados en nuevas carpetas que contengan los termogramas por separado, a estas carpetas se les asignará el nombre de "confallo" para los termogramas que tengan fallas y "sinfallo" para los termogramas que no presenten fallas.

Figura 1-2: Base de datos

Realizado por: Edwin Guzmán

2.1.2. Limpieza de la base de datos de termogramas

Una vez clasificados en las respectivas carpetas es necesario observar de manera minuciosa cada clasificación de los termogramas para determinar cuál es la característica que los diferencia, en este caso se observó que la parte donde se encontraba un aumento de temperatura notorio, es en la chumacera intermedia que sirve de soporte tanto para el eje del acople y para el eje donde va ubicado la carga, esto se puede asumir de acuerdo al cambio de color ya que presenta un color blanco en el termograma con fallo y un color oscuro cuando no está en fallo, como se mencionó en el capítulo I, el punto más caliente en termografía se representa con el color blanco. Para ambos casos el motor siempre presenta un color blanco, es decir, presenta alta temperatura, esto se debe al flujo de corriente que existe en las bobinas y al movimiento rotacional del mismo.

Teniendo en cuenta esta característica se debe eliminar los termogramas que son similares gráficamente, ya que esto puede generar confusión al momento de entrenar la red neuronal. También es recomendable que el formato de las imágenes sea el mismo, en este caso todos los termogramas se encuentran en formato jpg.

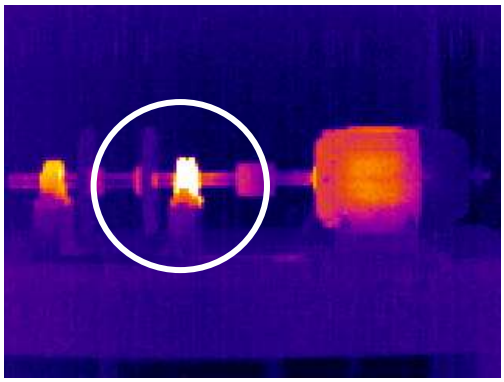


Figura 2-2: Termograma con fallo

Realizado por: Andrés Valencia, 2022

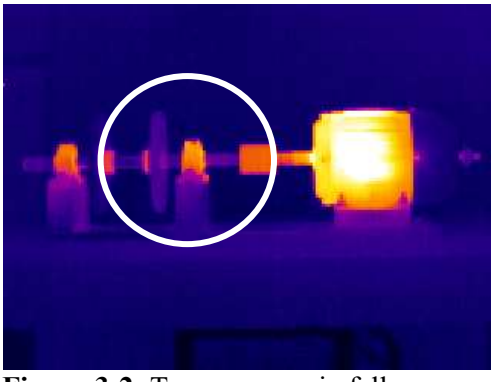


Figura 3-2: Termograma sin fallo

Realizado por: Andrés Valencia, 2022

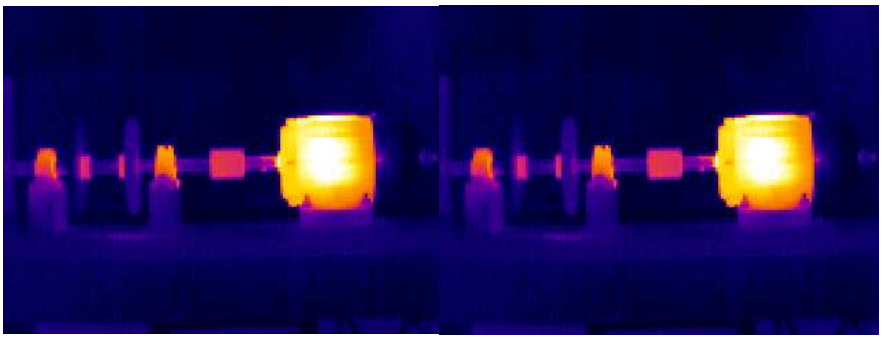


Figura 4-2: Termogramas con fallo y sin fallo similares

Realizado por: Andrés Valencia

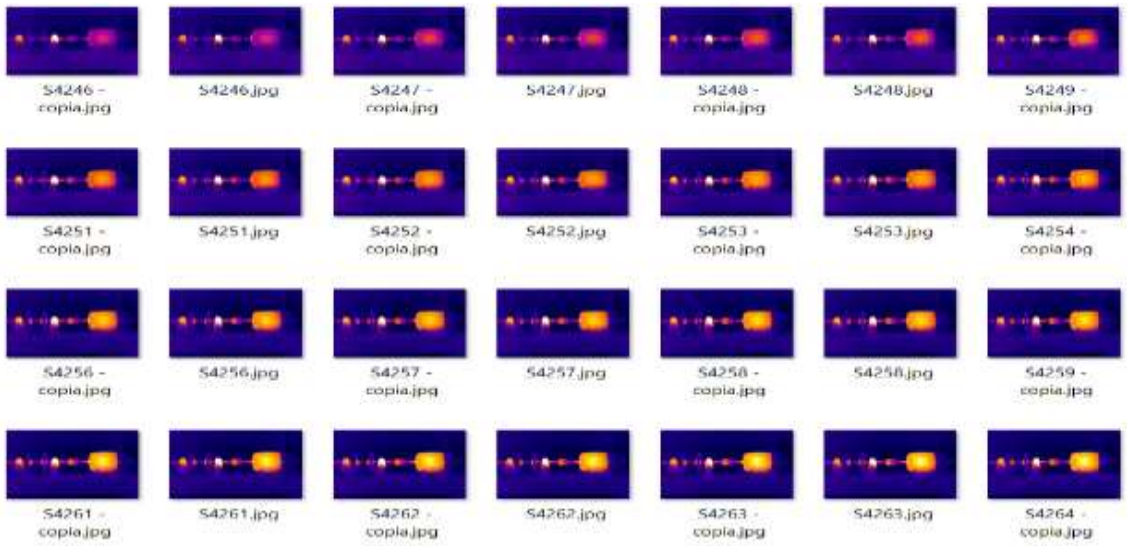


Figura 5-2: Carpeta de termogramas con fallo

Realizado por: Andrés Valencia

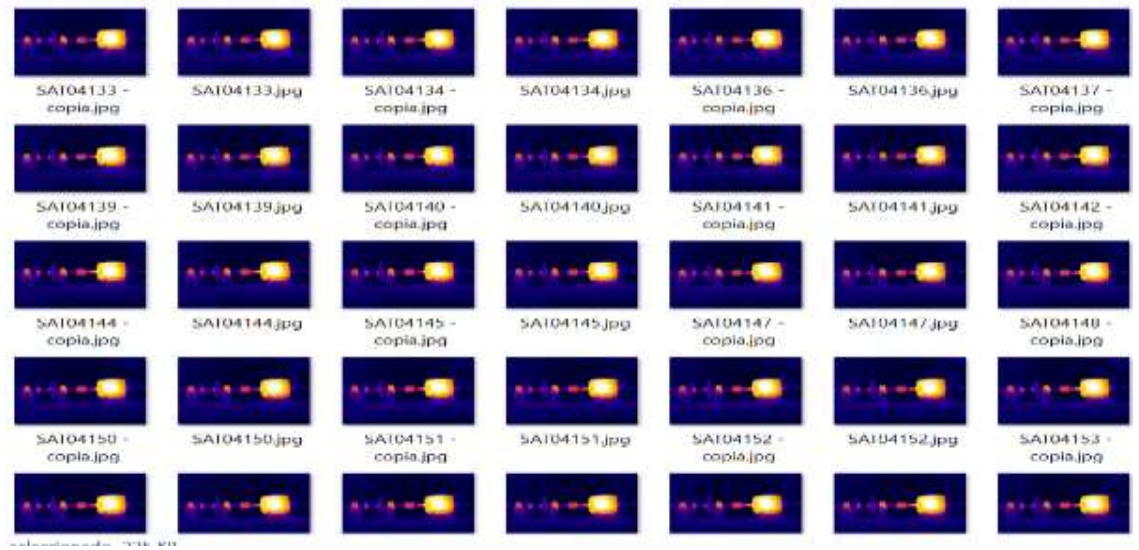


Figura 6-2: Carpeta de Termogramas sin fallo

Realizado por: Andrés Valencia

2.1.3. Creación de carpetas para termogramas de entrenamiento, validación y prueba.

En el set de datos limpio y actualizado se dispone de un total de 732 termogramas, 366 con fallo y 366 sin fallo, en este caso se aplicará la técnica de hold-out explicada en el primer capítulo, que consiste en dividir el set de datos en 60% para entrenamiento, 20% para validación y 20% para prueba. Teniendo en cuenta que los datos de validación están contenidos en el proceso de entrenamiento, se obtiene un total de 80% para entrenamiento y 20% para prueba.

Se designa el nombre de train a la carpeta que tendrá los termogramas para entrenamiento, test para prueba y validation para validación, en cada carpeta existirán subcarpetas con los nombres de “confallo” y “sinfallo” las cuales a su vez contienen los termogramas con fallo y sin fallo respectivamente.

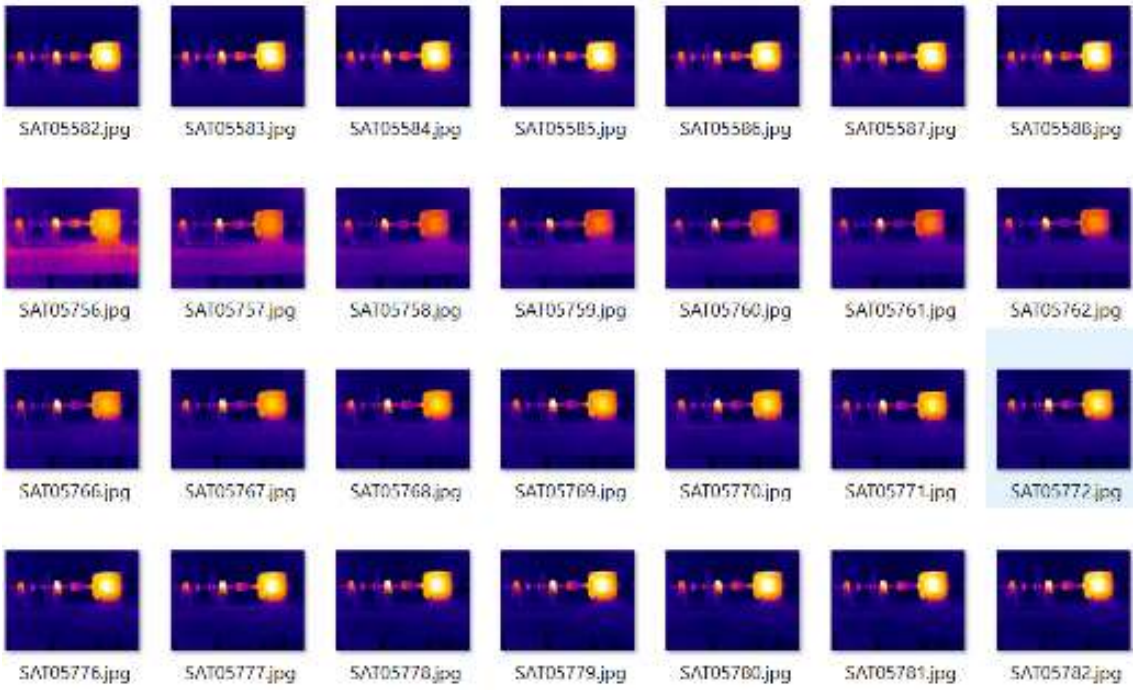


Figura 7-2: Termogramas en las carpetas de Train, test y validation

Realizado por: Andrés Valencia

2.2. Estructura del algoritmo de clasificación

Para poder crear el algoritmo de clasificación se utilizará el lenguaje de programación Python, ejecutado en la plataforma de jupyter notebook la cual a su vez se desarrolla en el entorno de anaconda, y se creará un nuevo cuaderno en el cual se ejecutará el código de programación.

2.2.1. Importar librerías

Para este algoritmo se utilizará las principales librerias como os, matplotlib, cv2, numpy, tensorflow, keras y sklearn, las cuales permitiran importar los termogramas desde el escritorio, visualizarlas, modificarlas y procesarlas, tambien se analizará los resultados de manera gráfica para una mejor comprensión.

```
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, BatchNormalization, Input
from keras.optimizers import Adam
from keras.callbacks import TensorBoard, ModelCheckpoint
from keras.utils import np_utils
import os
import numpy as np
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, f1_score, roc_curve, precision_score, recall_score, accuracy_score, roc_auc_score
from sklearn import metrics
from mlxtend.plotting import plot_confusion_matrix
from keras.models import load_model
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import mlxtend
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
```

Figura 8-2: Importación de librerías

Realizado por: Andrés Valencia

2.2.2. Configuración de parámetros

La red neuronal recibe información del número de pixeles que contenga el termograma, cada pixel ingresará en una neurona, es decir el número de pixeles que se defina en la imagen será el número de neuronas de entrada que tendrá la red neuronal, en este caso, se ingresó un tamaño de 224 x 224 pixeles de ancho(width) y alto(height), salvo para el modelo 2, ya que en este modelo se ingreso un tamaño de 150x150 pixeles; El clasificador es de tipo binario, es decir posee 2 etiquetas de clasificación “con fallo” y “sin fallo”, por lo cual el número de neuronas de salida de la red neuronal serán 2.

El número de épocas(epochs) o vueltas que realizará en el entrenamiento es opcional pero se debe tener en cuenta que, el número de parámetros que genere el modelo influirá en el tiempo de entrenamiento de la misma, de igual manera mientras más epocas se aplique mayor será el tiempo de entrenamiento de la red neuronal, por otro lado también se debe visualizar los datos que se generen en el entrenamiento ya que puede ser que por más epocas que se asigne, llegará a un punto en el cual la red ya no aprenda más, en este caso se asignará un total de 30 epocas para los 3 primeros modelos, y para el modelo 4 se asignará 100 épocas.

Para que el entrenamiento se realice rápidamente se agrupa los termogramas en grupos, conjuntos o lotes de datos, en este algoritmo se agrupará en lotes(batch) de 32 termogramas cada uno.

```
ancho = 224
alto = 224
clases = 2
epocas = 100
conjunto = 32
epochs = 30
width_shape = 150
height_shape = 150
```

Figura 9-2: Configuración de parámetros

Realizado por: Andrés Valencia

2.2.3. Ruta de la base de datos

Debido a que la base de datos se encuentra en el escritorio de la PC, se debe definir variables la cuales tengan acceso a las carpetas donde se encuentran los termogramas, para ello se copiará la ruta de acceso y se la definirá en las variables correspondientes, siendo así, la de "train_data_dir" para la carpeta que tenga los termogramas de entrenamiento y la variable "validation_data_dir" para la carpeta que tenga los termogramas de validación.

```
train_data_dir = 'C:/Users/ASUS VivoBook/Desktop/ULTIMO/train'
validation_data_dir = 'C:/Users/ASUS VivoBook/Desktop/ULTIMO/validation'
```

Figura 10-2: Ruta de acceso

Realizado por: Andrés Valencia

Con la ayuda de las herramientas de matplotlib se puede visualizar algunas imágenes del set de datos.

```
plt.figure(figsize=(20,20)) #PLT.FIGURE SIRVE PARA PODER MODIFICAR EL TAMAÑO DE LA IMAGEN

carpeta ='/content/confallo'
imagenes = os.listdir(carpeta)

for i, nombreimg in enumerate(imagenes[:20]):
  plt.subplot(5,5, i+1)
  imagen = mpimg.imread(carpeta + '/' + nombreimg)
  plt.imshow(imagen)
```

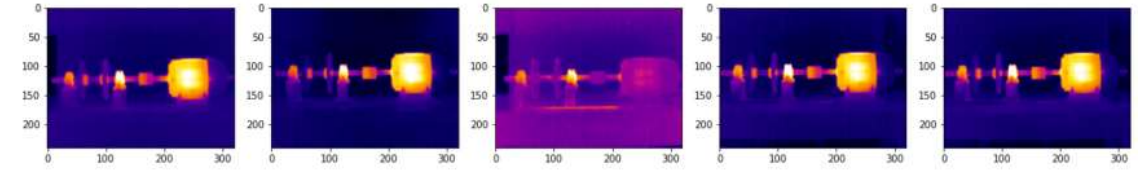


Figura 11-2: Termogramas con fallo en RGB

Realizado por: Andrés Valencia

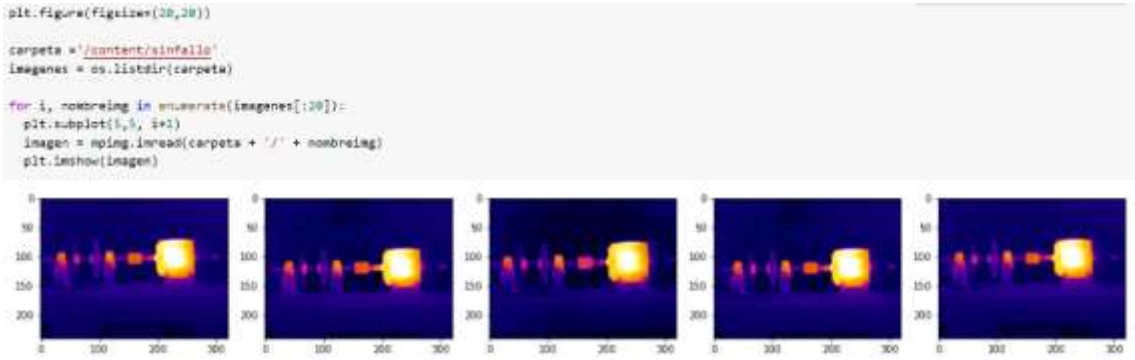


Figura 12-2: Termogramas sin fallo en RGB

Realizado por: Andrés Valencia

Para poder diferenciar de mejor manera el punto caliente en la chumacera se puede cambiar las imágenes a escala de grises en la cual el punto indicado es más notorio.

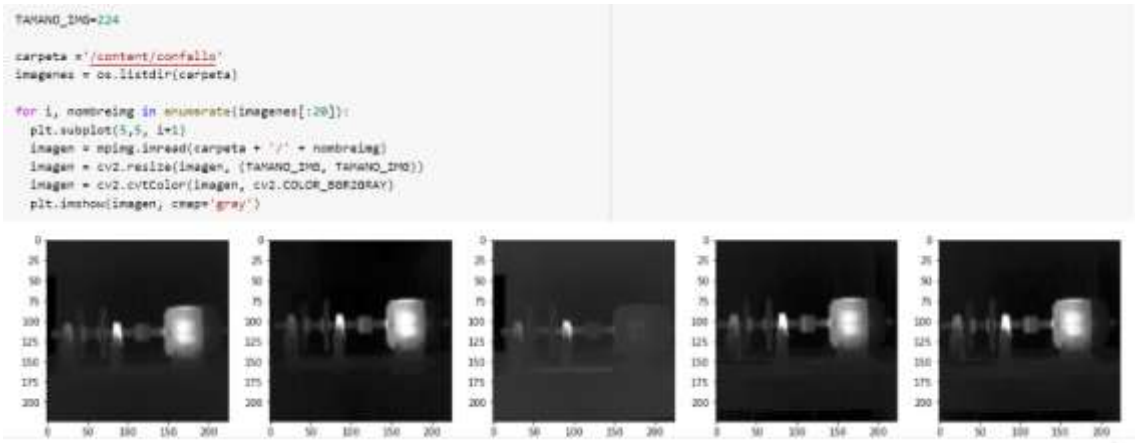


Figura 13-2: Termogramas con fallo en escala de grises

Realizado por: Andrés Valencia

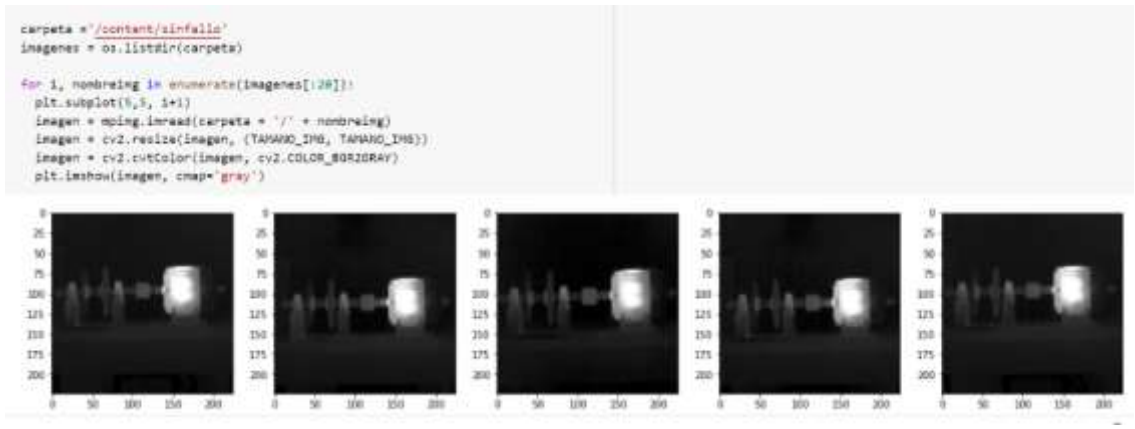


Figura 14-2: Termogramas sin fallo en escala de grises

Realizado por: Andrés Valencia

2.2.4. Aumento de datos

Debido a que la base de datos de la que se dispone es relativamente pequeña, se utiliza una técnica de aumento de datos utilizando la herramienta de "Image_Data_Generator" que proporciona la librería de Keras, esta herramienta se encarga de realizar cambios en las imágenes, como rotarlas, moverlas a la izquierda, derecha, darlas la vuelta e incluso alejarlas o acercalas mediante zoom. Una vez aplicada está técnica se obtiene una variedad de termogramas, la idea es aumentar la cantidad de imágenes en la base de datos con dichas modificaciones y que la red neuronal obtenga la mayor cantidad de características sin importar la posición en la que se encuentre el termograma.

```
train_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input)

valid_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    preprocessing_function=preprocess_input)
```

Figura 15-2: Aumento de datos con Image_Data_Generator

Realizado por: Andrés Valencia

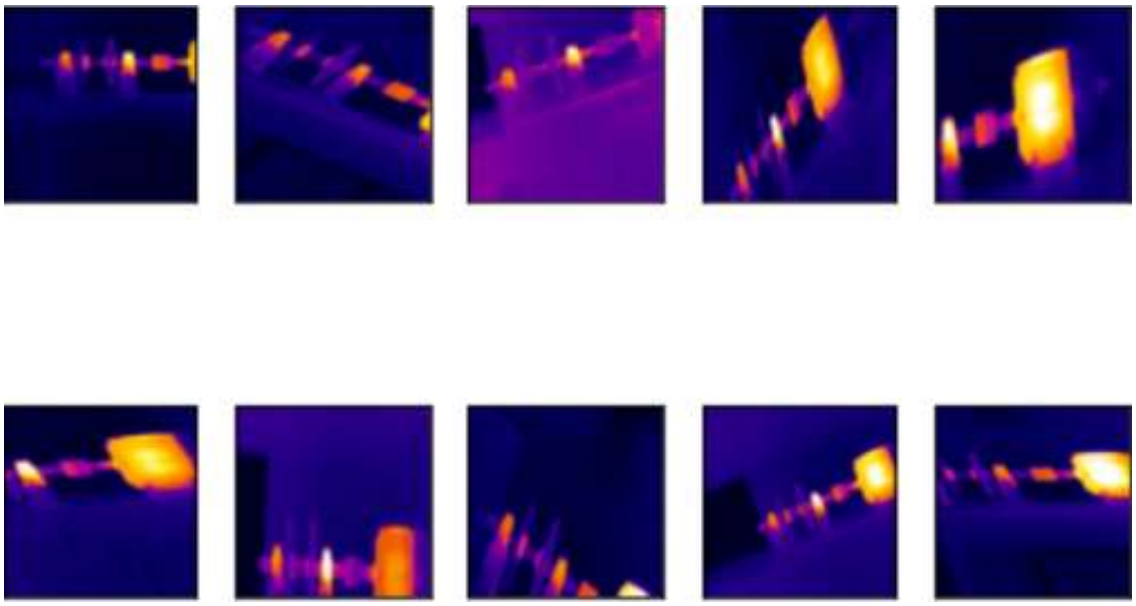


Figura 16-2: Termogramas modificados

Realizado por: Andrés Valencia

2.2.5. Generadores de entrenamiento y validación con shuffle

El nuevo set de datos en el cual consta los termogramas originales y los que se generaron con las modificaciones que se realizó gracias a la ayuda de la técnica de aumento de datos y con la herramienta de ImageDataGenerator, se agregarán en nuevas variables las cuales posteriormente se ingresarán a las neuronas de entrada de la red neuronal respectiva.

Con la ayuda de "flow_from_directory" se especifica utilizar como base el directorio especificado y aplicamos la técnica de shuffle que consiste en mezclar los datos y tomarlos al azar al momento del entrenamiento en cada época que se realice, en dicho directorio viene incluido por defecto esta herramienta, pero también se lo puede activar con el código: "shuffle=True" o a su vez ser desactivada con el código "shuffle=False", en cada época se realiza 18 mezclas de los datos aleatoriamente al ser activada esta herramienta debido al tamaño de lote que está definido inicialmente en las variables. Se recomienda activar dicha herramienta para de cierta manera evitar que la red neuronal no genere un sobreajuste, cabe recalcar que todo este proceso se realiza internamente en el programa de manera automática al momento del entrenamiento de la red neuronal que se ha creado.

```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    #shuffle=True
    class_mode='categorical')

validation_generator = valid_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(width_shape, height_shape),
    batch_size=batch_size,
    #shuffle=True
    class_mode='categorical')
```

Figura 17-2: Entrenamiento y validación con Shuffle

Realizado por: Andrés Valencia

2.2.6. Diseño de la red neuronal

Para obtener un modelo de red neuronal que sea eficiente se diseñó cuatro modelos en base al tema de investigación, para comprobar cuál de ellos se desenvuelve de mejor manera en el entrenamiento, y proporciona mejores resultados, la red que se seleccione será utilizada para predecir varios termogramas que no han sido mostrados al algoritmo y de esta manera comprobar si los clasifica de manera adecuada.

2.2.6.1. Modelo 1

El primer modelo de red neuronal consta de una configuración de 224 x 224 pixeles en la entrada de la red neuronal, es decir un total de 50.176 neuronas, con canal de colores RGB, seguido de ello en las capas ocultas se añadió un arreglo de redes neuronales convolucionales con la siguiente secuencia: 32, 64, 128, 128, 64 y 32, el kernel de todas las redes convolucionales es de 3x3, seguido de una función de maxpooling de 2x2, en cada convolución se escogió el tipo de activación relu.

Para evitar el sobreajuste se añadió un dropout del 80% en la última capa de redes convolucionales, seguido de una operación Flatten antes de ser entregada a la capa densa de 224 neuronas, la cual será la encargada de entregar la información a la salida, que contiene el número de clases o etiquetas, estás neuronas de salida se activaran con la función softmax. Para la compilación del modelo se utilizará el optimizador "adam", la métrica de "accuracy" y para pérdida "categorical_crossentropy".

```
nb_train_samples = 586
nb_validation_samples = 146

model = Sequential()

inputShape = (height_shape, width_shape, 3)
model.add(Conv2D(32,(3,3), activation='relu', input_shape=inputShape))
model.add(MaxPool2D(2,2))
model.add(Conv2D(64,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Conv2D(128,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Conv2D(128,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Conv2D(64,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Conv2D(32,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))

model.add(Dropout(0.8)),
model.add(Flatten()),
model.add(Dense(224,activation='relu')),
model.add(Dense(num_classes,activation='softmax', name='output'))

model.summary()
```

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Figura 18-2: Modelo 1 de red neuronal

Realizado por: Andrés Valencia

Tabla 1-2: Arquitectura de la red neuronal modelo 1

Layer (type)	Output Shape	Param #
conv2d (Conv2d)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2d)	(None, 109, 109, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2d)	(None, 52, 52, 128)	73856
max_pooling2d_2(MaxPooling2D)	(None, 26, 26, 128)	0
conv2d_3 (Conv2d)	(None, 24, 24, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 128)	0
conv2d_4 (Conv2d)	(None, 10, 10, 64)	73792
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_5 (Conv2d)	(None, 3, 3, 32)	18464
max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 32)	0
dropout (Dropout)	(None, 1, 1, 32)	0
flatten (Flatten)	(None, 32)	0
dense (Dense)	(None, 224)	7392
output (Dense)	(None, 2)	450
Total params: 340,930		
Trainable params: 340,930		
Non-trainable params: 0		

Realizado por: Andrés Valencia

2.2.6.2. Modelo 2

El segundo modelo de red neuronal consta de una configuración de 150 x 150 pixeles en la entrada de la red neuronal, es decir un total de 22.500 neuronas, con canal de colores RGB, seguido de ello en las capas ocultas se añadió un arreglo de redes neuronales convolucionales con la siguiente secuencia: 32, 32, cada una con un kernel de 3x3 y un maxpooling de 2x2, en la siguiente secuencia se dispone de 3 filas de 64 neuronas cada una con un kernel de 3x3 y un maxpooling de 2x2.

Finalmente se aplica un Flatten antes de ser entregada a la capa densa de 64 neuronas, seguida de otra capa densa de 32 neuronas, está será encargada de entregar la información a la salida que contiene el número de clases o etiquetas, activada con la función de softmax; Para la compilación del modelo se utilizará el optimizador “adadelta”, la métrica de “accuracy” y para pérdida “categorical_crossentropy”.

```
model = Sequential()

inputShape = (height_shape, width_shape, 3)
model.add(Conv2D(32,(3,3), input_shape=inputShape))
model.add(Conv2D(32,(3,3)))
model.add(MaxPool2D())

model.add(Conv2D(64,(3,3)))
model.add(Conv2D(64,(3,3)))
model.add(Conv2D(64,(3,3)))
model.add(MaxPool2D())

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(num_classes,activation='softmax', name='output'))

model.summary()
```

```
model.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=['accuracy'])
```

Figura 19-2: Modelo 2 de red neuronal

Realizado por: Andrés Valencia

Tabla 2-2: Arquitectura de la red neuronal Modelo 2

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2d)	(None, 148, 148, 32)	896
conv2d_11 (Conv2d)	(None, 146, 146, 32)	9248
max_pooling2d_4 (MaxPooling2D)	(None, 73, 73, 32)	0
conv2d_12 (Conv2d)	(None, 71, 71, 64)	18496
conv2d_13 (Conv2d)	(None, 69, 69, 64)	36928
conv2d_14 (Conv2d)	(None, 67, 67, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(None, 33, 33, 64)	0
flatten_2 (Flatten)	(None, 69696)	0
dense_4 (Dense)	(None, 64)	4460608
dense_5 (Dense)	(None, 32)	2080
output (Dense)	(None, 2)	66
Total params: 4´565,250		
Trainable params: 4´565,250		
Non-trainable params: 0		

Realizado por: Andrés Valencia

2.2.6.3. Modelo 3

Para este modelo se aplicó la técnica de Transfer Learning utilizando la red preentrenada VGG16, a la cual se agrega en la capa de entrada una configuración de 224x224 pixeles, obteniendo un total de 50.176 neuronas de entrada, se añade la arquitectura de VGG16 y se elimina la última capa para añadir el número de etiquetas para el algoritmo de clasificación que se necesita, en este caso serían 2 clases o etiquetas con activación softmax.

Para la compilación del modelo se utilizará el optimizador “adam”, la métrica de “accuracy” y

para pérdida “categorical_crossentropy”.

```
image_input = Input(shape=(width_shape, height_shape, 3))

model = VGG16(input_tensor=image_input, include_top=True,weights='imagenet')

last_layer = model.get_layer('fc2').output
out = Dense(num_classes, activation='softmax', name='output')(last_layer)
custom_vgg_model = Model(image_input, out)

for layer in custom_vgg_model.layers[:-1]:
    layer.trainable = False


custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

custom_vgg_model.summary()
```

Figura 20-2: Modelo 3 de red neuronal con VGG16

Realizado por: Andrés Valencia

Tabla 3-2: Arquitectura de la red neuronal Modelo 3 con VGG16

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	(None, 222, 222, 32)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 256)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 512)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
output (Dense)	(None, 2)	8194
Total params: 134´268,738		
Trainable params: 8,194		
Non-trainable params: 134´260,544		

Realizado por: Andrés Valencia

2.2.6.4. Modelo 4

En el modelo 4 se utilizó la misma técnica de transferencia de aprendizaje que se aplicó en el modelo 3, la diferencia es que en este caso el número de épocas fueron 100, de igual manera se utilizó la red preentrenada VGG16, añadiendo las etiquetas necesarias para este tema de estudio; este modelo se generó debido a que al momento de realizar pruebas agregando más épocas se pudo observar que los resultados eran mejores, sin embargo a partir de la época 100, los resultados ya no mejoraban sino que eran menores, por ello, se optó por general el modelo con 100 épocas.

2.2.7. Entrenamiento

Para entrenar el modelo de red neuronal que se ha creado se utilizará la función de "fit_generator" anteponiendo el nombre de la variable en la que se creó el modelo, dentro de esta función definiremos parámetros que inicialmente fueron agregados, dentro de ellos se encuentra el número de épocas(epochs), "train_generator", "validation_generator" y "batch_size, conjunto o lote".

2.2.7.1. Entrenamiento modelo 1

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

model_history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    steps_per_epoch=nb_train_samples//conjunto,
    validation_steps=nb_validation_samples//conjunto)
```

```
18/18 [==============================] - 23s 1s/step - loss: 1.3777 - accuracy: 0.5036 - val_loss: 0.6923 - val_accuracy: 0.5
000
Epoch 2/30

18/18 [==============================] - 22s 1s/step - loss: 0.6954 - accuracy: 0.5018 - val_loss: 0.6930 - val_accuracy: 0.5
078
Epoch 3/30

18/18 [==============================] - 24s 1s/step - loss: 0.6932 - accuracy: 0.5054 - val_loss: 0.6937 - val_accuracy: 0.4
844
Epoch 4/30
```

…

```
18/18 [==============================] - 30s 2s/step - loss: 0.6931 - accuracy: 0.5126 - val_loss: 0.6932 - val_accuracy: 0.4
922
Epoch 26/30

18/18 [==============================] - 29s 2s/step - loss: 0.6933 - accuracy: 0.5000 - val_loss: 0.6932 - val_accuracy: 0.5
000
Epoch 27/30

18/18 [==============================] - 26s 1s/step - loss: 0.6934 - accuracy: 0.4982 - val_loss: 0.6931 - val_accuracy: 0.5
078
Epoch 28/30

18/18 [==============================] - 26s 1s/step - loss: 0.6932 - accuracy: 0.4711 - val_loss: 0.6931 - val_accuracy: 0.5
156
Epoch 29/30

18/18 [==============================] - 30s 2s/step - loss: 0.6932 - accuracy: 0.4783 - val_loss: 0.6931 - val_accuracy: 0.4
922
```

Figura 21-2: Entrenamiento modelo 1

Realizado por: Andrés Valencia

A medida que el modelo 1 entrena, se puede visualizar en la métrica de exactitud o acuraccy que el modelo solo obtiene un valor de 0.5, es decir el 50% lo cual no es un buen indicio para poder utilizar este modelo de red neuronal, sería como lanzar una moneda al aire y esperar a la suerte el valor que predice.

2.2.7.2. Entrenamiento modelo 2

```
model2.compile(loss='categorical_crossentropy',optimizer='adadelta',metrics=['accuracy'])

model_history = model2.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    steps_per_epoch=nb_train_samples//conjunto,
    validation_steps=nb_validation_samples//conjunto)
```

```
Epoch 1/30
18/18 [==============================] - 21s 1s/step - loss: 6.1461 - accuracy: 0.5415 - val_loss: 5.2261 - val_accuracy: 0.539
1
Epoch 2/30
18/18 [==============================] - 22s 1s/step - loss: 4.3521 - accuracy: 0.6047 - val_loss: 4.9245 - val_accuracy: 0.539
1
Epoch 3/30
18/18 [==============================] - 27s 1s/step - loss: 3.5631 - accuracy: 0.6047 - val_loss: 4.7809 - val_accuracy: 0.531
2
Epoch 4/30
18/18 [==============================] - 28s 2s/step - loss: 2.8546 - accuracy: 0.6516 - val_loss: 3.2592 - val_accuracy: 0.554
7
Epoch 5/30
18/18 [==============================] - 23s 1s/step - loss: 2.5768 - accuracy: 0.6625 - val_loss: 2.5447 - val_accuracy: 0.648
4
Epoch 6/30
18/18 [==============================] - 23s 1s/step - loss: 2.9210 - accuracy: 0.6264 - val_loss: 3.3746 - val_accuracy: 0.562
5
Epoch 7/30
18/18 [==============================] - 24s 1s/step - loss: 1.9032 - accuracy: 0.6931 - val_loss: 2.0026 - val_accuracy: 0.671
9
Epoch 8/30
18/18 [==============================] - 24s 1s/step - loss: 2.1676 - accuracy: 0.6625 - val_loss: 2.7779 - val_accuracy: 0.632
8
Epoch 9/30
18/18 [==============================] - 24s 1s/step - loss: 1.6280 - accuracy: 0.7292 - val_loss: 2.8126 - val_accuracy: 0.617
2
```

...

```
18/18 [==============================] - 26s 1s/step - loss: 0.9807 - accuracy: 0.7527 - val_loss: 1.8249 - val_accuracy: 0.656
2
Epoch 23/30
18/18 [==============================] - 26s 1s/step - loss: 0.7288 - accuracy: 0.8195 - val_loss: 1.4119 - val_accuracy: 0.765
6
Epoch 24/30
18/18 [==============================] - 26s 1s/step - loss: 0.8922 - accuracy: 0.7635 - val_loss: 1.3160 - val_accuracy: 0.757
8
Epoch 25/30
18/18 [==============================] - 26s 1s/step - loss: 0.7769 - accuracy: 0.7996 - val_loss: 1.7489 - val_accuracy: 0.687
5
Epoch 26/30
18/18 [==============================] - 27s 1s/step - loss: 1.0124 - accuracy: 0.7617 - val_loss: 1.5234 - val_accuracy: 0.734
4
Epoch 27/30
18/18 [==============================] - 28s 2s/step - loss: 0.6038 - accuracy: 0.8368 - val_loss: 1.4346 - val_accuracy: 0.710
9
Epoch 28/30
18/18 [==============================] - 27s 1s/step - loss: 0.8925 - accuracy: 0.7726 - val_loss: 1.2893 - val_accuracy: 0.750
0
Epoch 29/30
18/18 [==============================] - 26s 1s/step - loss: 0.6981 - accuracy: 0.8069 - val_loss: 0.9929 - val_accuracy: 0.757
8
Epoch 30/30
18/18 [==============================] - 27s 1s/step - loss: 0.9268 - accuracy: 0.7527 - val_loss: 1.1397 - val_accuracy: 0.734
4
```

Figura 22-2: Entrenamiento modelo 2

Realizado por: Andrés Valencia

Para el modelo 2, conforme va pasando las épocas se puede visualizar que el valor de exactitud o acuraccy, va mejorando y el valor de perdida va disminuyendo, sin embargo, se mantiene alrededor de 0.75, es decir el 75% de exactitud, este valor es mejor que el modelo anterior pero lo ideal es sobrepasar el 80 % y aproximarse al 100% de exactitud y precisión, sin generar un overfitting o sobreajuste.

2.2.7.3. Entrenamiento modelo 3

```
model_history = custom_vgg_model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    steps_per_epoch=nb_train_samples//batch_size,
    validation_steps=nb_validation_samples//batch_size)
```

```
Epoch 1/30
18/18 [==============================] - 81s 5s/step - loss: 0.2651 - accuracy: 0.8767 - val_loss: 0.1467 - val_accuracy: 0.9
609
Epoch 2/30
18/18 [==============================] - 84s 5s/step - loss: 0.1016 - accuracy: 0.9603 - val_loss: 0.3182 - val_accuracy: 0.9
062
Epoch 3/30
18/18 [==============================] - 86s 5s/step - loss: 0.0906 - accuracy: 0.9675 - val_loss: 0.3553 - val_accuracy: 0.9
141
Epoch 4/30
18/18 [==============================] - 86s 5s/step - loss: 0.0532 - accuracy: 0.9783 - val_loss: 0.0782 - val_accuracy: 0.9
766
Epoch 5/30
18/18 [==============================] - 86s 5s/step - loss: 0.0455 - accuracy: 0.9801 - val_loss: 0.1569 - val_accuracy: 0.9
453
Epoch 6/30
18/18 [==============================] - 87s 5s/step - loss: 0.0535 - accuracy: 0.9747 - val_loss: 0.2378 - val_accuracy: 0.9
453
Epoch 7/30
```

…

```
Epoch 25/30
18/18 [==============================] - 82s 5s/step - loss: 0.0218 - accuracy: 0.9946 - val_loss: 0.2180 - val_accuracy: 0.9
453
Epoch 26/30
18/18 [==============================] - 81s 5s/step - loss: 0.0184 - accuracy: 0.9910 - val_loss: 0.1871 - val_accuracy: 0.9
297
Epoch 27/30
18/18 [==============================] - 80s 5s/step - loss: 0.0125 - accuracy: 0.9946 - val_loss: 0.2002 - val_accuracy: 0.9
609
Epoch 28/30
18/18 [==============================] - 82s 5s/step - loss: 0.0213 - accuracy: 0.9928 - val_loss: 0.1385 - val_accuracy: 0.9
531
Epoch 29/30
18/18 [==============================] - 82s 5s/step - loss: 0.0093 - accuracy: 0.9982 - val_loss: 0.3008 - val_accuracy: 0.9
297
Epoch 30/30
18/18 [==============================] - 83s 5s/step - loss: 0.0095 - accuracy: 0.9964 - val_loss: 0.1610 - val_accuracy: 0.9
531
```

Figura 23-2: Entrenamiento Modelo 3 con VGG16

Realizado por: Andrés Valencia

El modelo 3 es bastante aceptable en exactitud ya que en cada época va mejorando el valor de esta métrica, aproximándose al 100% y manteniéndose en un rango bastante aceptable sin caer en el error de overfitting como se mencionó anteriormente, sin embargo, hay que analizar otras métricas de evaluación para poder seleccionarlo como un algoritmo de clasificación.

2.2.7.4. Entrenamiento modelo 4

```
custom_vgg_model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

model_history = custom_vgg_model.fit_generator(
    train_generator,
    epochs=epocas,
    validation_data=validation_generator,
    steps_per_epoch=nb_train_samples//conjunto,
    validation_steps=nb_validation_samples//conjunto)
```

```
Epoch 1/100
```

…

```
18/18 [==============================] - 100s 6s/step - loss: 0.0125 - accuracy: 0.9964 - val_loss: 0.6834 - val_accuracy: 0.
8672
Epoch 5/100
```

…

```
18/18 [==============================] - 103s 6s/step - loss: 0.0121 - accuracy: 0.9964 - val_loss: 0.5277 - val_accuracy: 0.
9219
Epoch 49/100
```

…

```
18/18 [==============================] - 118s 7s/step - loss: 0.0041 - accuracy: 0.9964 - val_loss: 0.1648 - val_accuracy: 0.
9688
Epoch 71/100
```

…

```
18/18 [==============================] - 152s 9s/step - loss: 0.0163 - accuracy: 0.9946 - val_loss: 0.3180 - val_accuracy: 0.
9375
Epoch 91/100
```

Figura 24-2: Entrenamiento modelo 4

Realizado por: Andrés Valencia

El modelo 4 presenta valores similares a los del modelo 3 en los valores de pérdida y exactitud a pesar de que el número de épocas es mucho mayor, sin alcanzar el 100% en exactitud, por lo que se puede manifestar tentativamente que no se está produciendo un error de overfitting sin embargo las métricas de evaluación son las que determinarán que modelo seleccionar.

2.2.8. Guardar modelos

Una vez finalizado el entrenamiento de cada modelo, se procede a guardarlos con el comando "save. ("nombre con el que se desea guardar el modelo")", anteponiendo el nombre del modelo que fue entrenado, para posteriormente utilizarlo en predicciones y poder evaluarlo mediante ciertas métricas las cuales nos indicaran que tan eficiente es el modelo creado.

```
model.save("model.Valencia1")
```

Figura 25-2: Guardar modelo 1

Realizado por: Andrés Valencia

```
model.save("model.ValenciaA1")
```

Figura 26-2: Guardar modelo 2

Realizado por: Andrés Valencia

```
custom_vgg_model.save("modelo.VG16.VALENCIA")
```

Figura 27-2: Guardar modelo 3

Realizado por: Andrés Valencia

```
custom_vgg_model.save("modelo.VG16.VALENCIA100")
```

Figura 28-2: Guardar modelo 4

Realizado por: Andrés Valencia

CAPÍTULO III

3. MARCO DE RESULTADOS Y DISCUSIÓN DE LOS RESULTADOS

En este capítulo se analizará y se comparará las métricas de evaluación de cada modelo, para poder seleccionar el que obtenga mejores resultados y posteriormente proporcionarle un grupo de termogramas a dicho modelo y así determinar que tan bien los clasifica. Las métricas que se utilizarán serán la matriz de confusión, precisión, exactitud, f1-score, sensibilidad y exhaustividad, además se obtendrá el valor AUC, y se graficará la curva de ROC para analizar el comportamiento entre la sensibilidad y la exhaustividad.

En base a estás métricas y analizando los resultados y predicciones de cada modelo se podrá aceptar o rechazar la hipótesis que se planteó al inicio de este tema de estudio, agregando una hipótesis nula y una hipótesis alternativa.

3.1. Gráficos de perdida y exactitud en el entrenamiento

Con ayuda de las herramientas de Matplotlib se puede graficar el comportamiento de cada modelo al momento del entrenamiento, en este caso se visualizará las gráficas de pérdida y exactitud de cada modelo en función de cada época que pasa.

3.1.1. Modelo 1

En el Gráfico 6-3 se puede observar el comportamiento del modelo 1 al momento del entrenamiento, en el gráfico izquierdo se encuentra la función de pérdida vs número de épocas y en el gráfico derecho se tiene la exactitud vs número de épocas.

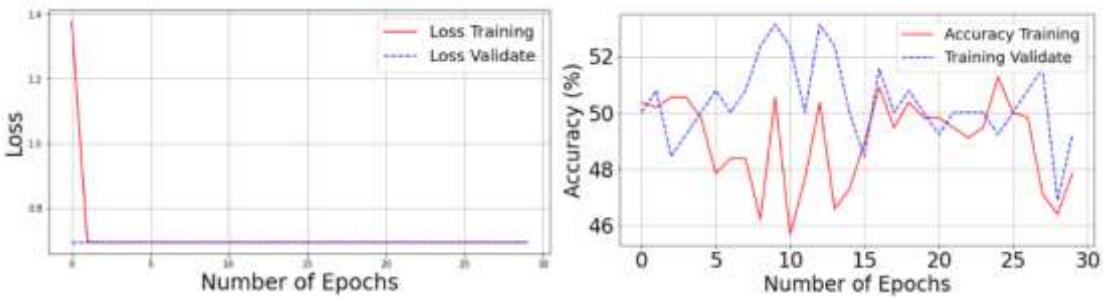


Gráfico 1-3: Gráfico de pérdida y exactitud - Modelo 1

Realizado por: Andrés Valencia

En el gráfico de pérdida, la línea roja representa el valor de pérdida con los datos de entrenamiento y la línea azul representa el valor de pérdida con los datos de validación. Si se analiza el gráfico se observa que la pérdida con los termogramas de entrenamiento es alta en las 2 primeras épocas, y la pérdida con los datos de validación es baja desde la época 1, a partir de la época 2 las 2 funciones son horizontales y paralelas, tendiendo el valor a cero, esto implica un error ya que no es posible que el modelo sepa de manera tan precisa, que tipo de termograma es sin antes haber visto ningún termograma.

En el gráfico de exactitud, la línea roja representa el valor de exactitud que presenta el modelo con los datos de entrenamiento y la línea azul representa el valor de exactitud con los datos de validación. Se puede observar que el gráfico es bastante distorsionado y conforme pasa las épocas el valor no mejora y se mantiene alrededor del 50%.

3.1.2. Modelo 2

En el Gráfico 2-3 se puede observar el comportamiento del modelo 2 al momento del entrenamiento, en el gráfico izquierdo se encuentra la función de pérdida vs número de épocas y en el gráfico derecho se tiene la exactitud vs número de épocas.

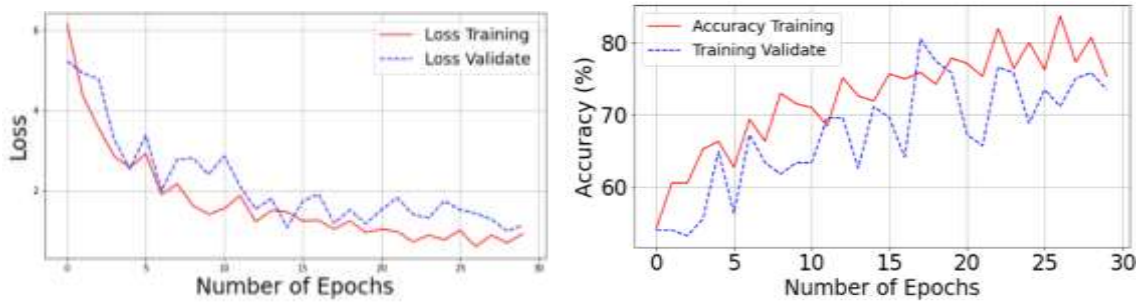


Gráfico 2-3: Gráficos de pérdida y exactitud- modelo 2

Realizado por: Andrés Valencia

En el gráfico de pérdida, la línea roja representa el valor de pérdida con los datos de entrenamiento y la línea azul representa el valor de pérdida con los datos de validación. Al analizar el gráfico se observa que la pérdida y la validación presentan una gráfica suave casi uniforme, que conforme pasa el tiempo o épocas disminuye el valor de pérdida, tratando de ajustarse los valores de pérdida en la validación a los valores de pérdida del entrenamiento, se puede asumir que el algoritmo no se memoriza los datos, sino que extrae más características en cada época que pasa.

En el gráfico de exactitud, la línea roja representa el valor de exactitud que presenta el modelo con los datos de entrenamiento y la línea azul representa el valor de exactitud con los datos de

validación. De igual manera como en la función de pérdida la gráfica es más suave, y conforme pasa las épocas el modelo va mejorando y subiendo su valor de exactitud, posicionándose alrededor del 80% de exactitud.

3.1.3. Modelo 3

En el Gráfico 3-3 se puede observar el comportamiento del modelo 2 al momento del entrenamiento, en el gráfico izquierdo se encuentra la función de pérdida vs número de épocas y en el gráfico derecho se tiene la exactitud vs número de épocas.

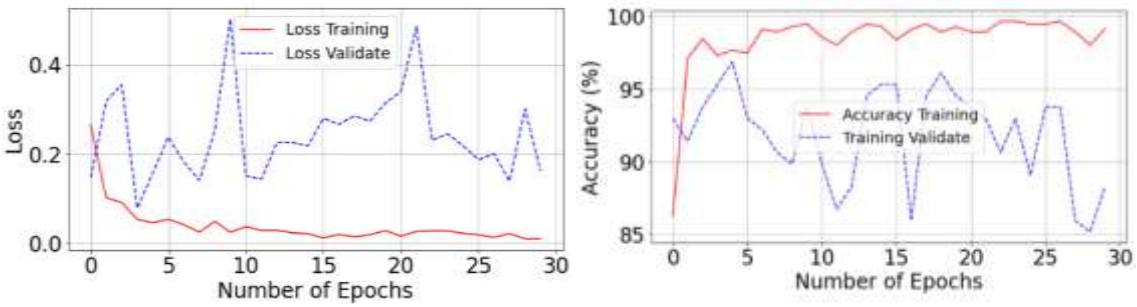


Gráfico 3-3: Gráficos de pérdida y exactitud - modelo 3 con VGG16

Realizado por: Andrés Valencia

En el gráfico de pérdida, la línea roja representa el valor de pérdida con los datos de entrenamiento y la línea azul representa el valor de pérdida con los datos de validación. Se puede observar en que la línea de los valores de pérdida en la validación es distorsionada en comparación de la línea roja, pero si se analiza la escala de valores se puede ver que los valores oscilan entre 0 y 0.4 es decir tiende a cero en la mayoría de épocas.

En el gráfico de exactitud, la línea roja representa el valor de exactitud que presenta el modelo con los datos de entrenamiento y la línea azul representa el valor de exactitud con los datos de validación. Se puede observar que el valor de exactitud se encuentra entre el 90 y 95% en lo datos de validación, mientras que en los datos de entrenamiento están cercanos al 100%, el modelo no se sobreajusta y se mantiene en ese rango, tratando de extraer la mayor cantidad de características posibles, entonces se puede asumir que el modelo en realidad está aprendiendo y no se está memorizando los termogramas.

3.1.4. Modelo 4

En el Gráfico 4-3 se puede observar el comportamiento del modelo 4 al momento del entrenamiento, en el gráfico izquierdo se observa la función de pérdida vs número de épocas y en

el gráfico derecho se tiene la exactitud vs número de épocas.

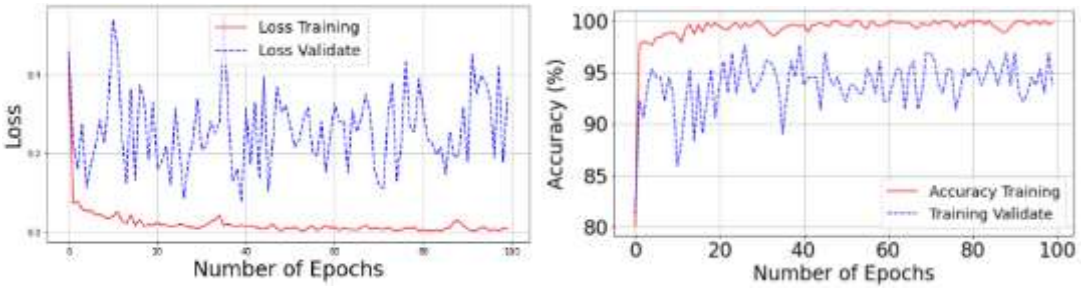


Gráfico 4-3: Gráficos de pérdida y exactitud – modelo 4

Realizado por: Andrés Valencia

Se puede observar que a medida que pasa las épocas las pérdidas son casi constantes sin embargo la exactitud va mejorando poco a poco generando una gráfica cada vez más suave y no muy distorsionada.

3.2. Predicción de cada modelo

Antes de generar la matriz de confusión para cada modelo, se indicará el mismo termograma a cada uno para poder analizar de qué manera lo clasifica.

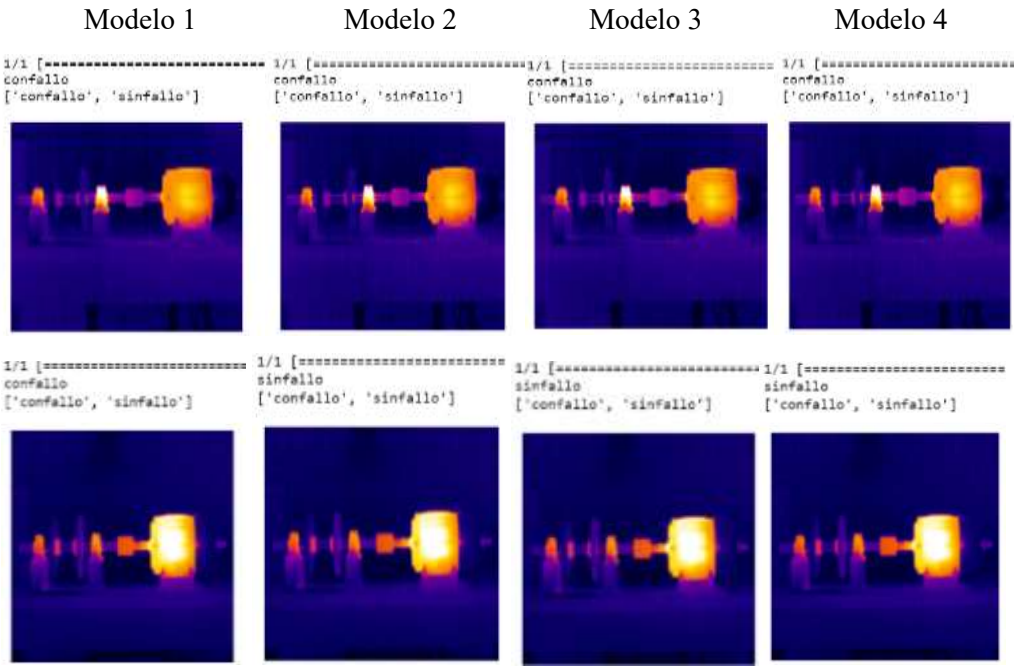


Figura 1-3: Predicción modelo 1,2 y 3

Realizado por: Andrés Valencia

Cómo se puede observar en la Figura 1-3, se proporcionó un termograma con fallo y sin fallo para

cada modelo, el modelo 2, 3 y 4 clasificaron correctamente los termogramas, sin embargo el modelo 1 clasificó de manera correcta el termograma con fallo y de manera incorrecta el termograma sin fallo, esto se debe a que en el entrenamiento dicho modelo alcanzo una exactitud del 50%, por lo que se podría manifestar que el modelo simplemente se memorizó el tipo de fallo y no aprendió a extraer la característica que los diferencia. Por ello se generará la matriz de confusión para cada modelo y en ella se visualizará de mejor manera los errores de clasificación que comete cada uno de ellos.

3.3. Análisis de la Matriz de confusión

A continuación, se indicará la matriz de confusión de cada modelo; Para poder generarla se proporcionará los termogramas de prueba a cada modelo y estos realizarán la predicción respectiva en base a lo aprendido en el entrenamiento, con estos valores se puede observar el número de errores que comete el algoritmo al momento de clasificar termogramas que nunca han sido visualizados.

3.3.1. Matriz de confusión del modelo 1

Tabla 1-3: Matriz de confusión del modelo 1

		Predicted label	
True label		**Con fallo**	**Sin fallo**
	Con fallo	TN = 73	FP = 0
	Sin fallo	FN = 73	TP = 0

Realizado por: Andrés Valencia

En la matriz de confusión del modelo 1 se puede observar que existe una cantidad de 73 termogramas clasificados como verdaderos negativos (TN) y 0 termogramas clasificados como verdaderos positivos(TP), mientras tanto el número de termogramas con falsos negativos(FN) es 73 y el número de termogramas con falsos positivos(FP) es 0, es decir que el algoritmo predijo de manera correcta todos los termogramas que tenían fallo sin cometer ningún error, sin embargo para los termogramas sin fallo no fue así, ya que se equivocó en todos los termogramas y no acertó uno solo, clasificándolos como que si tuvieran fallo. Teniendo en cuenta este análisis es comprensible el bajo valor de exactitud que obtuvo al momento del entrenamiento, se puede decir que el algoritmo simplemente se memorizó los termogramas que tenían fallo.

Se puede comparar los termogramas reales con los de la predicción imprimiendo el resultado de cada uno de ellos, y expresarlos en valores de 0 y 1, en este caso será 0 para los termogramas con fallo y 1 para los termogramas sin fallo.

```
print(y_real)

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

print(y_pred)

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Figura 33-3: Valores reales y predicción del modelo 1

Realizado por: Andrés Valencia

3.3.2. Matriz de confusión del modelo 2

Tabla 2-3: Matriz de confusión del modelo 2

		Predicted label	
True label		**Con fallo**	**Sin fallo**
	Con fallo	TN = 29	FP = 44
	Sin fallo	FN = 1	TP = 72

Realizado por: Andrés Valencia

En la matriz de confusión del modelo 2 se puede observar que existe una cantidad de 29 termogramas clasificados como verdaderos negativos (TN) y 72 termogramas clasificados como verdaderos positivos(TP), mientras tanto el número de termogramas con falsos negativos(FN) es 1 y el número de termogramas con falsos positivos(FP) es 44, es decir que el algoritmo clasificó de manera correcta 29 termogramas con fallo y 72 termogramas sin fallo, sin embargo también tuvo errores ya que clasificó 44 termogramas como que no tuvieran fallo cuando en realidad si tenían fallo, y 1 termograma que no tenía fallo lo clasificó como que en realidad lo tuviera, aquí se puede observar que el error de tipo I es muy notorio por lo tanto se debe tratar de disminuir ese tipo de error.

Se puede comparar los termogramas reales con los de la predicción imprimiendo el resultado de cada uno de ellos, y expresarlos en valores de 0 y 1, en este caso será 0 para los termogramas con

fallo y 1 para los termogramas sin fallo.

```
print(y_real)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
print(y_pred)
```

```
[0 0 0 1 1 1 0 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1
 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1
 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0
 0 1 1 1 1 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 1]
```

Figura 34-3: Valores reales y predicción del modelo 2

Realizado por: Andrés Valencia

3.3.3. Matriz de confusión del modelo 3 con VGG16

Tabla 3-3: Matriz de confusión del modelo 3 con VGG16

		Predicted label	
True label		**Con fallo**	**Sin fallo**
	Con fallo	TN = 72	FP = 1
	Sin fallo	FN = 15	TP = 58

Realizado por: Andrés Valencia

En la matriz de confusión del modelo 3 se puede observar que existe una cantidad de 72 termogramas clasificados como verdaderos negativos (TN) y 58 termogramas clasificados como verdaderos positivos(TP), mientras tanto el número de termogramas con falsos negativos(FN) es 15 y el número de termogramas con falsos positivos(FP) es 1, es decir que el algoritmo clasificó de manera correcta 72 termogramas con fallo y 52 termogramas sin fallo, en comparación con los otros modelos este modelo es bastante aceptable ya que los errores de tipo I y II son bajos en comparación a las predicciones acertadas.

Se puede comparar los termogramas reales con los de la predicción imprimiendo el resultado de cada uno de ellos, y expresarlos en valores de 0 y 1, en este caso será 0 para los termogramas con fallo y 1 para los termogramas sin fallo. En la Figura 65.3 se puede observar que los valores de la predicción se asemejan más a los valores reales, lo que indica que el modelo 3 es el que mejor rendimiento presenta en comparación a los dos primeros modelos.

```
print(y_real)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
print(y_pred)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1]
```

Figura 4-3: Valores reales y predicción del modelo 3 con VGG16

Realizado por: Andrés Valencia

3.3.4. Matriz de confusión del modelo 4

Tabla 4-3: Matriz de confusión del modelo 4

		Predicted label	
True label		**Con fallo**	**Sin fallo**
	Con fallo	TN = 71	FP = 2
	Sin fallo	FN = 1	TP = 72

Realizado por: Andrés Valencia

En la matriz de confusión del modelo 4 se puede observar que existe una cantidad de 71 termogramas clasificados como verdaderos negativos (TN) y 72 termogramas clasificados como verdaderos positivos(TP), mientras tanto el número de termogramas con falsos negativos(FN) es 1 y el número de termogramas con falsos positivos(FP) es 2, es decir que el algoritmo clasificó de manera correcta 71 termogramas con fallo y 72 termogramas sin fallo, pese a que el modelo 3 fue aceptable, el modelo 4 presenta mejores resultados de predicción y menor número de falsos positivos y falsos negativos.

De igual manera se puede comparar los termogramas reales con los de la predicción imprimiendo el resultado de cada uno de ellos, y expresarlos en valores de 0 y 1, en este caso será 0 para los termogramas con fallo y 1 para los termogramas sin fallo. Por lo que se puede visualizar y corroborar los resultados que arrojó la matriz de confusión.

```
print(y_real)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

```
print(y_pred)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

Figura 5-3: Valores reales y predicción del modelo 4

Realizado por: Andrés Valencia

3.4. Precisión

En el modelo 1 la precisión es cero, ya que si se analiza los valores obtenidos en la matriz de confusión se puede observar que genera una razón de 0 dividido entre 0, esto analizando los valores verdaderos positivos y falsos positivos, sin embargo, si se analiza con los valores de verdaderos negativos y falsos negativos obtenemos un valor de 0.5, es decir el 50% de precisión para el modelo 1 con estos parámetros.

En el modelo 2 el valor de la precisión es de 0.62068, es decir el 62,068%, cabe recalcar que cuando se analizó la gráfica de exactitud de este modelo, fue bastante aceptable, sin embargo, se puede observar que la precisión es baja.

En el modelo 3 el valor de la precisión es de 0,9830, es decir el 98,30%, por lo tanto, el modelo tiene buena precisión sin embargo hay que analizar las demás métricas de evaluación, ya que puede ser que tenga alta precisión, pero muy mala exactitud.

En el modelo 4 el valor de la precisión es de 97,29%, es un valor más bajo a comparación del modelo 3, pero bastante aceptable.

3.5. Exactitud

En el modelo 1, el valor de la exactitud es de 0,5, es decir el 50%, por lo tanto, este modelo no tiene buena precisión ni exactitud, ni en el entrenamiento ni en las predicciones, por consiguiente, este modelo será rechazado para utilizarlo como algoritmo de clasificación de termogramas.

En el modelo 2, el valor de la exactitud es de 0,6917, es decir el 69,17%, pese a que en el entrenamiento las gráficas y los valores fueron bastante aceptables, al momento de las predicciones no se obtiene los resultados esperados, siendo aun ineficiente para poder utilizarlo como un algoritmo de clasificación. Este modelo se lo puede mejorar y analizarlo nuevamente para comprobar que tan eficiente resulta.

En el modelo 3, el valor de la exactitud es de 0,89041, es decir el 89,041%, si comparamos con el valor de precisión verificamos que los porcentajes son aceptables tanto en el entrenamiento como en la predicción, esto quiere decir que puede ser seleccionado como un algoritmo de clasificación.

En el modelo 4, el valor de la exactitud es de 0.9794, es decir de 97,94%, en este caso tanto el valor de precisión y exactitud son similares y bastante aceptables.

3.6. Sensibilidad

La sensibilidad del modelo 1 es 0, esto quiere decir que el modelo no acertó a ningún termograma que no presentaba fallo.

La sensibilidad del modelo 2 es 0,9863, esto quiere decir que el modelo acertó en el mayor número de predicciones en termogramas que no tenían fallo y efectivamente los clasificó como tal.

La sensibilidad del modelo 3 es 0,7945, lo que significa que el modelo tuvo falencias al momento de predecir los termogramas sin fallo.

La sensibilidad del modelo 4 es 0,9863, lo que significa que el modelo predijo de manera correcta alrededor del 98% de termogramas sin fallo.

3.7. Especificidad

La especificidad del modelo 1 es 1, esto significa que el modelo acertó de manera correcta a todos los termogramas con fallo.

La especificidad del modelo 2 es 0,3972, lo que significa que el modelo tuvo muchos desaciertos al momento de clasificar los termogramas con fallo.

La especificidad del modelo 3 es 0,9863, lo que significa que el modelo clasificó correctamente

el mayor número de termogramas con fallo.

La especificidad del modelo 4 es 0,9726, lo que significa que el modelo clasificó de manera adecuada la mayoría de termogramas con fallo.

3.8. F1-score

El valor f1-score del modelo 1 es 0, lo cual se puede interpretar como un tipo de modelo sin buena precisión ni recall y no puede ser utilizado para un problema de clasificación de termogramas.

El valor f1-score del modelo 2 es 0,7619, esto implica que el modelo tiene una precisión y recall poco aceptables, sin embargo, se busca obtener un valor superior al 80%.

El valor f1-score del modelo 3 es 0,8787, es decir el 87,87%, esto indica que el modelo aún tiene un desbalance entre los falsos positivos y los falsos negativos.

El valor f1-score del modelo 4 es 0,9795, es decir el 97,95%, esto nos indica que obtuvo la menor cantidad y el mejor balance entre los falsos positivos y falsos negativos.

3.9. Valor AUC

Con el AUC se puede determinar qué tan bueno es el clasificador, siendo así que mientras el valor sea 0,5 se lo puede considerar como un clasificador aleatorio y si es 1 se lo puede considerar como un clasificador perfecto.

El valor AUC del modelo 1 es 0,5, por lo tanto, se puede definir como un clasificador aleatorio es decir que elije los resultados al azar, esto se debe al bajo valor que obtuvo tanto en exactitud como en precisión, por ende, las predicciones resultaron ser erróneas.

El valor AUC del modelo 2 es 0,6917, pese a que las métricas anteriores arrojaron mejores resultados que el modelo 1, se puede observar que el valor es alrededor de 0,5, describiéndolo, así como clasificador de bajo rendimiento.

El valor AUC del modelo 3 es 0,89041, este es un valor cercano a 1, por lo tanto, si se analiza con las demás métricas de evaluación podría considerarse como un clasificador aceptable.

El valor AUC del modelo 4 es 0,97945, este es un valor más próximo a 1, superando incluso al

modelo 3, por lo tanto, se lo determinará como un clasificador casi perfecto.

3.10. Curva de ROC

A continuación, se graficará la curva de Roc, para poder visualizar que tan bueno es el rendimiento de cada modelo.

3.10.1. Modelo 1

Cómo se puede observar en el Gráfico 5-3. la línea que representa los resultados coincide con la diagonal principal de la curva de ROC, si se compara con la Figura 32-1, se puede analizar que este tipo de gráfica corresponde a un tipo de clasificador malo.

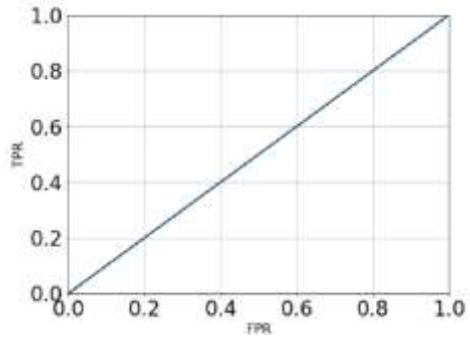


Gráfico 5-3: Curva de ROC - modelo 1

Realizado por: Andrés Valencia

3.10.2. Modelo 2

En el Gráfico 6-3 se puede observar que la línea de resultados se asemeja a la curva de la Figura 32-1, la cual indica que es un clasificador regular, y se puede corroborar con el resultado que se obtuvo del valor AUC.

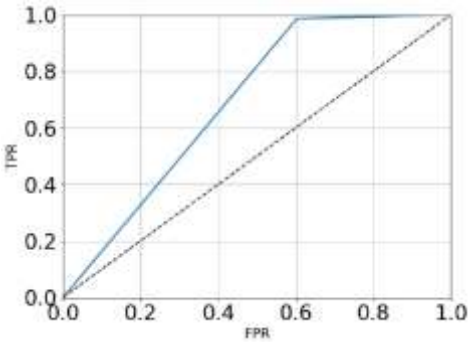


Gráfico 6-3: Curva de ROC - modelo 2

Realizado por: Andrés Valencia

3.10.3. Modelo 3

En el Gráfico 7-3 se observa la curva de ROC del modelo 3, si se compara dicha gráfica con la Figura 32-1, la cual nos indica que tan bueno es el clasificador, se puede resumir que es un clasificador bueno, de igual manera coincide con la respuesta del valor AUC, calculado anteriormente que asimismo lo definió como un clasificador casi perfecto.

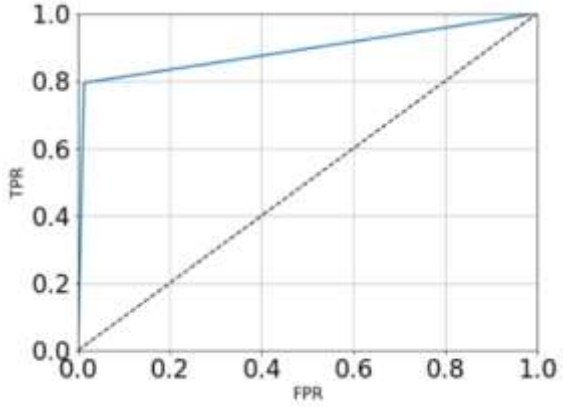


Gráfico 7-3: Curva de ROC modelo 3 con VGG16

Realizado por: Andrés Valencia

3.10.4. Modelo 4

En el Gráfico 8-3 se puede observar que la curva de ROC del modelo 4 es casi perfecta si se la compara con la Figura 32-1, de igual manera los valores del área bajo la curva para dicho modelo son cercanos a 1, lo que justifica la forma de esta gráfica.

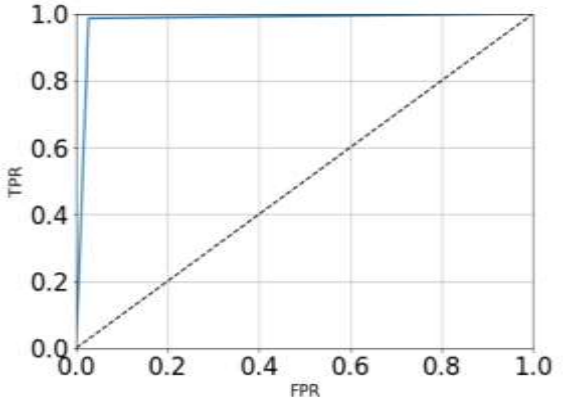


Gráfico 8-3: Curva de ROC modelo 4

Realizado por: Andrés Valencia

3.11. Comparación de las métricas de evaluación

Una vez calculadas todas las métricas de evaluación para cada modelo, se realizará una tabla donde se pueda comparar todos los valores que se obtuvo de cada uno, para así seleccionar el modelo que mejor resultados obtuvo.

Tabla 5-3: Comparación de métricas de evaluación

	Precisión	**Exactitud**	**Sensibilidad**	**Especificidad**	**F1-score**	**Valor AUC**
Modelo 1	0	0,5	0	1	0	0,5
Modelo 2	0,6206	0,6917	0,9863	0,3972	0,7619	0,6917
Modelo 3	0,9830	0,89041	0,7945	0,9863	0,8787	0,89
Modelo 4	0,9729	0,9794	0,9863	0,9726	0,9795	0,9794

Realizado por: Andrés Valencia

Al analizar la Tabla 5-3. se puede observar que el modelo que mejores resultados obtuvo es el modelo 4, en el cual se utilizó la técnica de transfer learning con 100 épocas, aplicando el algoritmo preentrenado VGG16. Por lo tanto, este modelo será el seleccionado para ser utilizado como clasificador binario de termogramas.

3.12. Comprobación del algoritmo de clasificación seleccionado.

Para poder comprobar el correcto funcionamiento del modelo seleccionado, se entregará cierta cantidad de termogramas que nunca han sido visualizados por el algoritmo, y se verificará la cantidad de termogramas que fueron clasificados correctamente y cuales no lo fueron.

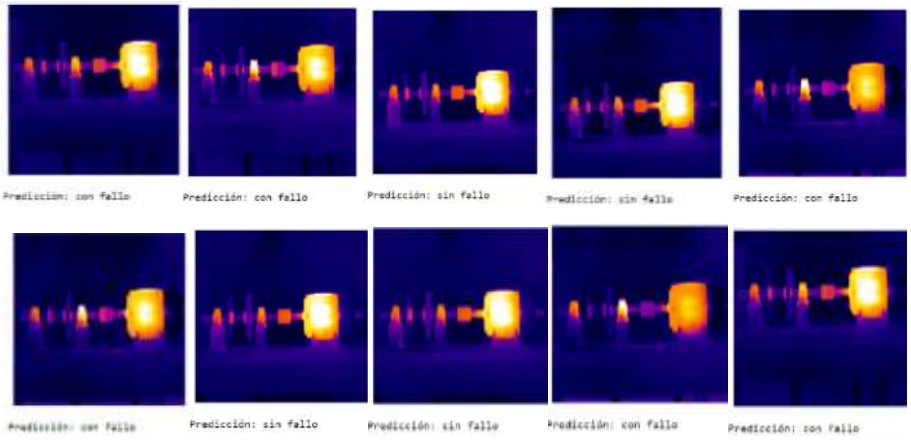


Figura 6-3: Predicciones realizadas por el modelo seleccionado

Realizado por: Andrés Valencia

3.13. Constatación de la Hipótesis

La hipótesis que se planteó inicialmente en el presente trabajo de integración curricular es la siguiente:

"Utilizando el método de aprendizaje profundo se detecta fallas en imágenes termográficas", en este caso esta sería la hipótesis nula, por lo tanto, se debe plantear la hipótesis alternativa. Por lo tanto, las hipótesis quedarían establecidas de la siguiente manera:

H_0: Utilizando el método de aprendizaje profundo se detecta fallas en imágenes termográficas.
H_a: Utilizando el método de aprendizaje profundo no se detecta fallas en imágenes termográficas

Al analizar los valores de las métricas de evaluación que se encuentran en la Tabla 7-3 del modelo 4, el cual fue seleccionado como algoritmo de clasificación para este tema de estudio, se puede analizar que todas las métricas cómo: precisión, exactitud, sensibilidad, especificidad, F1 score y el valor AUC son similares, estableciendo resultados entre el 97% y el 99% sin distorsionarse de manera abrupta, por lo que se puede decir que el modelo seleccionado presenta un balance adecuado y con el menor número entre falsos positivos y falsos negativos, de igual manera al obtener un excelente porcentaje entre precisión y exactitud podemos comprobar que el algoritmo no sufre un sobreentrenamiento y tampoco tiene deficiencia sino que predice de manera correcta la mayor cantidad de termogramas ya sean con fallo o sin fallo, además el valor estadístico F1-score es cercano a ,1 por lo tanto, se puede aceptar la hipótesis nula la cual manifiesta que se puede detectar fallas en imágenes termográficas utilizando aprendizaje profundo.

CONCLUSIONES

Al procesar la base datos se puede observar la cantidad de elementos que se dispone, aparentemente pueden ser bastantes, pero mientras más imágenes, es mejor ya que la red neuronal aprenderá de mejor manera y no se genera el error de underfiting, por otro lado, si no se dispone de muchas imágenes se puede ampliar la cantidad de las mismas con la función de ImageDataGenerator, que consiste en realizar ciertas modificaciones en los termogramas, por ejemplo: rotación. inclinación, zoom, entre otras.

Para dividir la base datos en entrenamiento y prueba se debe considerar el porcentaje de elementos que se asignará a cada una ya que se puede considerar o no una validación, esto dependerá de la cantidad de datos que se disponga.

Las mejores redes neuronales para clasificación de imágenes han resultado ser las redes convolucionales, ya que extraen un gran número de características permitiéndonos incluso añadir capas de maxpool para generar un mayor número de extracción de características, sin embargo, en caso de ser necesario se puede delimitar las zonas de estudio para ser más preciso el análisis que se realice.

Al aplicar el algoritmo de clasificación seleccionado, en un set de datos el cual contiene termogramas con fallo y sin fallo que nunca fueron visualizados por dicho algoritmo se puede diferenciar la cantidad de termogramas que fueron clasificados correctamente y cuales no, observando que fue poca la cantidad de errores que cometió, siendo así una opción para utilizarlo en una futura clasificación para detectar termogramas con fallo y sin fallo.

Realizando predicciones con el algoritmo entrenado y seleccionado con redes convolucionales se obtiene una gran cantidad de aciertos verificando así la efectividad de este y se puede corroborar visualmente que el modelo detecta adecuadamente los termogramas con fallo y sin fallo.

Mediante un enlace que permita aplicar el algoritmo diseñado mediante vía internet se puede crear una aplicación para ser utilizada en teléfonos celulares la cual sea capaz de clasificar de manera correcta termogramas con fallo y sin fallo, siempre y cuando se trate del mismo modo de fallo.

RECOMENDACIONES

Adquirir la mayor cantidad posible de termogramas al momento de crear el dataset y aumentar los parámetros, para que la red neuronal sea capaz de extraer la mayor cantidad de características y obtener una alta precisión.

Realizar una limpieza de datos de manera manual, ya que los termogramas pueden tener similitudes para el reconocimiento de imágenes y esto puede generar una baja precisión al momento de entrenar la red neuronal.

Actualizar la información de las librerías con las cual se trabaja para la creación del algoritmo, ya que estás constantemente presentan cambios, y algunos códigos son eliminados y el programa presenta fallos al momento de ejecutarlo.

Cuando se aplica la técnica de transfer learning se debe analizar la arquitectura con la cual fue diseñado el algoritmo para poder comprender de mejor manera lo que está realizando la red internamente y poder ingresar el tamaño de la imagen que sea compatible con el algoritmo preentrenado.

Cuando el modelo creado se demora en el entrenamiento, se puede utilizar la plataforma de Google Colab, ya que está permite acceder a una cierta capacidad de memoria dentro de una GPU y el entrenamiento lo realiza más rápidamente, pero algunas herramientas deben ser modificados ya que se acceden con un código diferente.

Que el Área de Mantenimiento Industrial se enfoque en aprovechar al máximo las nuevas tecnologías de la Industria 4.0, desarrollando mas temas de investigación relacionados a la inteligencia artificial, que puedan ser adaptados a los diferentes métodos de mantenimiento preventivo que existen y así aplicarlo a las necesidades de la industria y explotando al máximo las habilidades de los Ingenieros en Mantenimiento Industrial.

BIBLIOGRAFÍA

ALVERCA, BP Núñez. Clasificación automática de parásitos de reptiles mediante redes neuronales convolucionales (Trabajo de Integración Curricular). (Ingeniería) [En línea] Escuela Politécnica Nacional, Quito, Ecuador, 2022. pp. 20 [Consulta: 2022-08-06]. Disponible en: http://bibdigital.epn.edu.ec/handle/15000/22711

TAYUPANTA, Eduardo. Análisis comparativo de arquitecturas de aprendizaje profundo para odometría visual, (Trabajo de Titulación). (Maestría) [En línea] Universidad Internacional de la Rioja, Logroño, España, 2020. pp. 20 [Consulta: 2022-05-02]. Disponible en: .https://reunir.unir.net/handle/123456789/10820

GARCÍA, Juan. *Redes neuronales artificiales: aplicación a la regionalización de la precipitación y temperaturas diarias.* [En línea]. Madrid-España, Agencia Estatal de Metereología, 2021. [Consulta: 2022-07-11]. Disponible en: https://repositorio.aemet.es/handle/20.500.11765/12707

ATANASSOVA, Iana; et al. "Editorial: Mining Scientific Papers: NLP-enhanced Bibliometrics". *Frontiers in Research Metrics and Analytics* [En línea], 2019, (United States) 4(1), pp. 1-3. [Consulta: 2022-07-11]. Disponible en: https://www.frontiersin.org/articles/10.3389/frma.2019.00002/full.

BOBADILLA, J. *Machine Learning y Deep Learning: Usando Python, Scikit y Keras*. [En línea]. Bogotá-Colombia: Editorial Ra-ma, 2020 [Consulta: 2022-08-06]. Disponible en: https://books.google.es/books?hl=es&lr=&id=iAAyEAAAQBAJ&oi=fnd&pg=PA11&dq=tipos+de+aprendizaje+en+machine+learning&ots=QhAbz0tK7q&sig=xEKDcyoE9zP57I0jlaus7h4vaTc

CABRERA, Juan. Aprendizaje Profundo para el procesamiento de Imágenes-Optimización del conjunto de datos de entrenamiento. (Tesis de grado) (Licenciatura) [En línea] Universidad de la República (Uruguay), Facultad de Ingeniería, Montevideo, Uruguay. 2021. pp. 20 [Consulta: 2022-05-02]. Disponible en: https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/30313

GUZMÁN, Edwin. Obtención de una base de datos de temperatura, corriente y caída de tensión causada por el desbalanceo mecánico en el laboratorio de diagnóstico técnico. (Trabajo de Integración Curricular) (Ingeniería) [En línea] Escuela Superior Politécnica de Chimborazo,

Facultad de Mecánica, Carrera de Mantenimiento Industrial, Riobamba, Ecuador. 2022. pp. 85 [Consulta: 2022-12-13]. Disponible en: http://dspace.espoch.edu.ec/handle/123456789/17781

ARIAS, César. Detección y Diagnóstico de Fallas en Rodamientos con Velocidad Variable Aplicando Aprendizaje Profundo. (Trabajo Fin de Máster) (Maestría) [En línea] Universidad Internacional de la Rioja, Bogotá, Colombia. 2020. pp. 85 [Consulta 2022-05-02]. Disponible en: https://reunir.unir.net/handle/123456789/10700

FRANCISCO, Yagüe; et al. *La robótica y la inteligencia artificial en la nueva era de la revolución industrial 4.0: los desafíos jurídicos, éticos y tecnológicos de los robots inteligentes*. [En línea]. Madrid-España: Editorial Dykinson, 2021 [Consulta: 2022-05-03]. Disponible en: https://elibro.net/es/lc/espoch/titulos/189569?fs_q=inteligencia__artificial&fs_page=3&prev=fs

TUME, María. Estado del arte de la inteligencia artificial y su aplicación en el mantenimiento. (Tesis) (Ingeniería) [En línea] Universidad de Piura, Piura, Perú. 2022. pp. 85 [Consulta 2022-05-02]. Disponible en: https://pirhua.udep.edu.pe/handle/11042/5490

GARCÍA, Roberto. "EL PERCEPTRÓN: UNA RED NEURONAL ARTIFICIAL PARA CLASIFICAR DATOS". *Revista de investigación en modelos matemáticos aplicados a la gestión y la economía.* [En línea], 2022, (Argentina) 1(8), pp. 3-10 [Consulta: 2022-08-06]. Disponible en: http://www.economicas.uba.ar/wp-content/uploads/2016/04/Garcia-Roberto-1.pdf

GIL-VERA, VD, & SEGURO-GALLEGO, C. "Machine learning aplicado al análisis del rendimiento de desarrollos de software". *Revista Politécnica.* [En línea], 2022, (Colombia) 18(35), pp. 128-139 [Consulta: 2022-08-06]. Disponible en: https://revistas.elpoli.edu.co/index.php/pol/article/view/1976

NEGRETE, Ana. "Redes Neuronales". *Docplayer* [En línea], 2021, 1(1), pp. 1-5[Consulta: 2022-05-03]. Disponible en: http://ofeliayorquesta.com/articulos/Redes_Neuronales_001.pdf

PELEGERO, Lorena; et al. *Clasificación de imágenes de cáncer de cerebro mediante aprendizaje profundo*. [blog]. 2022 [Consulta 2022-07-12]. Disponible en: http://ocw.uoc.edu/webapps/o2/handle/10609/138146

PERALTA, Liz; et al. "Exactitud y precisión de los métodos dentales para estimar la edad basados en la transparencia de la dentina radicular". *Revista Ciencias de la Salud* [En línea], 2022, 20(2), pp. 1-16. [Consulta: 2022-08-06]. Disponible en:

https://revistas.urosario.edu.co/index.php/revsalud/article/view/9555

PÉREZ, Daniel; et al. "Transfer learning en la clasificación binaria de imágenes térmicas". *Ingenius: Revista de Ciencia y Tecnología* [En línea], 2021, (Ecuador), pp. 1-30. [Consulta: 2022-05-02]. ISSN:1390-650X. Disponible en: https://www.redalyc.org/journal/5055/505567902007/

PINEDA, Carlos. *Aprendizaje automático y profundo en python: Una mirada hacia la inteligencia artificial.* [En línea]. Bogotá-Colombia: Ediciones de la U, 2021 [Consulta: 2022-08-03]. Disponible en: https://books.google.es/books?hl=es&lr=&id=mgNcEAAAQBAJ&oi=fnd&pg=PA11&dq=que+es+python&ots=UOyEv2JTWQ&sig=MH7dvMaVxwwyIvBPGvVdcsQbRIA

LONCOMILLA, Patricio. *Deep learning: Redes convolucionales*. [blog]. [Consulta: 2022-05-03]. Disponible en: https://ccc.inaoep.mx/~pgomez/deep/presentations/2016Loncomilla.pdf

LÓPEZ, Juan; & PURIZACA, Miguel. *Modelo tecnológico para optimizar el proceso de detección de leucemia utilizando el algoritmo canny, a través de la microscopía digital* (Tesis) (Ingeniería) [En línea] Universidad Peruana de Ciencias Aplicadas, Facultad de Ingeniería, Lima, Perú. 2021. pp. 46 [Consulta: 2022-05-05]. Disponible en: http://hdl.handle.net/10757/655225

QADIR, Sumair; & MEHRA, Mónica. "Personality Prediction Using Sklearn". *SSRN* [En línea], 2022, (India) 9(2), pp. 29-32. ISSN 2350-0557 [Consulta: 2022-08-2022]. Disponible en: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4090780

RIVERA DEMANUEL, Diego Richard, HUANCARA, Cleofe Huamani and CCAMA, Alfredo Charca, 2022. Sistema automático para calificación de vino mediante Redes Neuronales. [En línea]revistas.ulasalle.edu.pe. Online. 2022. Vol. 3, no. 1. [Consulta: 2022-08-03]. Disponible en: https://revistas.ulasalle.edu.pe/innosoft/article/view/51

SÁNCHEZ, A López, 2021. Aplicación de aprendizaje profundo por refuerzo a problemas de robótica aérea. [En línea]. 2021. [Consulta: 2022-05-02]. Disponible en: https://oa.upm.es/id/eprint/68638

SCHMETTERER, L. Archivos Argentinos and 2021, 2021. Aprendizaje profundo, la atención oftalmológica del futuro [En línea]. 2021. Vol. 19, pp. 7–10. [Consulta: 2022-05-03]. Disponible en: https://archivosoftalmologia.com.ar/index.php/revista/article/download/142/160

UNE-EN 13306:2018 Mantenimiento. Terminología del mantenimiento. [En línea]. [Consulta: 3 May 2022]. Disponible en: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0060338

WAN, Z, YANG, R, HUANG, M, ZENG, N, NEUROCOMPUTING, X Liu. undefined. A review on transfer learning in EEG signal analysis. Elsevier. [En línea]. [Consulta: 2022-08-06]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0925231220314223

XIAO, Aoran, HUANG, Jiaxing, GUAN, Dayan, ZHAN, Fangneng and LU, Shijian. Transfer Learning from Synthetic to Real LiDAR Point Cloud for Semantic Segmentation. [En línea]. 2022. [Consulta: 2022-05-03]. Disponible en: www.aaai.org

ANEXOS