Alejandra Vilchis Yubi

Sistema embebido basado en termografía infrarroja de bajo costo e inteligencia artificial para la detección de anomalías en miembro inferior

2025

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Sistema embebido basado en termografía infrarroja de bajo costo e inteligencia artificial para la detección de anomalías en miembro inferior

Tesis

Que como parte de los requisitos para
obtener el Grado de
Maestro en Ciencias en Inteligencia
Artificial

Presenta

Alejandra Vilchis Yubi

Dirigido por:

Dr. Luis Alberto Morales Hernández

Co-dirigido por:

Dr. Irving Armando Cruz Albarran

Querétaro, Qro. a Septiembre 2025

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias en Inteligencia Artificial

Sistema embebido basado en termografía infrarroja de bajo costo e inteligencia artificial para la detección de anomalías en miembro inferior

TESIS

Que como parte de los requisitos para obtener el Grado de Maestro en Ciencias en Inteligencia Artificial

Presenta:

Alejandra Vilchis Yubi

Dirigido por:

Dr. Luis Alberto Morales Hernández

Co-dirigido por:

Dr. Irving Armando Cruz Albarran

Dr. Luis Alberto Morales Hernández
Presidente

Dr. Irving Armando Cruz Albarran
Secretario

Dr. Saul Tovar Arriaga
Vocal

Dr. Andras Takacs
Suplente

M. en C. Enoc Tapia Méndez
Suplente

Centro Universitario, Querétaro, Qro.
Fecha de aprobación por el Consejo Universitario (septiembre 2025)
México

DEDICATORIA

A mi familia, por su apoyo constante, por siempre acompañarme, por enseñarme a superarme, por creer siempre en mí y por regalarme su amor incondicional.

AGRADECIMIENTO

Deseo expresar mi agradecimiento a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por el financiamiento otorgado para el desarrollo de esta investigación y para mi formación académica. También a la Universidad Autónoma de Querétaro y a la Dirección de Investigación y Posgrado de la Facultad de Ingeniería, por brindar los recursos y apoyos necesarios que hicieron posible la culminación de mis estudios de posgrado. Extiendo mi reconocimiento a mis asesores, en especial al Dr. Luis Alberto Morales Hernández, por la dirección de este proyecto, y al Dr. Irving Armando Cruz Albarrán, por su guía, enseñanzas y valiosas oportunidades durante este proceso.

Por último, agradezco a mis papás por alentarme a iniciar mis estudios de posgrado, por su apoyo y motivación durante todo este proceso, por creer en mí y enseñarme con su ejemplo la importancia del esfuerzo, la perseverancia y la disciplina, no solo en el ámbito profesional, sino también en el deportivo; y, sobre todo, por el amor que me han brindado. A mis hermanitos, por compartir conmigo largas horas de estudio y alegrarme siempre con sus abrazos, sus chistes y pláticas que me hacen sentir querida. Este logro no es solo mío, sino también de ellos. A Buster, mi perrito, por recibirme siempre con alegría y hacerme sonreír cada vez que llego a casa. A mi pareja, por ser mi compañía constante, escucharme, quererme, celebrar mis logros y recordarme que puedo lograr lo que me propongo, no solo durante este proceso, sino también en natación. Y a mis amigos, cuyo apoyo fue fundamental para alcanzar este objetivo.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

ABREVIATURAS Y SIGLAS

ROI – Región de interés.

IA – Inteligencia Artificial

SDPF – Síndrome Patelofemoral

CNN – Red neuronal convolucional

SVM – Support Vector Machines (Máquina de vectores de soporte)

KNN - k-Nearest Neighbors

ReLU – Rectified Linear Unit (Unidad Lineal Rectificada)

ECG - Electrocardiograma

RESUMEN

Las lesiones en el miembro inferior son una de las afecciones más comunes en el ámbito clínico, siendo las articulaciones del tobillo y la rodilla las que sufren más. Investigaciones anteriores indican que entre el 68% y el 72% de las lesiones se localizan en esta área. Estas lesiones a menudo requieren métodos manuales o técnicas como la resonancia magnética y la tomografía, las cuales son costosas. Por lo tanto, la termografía infrarroja se presenta como una herramienta que permite una captura rápida y económica, proporcionando datos sobre el estado vascular y metabólico del cuerpo humano sin ser invasiva. Para abordar este desafío, se creó un sistema embebido que utiliza imágenes termográficas y modelos de aprendizaje profundo para identificar y clasificar anomalías del miembro inferior. El sistema se desarrolló en una placa Jetson Nano, incorporando una cámara infrarroja (Lepton 3. 5), una digital (Raspberry Pi v2. 1), una interfaz gráfica y una carcasa fabricada en 3D. Se realizaron pruebas de ablación y se aplicaron varias técnicas de aumento de datos en diversas configuraciones del modelo CNN, eligiendo el que mostró el mejor rendimiento. Este alcanzó un nivel de precisión de hasta el 98% en pruebas clínicas realizadas con el apoyo de personal fisioterapeuta. Por lo que la implementación de este tipo de soluciones embebidas es una opción viable y eficaz para el monitoreo fisiológico, con posibilidades de uso en el ámbito clínico. Como trabajo futuro, se plantea ampliar la base de datos termográfica con nuevas adquisiciones, procurando un balance adecuado entre clases. Esto permitirá mejorar la capacidad de generalización del modelo.

ABSTRACT

Lower limb injuries are one of the most common conditions in the clinical setting, with the ankle and knee joints suffering the most. Previous research indicates that between 68% and 72% of injuries are located in this area. These injuries often require manual methods or techniques such as MRI and CT scanning, which are expensive. Therefore, infrared thermography presents itself as a tool that allows rapid and cost-effective capture, providing data on the vascular and metabolic state of the human body without being invasive. To address this challenge, an embedded system was created that uses thermographic imaging and deep learning models to identify and classify lower limb abnormalities. The system was developed on a Jetson Nano board, incorporating an infrared camera (Lepton 3. 5), a digital camera (Raspberry Pi v2. 1), a graphical interface and a 3D fabricated housing. Ablation tests were performed, and several data augmentation techniques were applied on various configurations of the CNN model, choosing the one that showed the best performance. This achieved an accuracy level of up to 98% in clinical tests performed with the support of physiotherapist staff. Thus, the implementation of this type of embedded solution is a viable and effective option for physiological monitoring, with possibilities of use in the clinical setting. As future work, it is proposed to expand the thermographic database with new acquisitions, seeking an adequate balance between classes. This will improve the generalization capacity of the model.

CAPITULO 1

INTRODUCCION

1.1. Justificación

En los últimos años la termografía infrarroja se ha utilizado en la práctica clínica como una herramienta para medir la temperatura en la superficie corporal mediante la captura de imágenes, debido a que existe una correlación entre la actividad muscular y el aumento de la temperatura en la piel, la cual se considera térmicamente simétrica si la diferencia de temperatura es debajo de 0.5°C, siendo una asimetría de 0.5°C a 0.7 ° C, lo cual se asocia a una anomalía.

La termografía infrarroja presenta rapidez de captura, es de bajo costo, y obtiene información del estado vascular y metabólico del cuerpo humano sin ser un método invasivo. Por estos motivos, en la actualidad se utiliza como herramienta auxiliar para el diagnóstico, prevención y monitorización de fisiopatologías relacionadas a las anomalías en el cuerpo (Escamilla-Galindo et al., 2017; Trejo-Chavez et al., 2023).

Las lesiones afectan negativamente la vida de las personas, provocando dolor, limitaciones en las actividades de la vida diaria, deterioro de la condición física, es decir, es una disminución en la calidad de vida de la persona lesionada. Por lo tanto, es necesario desarrollar métodos para la detección de enfermedades musculares sin necesidad de una intervención médica. Debido a las ventajas que presenta la termografía en la capacidad de detección de anomalías fisiológicas, surge esta propuesta de investigación para desarrollar un sistema embebido no invasivo, de bajo costo, robusto y de poca carga computacional que sea capaz de clasificar de forma inteligente, mediante un modelo de aprendizaje profundo, anomalías en extremidad inferior por medio de termografía infrarrojo, reduciendo así el margen de error e incrementando la capacidad de análisis de datos.

En la Universidad Autónoma de Querétaro, se han realizado sistemas de visión artificial en termografía enfocados en miembro inferior (Vega Mancilla, 2022). La presente propuesta de investigación complementa los trabajos anteriores, pero se centra en la creación de un sistema embebido con técnicas de aprendizaje profundo que permita la mejora de detección de alteraciones fisiológicas, además, esta metodología reduce la carga computación del sistema embebido

permitiendo procesar más rápidamente, lo que permite un diagnóstico más rápido y oportuno. Del mismo modo, sirve como ayuda en la detección de lesiones en miembro inferior y beneficia a los pacientes al ser un proceso no invasivo y de bajo costo, ayudando a reducir el costo, el tiempo y los errores en la detección de lesiones en miembro inferior

1.2. Descripción del problema

Las lesiones de miembro inferior son un fenómeno bastante común en la vida, de acuerdo con varias investigaciones especializadas en epidemiología concluyeron que la parte del cuerpo humano que es mayormente afectada por lesiones es la extremidad inferior, según Moreno Pascual et al., (2008) concluyó que la incidencia de lesiones en miembro inferior fue de 71.9%, siendo las articulaciones del tobillo y la rodilla las más afectadas, mientras que (García González et al., 2015) el porcentaje de lesiones en la extremidad inferior fue del 68%, y en el miembro superior fue de 22%.

La termografía infrarroja ha sido utilizada como una herramienta en el diagnóstico de alteraciones fisiológica, aumentando la utilidad de esta técnica en conjunto con modelos de inteligencia artificial para mejorar la precisión, exactitud, sensibilidad y especificidad al clasificar termogramas.

Sin embargo, el uso de modelos de inteligencia artificial en la detección de anomalías es requerido para el diagnóstico sustituyendo otras técnicas que actualmente se usan como resonancias magnéticas, tomografías, etc.; las cuales son costosas y tardadas tanto para la obtención de la imagen como para la entrega de los estudios, atrasando el diagnóstico del paciente.

La implementación del modelo en un sistema embebido beneficia en la detección de enfermedades al permitir tomar en tiempo real los datos del paciente, procesar y diagnosticar. Pero son pocos los modelos de inteligencia artificial que han sido aplicados a sistemas embebidos debido a que requieren un gran costo computacional.

Además de ello, la termografía como método de clasificación se utiliza principalmente en enfermedades como el cáncer de mama y la diabetes. Mientras que la clasificación de lesiones en miembro inferior ha sido principalmente estudiada con métodos manuales, ocasionando errores y

pérdida de información al momento de clasificar, y, por consiguiente, la implementación en sistemas embebidos es escasa.

Por lo que, la implementación de un modelo de aprendizaje profundo en un sistema embebido para detección de anomalías en miembro inferior es un área de oportunidad grande debido a la limitante que presenta el sistema embebido en la carga computacional, además de la escasa investigación del miembro inferior con modelos de inteligencia artificial. Además, la carga económica y el tiempo de espera de los pacientes para obtener un diagnóstico es grande.

De acuerdo con lo mencionado anteriormente surge la siguiente pregunta: ¿Puede un modelo de aprendizaje profundo aplicado en un sistema embebido lograr la identificación, detección y clasificación de anomalías fisiológicas en miembro inferior?

1.3. Hipótesis

Un sistema embebido basado en un modelo de aprendizaje profundo permitirá identificar y detectar anomalías fisiológicas en el miembro inferior mediante imágenes termográficas con métricas de desempeño comparables a los métodos manuales.

Nota: Para fines de este estudio se entiende como métricas a la precisión, exactitud, sensibilidad y especificidad.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un sistema embebido con termografía infrarroja para la detección y la clasificación de anomalías fisiológicas en el miembro inferior por medio de modelos de aprendizaje profundo.

1.4.2. Objetivos específicos

- Seleccionar el mejor modelo de aprendizaje profundo probado que se adecue al sistema embebido para desarrollar el modelo de clasificación probando diferentes modelos.
- Desarrollar un sistema embebido con inteligencia artificial para la obtención en tiempo real

de termogramas.

- Realizar la implementación de una cámara termografía mediante la configuración y conexión de componentes necesarios para la adquisición de termogramas.

- Implementar módulo mediante la conexión de todos los componentes para la detección de anomalías fisiológicas

- Diseñar e implementar la carcasa para el sistema embebido mediante diseño e impresión 3D.

- Realizar pruebas clínicas con el sistema embebido desarrollado para la validación de este.

CAPITULO 2

ANTECEDENTES

2.1. Termografía infrarroja y miembro inferior

El ser humano tiende a padecer diversas enfermedades fisiológicas y patológicas que pueden influir en su rendimiento en las tareas cotidianas y, por lo tanto, en su bienestar general. Según diversos estudios epidemiológicos, las partes del cuerpo más susceptibles a sufrir lesiones y anomalías fisiológicas son las extremidades inferiores, seguidas de las superiores (Moreno Pascual et al., 2008).

Normalmente cuando ocurre una lesión, se producen se alteran el flujo de sangre y en la transferencia de calor de los tejidos, lo que afecta la temperatura en la piel. Por esta razón, los cambios de la temperatura superficial debido al estrés térmico aumentan la capacidad de detectar y monitorear diversas anormalidades o patologías como: el cáncer, complicaciones diabéticas, lesiones musculares, entre otros (Trejo-Chavez et al., 2022; Vardasca et al., 2012).

La termografía infrarroja es considerada como una herramienta enfocada en analizar la temperatura en la piel, ya que capta la energía irradiada de un cuerpo. El área clínica se ha beneficiado de esta herramienta al ser un método no invasivo, de bajo costo y por su rendimiento. Además, brinda una imagen completa de la distribución de la temperatura corporal, lo que ayuda a obtener información adicional sobre la salud general del paciente (Farooq & Corcoran, 2020; Vega Mancilla, 2022).

Un enfoque en la investigación sobre la aplicación de la termografía infrarroja en el ámbito de la salud es la identificación y prevención de lesiones musculares en los miembros inferiores, especialmente en deportistas. Por ejemplo, Gómez-Carmona et al., (2020) Reunieron y examinaron termogramas de 24 futbolistas profesionales de la primera división española para averiguar si este novedoso enfoque podría disminuir la incidencia de lesiones. Para ello, en la primera pretemporada emplearon un sistema estándar para prevenir lesiones, mientras que en la segunda pretemporada utilizaron un programa de prevención de lesiones con termografía infrarroja. Usaron el software "Thermocan Researcher Professional" para extraer las temperaturas de 25 regiones de interés (ROI, por sus siglas en ingles). Al finalizar las dos pretemporadas, el número de lesiones

disminuyó de 15 lesiones en la primera pretemporada a 6 lesiones en la segunda, y también disminuyó el número de días de baja por lesión. En otro estudio de futbolistas profesionales, Rodríguez-Sanz et al., (2017), analizaron a 21 participantes masculinos entre 23 y 26 años que padecían o no una dolencia en el gastrocnemio-sóleo (IGE), para detectar cambios de temperatura en el musculo gastrocnemio, en el tibial anterior y en el tendón de Aquiles. Nuevamente se utilizó el software "Thermocan Researcher Professional" para encontrar los valores máximos, mínimos y el promedio de los músculos seleccionados. Identificaron cambios significativos de temperatura en el músculo tibial anterior entre los participantes con y sin IGE, por lo tanto, recomendaron el uso de termografía para prevenir lesiones en atletas.

Otro enfoque en el área de la salud se centra en la detección de anomalías en la simetría térmica entre dos regiones de interés. La simetría térmica se define como el grado de similitud entre dos regiones de interés y se mide mediante la media y de la desviación estándar (Vardasca et al., 2012; Vega Mancilla, 2022). Cuando hay una diferencia significativa de temperatura se asocia a disfunciones fisiológicas como: tumores, inflamación, infección, lesiones traumáticas, etc. (Trejo-Chavez et al., 2023). Vardasca et al., (2012), realizaron un estudio en 29 pacientes sanos de acuerdo con la simetría correspondiente del lado izquierdo y derecho se encontró que, en pacientes sanos, la diferencia entre ambos lados del cuerpo es de máximo $0.4°C$. Para que se considere térmicamente asimétrico, el grado de similitud entre las dos áreas de interés y su desviación estándar deben ser mayor a $0.5°C$ y $0.3°C$ respectivamente, de lo contrario se considera que es simétricamente térmico (Vardasca et al., 2012).

En un estudio piloto (Côrte et al., 2019) evaluaron termogramas de 28 futbolistas profesionales de la liga brasileña para evaluar la termografía como un método auxiliar de prevención de lesiones, para ello utilizaron el software "FLIR Research IR Max 4" y dividieron en 12 regiones de interés, 7 regiones en la parte anterior y 5 regiones en la parte posterior del miembro inferior. Durante la temporada de 2015 se detectaron 11 lesiones en total. Mientras que en la temporada de 2016 se aplicó un método de prevención basado en la termografía infrarroja, donde las anomalías eran registradas dependiendo el grado de asimetría, resultando en un total de 4 lesiones musculares. De acuerdo con los datos obtenidos se concluyó que esta nueva técnica redujo en un 64% las lesiones en los futbolistas. Trejo-Chavez et al., (2023) recopilaron y analizaron 48 termogramas de pacientes sanos y pacientes con síndrome de dolor patelofemoral bilateral (SDPF), los cuales se

dividieron en 2 grupos, un grupo de control y un grupo experimental; y cada grupo se subdividieron en dos grupos de 12 personas, donde se aplicó calor a un grupo y frío al otro. Se utilizó el software "FLIR TOOLS" para obtener la temperatura de la piel. Se observó que no es posible detectar SDPF en el estado basal mediante termografía y tampoco es evidente en el estrés por frio. Sin embargo, tras el estrés térmico la recuperación térmica es menor en el grupo con SDPF que en el grupo de control, por lo que sería susceptible de detección.

2.2. Inteligencia artificial aplicada en termografía infrarroja

En los estudios anteriores las técnicas de detección de las regiones de interés se realizaron de manera manual, con figuras geométricas se seleccionaban las regiones deseadas y a partir de ello analizaban las termografías junto con la ayuda de los software utilizados, los cuales brindan información sobre la temperatura mostrada en los termogramas, de esta manera ambas técnicas se complementan para poder detectar asimetrías térmicas entre las dos partes del cuerpo o simplemente para notar una diferencia de cambio de temperatura durante los entrenamientos de los atletas.

A pesar de ello, el análisis de imágenes termográficas no es una tarea simple, pequeñas asimétricas pueden representar una anormalidad aun cuando las imágenes se ven casi simétricas. Estas diferencias pueden ser difíciles de identificar para la interpretación humana, es necesario considerar que la visión humana presenta limitaciones que afectan la precisión del diagnóstico. Por lo tanto, es necesario el desarrollo e implementación de un método automático que elimine los factores humanos ya que es importante para mejor la eficacia y el rendimiento de estas técnicas (Alshehri & AlSaeed, 2022; de Freitas Barbosa et al., 2020).

Varias investigaciones se están centrando en modelos de inteligencia artificial como un método automático. Una de las investigaciones enfocadas a esto es la realizada por Farooq & Corcoran, (2020) donde proponen un sistema autónomo de clasificación de tumores de mama a partir de imágenes térmicas empleando redes neuronales de convolución (CNN), utilizaron una base de datos pública llamada "DMR - Database for Mastology Research" donde usaron los datos de 40 pacientes (22 sanos y 18 con cáncer). El modelo de CNN que se utilizó fue "Inception V3 DL". Para la validación de este modelo se consideró el 20% de los datos y 10% para la prueba, se obtuvo una exactitud de 80%, sensibilidad de 83.33%, precisión de 71.42% y especificidad de 77.77%.

Cruz-Vega et al., (2020) realizó una comparación de modelos para la detección de diabetes, entre ellos se incluye AlexNet, GoogleNet y diseñaron una nueva estructura basada en aprendizaje profundo llamada “Diabetic Foot Thermogram Network (DFTNet)” con características de 9 capas, 100 épocas y un “batch” de 64. Para estos modelos se utilizó una base de datos pública "Plantar Thermogram Database for the Study of Diabetic Foot Complications" y para aumentar el tamaño de la data se usaron técnicas como rotación, mejora de contraste, entre otros. El modelo que obtuvo los mejores resultados fue el DFTNet con una sensibilidad de 91.61% y una exactitud de 85.3%. Otro estudio, realizado por Mohamed et al., (2022), propone su propio modelo de aprendizaje profundo en dos clases para la clasificación de tejidos mamarios normales y anormales, donde la red U-Net se utiliza para extraer y aislar automáticamente la zona del pecho del resto del cuerpo. El modelo propuesto es aprendizaje profundo con U-Network/CNN con 9 capas y entrenado con 30 épocas. Este modelo da una precisión de 99.33%, sensibilidad de 100% y especificidad de 98.67%.

Estudios más recientes se han centrado en crear varios modelos con diferentes características para observar que modelo clasifica mejor los termogramas, en lugar de compararlos con modelos existentes como se ha mostrado en estudios anteriores. En 2022, Filipe et al., (2022) crearon dos modelos para la clasificación de termogramas de personas sanas y personas con pie diabético. El primer modelo usa regresión logística (RL) con K vecinos más cercanos (k-NN), y clasifica en 4 clases: 1 clase para personas sanas y 3 clases dependiendo la gravedad de diabetes. El segundo modelo usa máquinas de vectores de soporte (SVM) y con k-NN, a diferencia del primero, este modelo tiene dos etapas, en la primera se clasifica si el pie pertenece a una persona sana o con diabetes; y en la segunda etapa se clasifica en tres clases dependiendo el grado de progresividad. En ambos se utilizó una base pública "Plantar Thermogram Database for the Study of Diabetic Foot Complications". Se concluyó que el segundo modelo clasificó mejor que el modelo 1 dando una exactitud de 93.2% contra 88.6%; una especificidad de 95.4% contra 92.3%.

Alshehri & AlSaeed, (2022) crearon 4 modelos diferentes para detectar cáncer de mama usaron en redes neuronales profundas con mecanismo de atención (AMs) utilizando imágenes térmicas de la base de datos “Database for Research Mastology with Infrared Image (DMR-IR)". El primer modelo es una CNN, la segunda es una CNN con “self-attention” (SL) como AMs, la tercera es una CNN con “soft-attention” (SF) como AMs y finalmente una CNN con “hard-attention” (HD)

como AMs. Todos los modelos fueron entrenados con 1302 imágenes y probados con 240 imágenes, las épocas de entrenamientos fueron de 50 y el batch fue de 64. El modelo CNN sin AM obtuvo el rendimiento más bajo entre los modelos, sin embargo, mostró resultados aceptables. Los resultados también mostraron que el modelo CNN-HD tuvo un impacto ligeramente mayor en métricas como la exactitud (99.49%), la especificidad (99.71%) y la sensibilidad (99.71%).

Para el miembro inferior, Trejo-Chavez et al., (2022) crearon una metodología con CNN para diferenciar automáticamente entre una rodilla lesionada y una rodilla sana. Para ello, generaron una base de datos, hicieron el procesamiento de imágenes y diseñaron y validaron el modelo CNN para identificar un paciente con rodilla lastimada, el cual consistía en un número de filtros de 8, un batch de 25 y épocas de 30. Para comprobar la eficiencia del modelo se probó con varias imágenes termográficas con cambios en la rotación y en diferentes niveles de brillo, el resultado que se obtuvo fue de una exactitud de 97.44%.

Los modelos de inteligencia artificial aplicados a la termografía infrarroja han mostrado resultados positivos, ya que la mayoría de los estudios mostrados obtuvieron un rendimiento superior al 90% en cuanto a exactitud, precisión y sensibilidad, demostrando así ser una herramienta útil para la detección de afecciones por medio de termografía infrarroja.

2.3. Sistemas embebidos con modelos de inteligencia artificial

En la actualidad, se están utilizando modelos de inteligencia artificial en sistemas embebidos, lo que facilita el desarrollo de sistemas inteligentes y autónomos. Además, proporcionan diversas aplicaciones, siendo una de ellas la visión por computadora, donde una de las tareas principales es la detección y clasificación de objetos. La implementación de algoritmos “machine learning” en sistemas embebidos pueden generar resultados positivos, como productos de bajo costo, además de ofrecer respuestas rápidas en tiempo real. Sin embargo, también presentan un desafío ya que la mayoría de estos algoritmos consumen recursos computacionales debido a la complejidad de las operaciones realizadas (Millán Duque, 2021; Sánchez Granja & Asanza, 2021).

Algunas de las investigaciones que se han realizado acerca de modelos de inteligentes aplicados a sistemas embebidos es el de Azariadi et al., (2016) donde desarrollaron un algoritmo para el análisis y la clasificación de ECG para el diagnóstico de latidos cardíacos, que se implementó en

una plataforma embebida basada en IoT. La implementación se realizó en la plataforma integrada Galileo de Intel, el data que se usó fue el de MIT-HIT "Arrhythmia database by PhysioNet", para el análisis se usó la transformada discreta de Wavelet (DWT) y para la clasificación se usó SVM. Este algoritmo propuesto logró una precisión superior a 97%, y proponen que sea utilizada en un dispositivo de diagnóstico de ECG portátil para la monitorización continua del paciente.

Maret et al., (2018) presentan un sistema que puede identificar una secuencia de los siguientes gestos en tiempo real: neutral, amistoso, neutral, angustia, neutral, enfadado. Para el sistema embebido se utilizó la tarjeta Odrior-XU4 con Linux. Las características fueron extraídas de un esqueleto 3D mediante el sensor Kinect v2 y se clasificaron usando un método SVM. Los resultados mostraron una confianza por encima del 85% en el modelo SVM para la detección de cada gesto y la secuencia de gestos se reconoció correctamente al 100 % en tiempo real para cada imagen. Millán Duque, (2021) desarrolló un sistema embebido capaz de clasificar imágenes en tiempo real, para ello utilizaron una colección de imágenes de "Café Robusta". El sistema embebido se implementó en la tarjeta NVIDIA Jetson Nano, en donde se entrenaron y se validaron los 3 modelos de redes neuronales (SVM, KNN y CNN), donde el modelo CNN obtuvo un desempeño significativamente mayor que los modelos clásicos k-NN y SVM, con una precisión del 98.21% para el entrenamiento y una precisión del 99.36% para validación.

2.4. Investigaciones en Universidad Autónoma de Querétaro

En la Universidad Autónoma de Querétaro, Vega Mancilla diseñó y desarrolló sistema de visión artificial que utiliza termografía para identificar irregularidades musculares a través de la medición automática de asimetría térmica para el tronco inferior del cuerpo. El principal problema que atendió es la segmentación de regiones anatómicas, donde se desarrolló un método basado en el centroide el cual mostró su capacidad para diferenciar entre las principales regiones del miembro inferior en vista frontal y posterior (Vega Mancilla, 2022).

Como se revisó anteriormente, la termografía ha sido el centro de estudio durante varios años y ha pasado por diversas etapas en un intento de reducir las áreas de oportunidad presentadas en estudios anteriores, por ejemplo, la segmentación manual de regiones de interés. Lo que lleva a probar nuevos métodos como los modelos de inteligencia artificial, que sugiere proporcionar mayor precisión y exactitud en la detección de anomalías fisiológicas. Al ser un área relativamente

nueva, muchos de estos estudios se centran en la detección de tumores en cáncer de mama o en la detección de diabetes mediante el pie diabético.

La escasa investigación en lesiones en miembro inferior y en la obtención de termogramas mediante modelos de inteligencia artificial presentan una gran área de oportunidad, que puede abarcarse al aplicarse en un sistema embebido que permita la detección de lesiones en tiempo real. Por lo que este proyecto de investigación se centrara en el desarrollo de un modelo de inteligencia artificial aplicado a un sistema embebido, implementado en una Jetson Nano NVIDIA; capaz de detectar e identificar en tiempo real imágenes termográficas para la detección de anomalías en el miembro inferior.

2.5. Estado del arte

Durante los últimos años se ha utilizado la termografía como una herramienta complementaria en diversas áreas, especialmente en el área médica. Ha sido utilizada para reducir y prevenir lesiones musculares en deportistas, para la prevención o detección de enfermedades o anomalías como la diabetes, cáncer, infecciones, lesiones traumáticas, inflamación, etc.

No obstante, su aplicación como herramienta ha cambiado con el tiempo debido a innovaciones tecnológicas y a la necesidad de mejorar su desempeño, además de aplicarse a nuevos enfoques. Un nuevo enfoque de la termografía son las lesiones musculares, donde, en una primera aproximación al tema, se ha utilizado softwares para estudiar los termogramas obteniendo la temperatura del área de interés y, de este modo, evaluar la variación térmica. En Trejo-Chavez et al., (2023) se investigó la región de interés (ROI) se analizó para cada rodilla, la cual tiene la forma de una elipse vertical con centro en la parte central de la rótula sin tocar los bordes lateral y medial de la rodilla, como se observa en la Figura 1a. En Gómez-Carmona et al., (2020) se obtuvo la temperatura de la piel, para 25 regiones de interés (ROIs) del miembro inferior y espalda, tanto dorsal como anterior, como se puede ver en la Figura 1b. En Côrte et al., (2019) se investigó el desarrollo de la misma sección y se analizaron las diferencias entre las ROIs contralaterales (es decir, el lado izquierdo comparado con el lado derecho), Figura 1c. Sin embargo, la definición de los ROIs se realizó manualmente, lo que puede incluir algún sesgo humano, como la perdida de información por la limitante de las figuras geométricas (Trejo-Chavez et al., 2023).

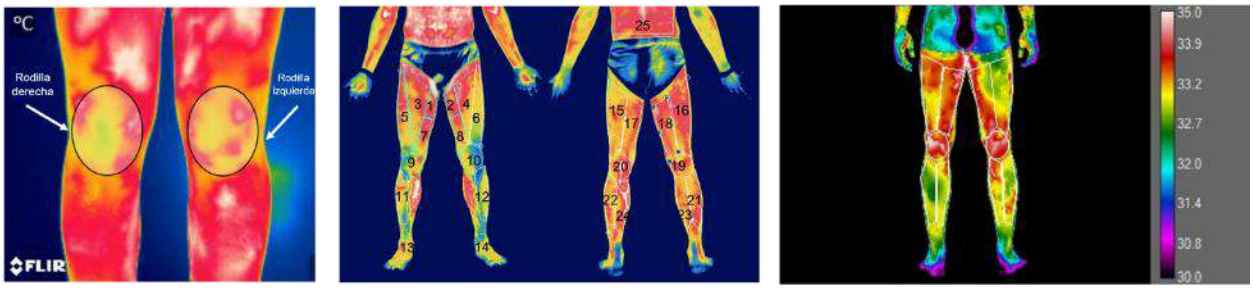

Figura 1. Termogramas con áreas de interés resaltadas mediante figuras geométricas. a) Se seleccionaron con óvalos las rótula, obtenido de (Gómez-Carmona et al., 2020), b) Selección de 25 ROIS en miembro inferior y espalda baja, obtenida de (Gómez-Carmona et al., 2020), c) Selección de 5 ROIs del miembro inferior, obtenido de (Côrte et al., 2019).

A medida que la tecnología ha progresado, se han creado nuevas formas de examinar termogramas, lo que ha dado paso al uso de inteligencia artificial para detección y clasificación de enfermedades o anomalías a través de los termogramas. En los últimos años, la IA ha sido utilizada principalmente en enfermedades como diabetes, específicamente el pie diabético (Cruz-Vega et al., 2020); así como en tumores, especialmente el cáncer de mama (Farooq & Corcoran, 2020; Mohamed et al., 2022). El enfoque principal de estos estudios ha estado en los modelos de aprendizaje profundo, por ejemplo, en Cruz-Vega et al., (2020)se desarrolló un modelo propio denominado DFTNet con una exactitud de 85.3%. En Farooq & Corcoran, (2020)se hizo uso del modelo Inception V3 con una exactitud de 80% y en Mohamed et al., (2022)usaron un modelo U-Net y una CNN la cual obtuvo una precisión 99.3%. Sin embargo, el análisis de lesiones musculares no ha sido el enfoque principal. Se han llevado a cabo investigaciones como la de Trejo-Chavez et al., (2022), el cual se centra en lesiones musculares de rodilla clasificando en dos grupos: lesión de rodilla o rodilla sana, usando una red neuronal convolucional (CNN) obteniendo una exactitud de 97.44% y 100.00% de precisión.

La Tabla 1 presenta un compendio de los estudios previamente citados, en el que se incluye la base de datos empleada, las enfermedad o lesión que buscan abordar, los dispositivos usados (software, computadora), modelos de aprendizaje automático aplicados y los resultados de cada trabajo, con la finalidad de proporcionar una perspectiva precisa de las propuestas actuales en el campo.

A través de esta breve reseña de algunos de los proyectos realizados sobre la termografía infrarroja como recurso para identificar anomalías mediante inteligencia artificial. Se puede observar las áreas de oportunidad que se presenta al analizar termogramas de miembro inferior con inteligencia artificial, al ser un área poco investigada. Así como, el utilizar un sistema embebido que sea capaz

de detectar en tiempo real anomalías en miembro inferior mediante el uso inteligencia artificial.

Tabla 1. Estado del arte.

Título	Enfermedad / Lesión	Base de datos	Aportación	Técnicas / Dispositivo	Métricas
(Trejo-Chavez et al., 2023)	Síndrome Patelofemoral Bilateral	Base de datos propia (n= 48, donde 24 pertenecen al grupo de control y 24 al grupo experimental)	Valorar la efectividad de identificación de termografía infrarroja en la línea de base de los PFPS, así como después de la aplicación de estrés térmico (calor y frio).	Software: FLIR Tools Cámara: FLIR GF320, termómetro infrarrojo FLUKE 61	ANOVA
(Gómez-Carmona et al., 2020)	Lesiones musculares en miembro inferior en futbolistas	Base de datos propia (n= 24)	Implementar un innovador sistema de prevención de lesiones utilizando termografía infrarroja y evaluar su impacto en la frecuencia de lesiones entre futbolistas profesionales durante la pretemporada.	Software: ThermaCAM Researcher Cámara: ThermaCAM SC660	Pruebas estadísticas: t de Student, Shapiro–Wilk, Wilcoxon, Friedman, χ^2, Cramer V, Somers D
(Côrte et al., 2019)	Lesiones musculares en miembro inferior en futbolistas	Base de datos propia (n= 18)	Analizar el uso de la termografía como técnica complementario en la prevención de lesiones musculares en jugadores de fútbol profesionales.	Software: Flir ResearchIR Max 4 Cámara: FLIR T450sc	Prueba de significancia (p-value)
(Trejo-Chavez et al., 2022)	Lesión de rodilla	Base de datos propia (n total = 48, n control = 24, n experimental = 24). 96 imágenes (80 x80 pixeles)	Metodología basada en termografía infrarroja (TI) y redes neuronales convolucionales (CNNs) para diferenciar automáticamente entre una rodilla sana y una rodilla lesionada	CNN (30 x 30, tamaño de filtro de 5 x 5, 8 filtros, batch size de 25, y 30 épocas)	Exactitud: 97.44%, Precisión:100.00% Recall: 95.12% F1-score: 97.50%
(Cruz-Vega et al., 2020)	Diabetes Mellitus	110 termogramas de una base de datos pública: "Plantar Thermogram Database for the Study of Diabetic Foot Complications"	Analizar el uso de IA y DL para la clasificación de termogramas de pie diabético. Se propone el diseño de una nueva estructura DL llamada Diabetic Foot Thermogram Network (DFTNet).	Diabetic Foot Thermograms Network (DFTNet). (época: 100, batch: 64, capas: 9)	Una sensibilidad de 91.67% y 85.3% en exactitud.
(Farooq & Corcoran, 2020)	Cáncer de mama	DMR - Database for Mastology Research [	Se propone un sistema autónomo de clasificación de tumores de mama a partir de imágenes térmicas de mama empleando redes neuronales de convolución (CNN).	Inception V3 DL (Red neuronal profunda de 48 capas diseñada) (Tamaño 299 x 299 píxeles)	Exactitud: 80 %. Sensibilidad: 83.33%. Precisión: 71.42% Especificidad: 77.77%
(Mohamed et al., 2022)	Cáncer de mama	DMR-IR (Digital Medical Repository - Infrared)	Propone un sistema totalmente automático de detección de cáncer de mama. Utiliza la red U-Net para extraer y aislar automáticamente la zona del pecho del resto del cuerpo y se propone un modelo de aprendizaje profundo de dos clases.	Red U-Net. Modelo basado en CNN de dos clases (resolución = 228x 228 pixeles)	Sensibilidad: 100% Especificidad.: 98.67%. Precisión: 99.33%
(Azariadi et al., 2016)	ECG	Dataset de MIT-BIH Arrhythmia database por PhysioNet.	Desarrolló de un algoritmo de análisis y clasificación de ECG para el diagnóstico de latidos cardíacos. La implementación se realizó en la plataforma integrada Galileo de Intel, que es uno de los dispositivos IoT propuestos por Intel.	Modelos: SVM Dispositivo: Intel Galileo	Precisión: 97%
(Millán Duque, 2021)	Plaga en plantas	Colección de imágenes de Café Robusta. Consta de 1560 imágenes.	Desarrollo de un sistema embebido, con enfoque para clasificar imágenes de manera instantánea, y demostrar su operatividad mediante objetos	Modelos: SVM, KNN y CNN Dispositivo: NVIDIA Jetson Nano	Precisión: SVM: 50.73 %, KNN: 43.1% y CNN: 59.43%

			relacionados con el sector agrícola y/o industrial.		
(Saponara et al., 2021).	Detección de personas	Conjunto de datos I; 775 imágenes térmicas de personas capturadas en varios escenarios. Conjunto de datos II: 800 imágenes. Se trata de imágenes infrarrojas creadas por la empresa FLIR para cámaras térmicas.	Se sugiere un enfoque de aprendizaje profundo para identificar personas en combinación con un algoritmo implementado para la clasificación de distanciamiento social en imágenes térmicas. La técnica propuesta se ha desplegado en un sistema embebido de bajo coste (Jetson Nano) que se compone de una cámara fija.	Modelo : CNN	Dataset I: Exactitud: 95.6% Precisión: 95% Recall: 96% Dataset II: Exactitud: 94.5% Precisión: 94% Recall: 95%

MARCO TEORICO

2.6. Termografía infrarroja

Se define a la termografía infrarroja como el registro de la temperatura de un cuerpo a partir de la radiación infrarroja emitida por la superficie. El producto visible de la termografía se denomina imagen térmica y su registro gráfico se le conoce como termograma (Ammer & Ring, 2019).

La termografía infrarroja proporciona un mapa térmico de la superficie de la piel mediante la medición del calor radiante que emite. Actualmente los dispositivos de termografía infrarroja proporcionan temperaturas precisas de la superficie de la piel ($< 0{,}05°C$) (Diakides et al., 2012).

Las primeras conferencias sobre termografía medica se realizaron en Nueva York en 1963 y en Estrasburgo, Francia en 1966, donde se presentó por primera vez a las imágenes infrarrojas dentro del área médica, y aquellos primeros trabajos mostraron la variedad y el detalle con lo que se puede hacer con la termografía médica (Ammer & Ring, 2019).

Generalmente, el equipo de termografía infrarroja se calibra a ciertos valores de temperatura y, dependiendo del calor emitido por la superficie analizada, el color mostrado en la imagen variará. En la Figura 2 se presenta una termografía infrarroja del tren inferior de un cuerpo humano, con una escala de colores que abarca de 30.0 °C a 35.0 °C. Los tonos fríos (azules y violetas) corresponden a zonas con temperaturas más bajas, mientras que los tonos cálidos (amarillos y rojos) indican regiones con mayor temperatura superficial..

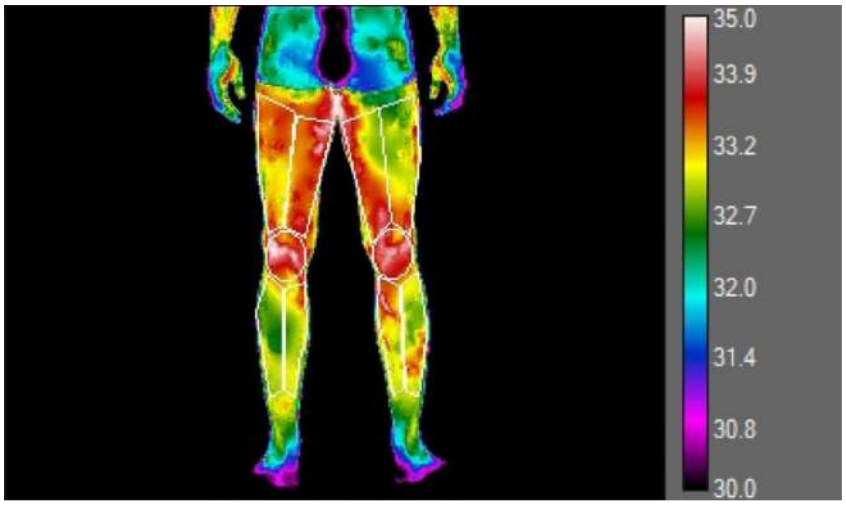

Figura 2. Ejemplo de una imagen termográfica. Obtenida de: (Côrte et al., 2019).

2.6.1. Factores que considerar en el uso de termografía en humanos

La radiación infrarroja comienza en longitudes de onda de 0.7 μm, los equipos generadores de imágenes termográficas funcionan en las longitudes de onda de $3 - 5\ \mu m$ (rango medio) o de $8 - 14\ \mu m$ (rango largo). En estas longitudes de onda, la capacidad de la piel para emitir calor radiante es de 0.98 en donde el 1 es el radiador perfecto, es decir, el cuerpo negro (Diakides et al., 2012).

Cuerpo negro. La idea del cuerpo negro surgió de Kirchhoff, quien lo describió como un cuerpo hipotético que emite radiaciones de todas las frecuencias y, por lo tanto, también debería de absorber todas las radiaciones, teniendo en cuenta que cualquier objeto por encima del 0 absoluto produce radiación térmica. Los cuerpos negros son idealizaciones y ningún objeto tangible puede liberar esta radiación térmica óptima a una temperatura específica (Ammer & Ring, 2019).

La emisividad de un cuerpo negro se describe por tres ecuaciones:

- Ley de Planck: Describe la potencia emisora ($W(\lambda)_b$) de un cuerpo negro en función de su longitud de onda, descrita en la Ecuación 1:

$$W(\lambda)_b = \frac{2\pi hc^2}{\lambda^5 \exp\left(\frac{hc}{\lambda kT}\right) - 1} \qquad [1]$$

Donde h es la constante de Planck ($6.6 \times 10^{-34} J \cdot s$), k es la constante de Boltzmann ($1.4 \times 10^{-23} J \cdot K^{-1}$), t es la temperatura absoluta de un cuerpo negro y c es la velocidad de la

luz.

- Ley de Stefan-Boltzmann: Esta ley explica la energía total emitida por un cuerpo negro W_b. Y es el resultado de integrar la ley de Planck de $\lambda = 0$ a $\lambda = \infty$.

$$W_b = \sigma T^4 \qquad [2]$$

Donde σ es la constante de Stefan-Boltzmann ($56.7 \times 10^{-9} Wm^{-2}K^{-4}$).

- Ley de desplazamiento de Wien: Esta ley explica la longitud de onda a la que la potencia emisiva de un cuerpo es máxima y varia inversamente con la temperatura del cuerpo. Resulta de derivar la ley de Planck respecto a λ: (Ammer & Ring, 2019)

$$\lambda_{max} = \frac{2898}{T} \qquad [3]$$

Emisividad. Los cuerpos negros son un concepto teórico y ningún objeto tangible puede generar esta radiación térmica máxima a una temperatura específica. No obstante, la emisividad real de cualquier objeto se puede determinar multiplicando la energía liberada por dicho objeto por la radiación que un cuerpo negro emitiría. Es un número entre 0 y 1, donde el 1 es el valor que corresponde al cuerpo negro, y caracteriza que tan bien un objeto emite radiación. Por lo tanto, emisividad, ε, representa la proporción entre la energía total radiante emitida por un objeto y la energía que un cuerpo negro libera a la misma temperatura. (Ammer & Ring, 2019; Vollmer & Möllmann, 2011).

La emisividad de un cuerpo humano es importante al momento de usar la termografía en el área clínica, donde tiene un valor de 0.98 (Vollmer & Möllmann, 2011).

2.6.2. Cámara termográfica

El principal objetivo de una cámara infrarroja es transformar la radiación en el rango infrarrojo en una imagen que sea visualmente interpretable, la cual debe mostrar la distribución en dos dimensiones de la radiación que un objeto emite. (Vollmer & Möllmann, 2011).

Los elementos principales de una cámara termográfica son los siguientes:

- Sistema óptico: Es el encargado en formar una imagen utilizando la radiación de longitudes de ondas térmicas de un cuerpo o escena externa.

- Detectores: Convierten la radiación a señales eléctricas proporcional a la radiación que incide sobre ellos.

- Sistema de exploración: Escanea la imagen térmica en un patrón regular a lo largo del detector. Muchas cámaras modernas ya no ocupan este sistema ya que tiene una matriz de detectores que cubren completamente el campo de visión de la cámara.

- Procesador electrónico: Procesa las salidas del detector en una señal de video.

- Unidad de visualización: Genera una imagen visual a partir de la señal de video (Williams, 2009).

2.7. Inteligencia artificial

La inteligencia artificial (IA) nació en la década de los 50´s al surgir preguntas sobre si se podían crear ordenadores que "pensaran". En 1956 se realizó un seminario en Dartmouth College, donde se presentaron diversos temas como el procesamiento del lenguaje natural, software que puede adquirir conocimiento a partir de ejemplos y las redes neuronales (Buduma & Papa, 2022; Kelleher, 2019).

A lo largo de la historia, la IA ha tenido cuatro enfoques diferentes, donde en cada uno se le ha dado una definición a inteligencia artificial. Durante los años 1950 hasta 1980 dominó un enfoque conocido como IA simbólica el cual consistía en automatizar tareas intelectuales comúnmente realizadas por humanos, aunque demostró ser adecuada para resolver problemas lógicos, al momento de resolver problemas más complejos y difusos, como el reconocimiento del habla y clasificación de imágenes, su desempeño fue malo. Por lo que surgió un nuevo enfoque llamado "machine learning" (Buduma & Papa, 2022).

2.7.1. Machine learning

Machine learning o aprendizaje automático es un proceso de búsqueda diseñado para seleccionar la función óptima con el fin de explicar las relaciones entre las características de un conjunto de datos. Este proceso involucra desarrollar y evaluar algoritmos que le permitan a una computadora extraer o aprender patrones o características significativas del conjunto de datos. Surge de la pregunta: ¿Podrá un ordenador ir más allá de "lo que le ordenamos que haga" y aprenderá por sí

mismo a realizar una tarea determinada? (Kelleher, 2019)

Para entender el aprendizaje automático se necesitan comprender tres términos esenciales:

- Conjunto de datos: Su forma más simple de representarse es en una tabla en la que las filas contienen la descripción de un dominio y las columnas las características del dominio.

- Algoritmo: Es un método que examina un grupo de datos y reconoce las tendencias dentro de este.

- Función: Es una asignación determinista, es decir, que para cualquier conjunto particular de entradas, siempre producirá las mismas salidas. (Kelleher, 2019).

Dependiendo de cómo se representan los datos y en cómo se definen las funciones se escoge el tipo de aprendizaje que mejor se acople al objetivo. Se divide en tres diferentes categorías de aprendizaje automático: supervisado, no supervisado y aprendizaje por refuerzo (Kelleher, 2019).

El aprendizaje automático supervisado es el tipo de aprendizaje que más se usa. Se distingue porque cada ejemplo dentro del conjunto de datos se marca con un valor de resultado anticipado, lo que puede ser un proceso complicado y caro. Sin embargo, la ventaja de ello es que, en el proceso de aprendizaje, compara las salidas que produce una función con los resultados etiquetados en el conjunto de datos y se utiliza la diferencia para calcular la exactitud de la función. Se considera supervisado porque el algoritmo recibe información de las etiquetas del conjunto de datos (Chollet, 2017).

El aprendizaje automático no supervisado suele emplearse para clasificar datos. A diferencia del tipo anterior, en este caso, las muestras dentro del conjunto de datos no reciben etiquetas. El algoritmo intenta encontrar características que agrupen muestras en “clusters”, donde las muestras dentro de un mismo grupo deben ser parecidas entre sí, y distintas de las de otros grupos (Kelleher, 2019).

Finalmente, el aprendizaje mediante refuerzo se aplica en actividades de gestión como la manipulación de un juego o la operación de un robot. En este tipo de aprendizaje, un agente recibe datos sobre su entorno y descubre cómo seleccionar acciones para optimizar la recompensa

que recibe (Chollet, 2017).

2.7.1.1. K-NN

K-nearest neighbor (KNN) es una técnica de clasificación de aprendizaje automático supervisado. Clasifica los nuevos datos según su ubicación en relación con los datos más próximos, lo cual es indicado por k, que establece cuántos vecinos cercanos deben ser analizados para definir la categoría de los nuevos datos. (Aized Amin Soofi & Arshad Awan, 2017a; Theobald, 2017).

Por ejemplo, en la Figura 3, los puntos de datos se han clasificado en dos clases, y se observa que hay un nuevo punto de datos, el cual se desconoce a la clase que pertenece. Se puede predecir la clase a la que pertenece basándonos en su posición respecto a los puntos de datos existentes (Theobald, 2017)

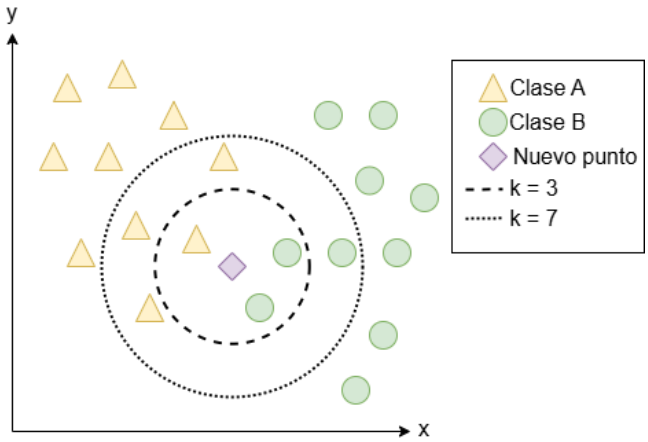

Figura 3. Ejemplo de k- NN para predecir la clase de un nuevo punto de datos. Elaboración propia

Primero, se determina k, para determinar con cuantos puntos de datos se utilizarán para clasificar el nuevo punto de dato. Si se selecciona $k = 3$, es decir, los tres puntos de datos más cercanos, el modelo clasificara el nuevo punto de dato, dentro de la clase B, debido a que hay dos puntos de la clase B y un punto de la clase A. Del contrario, si se selecciona $k = 7$, el modelo clasificará al nuevo punto de datos como Clase a, debido a que hay 4 puntos de la clase A y 3 puntos de la Clase B.

Por lo que la selección de k, es crucial para la determinación de resultados. Generalmente, se selecciona a través de la experimentación, se prueban varios valores de k. Y el valor de k que resulta con la mayor precisión de clasificación es el que se puede seleccionar. Se recomienda evitar

valores de k muy altos ya que resulta en una generalización, además de establecer k en un número impar ya que ayudará a eliminar la posibilidad de un estancamiento estadístico (Hamilton et al., 2020; Theobald, 2017).

Cuando se utiliza K-NN en imágenes, lo que hace es comparar el píxel nuevo con sus píxeles más próximos. Cada píxel se representa mediante n atributos, y la proximidad entre píxeles se determina por la similitud de sus atributos. Esta similitud se define en términos de una métrica de distancia, en este caso la distancia euclidiana. Entre los puntos $X_i = [x_{i1}, x_{i2}, \cdots, x_{in}]$ y $X_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]$ (Hamilton et al., 2020) :

$$Dist\,(x_i x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \qquad [4]$$

Las ventajas de KNN incluyen simplicidad, transparencia, robustez a datos de entrenamiento ruidosos, fácil de entender e implementar (Aized Amin Soofi & Arshad Awan, 2017b).

2.7.1.2. SVM

Support Vector Machine (SVM) se reconoce como uno de los métodos más útiles para abordar desafíos de clasificación y predicción, se utiliza para filtrar datos en una variable objetivo-binaria o multiclase. Se realiza la clasificación de los vectores de datos se lleva a cabo a través de un hiperplano en un espacio (Aized Amin Soofi & Arshad Awan, 2017a; Theobald, 2017).

Un ejemplo, se muestra en la Figura 4, donde el límite de la SVM (B) separa las dos clases (Clase A y Clase B), pero lo hace desde una posición de máxima distancia entre sí y las dos clases de puntos de datos. También se observa una zona gris que denota el margen, el cual es la distancia entre el límite de decisión y el punto de datos más cercano. El margen es una parte clave porque ofrece apoyo adicional para hacer frente a nuevos puntos de datos que pueden infringir el límite de decisión. El límite de la SVM también se puede modificar para ignorar los casos mal clasificados en los datos de entrenamiento utilizando un hiperparametros llamado C. Escoger el valor del hiperparametros C, se puede realizar a través de la experimentación, para regular que tanto se ignoran los casos mal clasificado. Hay que tener en cuenta que si se escoge un margen amplio habrá más errores, y de lo contrario, si se escoge un margen estrecho habrá menos errores(Passos & Rocha, 2022).

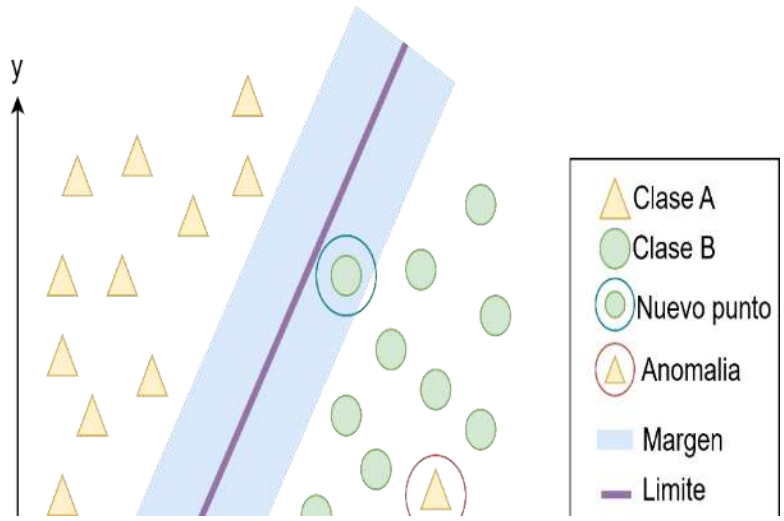

Figura 4. Ejemplo de SVM para predecir la clase de un nuevo punto de datos. Elaboración propia

Además, se puede observar una anomalía, es decir, un valor que no pertenece a su grupo. Afortunadamente la SVM es menos sensible antes estos puntos de datos, minimizando su impacto al momento de ubicar la línea límite.

La clasificación de imágenes en SVM se realiza convirtiendo los pixeles de las imágenes en forma de elementos de un vector, ya sea imagen de color o en grises. El entrenamiento del modelo se realiza desde la etiqueta en clases. Cada vector se coloca en la matriz X y su etiqueta en una matriz Y. Al entrenar el modelo, la SVM identifica un límite de decisión, es decir, un hiperplano; que divide los vectores en las clases correspondientes. Al aumentar la dimensionalidad de los datos dan como resultado hiperplanos de separación que tienen una dimensión menos que los datos. Desafortunadamente, hay hiperplanos que se encuentra cerca de los valores de entrenamiento e impiden generalizar bien (Hamilton et al., 2020).

Al agregar una imagen nueva, se clasifica de acuerdo con el hiperplano que creo la SVM, clasificando en función de qué lado del hiperplano cae la imagen, o vector en estricto sentido, no clasificado cuando se coloca en el espacio de entrada (Hamilton et al., 2020).

La característica más destacada de SVM es su habilidad para abordar diferentes tipos de problemas de clasificación, incluyendo aquellos que presentan alta dimensionalidad y que no se pueden separar de manera lineal. Sin embargo, uno de los principales inconvenientes es que se requiere ajustar varios parámetros importantes de manera adecuada para obtener resultados ideales en la clasificación (Aized Amin Soofi & Arshad Awan, 2017b).

2.7.2. Deep learning

El aprendizaje profundo surgió de la investigación sobre inteligencia artificial y aprendizaje automático. La diferencia principal entre aprendizaje automático y aprendizaje profundo es que en el primero se extrae manualmente las características para ser clasificadas posteriormente por un algoritmo de aprendizaje, mientras que en aprendizaje profundo las redes neuronales reconocen automáticamente las características y las clasifica en etiquetas (Elgendy, 2020; Kelleher, 2019).

Facilita la elección de opciones fundamentadas en la detección y obtención de patrones de datos que representan adecuadamente grupos de entradas complejas. El aprendizaje profundo se concentra en los modelos de redes neuronales profundas, que son formulaciones matemáticas inspiradas en la estructura del cerebro (Kelleher, 2019). La palabra "profundo" hace referencia al número de capas que conforma el modelo (Chollet, 2017).

La red neuronal más simple se le conoce como perceptrón, el cual consiste en una sencilla neurona y funciona de manera similar a una neurona biológica. La neurona es la unidad funcional del cerebro, una parte del cerebro del tamaño de un grano de arroz contiene alrededor de 10,000 neuronas, el cual nos permite experimentar el mundo a nuestro alrededor (Buduma & Papa, 2022; Elgendy, 2020).

La neurona está diseñada para capturar datos procedentes de otra neurona, procesar la información de una forma única y mandar el resultado a otra neurona. Recibe la información a través de las dendritas, que son una estructura en forma de antena, cada conexión se refuerza o se debilita dependiendo de cuán frecuentemente se utilice, y la fuerza de conexión determina cuánta influencia tiene la entrada sobre la salida de la a neurona. La fuerza de cada conexión se suma en el cuerpo de la célula y se convierte en una nueva señal que se extiende por el axón y se envía a otras neuronas (Buduma & Papa, 2022).

Esta información se intenta replicar en un modelo artificial, el cual surgió en 1943 por Warren S. McCulloch y Walter H. Pitts. El modelo se describe en la Figura 5:

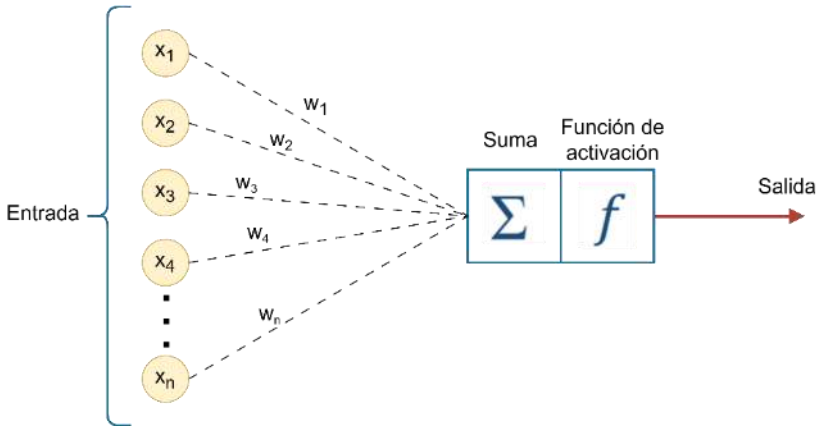

Figura 5. Estructura de perceptrón simple. Elaboración propia.

Donde:

- Vector de entrada (x) : Es la característica que alimenta la neurona, y suele representarse como un vector de entradas $x_1, x_2, \dots, x_n$.

- Vector de pesos (w): A cada valor del vector de entradas se le asigna un valor al peso, su función es distinguir los diferentes puntos de entrada, se representada como $w_1, w_2, \dots, w_n$.

- Función: Son los cálculos realizados dentro de la neurona para modular las señales de entrada: la suma ponderada (z) y la función de activación. En muchos casos estos cálculos incluyen el bias (b), el cual es una constante. Esta función está dada por:

$$z = \sum_{i=0}^{n} w_1 x_1 \qquad [5]$$

- Salida (y) : Esta controlada por el tipo de función de activación que se escoja para la red. Este representa la predicción del perceptrón. En redes neuronales más complejas esta salida puede conectarse a otra neurona. Se puede expresar de la siguiente (Buduma & Papa, 2022; Elgendy, 2020):

$$y = f(\boldsymbol{x} \cdot \boldsymbol{w} + b) \qquad [6]$$

La neurona tiene un proceso de dos etapas. La primera es el cálculo de la suma ponderada de las entradas de las neuronas (z). La segunda etapa es la llamada función de activación, que mapea los resultados de la suma ponderada al valor de la salida final de la neurona (Elgendy, 2020).

La importancia de la función de activación se debe a que muchas de las aplicaciones en la que se usan las redes neuronales no son lineales, sobre todo al utilizar redes neuronales más complejas como el perceptrón multicapas mostrado en la Figura 5 y compuesto por (Elgendy, 2020).

2.7.3. CNN (Convolutional Neural Network)

Una red neuronal convolucional (CNN) se compone de una serie de bloques que incluyen capas de convolución y capas de agrupamiento, seguidos de una o varias capas totalmente conectadas (FC), y en última instancia, una capa final.. Estas capas son etapas de extracción de características, reducción de dimensionalidad y finalmente las capas de clasificación. La capa de convolución es el núcleo (Ghosh et al., 2020; Teuwen & Moriakov, 2020). Además del estructura de las capas, el rendimiento de la CNN está influenciado por otros factores como la función de activación, función de pérdida, método de normalización, regularización, optimización y velocidad de procesamiento (Ghosh et al., 2020).

La CNNs se han aplicado ampliamente a diversos campos, entre ellos está la visión por computadora, procesamiento del habla, reconocimiento facial, etc. La estructura está inspirada por las neuronas del cerebro humano y del cerebro animal, específicamente del cerebro del gato (Alzubaidi et al., 2021).

El principal beneficio de una CNN es que identifica automáticamente las características más importantes sin la supervisión del humano. En el entorno de visión por computadora, los beneficios que brinda son:

- Es la característica de reparto de pesos, el cual reduce el número de parámetros entrenables de la red, lo que ayuda a mejorar la generalización y evita el sobreajuste.
- Es más fácil la implementación de redes a gran escala (Alzubaidi et al., 2021).

Para entender cómo está compuesta la arquitectura de una CNN, se tiene que aclarar los siguientes conceptos:

Kernel. Se define como una matriz de números o valores discretos, donde cada valor se conoce como peso del núcleo. Al comenzar el proceso de entrenamiento de un modelo CNN, los pesos del kernel se otorgan de forma aleatoria. A lo largo de cada ciclo de entrenamiento, los pesos

se modifican, permitiendo que el kernel aprenda a identificar las características más significativas (Alzubaidi et al., 2021; Ghosh et al., 2020).

Operación convolucional. La entrada de un modelo CNN es una imagen multicanal. Para entender la operación de convolución, se toma una imagen que está en escala de grises siendo una matriz 4x4 y un kernel de tamaño 2 x 2 con pesos inicializados aleatoriamente, como se observa en la Figura 6.

a)

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

b)

0	1
-1	2

Figura 6. a) Imagen en escala de grises de tamaño 4x4. b) Kernel de tamaño 2x2

La operación de convolución se explica si se imagina que el kernel se desliza sobre toda la imagen completa tanto horizontal como verticalmente y a lo largo de su recorrido, se lleva a cabo un producto punto entre el kernel y la imagen de entrada, donde se multiplican los valores correspondientes. Luego, todos estos valores se suman para producir un solo valor que aparecerá en el mapa de características de salida. Este procedimiento continúa hasta que el kernel ya no pueda deslizarse más, se ilustra en la Figura 7.

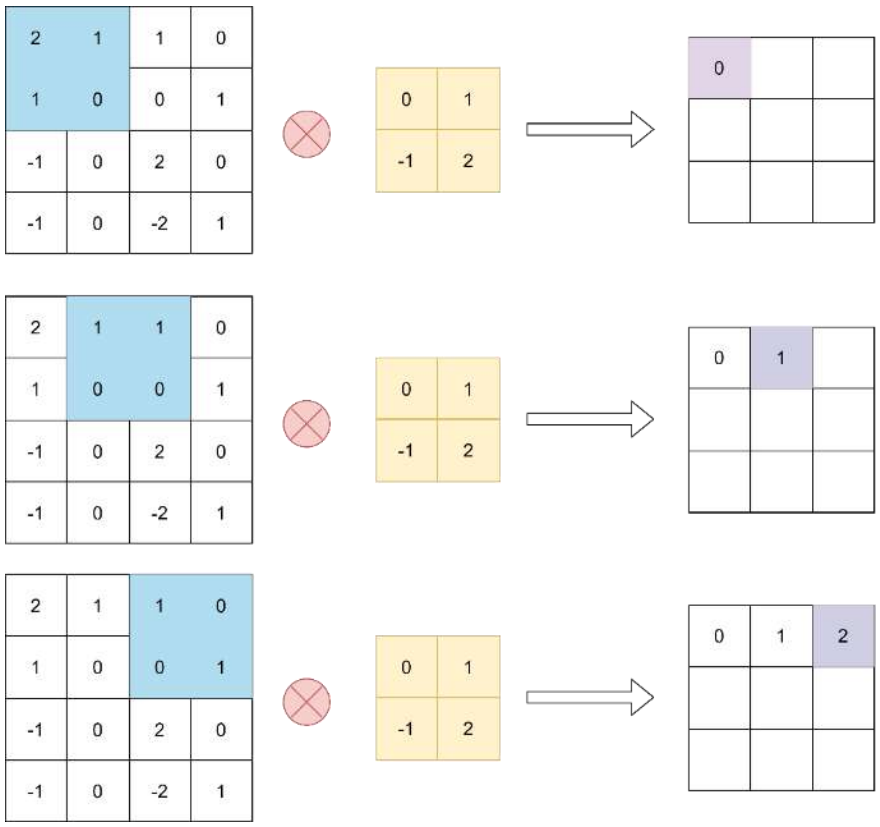

Figura 7. Ilustración de la operación convolución. La matriz 4x4 representa la imagen en escala de grises. La matriz 2x2 el kernel. El producto punto se realiza entre la imagen y el kernel. El resultado, denominado mapa de características, es la matriz de 3x3.

La arquitectura de una CNN se puede considerar un combinación de dos partes: la extracción de características y la clasificación, como se observa en la Figura 8. Tanto la capa de convolución como la capa de agrupación perteneces a la etapa de extracción de características. Mientras que las capas totalmente conectadas pertenecen a la etapa de clasificación, en el cual se asigna una probabilidad de cada clase a la imagen de entrada (Ghosh et al., 2020).

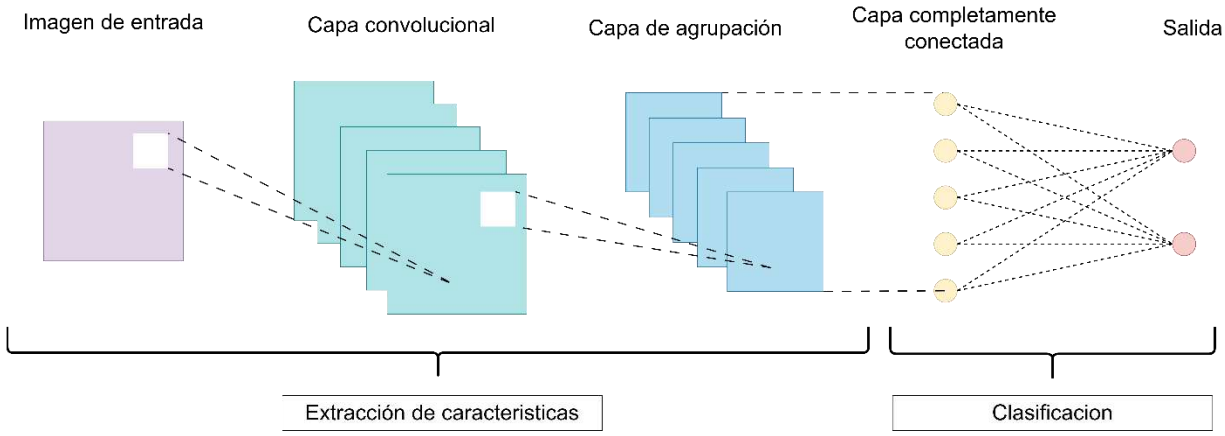

Figura 8. Esquema de la arquitectura de una red neuronal convolucional

Capa convolucional. El objetivo de esta capa es aprender representaciones de características de la entrada. Está formada por múltiples filtros de convolución, conocidos como kernel, que se utilizan para crear diversos mapas de características. La imagen de entrada, se aplica a estos filtros (kernel) a fin de producir el mapa de características de salida (Alzubaidi et al., 2021; Ghosh et al., 2020).

Capa de agrupación. Se aplica para reducir los mapas de características que se generan a partir de las capas de convolución, lo que significa que transforma los mapas de características grandes en versiones más pequeña. Al disminuir el tamaño de los mapas, se preservan las características más significativas en cada fase del agrupamiento. La operación de agrupación se lleva a cabo definiendo el tamaño de la región agrupada y el paso de la operación (Ghosh et al., 2020). Existen varios tipos de métodos de Pooling que pueden utilizarse en varias capas de Pooling. Estas incluyen: agrupación en árbol, agrupación cerrada, agrupación media, agrupación mínima, agrupación máxima, agrupación media global (GAP) y agrupación máxima global (Alzubaidi et al., 2021).

Capa totalmente conectada. Es la última parte de la CNN, puede componerse de una o varias capas. Se encarga de recibir datos de la última capa de convolución o de la última capa de agrupamiento y produce la salida final. Se utiliza para la clasificación. Su principal función es la clasificación, ya que está compuesta por capas completamente conectadas, donde cada neurona de una capa establece conexiones con cada neurona de la capa anterior. La capa final de estas capas se utiliza como salida, actuando como un clasificador (Ghosh et al., 2020).

Función de activación. Su función principal es asignar la entrada a la salida, donde el valor de entrada se obtiene calculando la suma ponderada de la entrada de la neurona y añadiendo el sesgo, si es que hay (Ghosh et al., 2020). La función de activación se coloca al final de cada perceptrón para decidir si la neurona se activa o no. El objetivo principal es introducir la no linealidad en la red y restringir el valor de salida a un cierto valor finito. Algunas de las funciones de activación que se usan son las siguientes (Elgendy, 2020):

$$Output = \begin{cases} 0, & \boldsymbol{x} < 0 \\ x, & \boldsymbol{x} \geq 0 \end{cases} \qquad [7]$$

Como se mencionó anteriormente, el desempeño de la CNN está influenciado por varios factores que ayudan a disminuir el tiempo necesario para el entrenamiento y aumentar la precisión del

modelo. Tales factores como: función de pérdida, regularización, optimización y velocidad de procesamiento se incluyen dentro del proceso de entrenamiento.

Función de pérdida. Se determina el error de predicción que surge del modelo CNN cuando se trabaja con las muestras de entrenamiento. Este error informa a la red cuán lejos está su predicción de la correcta, con el propósito de optimizar dicho error durante el proceso de aprendizaje. Para calcular el error, se utilizan dos elementos: la salida estimada del modelo (predicción) y la salida real (etiqueta) (Ghosh et al., 2020).

Regularización. Un modelo CNN tiene como principal reto adaptarse adecuadamente a entradas nuevas, esta capacidad es conocida como generalización. El principal problema de lograr una buena generalización es el sobreajuste (Ghosh et al., 2020). Cuando se dice que un modelo está sobre-ajustado, significa que funciona apropiadamente con los datos de entrenamiento, pero falla con datos de prueba o desconocidos. Un modelo se considera infra-ajustado si no ha aprendido adecuadamente de la información de entrenamiento. Por otro lado, si el modelo funciona bien tanto en los datos de entrenamiento como en los de prueba, se le denomina "just-fitted" (Alzubaidi et al., 2021; Ghosh et al., 2020).

La forma más fácil de evitar el sobreajuste es entrenar el modelo con una gran cantidad de datos con diversas variedades. Esto puede conseguirse mediante el aumento de datos o data augmentation.

Data augmentation. Un problema que surge cuando un modelo tiene pocas muestras de entrenamiento es el sobreajuste, volviendo al modelo demasiado específico para ese conjunto de datos, lo que impide entrenar un modelo que pueda generalizar nuevos datos. Si los datos fueran infinitos, el modelo estaría expuesto a todas las características posibles de la distribución de datos. Para ello, se utiliza la técnica de aumento de datos, generalmente usada cuando se procesan imágenes con aprendizaje profundo (Ammer & Ring, 2019).

El aumento de datos consiste en generar más datos para el entrenamiento a partir de las muestras de entrenamiento existentes (pertenecientes únicamente al conjunto de datos de entrenamiento), mediante una serie de transformaciones que crean imágenes nuevas, que se utilizan para el proceso de entrenamiento. Estas imágenes siguen estando interrelacionadas ya que proceden de la misma

muestra de imágenes, no se crea información nueva, sino información existente remezclada. El objetivo de esta técnica es que el modelo cuando se esté entrenando nunca vea la misma imagen dos veces (Ammer & Ring, 2019) (Ghosh et al., 2020). Hay varias operaciones de aumento de datos disponibles, tales como recortes, rotaciones, volteos, traslaciones, ajuste de contraste, escalado, etc. (Ghosh et al., 2020).

2.7.4. Grad-CAM

Para entender cómo funciona una CNN o qué decisiones toma al realizar tareas como la clasificación, se han desarrollado varios métodos de explicación visual. Que tienen como objetivo generar confianza al construir modelos transparentes que expliquen por qué predicen lo que predicen, además de identificar los modos de fallo del modelo, y por +ultimo, enseñar a una maquina a tomar mejores decisiones (Selvaraju et al., 2017).

Una de las técnicas propuesta es el Mapeo de Activación de Clases (CAM), el cual identifica regiones discriminativas utilizadas por una tipo especifico de CNNs de clasificación de imágenes, ya que requiere de propiedades específicas del modelo para poder aplicarse, como no contener ninguna capa totalmente conectada, y estar sustituida por una capa GAP (Global Average Pooling) y una estructura lineal, además es necesario para reentrenar el modelo (Selvaraju et al., 2017).

Funciona extrayendo el mapa de características de la última capa convolucional, que se pasa por GAP. Se obtiene un vector de escalares correspondiente al valor de cada mapa de características y, dependiendo de la clase de la imagen de entrada, se realiza una suma de la multiplicación de cada mapa de características por el peso correspondiente a la clase, obteniendo un mapa de calor correspondiente a las principales áreas o características de la imagen.

(Selvaraju et al., 2017) diseñaron una nueva técnica, es una generalización de CAM, la cual se puede aplicar a más tipos de CNN: 1) CNN con capas totalmente conectadas, (2) CNN utilizadas para resultados estructurados,(3) CNN utilizadas en tareas de aprendizaje por refuerzo. Además, no requiere que la CNN tenga cambios arquitectónicos ni reentrenamiento.

Grad-CAM utiliza la información de gradiente que llegan a la última capa convolucional de la CNN para determinar qué tan relevante es de cada neurona para una decisión de interés. Una breve explicación de cómo funciona se realiza a continuación.

Para obtener el mapa de calor de Grad-CAM, se necesita obtener el mapa de características ponderado, donde a_k^c es el peso que se le aplica al mapa de características A^k, donde c es la clase a la que pertenece y k es el número de mapa de características de la capa convolucional:

$$\sum_k a_k^c A^k \qquad [8]$$

Para calcular los pesos a_k^c, en Grad-CAM se utiliza el gradiente de la puntuación para la clase c, y^c (antes del softmax). Donde cada mapa de características A^k, es una matriz de elemento de tamaño mxn, el cual depende de las capas anteriores, y A_{ij}^k es el valor de un elemento arbitrario en el mapa de características, como se muestra en la Figura 9. Al usar back-propagation, se puede encontrar la derivada de y^c respecto a cada elemento del mapa de características $\frac{\partial y^c}{\partial A_{ij}^k}$.

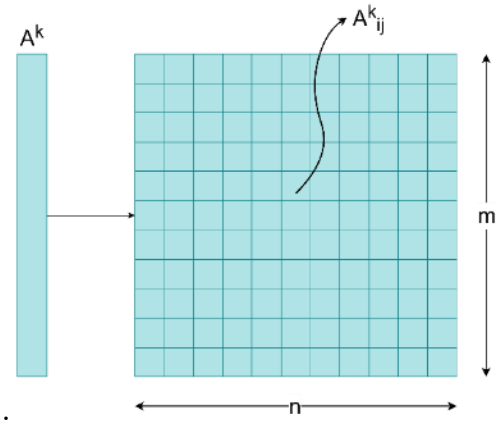

Figura 9. Diagrama de mapa de características (A^k), donde A_{ij}^k es cada elemento, y mxn es el tamaño del mapa de características.

El gradiente sirve para obtener informacion de que tanto varia y^c con respecto a pequeños cambios en cada elemento, Los gradientes se ponen a cero para todas las clases excepto la clase deseada. Al obtener el promedio de todos los gradientes, se puede saber que tan importante es el mapa de características

$$\sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \qquad [9]$$

Y se divide entre el número total de elementos Z. Con esto se obtiene el peso de cada mapa de características, es similar al global Average Pooling.

$$a_k^c = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \quad [10]$$

Se sigue el mismo procedimiento para cada mapa de características de la capa convolucional. Después, cada peso se multiplica por su mapa de características. Finalmente, se realiza la sumatoria de cada multiplicación. Obteniendo el mapa de características ponderado ($\sum_k a_k^c A^k$).

El siguiente paso es el ReLU que ayuda a eliminar los números negativos, volviéndolos cero. Aplicamos un ReLU a la combinación lineal de mapas ($\sum_k a_k^c A^k$) porque sólo interesan las características que influyen positivamente en la clase de interés. Debido a que es probable que los píxeles negativos pertenezcan a otras categorías de la imagen.

$$L_{Grad-CAM}^c = ReLU\left(\sum_k a_k^c A^k\right) \quad [11]$$

2.8. Visión artificial

La visión artificial inició en la década de 1970, se creía que sería fácil resolver el problema de la entrada visual para dar paso a la resolución de problemas más difíciles como el razonamiento y el planteamiento de alto nivel. Se distinguía por recuperar la estructura dimensional del mundo real en imágenes y así tener una compresión completa de la escena. Sin embargo, la visión artificial no fue un camino fácil para resolver estos problemas debido a la complejidad que presentan los datos visuales (Szeliski, 2022).

La visión artificial está basada en la visión humana, donde esta última es el nivel más alto de un sistema de visión. Un ejemplo es el de la Figura 10, donde se muestra cómo funciona un sistema de visión humana, en el cual se interpreta la imagen de un perro. Los ojos serían el dispositivo de detección, al ver la imagen nosotros entendemos directamente que la imagen consiste en varios perros, y podemos clasificar y detectar los objetos dentro de la imagen, por que conforme han pasado los años nos hemos entrenado para poder diferenciar un perro, este proceso de entendimiento de la imagen lo realiza el cerebro, el cual es el dispositivo de interpretación.

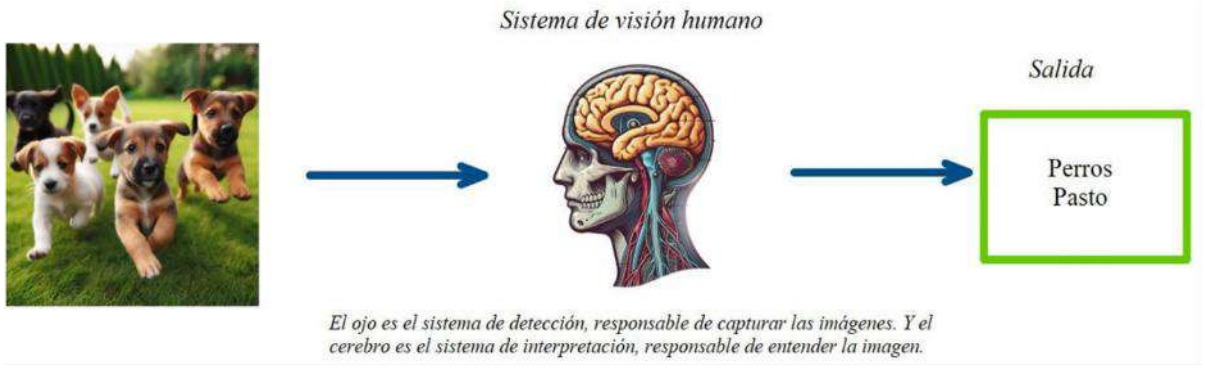

Figura 10. Sistema de la visión human. Adaptado de Generador de Imágenes de Microsoft Designer por https://www.bing.com/images/create?toWww=1&redig=A3AF1EFD160148DC9CEA919341559A22

Este mismo proceso sucede con las computadoras, las cuales se entrenan para aprender e identificar objetos. En la Figura 11 se muestra el sistema visual de inteligencia artificial que se ha ido desarrollando a lo largo de los años para imitar el sistema de visión humana, donde la cámara es el sistema de detección y el algoritmo imita la función del cerebro en interpretar y clasificar el contenido de la imagen (Elgendy, 2020).

Sin embargo, el sistema visual humano tiene una gran ventaja sobre la computadora debido a que el cerebro posee más 10^{10} de neuronas. Si cada neurona actúa como un microprocesador entonces se tiene un ordenador muy potente por lo que las tareas de procesamiento que tiene el sistema visual humano no se pueden replicar por los sistemas actuales diseñados por el humano (Davies, 2017).

Definiendo así, a la visión artificial como una ciencia que percibe y entiende el mundo a través de imágenes y videos construyendo un modelo físico para que los sistemas de IA puedan trabajar sobre ellos. Es una disciplina que adquiere, procesa y analiza imágenes del mundo real con el objetivo de producir información que pueda ser tratada por un sistema de (Borrella Petisco, 2022; Elgendy, 2020).

Figura 11. Sistema de visión por computador. Adaptado de Generador de imágenes de Microsoft Designer por

https://www.bing.com/images/create?toWww=1&redig=A3AF1EFD160148DC9CEA919341559A22

2.8.1. Componentes de un sistema de visión artificial

Un sistema de visión artificial está formado por los siguientes componentes:

- Fuente de luz o Iluminación: Su objetivo es controlar la forma en que el objeto se va a capturar por medio de la cámara, si hay una buena iluminación esto simplificaría el procesamiento de imagen debido a que los sistemas de visión artificial crean imágenes a través de la luz reflejada en el objeto

- Cámara: Es el sistema de detección, el lente de la cámara junto con el sensor de imagen también determinará la calidad con la que se captura la imagen. El trabajo del sensor de imagen es capturar la luz y convertirla en un ruido de imagen digital, sensibilidad y rango dinámico.

- Computadora: Es el sistema de procesamiento, recibe las imágenes e implementa algoritmos dependiendo del análisis que se quiera realizar (Cognex Corporation, 2018).

El proceso básico que realiza este sistema de visión artificial es el siguiente:

- Adquisición y digitalización: Se encarga de capturar la imagen y convertirla a un formato digital.

- Procesamiento: Se encarga de filtrar la información necesaria y de eliminar lo que no es de interés, mejora la calidad de la imagen, mediante la corrección de iluminación, normalización, eliminación del ruido, etc.

- Segmentación: Se extrae y se reconoce cada objeto de la imagen, las técnicas más usadas son la umbralización de regiones y la detección de bordes.

- Extracción de características: Se utiliza para identificar y extraer las características claves del objeto para describir y encontrar patrones en las imágenes.

- Identificación de objetos: Se utilizan modelos de aprendizaje profundo para la clasificación de objetos, donde se decide a que categoría pertenece cada objeto de la imagen (Borrella Petisco, 2022; Cognex Corporation, 2018).

2.8.2. Procesamiento de imágenes

El propósito del procesamiento de imágenes es preparar imágenes para modelos de aprendizaje profundo y facilitar el proceso de análisis y cálculo. La adquisición de imágenes usualmente suele tener ruido, para poder alimentar una red neuronal necesita estandarizarse.

Es necesario para mejorar el rendimiento o trabajar con limitaciones de la red neuronal, alguna de las técnicas usadas son la conversión de imágenes a escala de grises, la normalización, aumento de datos, cambio de tamaño de imagen (Elgendy, 2020).

- Imágenes a escala de grises: Se utiliza para ahorrar en complejidad computacional, en vez de utilizar tres matrices para representar imágenes de color, solo se ocupa una matriz.

- Cambio de tamaño de imagen: Uno de los problemas principales de las redes neuronales es que ocupan que todas las imágenes sean del mismo tamaño, por lo que se tiene que redimensionar la imagen para que pueda entrar a la red neuronal.

- Normalización: Es el proceso de ajustar el rango de pixeles de una imagen para que este dentro de los valores específicos, hay imágenes cuyos pixeles tienen valores de 0 a 255, y otras de 20 a 200. Por lo que es preferible normalizar los valores de los pixeles en el rango de 0 a 1, así aumentara el rendimiento de aprendizaje (Elgendy, 2020).

Imagen. Para comenzar con el procesamiento de imagen se necesita definir el concepto de imagen y sus variantes. Las imágenes más sencillas que se conocen son imágenes en blanco y negro, las cuales se representan como una función $f(x, y)$ donde $x\ e\ y$ son las coordenadas espaciales y el valor f en (x, y) es proporcional a la luminosidad de la escena en ese punto. Si la imagen es multiespectral, ese punto indica el brillo de la escena en la banda espectral correspondiente (Borrella Petisco, 2022; Petrou & Petrou, 2010).

Imagen digital. Una imagen digital es una imagen discretizada, se representa con una matriz de enteros de 2D, o en una serie de matrices 2D donde cada una representa una banda de color (Petrou & Petrou, 2010).

Cada uno de los puntos de una imagen se denomina píxel, y se identifican por la columna y la fila

a la que pertenece. Los pixeles cuantifican la intensidad de la imagen en ese punto y se representa como un valor numérico. También influyen en la resolución espacial o en la nitidez de la imagen (Borrella Petisco, 2022). Una imagen digital se representa de la siguiente manera:

$$f(x,y) = \begin{matrix} f(1,1) & \cdots & f(1,N) \\ \vdots & \ddots & \vdots \\ f(N,1) & \cdots & f(N,N) \end{matrix} \qquad [\,12\,]$$

Imagen en escalas de grises. Este tipo de imágenes contienen valores de gris que van desde el color negro al cual se le asocia el 0 y al color blanco siendo el valor de $2^n - 1$, donde n es el número de cortes diferentes. Por ejemplo, si $n = 5$, entonces el color negro es 0 y el color blanco es $= 2^5 - 1 = 32$, por lo que se considera que hay 32 diferentes intensidades de gris. La mayoría de las imágenes a escala de grises están en formato de 8 bits y los valores de los pixeles van de 0 a 255 (Borrella Petisco, 2022; Vega Mancilla, 2022).

Imagen multiespectrales. Las imágenes multiespectrales son una función con tantos componentes como bandas que tenga la imagen. Las imágenes que vemos diariamente están compuestas por tres bandas, el más conocido es el RGB (Red, Green, Blue) como se muestra en la Figura 12. Y se representa de la siguiente manera (Borrella Petisco, 2022):

$$f(x,y) = (red(x,y), green(x,y), blue(x,y)) \qquad [\,13\,]$$

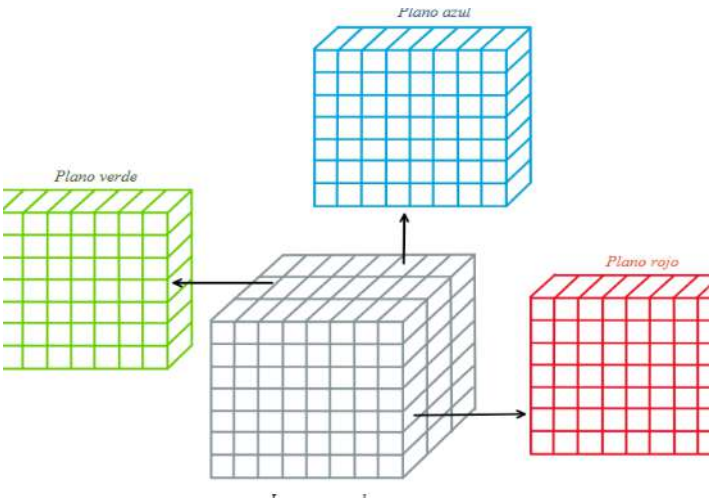

Figura 12. Bandas de una imagen RGB. Elaboración propia.

2.9. Sistema embebido

Es una computadora basada en un microprocesador con software que realiza funciones específicas

dentro de un sistema compuesto o en un sistema independiente (Millán Duque, 2021).

Se caracterizan por capacidad de procesamiento en tiempo real en aplicaciones específicas. Ya que son diseñadas específicamente para una aplicación o para funciones concretas. Estas aplicaciones deben ejecutarse dentro de las limitaciones del sistema embebido los cuales son la memoria y recursos informáticos estrictamente limitados, es decir, el hardware donde se ejecuta el programa restringe la cantidad de memoria disponible y la frecuencia del procesador (Katsaragakis et al., 2020; Sánchez Granja & Asanza, 2021).

2.9.1. Jetson Nano

El sistema embebido NVIDIA Jetson Nano es parte de la última generación de sistemas embebidos con la capacidad de procesamiento paralelo en GPU. Este producto está destinado a crear Internet de las cosas (IoT). La placa consta de un CPU QUAD $-$ core ARM A57 de 1.43 GHz y un GPU de 128 núcleos, memoria de 4 GB 64 $-$ bit $LPDDR4$ 25.6 GB/s y un almacenamiento microSD. Y posee un USB 2.0 Micro $-$ B, 4xUSB 3.0, HDMI 2.0 (Saponara et al., 2021).

Este sistema embebido se especializa en tareas de inteligencia artificial, puede ejecutar varias redes neuronales avanzadas, específicamente en la detección de objetos y la clasificación de imágenes, se puede ejecutar Pandas, Numpy, Pytorch Tensor Flow y Keras (Kurniawan, 2021; Millán Duque, 2021; Selvaraju et al., 2017) . Además, emplea librerías aceleradoras TensorRT, que incluyen paquetes Jetpack. Jetson nano se puede usar para aplicaciones en tiempo real (Selvaraju et al., 2017).

2.10. Sistema muscular

El movimiento del cuerpo humano resulta de la acción de los músculos al contraerse y relajarse, que constituyen entre un 40% y un 50% del peso total de un adulto. Las funciones de los músculos consisten en la fuerza muscular mediante la conversión de energía química en energía mecánica, en estabilizar la postura, regular el volumen de los órganos y en producir calor.

Se clasifica en tres tipos de tejidos musculares: músculo esquelético, músculo cardíaco y músculo liso. El tejido muscular esquelético está anclado a los huesos del esqueleto, el músculo cardíaco reside en el corazón y forma parte de su estructura, mientras que el músculo liso se localiza en las

paredes de los vasos sanguíneos, las vías respiratorias y en la mayoría de los órganos dentro de la cavidad abdominopélvica (T. et al., 2011).

El músculo esquelético constituye la mayor parte del tejido muscular en el cuerpo humano. Está compuesto por múltiples fibras con diámetros que oscilan entre 10 a 80 μm. Las fibras se extienden a lo largo de toda la longitud del músculo y usualmente solo reciben inervación de una terminación nerviosa que facilita su contracción y relajación (Hall, 2015). Los músculos en contracción contribuyen a la distribución de la temperatura en la superficie corporal debido a que el trabajo muscular es la fuente más importante de calor metabólico, por lo que los espasmos musculares pueden hacerse visibles en regiones de alta temperatura (Diakides et al., 2012).

2.10.1. Miembro inferior

El miembro inferior está compuesto por cinco segmentos: la cintura pélvica y la parte libre; este último dividido en 4 regiones: muslo, rodilla, pierna y pie. Su función principal es la bipedestación, sus características anatómicas hacen que tenga una finalidad de brindar apoyo, estabilidad y potencia (Dufour, 2012; Latarjet & Liard, 2004).

2.10.2. Lesiones comunes en miembro inferior

Las lesiones musculares se generan debido al esfuerzo excesivo de estos, siendo las más comunes, con una aparición que varía entre el 10% y el 55% de todas las lesiones. Entre estas, contusiones y distensiones representan el 90% de los casos de daño muscular.. La contusión de un músculo ocurre cuando el músculo es sometido a una fuerza repentina que generalmente predomina en deportes de contacto, mientras que la distención muscular predomina en los deportes que requieren cambios de dirección y saltos (Jiménez Díaz, 2006; Rafaela Rosas, 2011).

Las lesiones musculares se clasifican en dos, el primero se debe a lesiones producidas por un choque directo y se incluyen las contusiones, mientras que el segundo son lesiones ocasionadas por un traumatismo intramuscular:

- Lesiones musculares extrínsecas: Son traumatismos que generalmente afectan la extremidad inferior, son provocados por el choque de la masa muscular contra una superficie dura. Si recibe el impacto sobre un músculo mientras este se encuentra en fase de contracción, la lesión

afectas las fibras superficiales, mientras si fue en la etapa de relajación las lesiones más profundas son las afectadas. Su severidad depende de la fuerza de contacto y la fase de contracción del músculo.

- Lesiones musculares intrínsecas: Ocurren por un estiramiento excesivo al momento de la contracción del músculo (Jiménez Díaz, 2006; Muñoz Ch. et al., 2018).

Las lesiones más frecuentes en la extremidad inferior en el área del tobillo y del pie son la rotura del tendón de Aquiles y las lesiones del ligamento del tobillo. Mientras que en la rodilla son los esguinces y las lesiones en los ligamentos de la rodilla, tendinitis rotuliana y lesionas meniscales (Rafaela Rosas, 2011).

2.10.3. Síndrome Patelofemoral

El síndrome de dolor patelofemoral (SDPF) se refiere al malestar que se siente en la parte frontal de la rodilla y en torno a la rótula, el cual puede ser unilateral o bilateral (Instituto Nacional de Rehabilitación, 2021; Waryasz & McDermott, 2008).Las actividades que contribuyen al dolor son correr, permanecer mucho tiempo sentado, subir escaleras, saltar y ponerse en cuclillas. El SDPF también se denomina rodilla de corredor y síndrome de dolor anterior de rodilla (David Y. Gaitonde et al., 2019; Lankhorst et al., 2012).

El término dolor anterior de rodilla abarca todas las afecciones relacionadas con la parte frontal de esta. A excepción de las molestias en la parte anterior de la rodilla causadas por trastornos intraarticulares, síndromes de plica, enfermedad de Sinding-Larsen-Johansson, enfermedad de Osgood-Schlatter, bursitis o tendinitis, neuroma y otras patologías poco frecuentes, los pacientes que presentan dolor en la parte anterior de la rodilla pueden ser diagnosticados con SDPF (Lankhorst et al., 2012). Los factores de riesgo incluyen el uso excesivo, traumatismos, disfunción muscular, tensión en los ligamentos laterales, hipermovilidad rotuliana y poca flexibilidad de los cuádriceps (Dixit et al., 2007).

La rótula, el mayor hueso sesamoideo del ser humano, funciona para mejorar la eficacia de la flexión y proteger la articulación tibiofemoral. Este hueso actúa como una palanca en la pierna, reduciendo la fuerza que el cuádriceps necesita para extender la pierna en la articulación de la rodilla. La articulación femororrotuliana se mantiene estabilizada por el cuádriceps, el vasto

medial oblicuo, el tendón rotuliano, el ligamento femororrotuliano medial, los retináculos medial, el ligamento patelotibial medial, y lateral oblicuo, la banda patelotibial, las bandas epicondilopatelares y el retináculo externo (David Y. Gaitonde et al., 2019; Waryasz & McDermott, 2008).

La fuerza del cuádriceps se evalúa mediante el ángulo que se forma desde la espina ilíaca anterosuperior hasta el centro de la rótula y el tubérculo tibial, conocido como ángulo Q. Este ángulo mide aproximadamente 14 grados en los hombres y 17 grados en las mujeres. Dado que fuerzas laterales en la rótula aumentan con ángulos Q más grandes, se ha creído durante mucho tiempo que un ángulo Q elevado incrementa el riesgo de SDPF; sin embargo, en investigaciones más recientes no se ha demostrado que un ángulo Q alto sea una causa que contribuya claramente al SDPF(David Y. Gaitonde et al., 2019).

2.10.4. Inflamación y aumento de temperatura

La inflamación es un fenómeno complejo el cual puede ser desencadenado por diferentes lesiones tisulares. Ocurren una serie de cambios químicos y celulares que son destructivos para el tejido circundante. Bajo circunstancias normales, el proceso termina cuando se produce la cicatrización.

Ocurren una serie de procesos fisiológicos que afectan el tejido. Inicialmente, ocurre una breve constricción arteriolar, seguido de una prolongada dilatación de las arteriolas, capilares y vénulas, El incremento inicial de flujo sanguíneo causado por la dilatación de los vasos sanguíneos se vuelve lento y los leucocitos se acumulan en el endotelio vascular (Diakides et al., 2012).

El incremento de la permeabilidad de las proteínas plasmáticas provoca la formación de exudados, que son absorbidos lentamente por el sistema linfático. El fibrinógeno, remanente de la reabsorción, se polimeriza parcialmente en fibrina. El aumento de la permeabilidad en la inflamación se atribuye a la acción de varios mediadores, como histaminas, cininas y prostaglandinas. El proceso final se manifiesta como hinchazón causada por los exudados, enrojecimiento y aumento del calor en la zona afectada como consecuencia de la vasodilatación y el aumento del flujo sanguíneo. La pérdida de funcionalidad y el dolor acompañan a estos signos visibles (Diakides et al., 2012).

El trabajo muscular es la fuente más importante de calor metabólico. Por lo tanto, los músculos en

contracción contribuyen a la distribución de la temperatura en la superficie corporal de los deportistas. Las condiciones patológicas, como los espasmos musculares o los puntos gatillo miofasciales, pueden hacerse visibles en regiones de temperatura elevada (Diakides et al., 2012).

Las lesiones musculares agudas también pueden reconocerse por zonas de temperatura elevada debidas a la inflamación en el estado inicial del traumatismo. Sin embargo, las lesiones de larga duración y también las cicatrices aparecen en zonas hipotérmicas causadas por la reducción de la contracción muscular y, por tanto, de la producción de calor. Se han encontrado zonas similares de temperatura disminuida adyacentes a articulaciones periféricas con rango de movimiento reducido debido a inflamación o dolor (Diakides et al., 2012).

2.11. Métricas de desempeño

Las métricas de rendimiento se utilizan para cuantificar el desempeño del modelo de inteligencia artificial. Para ello se utilizan varios métodos de evaluación, en clasificación de imágenes los métodos que comúnmente se utilizan son: la matriz de confusión F1-score, precisión, exactitud, sensibilidad, especificidad.

Matriz de confusión. La matriz de confusión brinda información acerca del rendimiento de un modelo de clasificación durante la fase de entramiento y de validación. Se representa como una matriz de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN), donde:

- Verdaderos positivos (TP): Valores que son realmente positivos y que el modelo los clasificó como positivos.
- Verdaderos negativos (TN): Valores que son realmente negativos y que el modelo los clasificó como negativos.
- Falsos positivos (FP): Valores reales que son negativos pero el modelo los clasificó como valores positivos.
- Falsos negativos (FN): Valores reales que son positivos pero el modelo los clasificó como valores negativos.

Exactitud. Es la métrica de rendimiento más usada e indica cuantas veces el modelo hizo una predicción o clasificación correcta sobre el número total de objetos clasificados. Se representa como:

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \quad [14]$$

Precisión. Esta métrica indica cuantas veces el modelo clasificó incorrectamente un caso real negativo como positivo, un falso positivo (FP). Es decir, la cantidad de casos clasificados como verdaderamente positivos sobre todos los casos clasificados como positivos. Se representa como:

$$Precisión = \frac{TP}{TP + FP} \quad [15]$$

Sensibilidad o Recall. Esta métrica indica cuantas veces el modelo clasificó incorrectamente un valor real positivo como negativo, un falso negativo (FN). Es decir, la cantidad de casos clasificados realmente como positivos sobre todos los casos positivos. Se expresa como:

$$Sensibilidad = \frac{TP}{TP + FN} \quad [16]$$

F1-score. Esta métrica combina la precisión y la sensibilidad, generalmente se utiliza para expresar la media armónica entre estas dos métricas, y se expresa como:

$$F_1\ score = \frac{precision \times sensibilidad}{precision + sensibilidad} \quad [17]$$

Especificidad. Indica cuantos casos de toda la clase negativa han sido clasificados como negativos y se representa de la siguiente manera (González-Acuña & Chaparro-Romo, 2020):

$$Especificidad = \frac{TN}{TN + FP} \quad [18]$$

CAPITULO 3

METODOLOGÍA

Para el desarrollo del sistema embebido capaz de capturar imágenes termográficas en tiempo real con el objetivo de clasificar las lesiones musculares, específicamente, se presenta en la Figura 13 la metodología seguida, la cual se generalizó en tres bloques: i) Obtención del modelo, ii) Desarrollo del módulo y iii) Validación del sistema embebido.

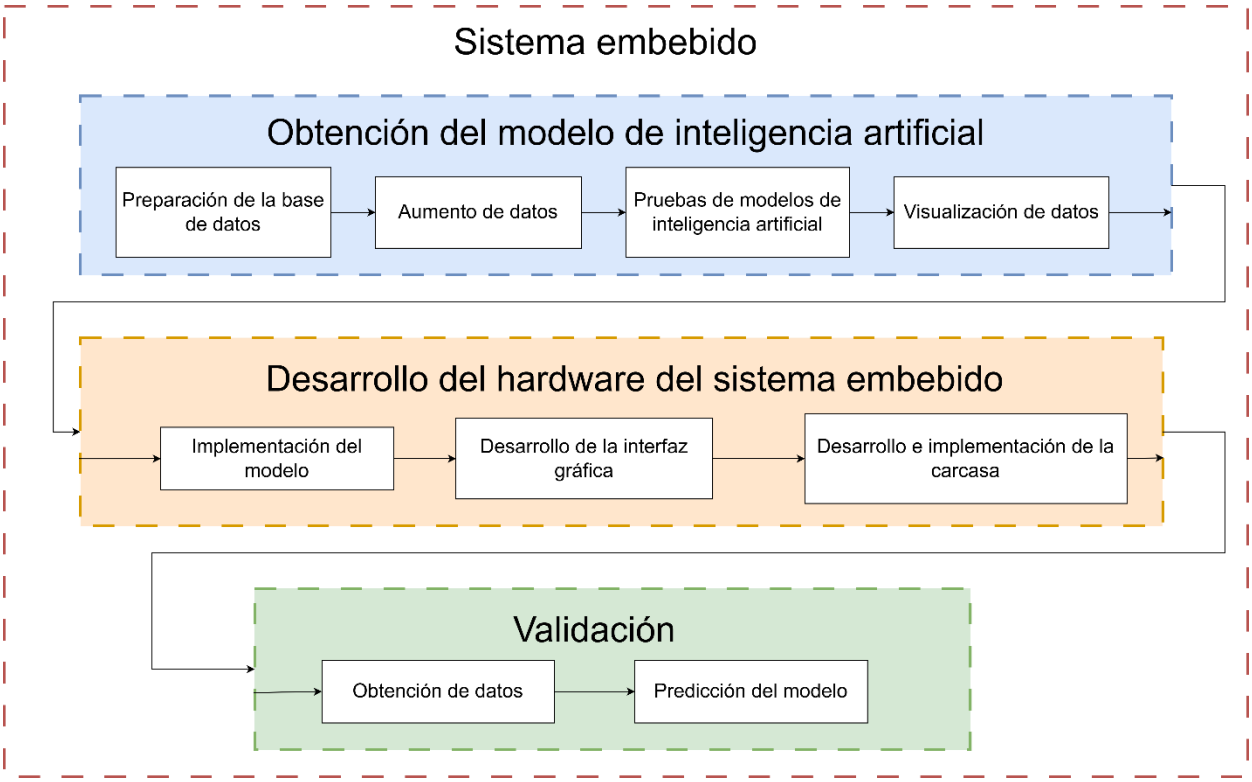

Figura 13. Diagrama de bloques de la metodología de investigación.

En el primer bloque se establece cómo se obtiene el modelo de inteligencia artificial, el cual clasifica las imágenes termográficas en tres clases: sano, síndrome patelofemoral bilateral, y síndrome patelofemoral unilateral. Iniciando con la preparación de imágenes de la base de datos, seguido del aumento de datos en el que se crearon 3 bases de datos con diferentes técnicas aplicadas a las imágenes original con la finalidad de obtener un modelo robusto. Después, se evaluaron distintos modelos de inteligencia artificial: SVM, KNN y varias configuraciones de CNNs. Una vez elegido el modelo más adecuado en función de las métricas (precisión y exactitud), se realizó la visualización de características del modelo utilizando la técnica Grad-CAM, para identificar cuáles áreas de las imágenes el modelo considera relevantes para la

clasificación de cada clase.

En el segundo bloque se realizó la implementación del hardware del sistema embebido, es decir, la conexión de todos los componentes electrónicos, asegurando el adecuado funcionamiento individual y colectivo de cada uno. Seguido, del diseño y desarrollo de la interfaz, que se encarga de capturar los datos del paciente, así como de obtener imágenes termográficas y digitales, realizar predicciones a partir de ellas y actualizar el modelo con las nuevas imágenes recibidas, además de elaborar un informe por cada paciente. El bloque termina con el diseño e implementación de la carcasa para el sistema embebido, el cual fue diseñado por estudiantes de la carrera Diseño industrial que forman parte del mismo grupo de investigación.

Por último, el tercer bloque, el cual es el proceso de validación del sistema embebido. Se realizaron pruebas clínicas con el sistema embebido verificar su correcto funcionamiento. Estas pruebas se realizaron en clínicas se efectuaron en clínicas de fisioterapia de la Universidad Autónoma de Querétaro, bajo la supervisión de personal especializado.

3.1. Base de datos

La base de datos usada fue obtenida de una investigación previa (Trejo-Chávez et al., 2022). Además, para la validación del sistema embebido nuevas imágenes termográficas fueron capturadas. A continuación, se explica a detalle sobre la base de datos. La obtención de nuevas imágenes se explicará a detalle más adelante.

La base de datos de (Trejo-Chávez et al., 2022) contiene imágenes del miembro inferior, específicamente con pacientes que padecen del síndrome patelofemoral. Estas imágenes fueron obtenidas mediante dispositivos de termografía infrarroja en clínicas de fisioterapia con la colaboración de expertos de la Universidad Autónoma de Querétaro. Todos los participantes aceptaron y firmaron las cartas de consentimiento informado y confidencialidad de datos. Además, el estudio recibió la aprobación del comité de ética de la Universidad Autónoma de Querétaro, bajo el número de registro CEAIFI-084-2020-TP. Así mismo, este proyecto cuenta con la aprobación del comité de ética de la Universidad Autónoma de Querétaro, bajo el número de registro CEAIFI-169-2023-TP.

Los participantes fueron seleccionados por expertos en el área de la salud, en este caso, personal

de la facultad de enfermería de la Universidad Autónoma de Querétaro, quienes evaluaron y diagnosticaron la condición de la rodilla de cada participante.

La base de datos consta de 48 participantes (25 hombres and 23 mujer entre la edad de 22.5 ± 4.5 años. Por lo que se juntaron los participantes en dos grupos: (1) grupo control, donde los participantes están sanos, y (2) grupo experimental, donde los participantes presentan algún tipo de alteración en el área de la patela.

Todas las imágenes fueron ajustadas a una temperatura de $30°C - 36°C$. Cabe mencionar que los participantes en el grupo experimental cumplen con los criterios de inclusión para el uso de la termografía en humanos. Además, los participantes de ambos grupos debieron de cumplir los siguientes criterios: (i) No realizar ninguna actividad física, al menos, 12 horas antes del test; (ii) Evitar usar maquillaje o crema; (iii) No consumir ninguna bebida alcohólica o energética; (iv) No fumar; y (v) No tomar medicamentos, cafeína, o alguna sustancia que pueda afectar el test. Las imágenes fueron obtenidas en un cuarto con la temperatura controlada entre $21 \pm 2\,°C$, se estableció una distancia entre el participante y la cámara termográfica fue de $1.2m$, y la orientación del paciente hacia la cámara fue frontal, como se muestra en la Figura 14. Un experto en termografía y un científico de la salud evaluaron, en todo momento, si se cumplió con el protocolo de adquisición.

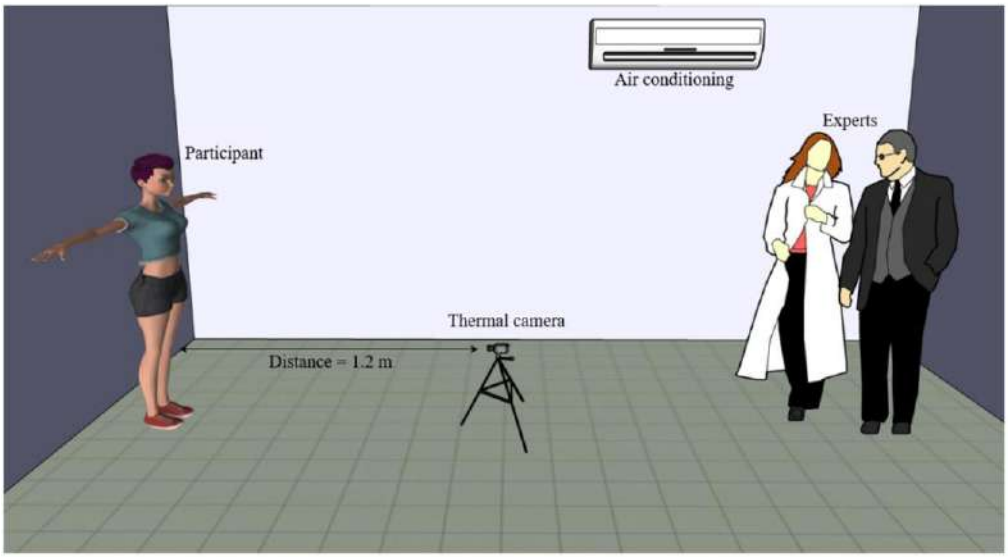

Figura 14. Diagrama de adquisición de imágenes termográficas. Obtenida de(Trejo-Chávez et al., 2022)

Posteriormente, el equipo de investigación realizó el mismo procedimiento de adquisición de

imágenes, para ampliar la base de datos. Sin embargo, estas nuevas imágenes pertenecen a pacientes que padecen Síndrome Patelofemoral Unilateral. Por lo que la base de datos usada en este trabajo consiste en tres clases: i) clase 0 o clase “Sano”; ii) clase 1 o clase "Síndrome Patelofemoral Bilateral" (Bilateral PS); y iii) clase 2 o clase "Síndrome Patelofemoral Unilateral" (Unilateral PS), un ejemplo de la base de datos se muestra en la Figura 15.

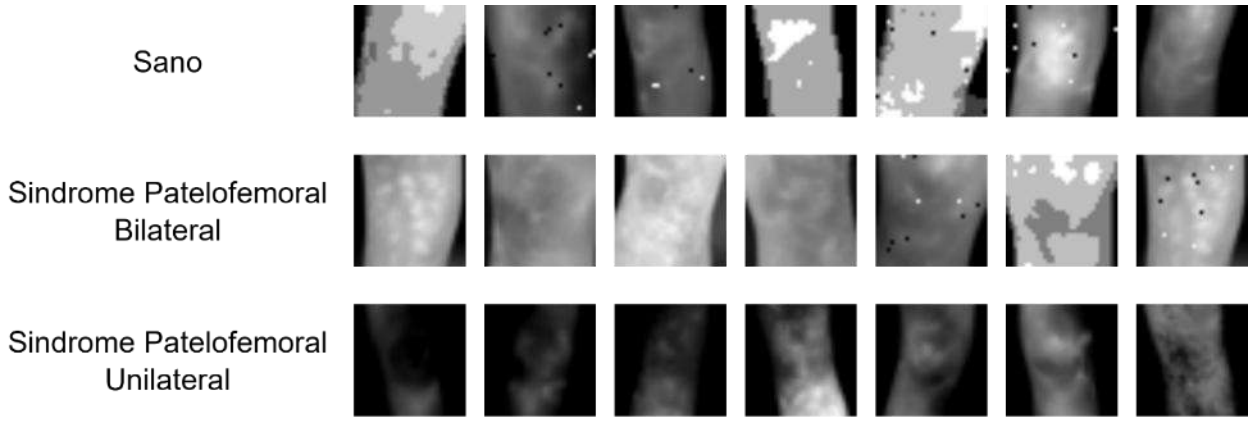

Figura 15. Ejemplo de imágenes por clase (Sano, Síndrome Patelofemoral Bilateral, Síndrome Patelofemoral Unilateral).

3.2. Preparación de datos

La preparación de los datos consistió en cambiar el tamaño de la imagen, etiquetar y crear un conjunto de datos para el modelo. El conjunto de imágenes provenía de una base de datos privada que contenía tres carpetas correspondientes a las tres clases en las que se etiquetaron las imágenes. En el presente trabajo se utilizaron directamente las imágenes térmicas que fueron generadas en el artículo de (Trejo-Chávez et al., 2022). Estas imágenes fueron sometidas previamente a un procesamiento de imágenes.

Las imágenes fueron redimensionadas a $30\ x\ 30$ píxeles. Se cargaron y etiquetaron en 3 clases: la clase “Sano” o clase 0, la clase “Síndrome Patelofemoral Bilateral” o clase 1, y la clase “Síndrome Patelofemoral Unilateral” o clase 2. Obteniendo 192 imágenes para cada clase, es decir un total de 576 imágenes.

El conjunto de imágenes fue dividido en 3 grupos: un conjunto de entrenamiento, un conjunto de validación y un conjunto de prueba, con el 75% de los datos asignados al conjunto de entrenamiento, el 15% al conjunto de validación y el 10% al conjunto de prueba. Primeramente, cada clase fue dividida en dos conjuntos: conjunto de entrenamiento (75%) y conjunto temporal

de prueba (25%), en el que aleatoriamente se seleccionaron 143 imágenes para el entrenamiento y 49 imágenes para la prueba de cada clase.

Al sumar cada conjunto de imágenes de cada clase se obtuvieron un total de 429 imágenes para el conjunto de imágenes de entrenamiento. Mientras que para el conjunto temporal de prueba se obtuvieron un total de 147 imágenes. Para la obtención del conjunto de validación y de prueba, el conjunto temporal de prueba fue dividido en 2, respetando el porcentaje. La selección de imágenes se realizó de manera aleatoria. Resultando en un total de 88 imágenes para el conjunto de validación y para el conjunto de test un total de 59 imágenes.

3.3. Aumento de datos

Tener una base de datos pequeña puede afectar negativamente al modelo, y limitar su capacidad de generalización. Por lo que el modelo puede presentar un sobreajuste de los datos, es decir, aprender bien las imágenes de conjunto de entrenamiento, pero fallar en imágenes de prueba o con imágenes. Además, la falta de variabilidad disminuye la robustez del modelo. Para evitar estos problemas, se usaron varias técnicas de aumento de datos, específicamente se utilizaron ocho técnicas, evitando técnicas que pudieran introducir ruido al modelo o información incompleta, ya que las imágenes son una parte anatómica del cuerpo que contiene información de la rodilla. Las técnicas utilizadas fueron: i) rotación más voltear, ii) voltear , iii) variación del contraste, iv) variación de brillo, v) variación de la saturación; y combinaciones de dos técnicas, como vi) variación del contraste más rotación, vii) variación del brillo más rotación y viii) variación de la saturación más rotación.

Los parámetros establecidos para cada técnica de aumento de datos fueron definidos de manera experimental; para rotación fue de una variación de $\pm 20\%$, para variación de brillo fue un factor de $+0.15$, para la variación del contraste se usaron los siguientes factores $lower = 0.1, upper = 0.9$, y finalmente para la variación de la saturación un contraste de 11. Para llevar a cabo el aumento de datos, se utilizaron las librerías de Python: Keras y Tensorflow.

Se crearon tres conjuntos de imágenes diferentes, compuestas por imágenes originales e imágenes generadas por técnicas de aumento de datos. El aumento de datos se utilizó únicamente en el conjunto destinado al entrenamiento, mientras que el conjunto de validación y el conjunto de prueba no se modificaron, es decir, están conformadas solo por las imágenes originales. Se

realizaron tres conjuntos diferentes con el objetivo de identificar el conjunto de datos que optimice el rendimiento del modelo, evitando así el sobreajuste. Los tres conjuntos de imágenes creados son los siguientes.

- Conjunto de imágenes aumentadas A: Contiene la imagen original más 2 imágenes adicionales generadas por el aumento de datos.

- Conjunto de imágenes aumentadas B: Contiene la imagen original más 5 imágenes adicionales generadas por el aumento de datos.

- Conjunto de imágenes aumentadas C: Contiene la imagen original más 8 imágenes adicionales generadas por el aumento de datos.

En la Tabla 2 se muestra las técnicas de aumento de datos usadas para la creación de cada conjunto de imágenes aumentadas.

Tabla 2. Técnicas de aumento de datos usadas para cada conjunto de datos

Técnica de aumento de datos	Conjunto de imágenes aumentadas A	Conjunto de imágenes aumentadas B	Conjunto de imágenes aumentadas C
Random rotación más voltear	Si	Si	Si
Voltear	Si	Si	Si
Variación del contraste	–	Si	Si
Variación de brillo	–	Si	Si
Variación de la saturación	–	Si	Si
Variación del contraste más rotación	–	–	Si
Variación del brillo más rotación	–	–	Si
Variación de la saturación más rotación.	–	–	Si

3.4. Primeras pruebas para la selección del modelo de aprendizaje automático.

Inicialmente para la selección del modelo se probaron con varios modelos de aprendizaje automático y de aprendizaje profundo. Los algoritmos evaluados incluyeron: (i) SVM (Support Vector Machine), (ii) KNN (K-Nearest Neighbor) y CNN (Convolutional Neural Network). Es importante mencionar que cada modelo se probó con los tres conjuntos de imágenes aumentadas (A, B y C), para comparar los resultados de acuerdo con la precisión de cada modelo. De este modo, se pudo determinar cuál conjunto de imágenes era el más efectivo para el entrenamiento del modelo, así como el modelo con el mejor rendimiento.

3.4.1. SVM (Support Vector Machine),

Para entrenar el modelo, solo fueron necesarios el conjunto de entrenamiento para el entrenamiento del modelo y para la predicción del modelo, el conjunto de prueba. Ambos conjuntos de imágenes se transformaron, es decir, de una imagen tridimensional ($30 \times 30 \times 3$) se aplana a un vector unidimensional de tamaño 2700. Obteniendo una lista de vectores $[n, 2700]$, donde n es el número de imágenes. Las pruebas fueron realizadas con imágenes normalizadas y sin normalizar. Este procedimiento fue el mismo para los tres conjuntos de imágenes aumentadas. Para la precisión del modelo se tomó en cuenta las predicciones correctas entre el total de predicciones.

3.4.2. K-NN (K-Nearest Neighbors)

Para el modelo K-NN, al igual que el modelo SVM, fueron necesarios el conjunto de entrenamiento para entrenar el modelo y para la predicción del modelo, el conjunto de prueba. El conjunto de imágenes se transformó, es decir, se aplanaron. Al igual que SVM, K-NN trabaja con vectores, por lo que las imágenes tridimensionales se transformaron a vectores de $[n, 2700]$. Para la predicción del modelo se utilizó $k = 3$, para que busque entre los 3 vecinos más cercanos para realizar la predicción. Del mismo modo, se hicieron pruebas con el conjunto de imágenes normalizado y sin normalizar. Este procedimiento fue el mismo para los tres conjunto de datos de imágenes aumentadas (A, B y C).

3.4.3. CNN (Convolutional Neural Network)

La arquitectura de CNN utilizada fue, en un primer acercamiento, una versión modificada del modelo LeNet-5. Debido a que, las lesiones en miembros inferiores no han sido abarcadas desde una perspectiva de inteligencia artificial. No hay investigaciones que nos ayuden a guiarnos sobre qué tipo de modelo de aprendizaje profundo usar. Y, además, al ser la arquitectura con la que se tenía un mayor acercamiento, se optó por comenzar desde ese punto.

Las modificaciones realizadas a la arquitectura constan de agregar capas convolucionales extras. La arquitectura modificada de este modelo CNN se muestra en la Tabla 3.

Tabla 3. Arquitectura modificada de LeNet-5

Nombre	Tipo de capa	Tamaño de salida
Input Layer	Imagen de entrada	$30 \times 30 \times 3$
Layer_1_Convolutional	Convolucional	$28 \times 28 \times 6$
Layer_2_Convolutional	Convolucional	$26 \times 26 \times 6$
Layer_1_Pooling	Average Pooling	$13 \times 13 \times 6$
Layer_3_Convolutional	Convolucional	$11 \times 11 \times 10$
Layer_4_Convolutional	Average Pooling	$9 \times 9 \times 10$
Layer_2_Pooling	Average Pooling	$4 \times 4 \times 10$
Layer_5_Convolutional	Convolucional	$2 \times 2 \times 20$
Flatten_49	Flatten	$None \times 80$
Layer_1_Dense	Dense	$None \times 128$
Layer_2_Dense	Dense	$None \times 3$

Además, el modelo CNN fue diseñado para recibir imágenes de tamaño 30×30. Las primeras dos capas convolucionales están diseñadas con 6 filtros de tamaño 3×3. Luego, se aplica una capa de agrupamiento promedio (Average Pooling) con un kernel de 2×2. Seguido de dos capas convolucionales con 10 filtros de tamaño 3×3, seguida de otra capa de agrupamiento promedio. Después, se añade una última capa convolucional con 20 filtro, también de tamaño 3×3. Todas las capas de convolución contienen la función de activación ReLU. El resultado de estas capas convolucionales se aplana para ser procesado por las capas completamente conectadas: una densa con 128 neuronas y activación ReLU, y una capa de salida con 3 neuronas y activación softmax.

El entrenamiento fue realizado de dos formas: con imágenes sin normalizar y con imágenes normalizadas. Para observar si hay una diferencia importante al entrenar el modelo de esta manera. Además, este proceso fue realizado con los tres conjuntos de imágenes aumentadas (A, B y C). Por lo que, el modelo está diseñado para clasificar imágenes de $30\ x\ 30$ pixeles en 3 clases: Sano, síndrome Patelofemoral Bilateral, síndrome Patelofemoral Unilateral.

Para calcular el desempeño de cada prueba del modelo de CNN, se utilizaron las métricas más usadas para redes convolucionales: exactitud, matriz de confusión y un reporte de clasificación que contiene las métricas: precisión, f1-score y Recall. Se utilizaron las librerías Tensorflow, Keras, Numpy, Matplotlib, Sklearn tanto para el entrenamiento, compilación y guardado del modelo, como para la obtención de las métricas.

3.5. Selección del modelo de aprendizaje profundo.

Al realizar una comparación del desempeño de los modelos K-NN y SVM, y de los modelos CNN

(Sección 3.4). Se optó por usar un modelo de redes neuronales convolucionales. Sin embargo, a diferencia del modelo CNN basado en LeNet-5, donde no se realizaron pruebas relacionadas con la arquitectura o configuración, esta vez sí se realizaron. Por consiguiente, la selección del modelo de CNN dependió de las diversas pruebas que se realizaron, es decir, del test de ablación.

Para ello, el procedimiento está basado en la metodología usada en (Trejo-Chávez et al., 2022), donde analizaron diferentes configuraciones, como el tamaño de la imagen de entrada, el número y tamaño de los filtros, los métodos de submuestreo y el tamaño del lote (batch size) para la CNN, con el fin de evaluar y seleccionar una estructura CNN mejor en términos de precisión y coste computacional. La metodología usada para la selección del modelo CNN, se muestra en la Figura 16.

Etapa 1. Esta etapa consiste en la selección de la arquitectura de la red neuronal convolucional. Para ello, se probaron dos CNN sencillas. La diferencia entre estos dos modelos está en la etapa de extracción de características, mientras que en la etapa de clasificación se mantuvieron igual. La primera arquitectura de CNN (CNN_A1) consta de una capa convolucional y una capa de agrupamiento. Mientras la segunda arquitectura de CNN (CNN_A2) está compuesta por dos capas convolucionales y una capa de agrupamiento.

Etapa 2. Ambos modelos (CNN_A1 y CNN_A2) se analizaron usando diferentes configuraciones con el objetivo de seleccionar la mejor estructura de CNN. Los hiperparametros que se modificaron fueron: el número y tamaño de los filtros. Se evaluaron combinaciones de tamaños de filtro $(3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6)$ y cantidades de filtros $(4, 6, 8, 10, 12)$, en ambos modelos. Donde el entrenamiento del modelo solo fue realizado con el conjunto de imágenes aumentadas A, y los hiperparametros de entrenamiento como tamaño de la imagen de entrada, la taza de aprendizaje (learning rate), las épocas y el tamaño de lote (batch size) permanecieron igual durante cada prueba de las combinaciones. Los valores asignados fueron los siguientes: 30×30 al tamaño de imagen de entrada, un learning rate de 0.0001, épocas de 10, y un batch size de 8. La selección del modelo fue realizada en términos de la exactitud y la función de pérdida del modelo.

Etapa 3. Una vez realizada la selección de la estructura del modelo de acuerdo con su desempeño, se realizaron más pruebas para la obtención de un modelo CNN más robusto. El objetivo principal es minimizar el error o la función de pérdida. Para reducir el error, los hiperparametros del modelo

se actualizan continuamente durante cada época de entrenamiento y el modelo busca iterativamente la solución localmente óptima en cada época de entrenamiento (Alzubaidi et al., 2021). El tamaño de los pasos de actualización de parámetros se denomina "tasa de aprendizaje" y una iteración completa de actualización de parámetros que incluye el conjunto de datos de entrenamiento se denomina "época de entrenamiento". Aunque la tasa de aprendizaje es un hiperparametros, pero tenemos que elegir con tanto cuidado que no afecte al proceso de aprendizaje (Alzubaidi et al., 2021). Esta vez se modificaron los hiperparametros utilizados para la optimización del modelo: como el tamaño del lote, épocas y la tasa de aprendizaje. Se evaluaron combinaciones de learning rate ($0.0001\ y\ 0.001$); batch size $(8, 10, 12)$; y épocas de entrenamiento $(10, 20, 30, 40, 50, 60, 70, 80, 90, y\ 100)$. Estas pruebas se realizaron con los 3 conjuntos de imágenes aumentadas $(A, B\ y\ C)$ con el fin de comparar cuál de estos 3 conjuntos da un mejor resultado al evaluar el modelo con el conjunto de prueba. La evaluación se realizó en términos de precisión, exactitud, Recall, F1-score, y con la matriz de confusión, durante cada evaluación se fueron guardando los mejores modelos en formato "$h5$".

Etapa 4. Se seleccionaron los mejores modelos obtenidos de cada conjunto de imágenes aumentadas, teniendo un total de 3 modelos finales. Cada modelo se probó con los conjunto de imágenes aumentadas con el que no fueron entrenados, es decir, el mejor modelo obtenido del entrenamiento con el conjunto de imágenes aumentadas A, se probó con el conjunto de imágenes aumentadas B y C. Y del mismo modo para los otros dos modelos. Se compararon los resultados de acuerdo con sus métricas, y se realizó la selección del modelo que mejor desempeño obtuviera.

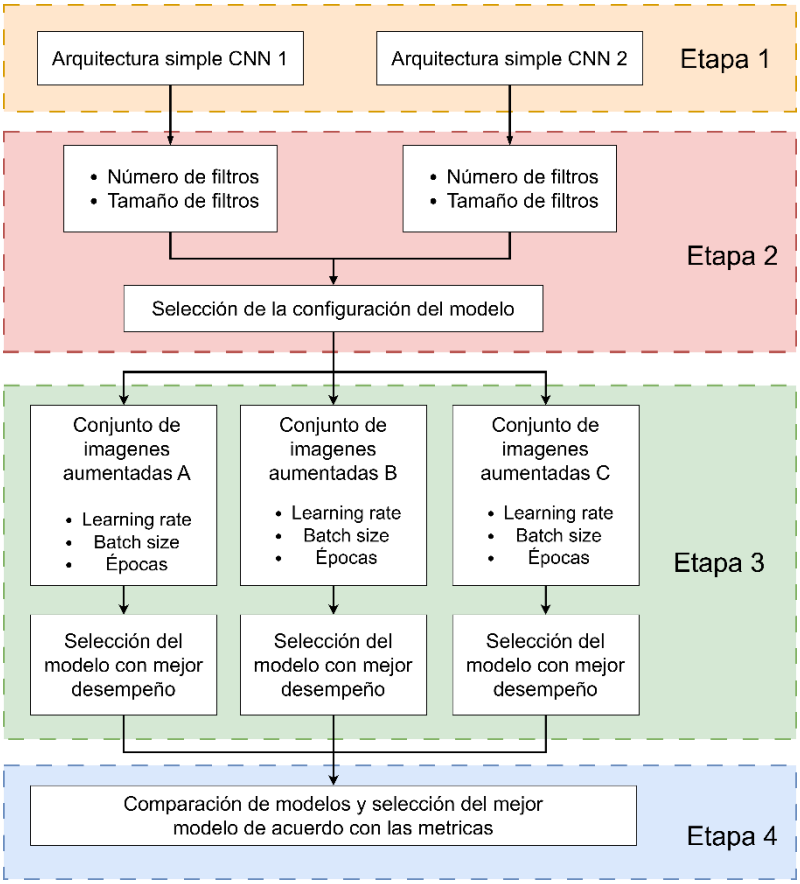

Figura 16. Metodología para la selección del modelo de CNN

3.6. Visualización de características

Para entender cómo funciona una red neuronal o qué decisiones toma al realizar tareas como la clasificación o la detección de objetos, se han desarrollado varias técnicas. Una de ellas es Grad-CAM que, a diferencia del método CAM, no restringe la arquitectura y no requiere una capa Global Average Pooling seguida de una capa Fully Connected. Normalmente utiliza la última capa convolucional, asumiendo que la última capa contiene información más crítica y de mayor nivel, cuanto más profundos sean los mapas de características, más rica será la información. La principal diferencia entre CAM y Grad-CAM es cómo se ponderan los mapas de características para producir el mapa final, el proceso se muestra en la Figura 17.

El diagrama describe el proceso que sigue Grad-CAM, donde la imagen de entrada pasa por el modelo CNN y se obtiene el mapa de características de la última capa convolucional, pasará por la capa completamente conectada y luego por softmax, que dependiendo de la clase a la que corresponda la imagen, se obtendrán los pesos de esa clase, donde cada peso corresponde a cada

mapa de características. Para obtener el mapa de calor, el gradiente se obtiene por backpropagation, se refiere a la derivada parcial de la puntuación de la clase con respecto a los mapas de características, donde cuanto más fuerte sea la derivada parcial más importante será la información que contiene. Crea una matriz (mapas de gradiente) con el mismo tamaño que los mapas de características. Pasa por un Global Average Pooling (GAP) donde obtiene escalares y se realiza la sumatorio de la multiplicación de cada feature map por el peso correspondiente. Luego pasa por el ReLU donde los valores negativos no pasan porque contribuyen a otra clase, por lo que el mapa de calor muestra la influencia más fuerte hacia una clase en particular, luego se redimensiona.

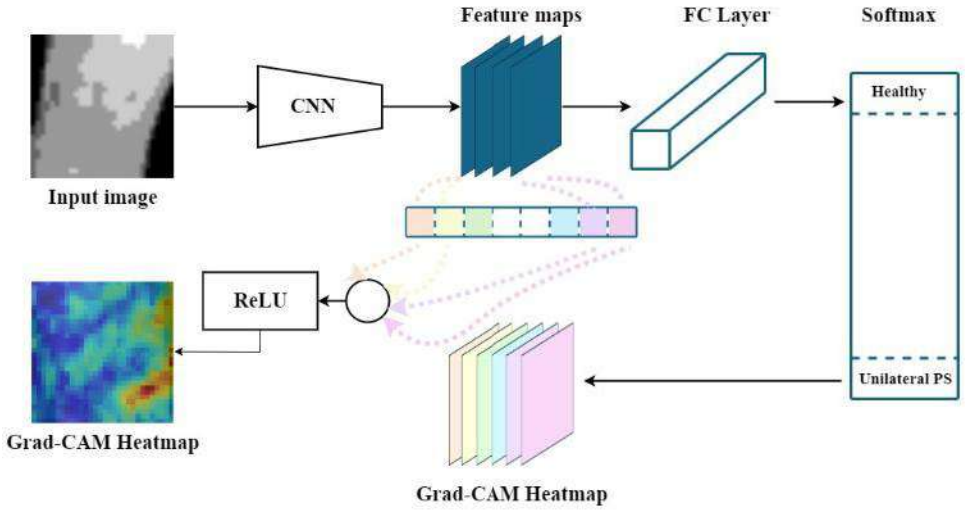

Figura 17. Metodología de Grad-CAM. Elaboración propia.

La librería usada para obtener los mapas de calor fue Tensorflow Keras, que contiene el módulo de Grad-CAM. Para ello, se utilizaron las imágenes pertenecientes a los conjuntos de datos de prueba y validación. El objetivo de utilizar esta técnica es encontrar un mapa de calor que pueda representar de forma general las zonas de las imágenes que identifican a cada clase. Por lo que al pasar cada imagen por el Grad-CAM se obtendrá un mapa de calor por cada imagen. Para generalizar qué zonas de la imagen de cada clase tienen mayor impacto según el modelo, se obtuvieron los mapas de calor correspondientes a la clase "Sano", los mapas de calor de la clase "Síndrome patelofemoral bilateral" y los mapas de calor de la clase "Síndrome patelofemoral unilateral", es decir, de cada imagen perteneciente a cada clase se obtuvo su mapa de calor.

A continuación, se sumaron los mapas de calor con su clase respectiva, es decir, se sumaron todos los mapas térmicos de la clase "Sano", todos los mapas térmicos de la clase "Bilateral" y todos los mapas térmicos de la clase "Unilateral". Obteniendo un total de 3 mapas térmicos, uno para cada

clase, y cada uno de ellos se normalizó entre 0 y 255. Este proceso fue realizado dos veces, la primera consta de la obtención de los mapas de calor pertenecientes a la capa "Layer_1_convolutional" como capa convolucional y la segunda con los mapas de calor de la capa "Layer_2_convolutional" como capa convolucional. Finalmente, se hizo un análisis de los mapas de calor para obtener cual representa mejor las características que hacen que cada imagen pertenezca a una clase.

3.7. Conexión y configuración de componentes electrónicos

El propósito del desarrollo del módulo es crear un sistema embebido que permita ejecutar en tiempo real el modelo de CNN seleccionado. Creando como primer prototipo una herramienta que se pueda utilizar en pacientes con lesiones en miembro inferior. El objetivo principal del sistema embebido es capturar imágenes termográficas del miembro inferior para clasificar en tiempo real la imágenes en las tres clases (Sano, síndrome patelofemoral Bilateral, síndrome patelofemoral Unilateral). Así mismo, el modelo se reentrenará en el sistema embebido cada que el usuario deseé, para generar un modelo robusto debido a que las imágenes capturadas provienen de una diferente cámara termográfica de la que generó la base de datos original. Consta de 4 partes fundamentales: i) Conexión y configuración de componentes electrónicos, ii) Modificación del modelo CNN seleccionado para ejecutarlo en el sistema embebido (Sección 3.1.8), iii) Diseño de la interfaz (Sección 3.1.9) y iv) Diseño de la carcasa (Sección 3.1.10).

Por lo que, para el desarrollo del módulo se ocuparon los siguientes componentes: i) Jetson Nano Developer Kit, ii) Raspberry Pi Modulo de cámara V2.0 como cámara digital, iii) FLIR Lepton® Breakout Board v2.0, iv) LWIR Micro Thermal Camera Module 3.5, v) HDMI 7inch Capacitive Touch Screen LCD (H). Y como componentes extras: el adaptador de corriente, cable ethernet, cable HDMI, dos cables USB-Micro USB y cables Dupont hembra-hembra.

En el Anexo D, se ofrece una explicación más detallada de la metodología utilizada para desarrollar el sistema embebido. No obstante, se puede resumir brevemente que el proceso incluye la configuración del dispositivo Jetson Nano, es decir, los preparativos para la instalación y la instalación de la Imagen en la tarjeta, así como la instalación de librerías esenciales, programas necesarios como Visual Code Studio, y la configuración de OpenCV con Cuda para reducir el costo computacional en las tareas de visión por computadora. Después, mediante test SPI, el

correcto funcionamiento del protocolo de comunicación SPI fue comprobado, es necesario para no tener problemas al momento de realizar la conexión de la cámara termográfica Lepton. Luego, se descargaron las librerías necesarias y la Raspberry Pi Modulo de cámara V2.0 fue configurada y conectada. Se siguió con la descarga de la librería necesaria para poder acceder a la cámara termográfica Lepton 3.5, llamada pylepton, obtenida de repositorios de GitHub, además, se realizaron modificación en la librería para que la cámara funcionara correctamente. Por medio de Python, se crearon diversos scripts o pruebas para familiarizarse con el funcionamiento de ambas cámaras.

Finalmente, en la Figura 19 se muestra la conexión completa del sistema embebido utilizando una Jetson Nano. Se observa la cámara Raspberry pi (A) conectada al puerto CSI (en la Figura 18 se muestra los conectores [$J49$] y [$J13$] siendo los puertos CSI), la cámara termográfica FLIR Lepton (B) está conectada mediante un Breakout Board (C) a los pines GPIO de la Jetson Nano (en la Figura 18 se muestra: [$J41$]). También se conecta a una red Wi-Fi (D) mediante el puerto Ethernet (en la Figura 18 se muestra: [$J43$]). En la parte inferior se conectan un teclado y mouse (E) por USB (en la Figura 18 se muestra:[$J32$]), así como una pantalla táctil (F) mediante HDMI (en la Figura 17 se muestra: [$J6$]) y USB para salida visual. Finalmente, la placa es alimentada a través de un adaptador de corriente (G) (en la Figura 18 se muestra: [$J25$]).

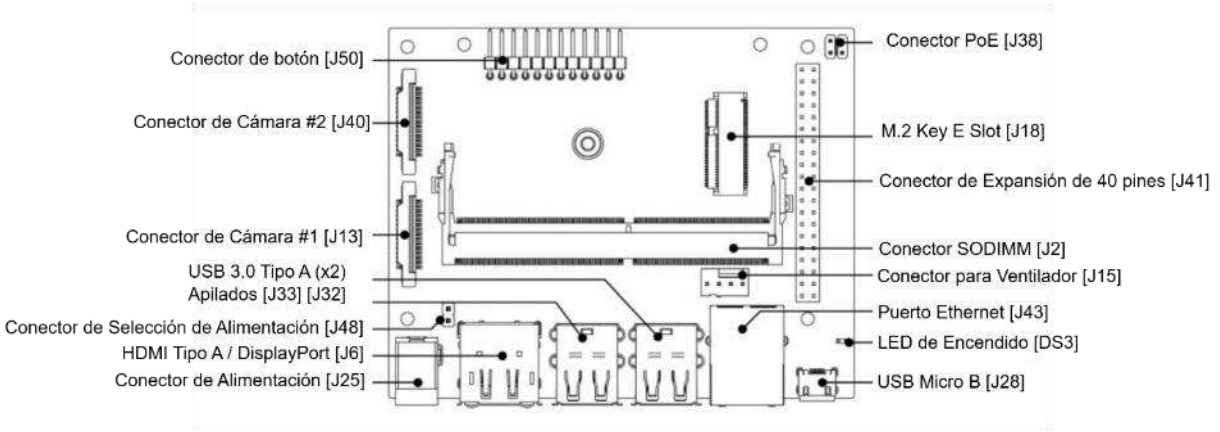

Figura 18. Diagrama de la tarjeta Jetson Nano. Obtenida de (NVIDIA, n.d.)

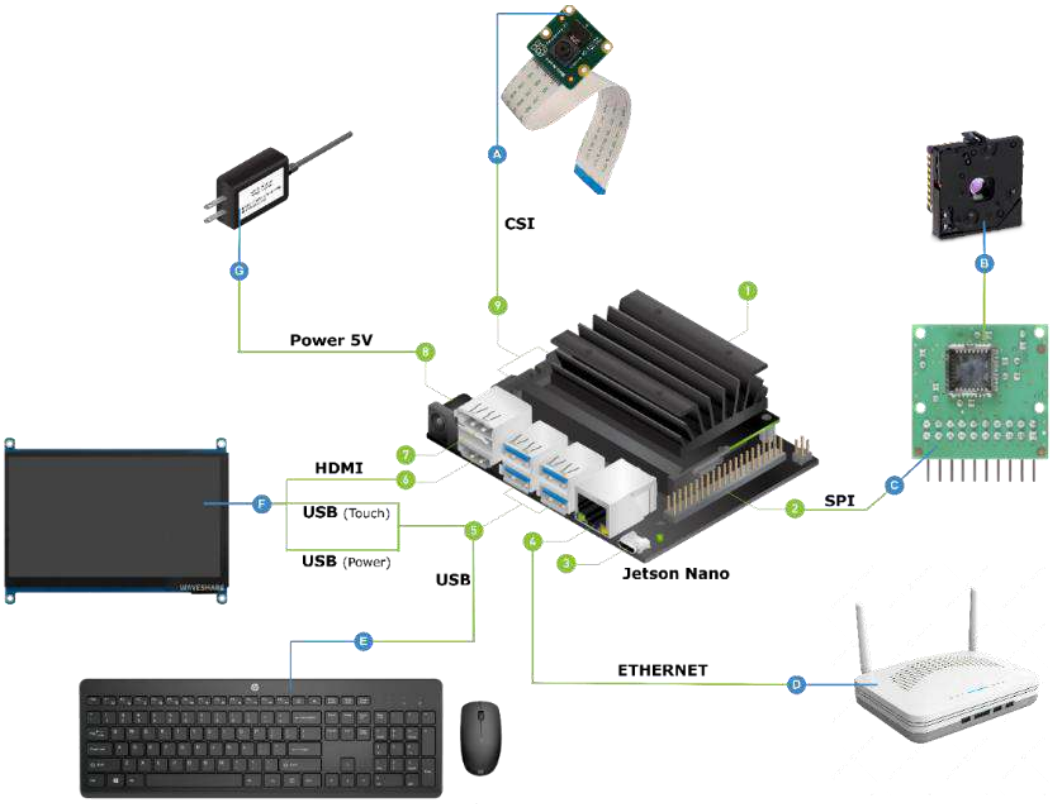

Figura 19. Diagrama de la conexión de componentes del sistema embebido. Elaboración propia.

3.8. Modelo Tensorflow a Tensorflow Lite

Las pruebas tanto para los modelos de aprendizaje automático (SVM y K-NN) como para los modelos de aprendizaje profundo (CNN) fueron realizadas utilizando el entorno de desarrollo Google Colab. Tensorflow y Keras fueron las principales librerías que se ocuparon, sin embargo, las versiones utilizadas son recientes: 2.18.0 para Tensorflow y 3.8.0 para Keras. Por lo que, al guardar el modelo CNN en un archivo ".h5", este se almacena con un formato que puede no ser compatible con versiones anteriores de estas bibliotecas.

Las versiones instaladas en la tarjeta Jetson Nano, siendo $2.5.0 + nv21.8$ para Tensorflow y 1. 0. 8 para Keras, son antiguas y no pueden actualizarse debido a la necesidad de una versión más reciente de Python, la cual ya no es compatible con Jetson Nano. Debido a esto, se optó por transformar el modelo a un formato más eficiente, como Tensorflow Lite.

Tensorflow Lite es una herramienta que permite ejecutar modelos en dispositivos incorporados, móviles o sistemas embebidos, se identifica con la extensión de archivo ".tflite". Sin embargo, un modelo tflite no pude reentrenarse, solo puede ejecutarse. Se realizaron comparaciones entre los

dos modelos: modelo CNN y modelo CNN convertido a tflite para corroborar si se perdía información o había algún cambio en el desempeño del modelo. Los pasos para convertir el modelo CNN de Tensorflow a Tensorflow Lite, así como, los pasos para cargar y ejecutar el modelo Tensorflow Lite se muestran en el Anexo A, específicamente en el Paso 13 y Paso 14.

3.9. Interfaz

En la Figura 20 se muestra el diagrama de bloques que describe el funcionamiento de la interfaz. El diseño de la interfaz se realizó con librerías de Python: Tkinter, CustomTkinter, docx, PIL, numpy y Tensorflow.

Inicialmente, la interfaz muestra una ventana con dos opciones: Ingresar al paciente y Cerrar. Al seleccionar “Ingresar Paciente” aparecerá una ventana en la que se tiene que llenar con los datos del paciente: nombre, apellido materno, apellido paterno, edad, estatura, género, estatura, y el diagnóstico del paciente. Si se ingresó correctamente los datos, y están todos los espacios llenos, pasará a la captura de imágenes, de lo contrario, se tiene que corregir los datos.

Una vez que se está en la pantalla de captura de imágenes, se procederá a tomar tanto la imagen térmica como la imagen digital. Si se capturaron correctamente las imágenes, se pasa a la siguiente ventana, de lo contrario se vuelve a intentar capturar las imágenes.

En la siguiente ventana, saldrán dos opciones: Predicción o Reentrenar modelo. Al seleccionar “Predicción”, se realizará la predicción de la imagen termográfica capturada y se podrá obtener el reporte del paciente. Si se selecciona “Reentrenar modelo” el sistema embebido empezará a entrenar el modelo con las nuevas imágenes capturadas. Una vez finalizado el entrenamiento, se podrá realizar la predicción de la imagen y finalmente obtener el reporte del paciente.

El reporte del paciente contiene los datos personales del paciente, así como las imágenes capturadas del paciente, y la pertenencia de la imagen a cada clase del modelo.

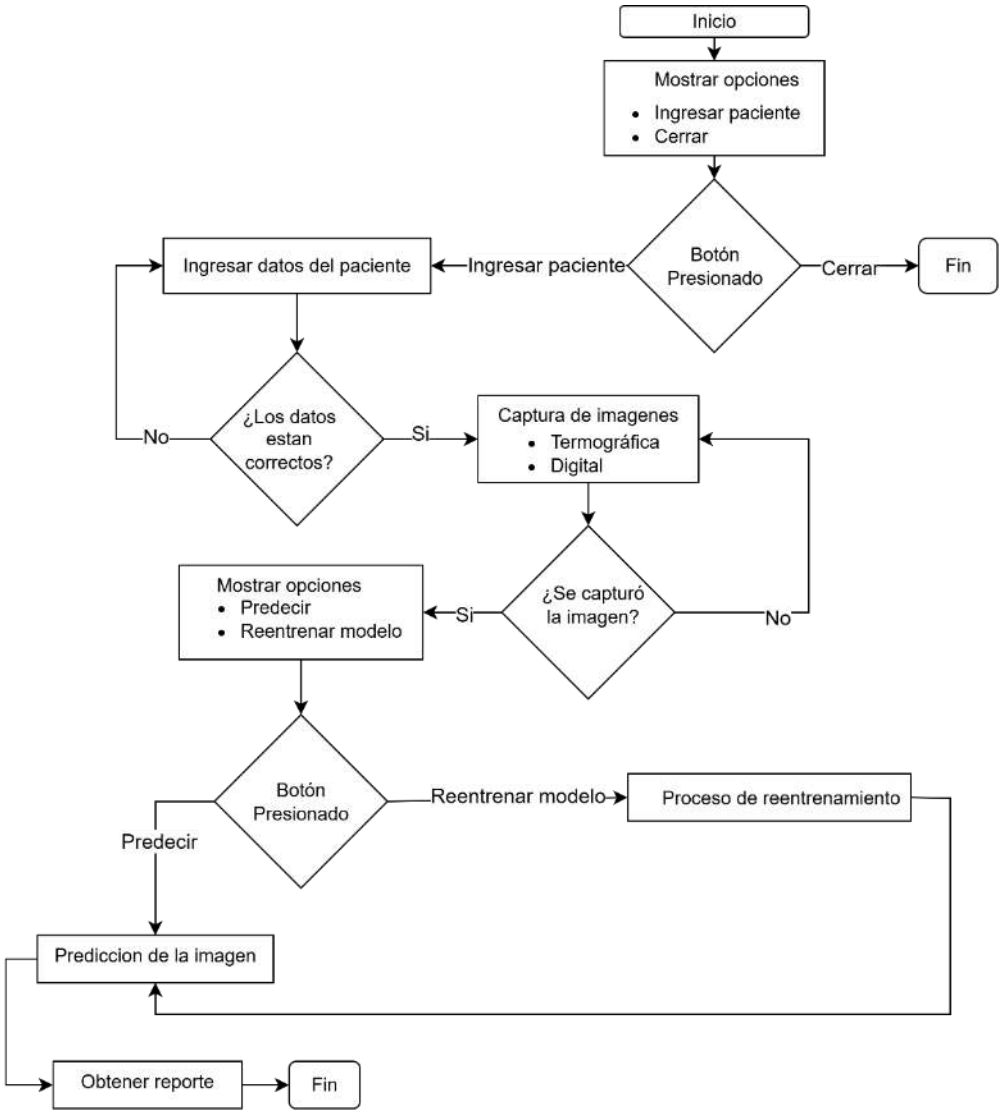

Figura 20. Diagrama de bloques sobre el funcionamiento de la interfaz. Elaboración propia.

3.10. Carcasa del modelo

El diseño de la carcasa fue realizado por estudiantes de la Universidad Autónoma de Querétaro, que pertenecen al grupo de investigación.

3.11. Obtención de nuevos datos

En esta investigación se examinó un grupo de 10 participantes, compuesto por tres hombres y siete mujeres, con un rango de edad de 22-27 años. Los participantes fueron elegidos por especialistas en el área, específicamente por personal de la facultad de enfermería de la Universidad Autónoma de Querétaro, quienes confirmaron el estado de la/las rodilla/s de cada participante. En este sentido, se generaron tres grupos: (i) el grupo control, donde los participantes presentan una rodilla sana,

(ii) el grupo bilateral, donde los participantes presentan cierto tipo de alteración en ambas rodillas y (iii) el grupo unilateral, donde los participantes presentan cierto tipo de alteración solo en una rodilla. Es importante mencionar que todos los participantes aceptaron y firmaron las cartas de consentimiento informado y confidencialidad de datos; además, esta investigación fue aprobada por el comité de ética de la Universidad Autónoma de Querétaro, bajo el número de registro CEAIFI-169-2023-TP.

Es importante mencionar que el diseño del experimento cumple con las pautas para la aplicación de la termografía en personas. Las imágenes se realizaron en un ambiente con una temperatura regulada de 28 grados Celsius y una humedad del 44%. La distancia establecida entre el participante y la cámara termográfica fue de $1.2\,m$, y la orientación del participante hacia la cámara fue frontal. Los pacientes que participaron cumplieron con los siguientes criterios:

Criterios de inclusión:

- Edad entre 18 y 28 años.
- Capacidad para entender y proporcionar un consentimiento informado.
- Ausencia de condiciones médicas que impidan la realización de imágenes termográficas como condiciones dermatológicas graves en las extremidades inferiores.
- Diagnóstico de enfermedad muscular en miembro inferior por un profesional de la salud.

Criterios de exclusión:

- Embarazo o lactancia en mujeres.
- Antecedentes de cirugía reciente en el miembro inferior (menos de 3 meses antes del reclutamiento).
- Presencia de úlceras o lesiones cutáneas graves en la región de interés.
- Antecedentes de enfermedades graves que puedan afectar la interpretación de las imágenes termográficas como enfermedades autoinmunes o neuropatía periférica.

- Uso de medicamentos vasodilatadores o vasoconstrictores.
- Contraindicaciones conocidas para la realización de imágenes termográficas como condiciones médicas que impidan permanecer inmóvil durante la toma de imágenes.

Después de dividir los grupos, se procedió a la terapia. Un subgrupo de los participantes en el grupo de control y PFPS fue sometido a calor durante 15 minutos con compresas calientes envueltas en una toalla grande. A cada individuo se le colocó una compresa caliente en una rodilla.

Las imágenes termográficas fueron obtenidas en la región anterior de las rodillas, con el participante de pie frente a la cámara y a una distancia de $0.5\ m$. El protocolo para la adquisición de imágenes termográficas, Figura 21, fue de acuerdo con los criterios para el uso de la termografía en humanos. Las imágenes se capturaron de la siguiente manera: la primera se capturó justo después de un periodo de adaptación a la habitación de 15 minutos en un ambiente controlado, se tomó después de finalizar el estrés térmico al que se les sometió, y posteriormente se recogieron imágenes durante 15 minutos con intervalos de 3 minutos para cada captura. En total, se lograron obtener 7 imágenes por participante. La cantidad de imágenes recogidas de cada paciente durante la primera captura variaba según el diagnóstico; en el grupo de control se tomó solo un termograma de una rodilla, en el grupo bilateral se obtuvieron termogramas de ambas rodillas y en el grupo unilateral se registraron termogramas de la rodilla afectada por la alteración.

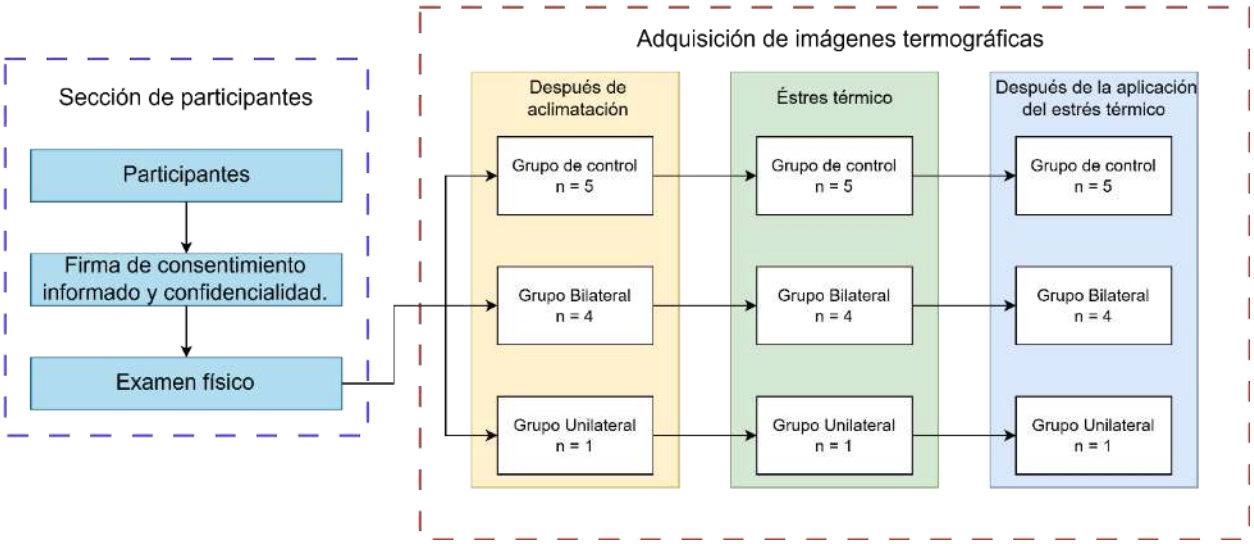

Figura 21. Metodología para la adquisición de imágenes termográficas. Elaboración propia.

Las imágenes termográficas se obtuvieron utilizando el sistema embebido que fue desarrollado en

este proyecto, el cual incluye (i) una tarjeta Jetson Nano, (ii) una cámara termográfica Lepton 3.5 de una resolución de $160x120$, (iii) cámara digital Raspberry Pi y (iv) display HDMI Waveshare de 7 pulgadas. Las imágenes fueron capturadas a través de la interfaz diseñada para este proyecto, cabe resaltar que las imágenes de la cámara digital tienen como objetivo servir de apoyo. La metodología usada para el reentrenamiento del modelo CNN, se muestra en la Figura 22.

Etapa1. Consiste en la preparación de las nuevas imágenes termográficas. Para optimizar la visualización en la interfaz, se ajustó la imagen a una resolución de $480x360$. Se obtuvieron tres imágenes: la imagen completa, la imagen térmica con un cuadrado superpuesto, y la imagen termográfica recortada según el cuadrado. El cuadrado superpuesto ayuda a generar una imagen de forma cuadrada, dado que el modelo fue entrenado con imágenes de $30x30$. El tamaño del cuadrado es de $180x180$. Cada recorte fue adaptado a $30x30$.

El modelo previamente escogido (Sección 3.5) se reentrena con el fin de desarrollar un modelo robusto capaz de funcionar adecuadamente sin depender del tipo de cámara termográfica utilizada. Para lograr esto, se llevaron a cabo cuatro pruebas distintas que consisten en evaluar diversas combinaciones de conjuntos de imágenes para entrenamiento, validación y prueba:

- Conjunto A. El conjunto de entrenamiento incluye únicamente imágenes nuevas. Los conjuntos de prueba y validación provienen de la base de datos original.

- Conjunto B. Este conjunto de entrenamiento consta de imágenes nuevas. Los conjuntos de prueba y validación se componen de imágenes mixtas, es decir, imágenes adquiridas mediante el sistema embebido junto con imágenes de la base de datos original.

- Conjunto C. Todos los conjuntos, incluyendo entrenamiento, prueba y validación, están formados por imágenes nuevas.

- Conjunto D. El conjunto de entrenamiento, así como los conjuntos de prueba y validación, están compuestos por imágenes mixtas.

Asimismo, se aplicaron las mismas técnicas de aumento de datos que se mejor se desempeñaron al entrenar el modelo.

Etapa 2. El proceso de reentrenamiento del modelo se llevó a cabo de dos formas: utilizando el modelo original sin cambios y aplicando fine tuning. Se optó por usar fine tuning debido a la estructura del modelo y a la dimensión de la nueva base de datos, ya que se trata de una arquitectura pequeña y una base de datos que, aunque pequeña, es semejante a la original, lo que hace que el fine tuning sea la opción más adecuada para el entrenamiento. Cada modelo fue evaluada con un rango de épocas (10, 20, 30, 40, 50, 60, 70, 80, 90 y 100) y con la técnica de "Early stopping". El modelo final será aquel que muestre el mejor rendimiento en aspectos como precisión, exactitud, recall y f1-score.

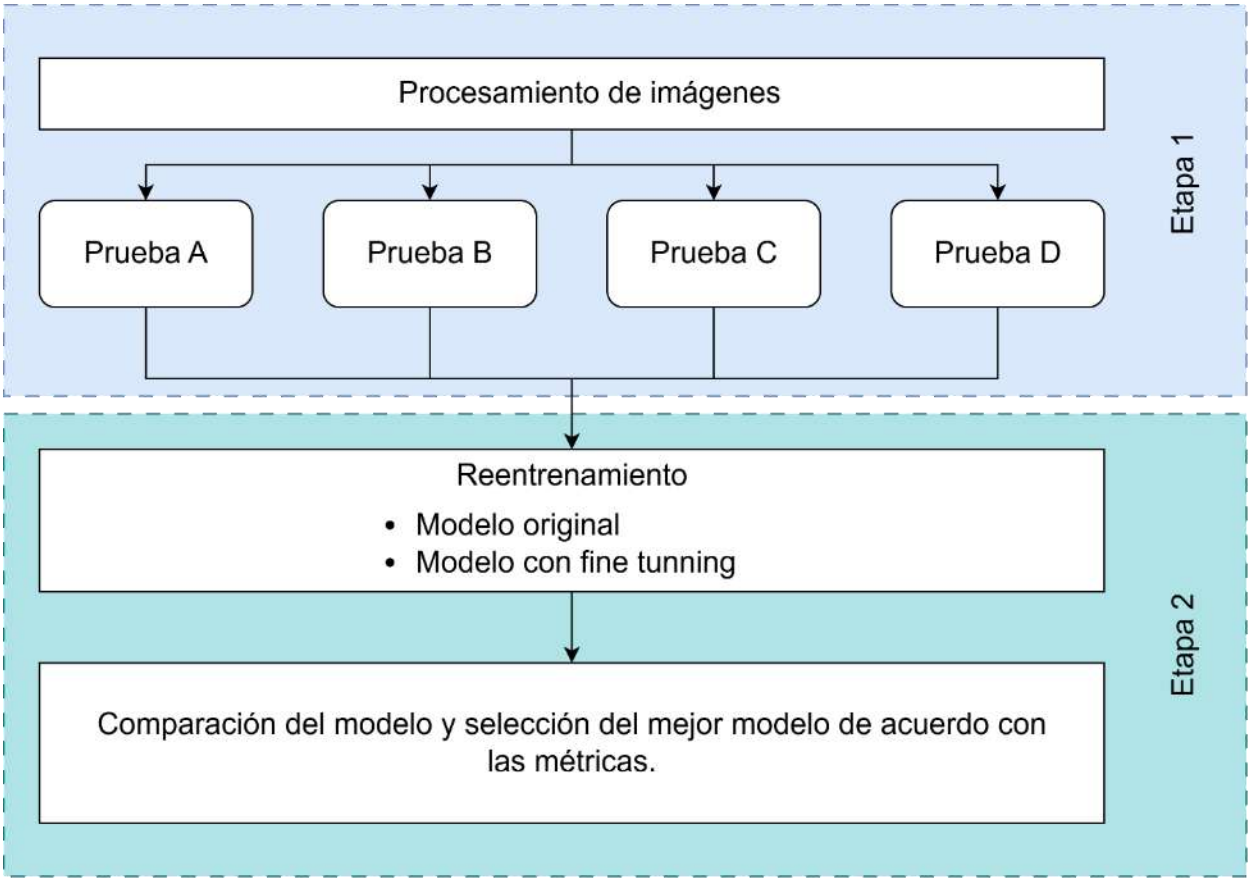

Figura 22. Metodología para el reentrenamiento del modelo. Elaboración propia.

CAPITULO 4

RESULTADOS

4.1. División de la base de datos

Las imágenes se redimensionaron a $30\ x\ 30$ píxeles. Los datos para el entrenamiento, la validación y la prueba se escogieron aleatoriamente, con un 75% de los datos para el entrenamiento, un 15% para la validación y un 10% para la prueba. Por último, para equilibrar el número de imágenes por clase, se tomaron 192 imágenes de la clase 0, 192 de la clase 1 y 192 de la clase 2. Por lo que la división final del conjunto de imágenes se obtuvo de la siguiente manera:

Tabla 4. División del conjunto de imágenes.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	429 imágenes	Sano: 143 Bilateral: 143 Unilateral: 143
Conjunto de validación	88 imágenes	Sano: 26 Bilateral: 32 Unilateral: 30
Conjunto de prueba	59 imágenes	Sano: 23 Bilateral: 17 Unilateral: 19

4.2. Aumento de datos

Es importante resaltar que las imágenes para el entrenamiento son imágenes originadas seleccionadas aleatoriamente más sus imágenes generadas por aumento de datos, mientras que el conjunto de imágenes utilizada para la validación y la prueba son todas imágenes originales, no se les aplicó ninguna técnica de aumento de datos.

Conjunto de imágenes aumentadas A. Este conjunto de imágenes está formado por la imagen original y dos imágenes adicionales generadas mediante técnicas de aumento de datos, como la rotación y el volteo de imágenes. La Figura 23 muestra un ejemplo de las imágenes del conjunto de datos, con la imagen original y las dos imágenes generadas.

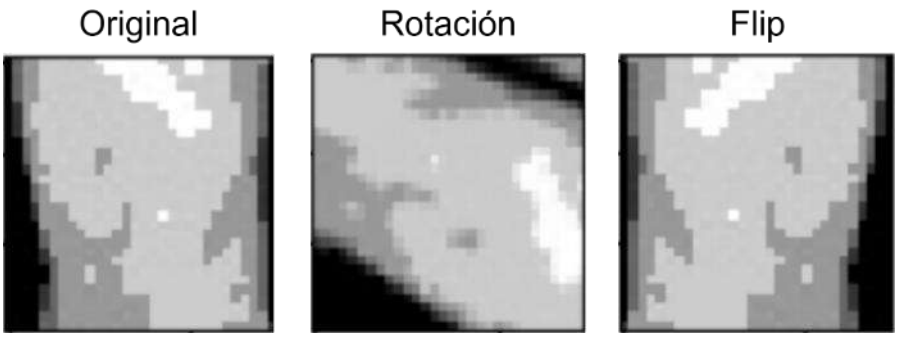

Figura 23. Ejemplo de imágenes del conjunto de imágenes aumentadas A. Elaboración propia

Conjunto de imágenes aumentadas B. Este conjunto de imágenes está formado por la imagen original y cinco imágenes adicionales generadas mediante cinco técnicas de aumento de datos, como la rotación de imágenes, el volteo de imágenes y la variación del contraste, el brillo y la saturación de las imágenes. La Figura 24 muestra un ejemplo de las imágenes del conjunto de datos, con la imagen original y las cinco imágenes generadas.

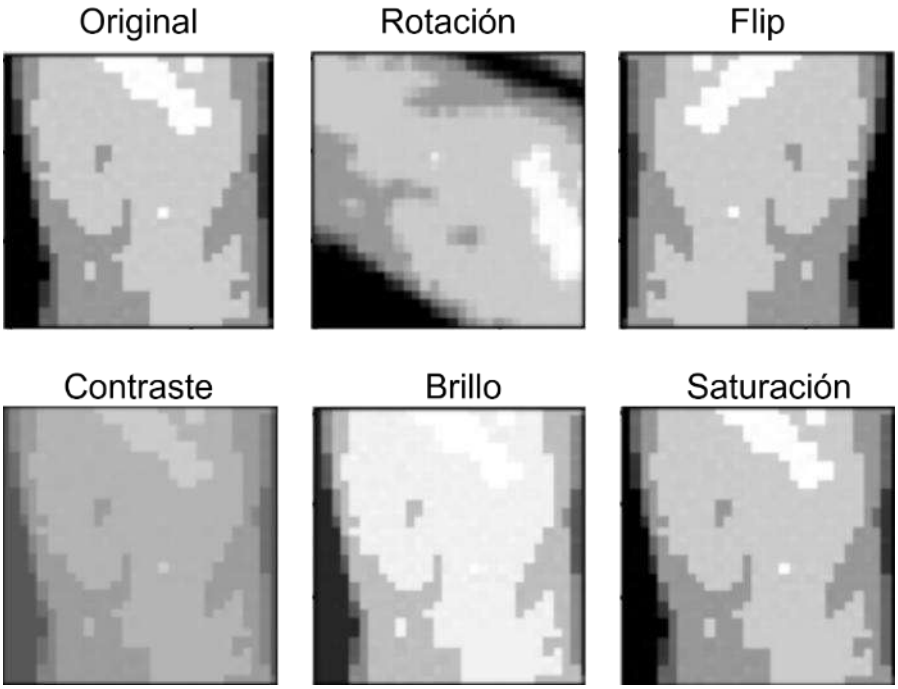

Figura 24. Ejemplo de imágenes del conjunto de imágenes aumentadas B. Elaboración propia

Conjunto de imágenes aumentadas C. Este conjunto de imágenes está formado por la imagen original y ocho imágenes adicionales generadas mediante cinco técnicas de aumento de datos, como la rotación de la imagen, el volteo de la imagen, la variación del contraste, el brillo y la saturación de las imágenes, y tres generadas mediante la combinación de técnicas aumentadas, en las que la imagen obtenida mediante la variación del contraste se giró y volteó, y se aplicó el mismo

procedimiento para la variación del brillo y la saturación. La Figura 25 muestra un ejemplo de las imágenes del conjunto de datos, con la imagen original y ocho imágenes generadas.

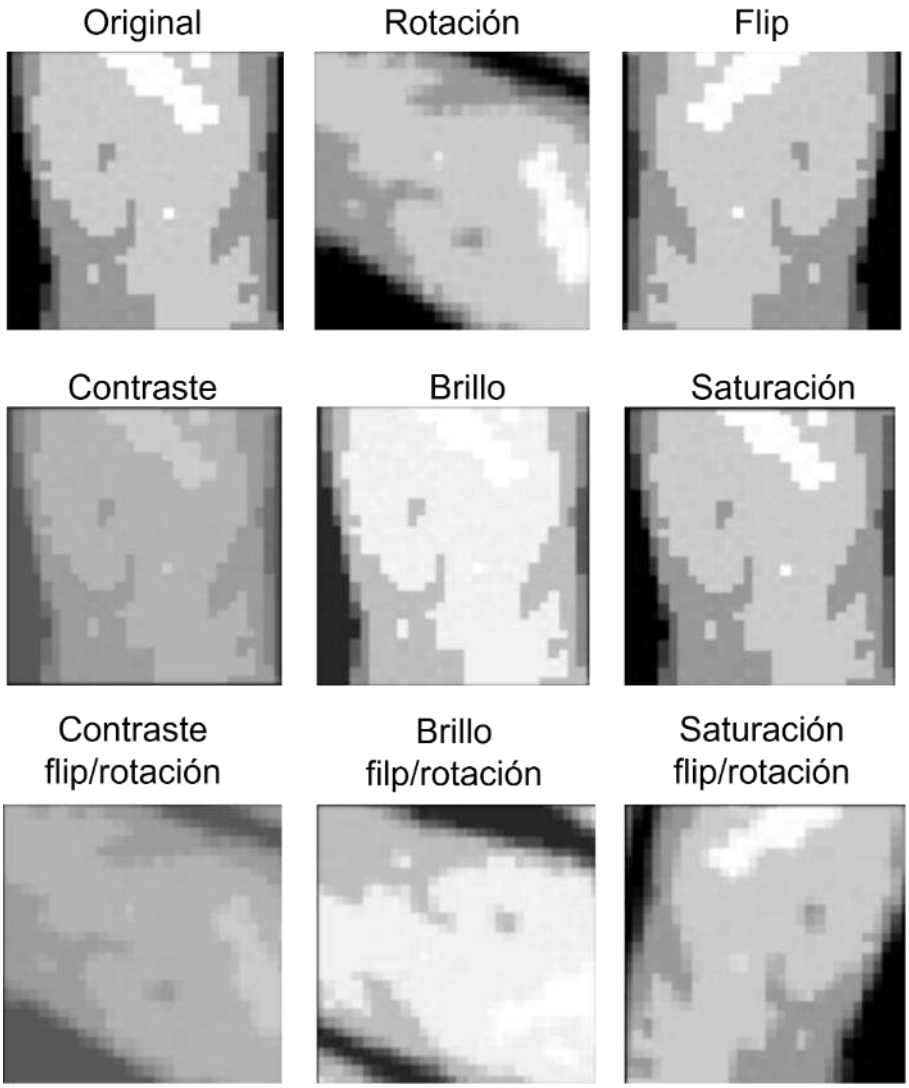

Figura 25. Ejemplo de imágenes del conjunto de imágenes aumentadas C. Elaboración propia

Dado que el aumento de datos sólo se aplicó al conjunto de entrenamiento, es el único conjunto que varía en tamaño, con 1287 generados para el conjunto de datos aumentado A, 2574 para el conjunto de datos aumentado B y 3861 para el conjunto de datos aumentado C. Los conjuntos de validación y prueba son los mismos en cada conjunto de imágenes. En la Tabla 5 se muestra la cantidad de imágenes utilizadas para cada conjunto de imágenes.

Tabla 5. Distribución de imágenes final usada para el entrenamiento de los modelos

Conjunto de entrenamiento usado	Núm. de imágenes para el conjunto de entrenamiento	Núm. de imágenes para el conjunto de validación	Núm. de imágenes para el conjunto de prueba
Conjunto de imágenes aumentadas A	1287 imágenes	88 imágenes	59 imágenes
Conjunto de imágenes aumentadas B	2574 imágenes	88 imágenes	59 imágenes
Conjunto de imágenes aumentadas C	3861 imágenes	88 imágenes	59 imágenes

4.3. Primeros modelos

Inicialmente varias pruebas fueron realizadas con diferente modelos de aprendizaje automático, con el propósito de seleccionar el mejor modelo. Para ello, se realizaron pruebas con tres modelos diferentes SVM, K-NN y CNN. Cada modelo se entrenó con los tres conjunto de imágenes aumentadas, además, las pruebas se realizaron con los datos normalizados y sin normalizar.

Los resultados obtenidos con el conjunto de imágenes aumentadas A se muestra en la Tabla 6. Con K-NN se obtuvo una presión de 100% con un valor de $k = 3$, se probaron más valores de k, sin embargo, a partir de 5, la precisión del modelo empieza a disminuir tanto con los datos normalizado como sin normalizar. Para SVM se muestra constante ante los datos, sin embargo, muestra un desempeño menor que K-NN, con un 89.83% de precisión. En contraste, el modelo CNN evidenció una mejora al aplicar normalización: la precisión aumentó de $93.22\ \%$ a $96.61\ \%$ y la función de pérdida disminuyó de 0.1546 a 0.1113.

Tabla 6. Precisión de cada modelo del conjunto A.

Modelo	***Precisión***	***Función de perdida***
KNN (k = 3) con normalización	100 %	–
KNN (k = 3)	100 %	–
SVM sin normalización	89.8305 %	–
SVM con normalización	89.8305%	–
CNN sin normalización	93.22 %	0.1546
CNN con normalización	96.61%	0.1113

Al analizar el rendimiento del modelo CNN por clase, Tabla 7, tanto con cómo sin normalización, se observa una mejora general al aplicar normalización de los datos. El modelo CNN sin normalizar se entrenó con una taza de aprendizaje de 0.001, 20 épocas y un batch size de 18.

Mientras que el modelo CNN con normalización fue con una taza de aprendizaje de 0.001, 60 épocas y 16 de batch size.

Tabla 7. Reporte de clasificación de los modelos CNN con el conjunto de imágenes aumentadas A.

Clase	***Modelo CNN sin normalización***			***Modelo CNN con normalización***		
	Precisión	Recall	F1-score	Precisión	Recall	F1-score
Sano	0.88	0.96	0.92	1.00	0.91	0.95
SP Bilateral	0.93	0.82	0.87	0.89	1.00	0.94
SP Unilateral	1.00	1.00	1.00	1.00	1.00	1.00

En la Tabla 8 se presentan los resultados obtenidos con el conjunto de imágenes aumentadas B. Para K-NN se alcanzó una precisión de 100% con un valor de $k = 3$. No obstante, se evaluaron otros valores de k, y a partir de $k = 11$, la precisión del modelo empieza a disminuir tanto con los datos normalizado como sin normalizar aunque la caída fue menos pronunciada que con el conjunto de imágenes A. En el caso de SVM se observó una mejora en la precisión respecto a los resultados del conjunto A, obteniendo una precisión de 93.22%, Sin embargo, con datos normalizados, la precisión descendió a 91.52%. Por su parte, el modelo convolucional también presentó un aumento en la precisión, llegando a 98.31%. No se observaron diferencias en la precisión al aplicar o no la normalización, aunque en la función de pérdida el mejor desempeño se obtuvo sin normalizar los datos.

Tabla 8. Precisión de cada modelo del conjunto B.

Modelo	***Precisión***	***Función de perdida***
KNN (k = 3) con normalización	100 %	–
KNN (k = 3)	100 %	–
SVM	93.2203 %	–
SVM con normalización	91.5254%	–
CNN sin normalización	98.31 %	0.0839
CNN con normalización	98.31%	0.0951

El modelo CNN sin normalizar se entrenó con una taza de aprendizaje de 0.001, 60 épocas y un batch size de 16. Mientras que el modelo CNN sin normalizar fue con una taza de aprendizaje de 0.001, 20 épocas y 16 de batch size. En la Tabla 9, se muestra el reporte de clasificación por clase de cada modelo, donde el resultado es el mismo para ambos modelos, pero es importante resaltar que la clase "Síndrome patelofemoral unilateral" es el que se tiene un mejor desempeño. Aunque, la función de perdida fue un poco mayor en el modelo normalizado, no presenta una disminución en el desempeño del modelo.

Tabla 9. Reporte de clasificación de los modelos CNN con el conjunto de imágenes aumentadas B.

Clase	***Modelo CNN sin normalización***			***Modelo CNN con normalización***		
	Precisión	Recall	F1-score	Precisión	Recall	F1-score
Sano	0.96	1.00	0.98	0.96	1.00	0.98
SP Bilateral	1.00	0.94	0.97	1.00	0.94	0.97
SP Unilateral	1.00	1.00	1.00	1.00	1.00	1.00

Los resultados del conjunto de imágenes aumentadas C se muestran en la Tabla 10. El modelo K-NN obtuvo una precisión del 100%, con un valor de $k = 3$, la precisión empieza a disminuir a partir del valor de $k = 11$. SVM disminuyó bastante su precisión obteniendo un 76.27% sin normalizar y 59.32% con los datos normalizados. Del mismo modo, sucede con el modelo CNN con normalización donde la precisión disminuyo a 96.61%, sin embargo, con normalización el modelo obtuvo el mejor desempeño con una precisión de 100%.

Tabla 10. Precisión de cada modelo del conjunto C.

Modelo	Precisión	Función de perdida
KNN (k = 3) con normalización	100 %	–
KNN (k = 3)	100 %	–
SVM	76.2711 %	–
SVM con normalización	59.3220%	–
CNN sin normalización	96.61	0.0639
CNN con normalización	100 %	0.0119

El modelo CNN sin normalizar se entrenó con una taza de aprendizaje de 0.001, 20 épocas y un batch size de 18. Mientras que el modelo CNN con normalización fue con una taza de aprendizaje de 0.001, 60 épocas y 16 de batch size. En la Tabla 11, se muestra un aumento tanto de precisión, Recall y f1-score en las tres clases.

Tabla 11. Reporte de clasificación de los modelos CNN con el conjunto de imágenes aumentadas C.

Clase	***Modelo CNN sin normalización***			***Modelo CNN con normalización***		
	Precisión	Recall	F1-score	Precisión	Recall	F1-score
Sano	1.00	0.91	0.95	1.00	1.00	1.00
SP Bilateral	0.89	1.00	0.94	1.00	1.00	1.00
SP Unilateral	1.00	1.00	1.00	1.00	1.00	1.00

Los resultados pasados muestran que el modelo K-NN se presenta constante ante los cambios del conjunto de datos de entrenamiento (tanto las imágenes aumentadas como con la normalización). Mientras que el modelo SVM es más susceptible a los cambios obteniendo el mejor desempeño

con el conjunto de datos aumentados B. Sin embargo, ninguno de los dos modelos es viable para la problemática que se está tratando. SVM principalmente es utilizada para clasificaciones binarias. K-NN puede tener un mal rendimiento en tiempo de ejecución para grandes conjuntos de entrenamiento, además ante espacios con alta dimensionalidad la distancia entre cada dato ya no es tan significativas debido a que los datos ya se encuentran lejos entre sí. Por lo que, el modelo que mejor se desempeña con la base de datos es la red neuronal convolucional, donde la normalización mejorar o mantiene igual el desempeño del modelo. Sin embargo, para la selección de la mejor configuración del modelo CNN, se realizaron varias pruebas empezando desde una CNN sencilla y cambiando varios hiperparametros, como se realizaron en la investigación de (Trejo-Chavez et al., 2022)

4.4. Hiperparametros tuning

Antes de pasar a la selección del modelo, es importante destacar que la función de pérdida utilizada tanto para la elección de la arquitectura y su configuración, como para el entrenamiento del modelo fue Sparse Categorical Crossentropy. Dado que las clases fueron codificadas como: “sano” : 0, “síndrome patelofemoral bilateral” : 1, y “síndrome patelofemoral unilateral” : 2, y considerando que se trata de un problema de clasificación multiclase, esta función de pérdida resulta adecuada. Sparse Categorical Crossentropy se emplea en problemas de clasificación multiclase cuando las etiquetas están codificadas como enteros (en lugar de codificación one-hot) y presenta una ventaja computacional al ser más ligera. El optimizador utilizado fue Adam, un algoritmo ampliamente usado en tareas de clasificación por combinar las ventajas de los métodos Momentum y RMSprop.

Las pruebas para obtener el modelo CNN con mejor desempeño fueron realizadas de manera experimental, donde se probaron dos arquitecturas de redes neuronal convolucional sencillas: La primera arquitectura de CNN (CNN_A1) consta de una capa convolucional y una capa de agrupamiento. Mientras la segunda arquitectura de CNN (CNN_A2) está compuesta por dos capas convolucionales y una capa de agrupamiento.

Ambos modelos ($CNN\_A1\ y\ CNN\_A2$) se analizaron usando diferentes configuraciones con el objetivo de seleccionar la mejor estructura de CNN. Se evaluaron combinaciones de tamaños de filtro ($3 \times 3, 4 \times 4, 5 \times 5, 6 \times 6$) y cantidades de filtros ($4, 6, 8, 10, 12$), en ambos modelos. El tamaño del filtro determina la capacidad del modelo para capturar patrones en las imágenes: los

filtros más pequeños permiten detectar detalles finos como bordes, mientras que los más grandes pueden capturar patrones más complejos y de mayor escala. Por otro lado, la cantidad de filtros define el número de mapas de características que se generarán. Cada filtro extrae una característica distinta de la imagen, por lo que a mayor cantidad de filtros, mayor riqueza representacional del modelo. Sin embargo, esto también conlleva un aumento en el costo computacional.

El entrenamiento del modelo solo fue realizado con el conjunto de imágenes aumentadas A, y los hiperparametros de entrenamiento como tamaño de la imagen de entrada, la taza de aprendizaje (learning rate), las épocas y el tamaño de lote (batch size) permanecieron igual durante cada prueba de las combinaciones. Los valores asignados fueron los siguientes: 30×30 al tamaño de imagen de entrada, un learning rate de 0.0001, 10 épocas, y un batch size de 8.

La selección del modelo fue realizada en términos de la exactitud y la función de pérdida del modelo. En la Tabla 12 se muestra los resultados de las pruebas de la primera arquitectura (CNN_Simple_A1). El mejor resultado obtenido, subrayado en naranja, para esta arquitectura es con la combinación de número de filtros y tamaño de filtro de 12 y 3×3, respectivamente, con una exactitud de 94.92% y una función de pérdida de 0.2447. En términos generales, esta arquitectura obtuvo una exactitud de entre $84.75\% - 91.53\%$, donde el peor desempeño se obtiene con la combinación de número de filtros: 4, y tamaño de filtros 3×3; con una exactitud de 76.27%. De lo contrario, la función de perdida tuvo una mayor variación, donde el valor máximo que se obtuvo fu de 1.1086 y el menor de 0.2416.

Tabla 12. Pruebas del modelo 1 capas convolucional, 1 capa de agrupamiento (CNN _A1).

Número de experimento	Learning rate	Batch size	Épocas	Número de filtro	Tamaño de filtro	Test Accuracy (%)	Test loss
1	0.0001	8	10	4	3×3	76.27	0.8763
2	0.0001	8	10	6	3×3	84.75	0.9963
3	0.0001	8	10	8	3×3	89.83	0.7656
4	0.0001	8	10	10	3×3	84.75	0.5283
5	0.0001	8	10	12	3×3	94.92	0.2447
6	0.0001	8	10	4	4×4	84.75	0.7547
7	0.0001	8	10	6	4×4	84.75	1.0936
8	0.0001	8	10	8	4×4	91.53	0.2548
9	0.0001	8	10	10	4×4	88.14	0.2315
10	0.0001	8	10	12	4×4	89.83	0.4420
11	0.0001	8	10	4	5×5	84.75	1.1086
12	0.0001	8	10	6	5×5	91.53	0.2416
13	0.0001	8	10	8	5×5	84.75	0.4045
14	0.0001	8	10	10	5×5	89.83	0.8366
15	0.0001	8	10	12	5×5	91.53	0.3181

16	0.0001	8	10	4	6×6	84.75	0.6370
17	0.0001	8	10	6	6×6	88.14	0.3789
18	0.0001	8	10	8	6×6	84.75	0.5727
19	0.0001	8	10	10	6×6	91.53	0.4710
20	0.0001	8	10	12	6×6	88.14	0.2833

Los resultados de las pruebas correspondientes a la segunda arquitectura se presentan en la Tabla 13. Los valores de exactitud varían de $71.19\% - 94.92\%$. A diferencia de la arquitectura anterior, en esta se registró un valor mínimo de exactitud más bajo, aunque se identificaron más configuraciones con valores elevados. Un patrón similar se observó en la función de pérdida, cuya máxima fue de 1.0294 y la mínima es de 0.1237. Según el análisis, tres configuraciones alcanzaron una exactitud del 94.92%, todas con una función de pérdida considerablemente baja. No obstante, la configuración con la menor función de pérdida de 0.1237, corresponde al modelo con un tamaño de filtro 5×5 y un número de filtro de 12, resaltada en color amarillo en la tabla, mientras que las otras dos configuraciones con igual exactitud se muestran en color naranja.

Tabla 13. Pruebas del modelo 2 capas convolucional, 1 capa de agrupamiento (CNN_ A2),

Número de experimento	Learning rate	Batch size	Épocas	Número de filtro	Tamaño de filtro	Test Accuracy (%)	Test loss
1	0.0001	8	10	4	3×3	76.27	1.0294
2	0.0001	8	10	6	3×3	71.19	0.9098
3	0.0001	8	10	8	3×3	88.14	0.2701
4	0.0001	8	10	10	3×3	89.83	0.5613
5	0.0001	8	10	12	3×3	91.53	0.3367
6	0.0001	8	10	4	4×4	89.83	0.1854
7	0.0001	8	10	6	4×4	81.36	0.9868
8	0.0001	8	10	8	4×4	86.44	0.4058
9	0.0001	8	10	10	4×4	89.83	0.2652
10	0.0001	8	10	12	4×4	84.75	0.4885
11	0.0001	8	10	4	5×5	79.66	0.5932
12	0.0001	8	10	6	5×5	91.53	0.2230
13	0.0001	8	10	8	5×5	88.14	0.4251
14	0.0001	8	10	10	5×5	94.92	0.1499
15	0.0001	8	10	12	5×5	94.92	0.1237
16	0.0001	8	10	4	6×6	76.27	0.8639
17	0.0001	8	10	6	6×6	79.66	0.5017
18	0.0001	8	10	8	6×6	81.36	0.5229
19	0.0001	8	10	10	6×6	94.92	0.1685
20	0.0001	8	10	12	6×6	88.14	0.2115

En la Figura 26 se muestra una representación visual de las pruebas, la primera gráfica pertenece a la primera arquitectura (CNN_Simple_A1), y la segunda gráfica a la segunda arquitectura (CNN_Simple_A2). Ambas graficas están divididas en bloques, representando el número de filtro, mientras que cada barra del bloque representa el número de filtro. Por lo que, la arquitectura seleccionada de acuerdo con las métricas es la segunda (CNN_Simple_A2), y la configuración

seleccionada por su alta exactitud y baja función de perdida corresponde al número de experimento 15, subrayado de naranja, con tamaño de filtro 5×5 y numero de filtros igual a 12.

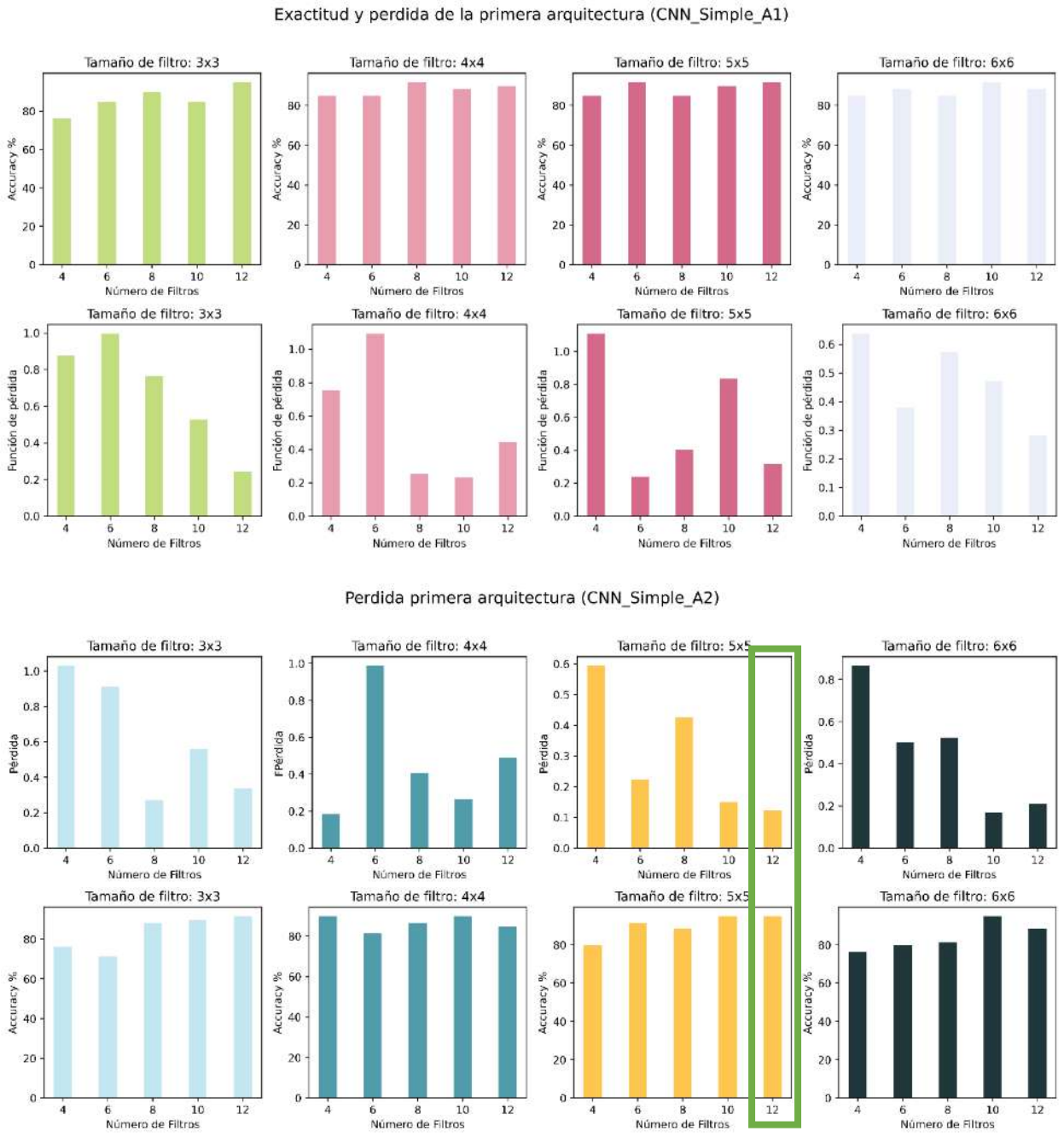

Figura 26. Experimentos para obtener el modelo de arquitectura. Elaboración propia

La arquitectura final consta de dos capas convolucionales seguidas de una activación ReLU (Rectified Linear Unit) para introducir no linealidad y mejorar la extracción de características. Las capas convolucionales aplican 12 filtros de tamaño 5×5 a las imágenes de entrada que tienen un

tamaño de 30×30 pixeles , lo que da como resultado mapas de características de salida con un tamaño de $26 \times 26 \times 12$ de la primera capa convolucional y $22 \times 22 \times 12$ de la segunda capa convolucional. A continuación, se utiliza el Average Pooling (capa de agrupación) con un tamaño de 2×2 para reducir las dimensiones de los mapas de características a $11 \times 11 \times 12$. El resultado se suaviza y se utiliza como entrada para las capas totalmente conectadas (densas). La primera capa densa reduce el vector de características a 128 unidades para que la red pueda aprender y representar patrones complejos. La salida final es procesada por una capa softmax que genera distribuciones de probabilidad sobre las tres clases: clase 0 (sano), clase 1 (síndrome patelofemoral bilateral) y clase 2 (síndrome patelofemoral unilateral). La arquitectura detallada se muestra en la Figura 27 e ilustra la estructura y el flujo de la red.

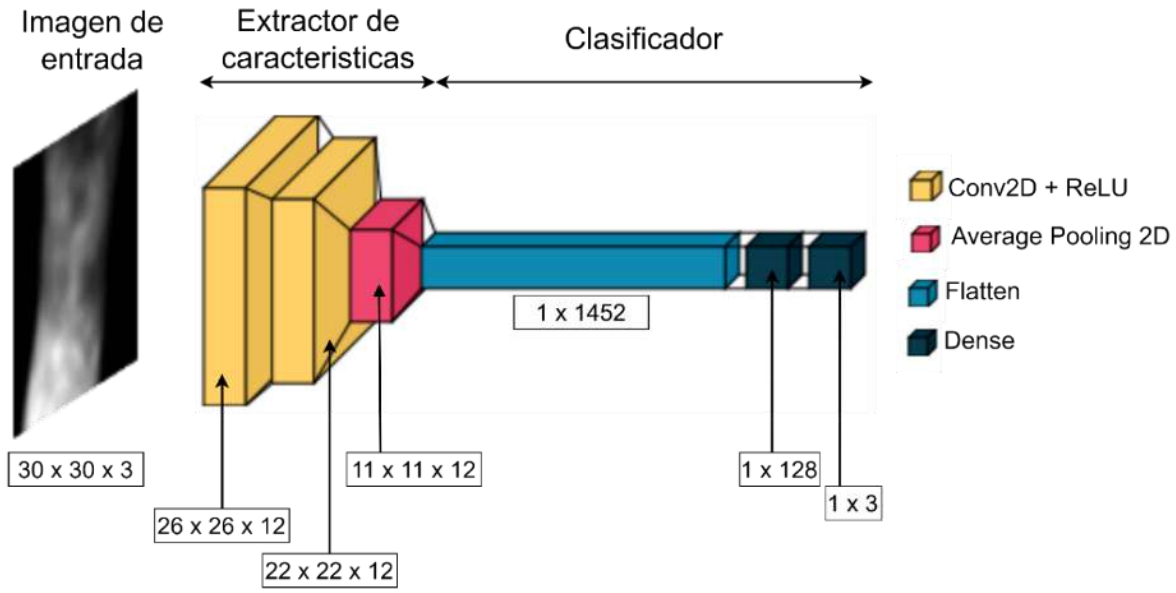

Figura 27. Arquitectura del modelo CNN. Elaboración propia.

Así, se obtuvo la arquitectura final del modelo CNN, que incluye 2 capas convolucionales, 1 capa de agrupamiento, 1 capa de aplanamiento, 1 capa densa y 1 capa densa final con activación softmax para la clasificación. Además, la Tabla 14 resume la arquitectura final.

Tabla 14. Arquitectura final del modelo CNN

Nombre de la capa	Tipo de capa	Salida
Input Layer	Image input	$30 \times 30 \times 3$
Layer_1_Convolutional	Convolucional	$26 \times 26 \times 12$
Layer_2_Convolutional	Convolucional	$22 \times 22 \times 12$
Layer_1_Pooling	Average Pooling	$11 \times 11 \times 12$
Flatten_49	Flatten	$None \times 1452$
Layer_1_Dense	Dense	$None \times 128$
Layer_2_Dense	Dense	$None \times 3$

4.5. Entrenamiento del modelo

El entrenamiento del modelo CNN se realizó experimentalmente cambiando los valores de varios hiperparametros como la tasa de aprendizaje, el tamaño del lote y las épocas. Los valores utilizados para la tasa de aprendizaje fueron $0{,}0001\ y\ 0{,}001$; para el tamaño de lote fueron: $8, 10\ y\ 12$; y para las épocas fueron: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$.

Este experimento se realizó con los 3 conjuntos de imágenes aumentadas con el fin de comparar cuál de estos 3 conjuntos da un mejor resultado al evaluar el modelo con el conjunto de prueba. Para la selección del mejor modelo, primero se seleccionaron los mejores modelos obtenidos de cada conjunto de imágenes aumentadas y se compararon entre ellos de acuerdo con sus métricas, teniendo en cuenta que la exactitud del conjunto de prueba y del conjunto de validación fueran lo más cercanos posible, para evitar la selección de un modelo de sobreajuste, que tenga una exactitud alta ante el conjunto de validación pero bajo con el conjunto de prueba, o viceversa. Además para una mejor selección se evaluaron los modelos con el conjunto de imágenes aumentadas con el que no se entrenó, finalmente se seleccionó el que obtuvo mejores resultados.

Los resultados de cada evaluación se muestran en el Anexo A, son 6 tablas que contienen información acerca de la exactitud (Accuracy) y la función de pérdida (Loss) de la última época de entrenamiento, junto con su validación. Y la exactitud y función de pérdida del conjunto de prueba. A continuación su muestra un resumen de las tablas, en la Figura 28 se muestra los resultados en términos de exactitud y función de perdida de la combinación (tasa de aprendizaje: 0.001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: $8, 10, 12$; y del conjunto de imágenes aumentas A). Donde el color naranja representa el batch size de 8, el color verde de 10, y el color rojo de 12. Las líneas continuas son de los resultados del conjunto de prueba, mientras que las líneas punteadas son del conjunto de validación. En la Figura 28a se muestran los resultados de la exactitud el valor máximo es de 98% aproximadamente, en la época 20 con batch size de 8; y en la época 50 con batch size de 12. Sin embargo, en la época 50 su exactitud de validación es de 97%, mientras que el de la época 20 es más bajo de 94%. En la Figura 28b, se muestra la función de pérdida durante cada época, se observa que durante la época 20 y 50 los valores son cercanos a 0, lo que indica que el ajuste de datos es bueno.

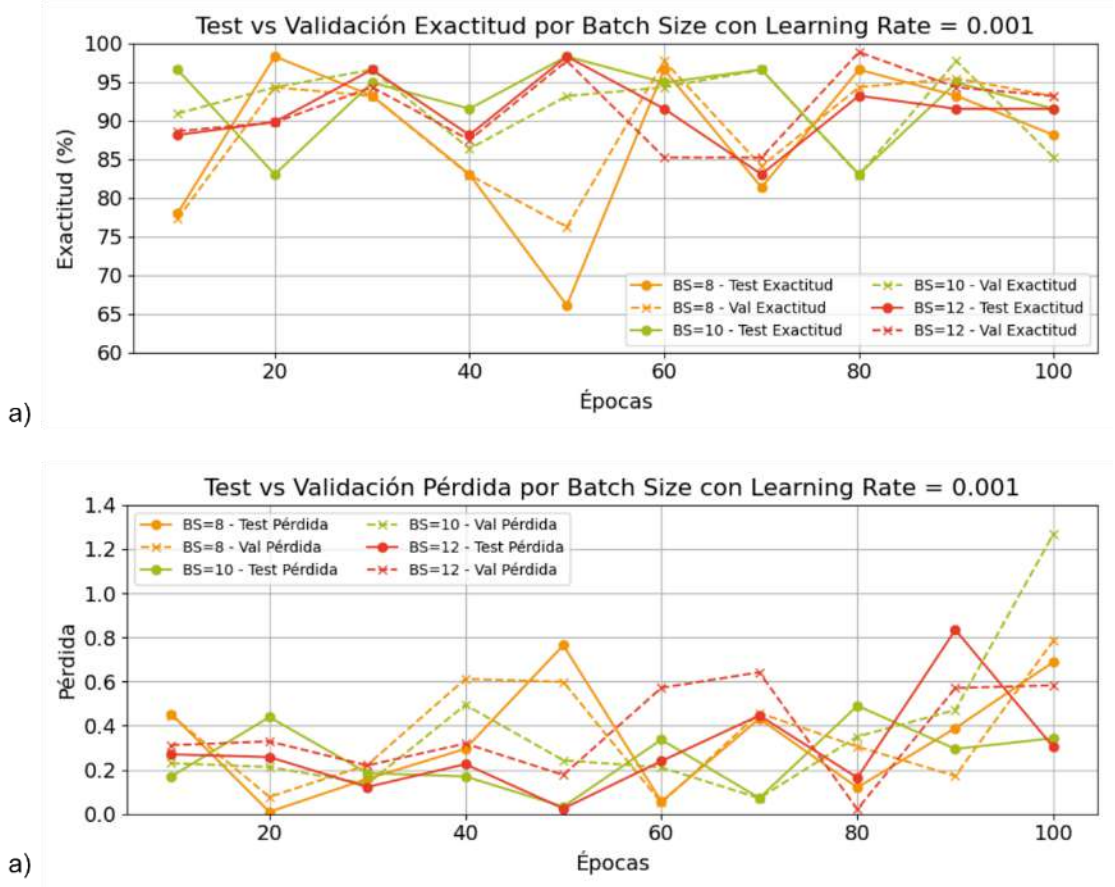

Figura 28. Resultados en términos de perdida y exactitud de los modelos del conjunto A (con learning rate 0.001).

En la Figura 29 se muestra los resultados en términos de exactitud y función de perdida de la combinación (tasa de aprendizaje: 0.0001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: 8,10,12; y del conjunto de imágenes aumentas A). Donde el color azul representa el batch size de 8, el color amarillo de 10, y el color rosa de 12. Las líneas continuas son de los resultados del conjunto de prueba, mientras que las líneas punteadas son del conjunto de validación. En la Figura 29a se muestran los resultados de la exactitud el valor máximo es de 98% aproximadamente, en la época 90 con batch size de 8; y en la época 30 con batch size de 10. Sin embargo, en la época 90 su exactitud de validación es de 93%, mientras que el de la época 30 es más bajo de 88%. En la Figura 29b, se muestra la función de pérdida durante cada época, se observa que durante la época 90 y 30 los valores son cercanos a 0, mientras que el loss de validación esta más alto entre $0.2 - 0.4$.

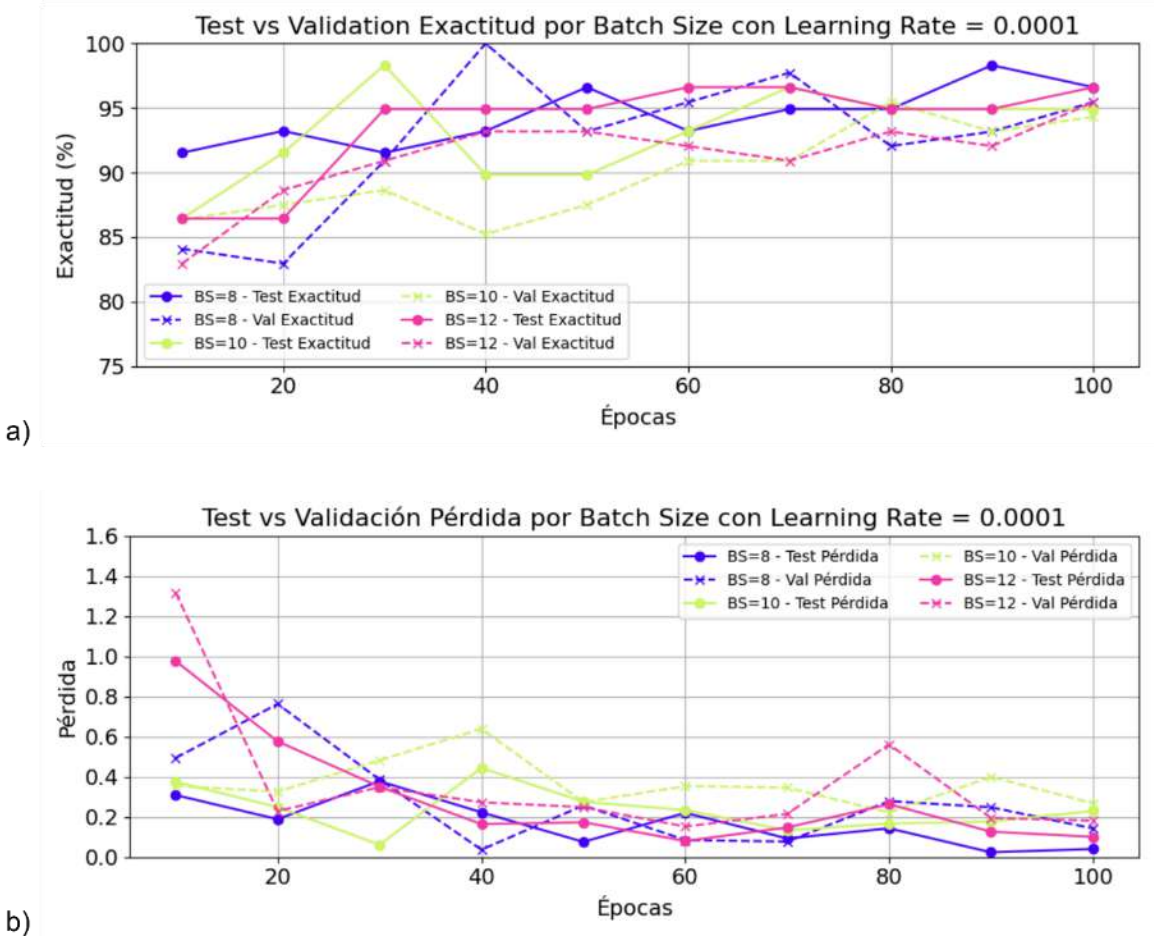

Figura 29. Resultados en términos de perdida y exactitud de los modelos del conjunto A (con learning rate 0.0001).

En la Tabla 15 se muestran los mejores modelos del conjunto de imágenes aumentadas A, el mejor modelo esta subrayado de amarillo, con la configuración de tasa de aprendizaje de 0.001, batch size de 12 y 50 épocas. Debido a que los demás modelos presentan una diferencia significativa entre la exactitud del conjunto de validación y del de prueba lo que indica que el modelo no se está ajustando de manera adecuado a la base de datos

Tabla 15. Mejores modelos del conjunto de imágenes aumentadas A.

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	8	20	94.32	0.0776	98.31	0.01
2	0.001	12	50	97.73	0.1769	98.31	0.0249
3	0.0001	8	90	93.18	0.2479	98.31	0.0236
4	0.0001	10	30	88.64	0.4822	98.31	0.0613

La Figura 30 se muestra los resultados en términos de exactitud y función de perdida de la combinación (tasa de aprendizaje: 0.001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: $8, 10, 12$; y del conjunto de imágenes aumentas B). En la Figura 30a se muestran los

resultados de la exactitud el valor máximo es de 100% aproximadamente, en la época 60 con batch size de 8 y una exactitud de validación de 100%; en la época 40 con batch size de 10 con exactitud de validación de 96%; y en la época 40 con batch size de 12 con exactitud de validación de 98%. En la única que se mantiene la misma exactitud es en la primera, en las demás disminuye. En la Figura 30b, se muestra la función de pérdida durante cada época, se observa que valores son cercanos a 0.

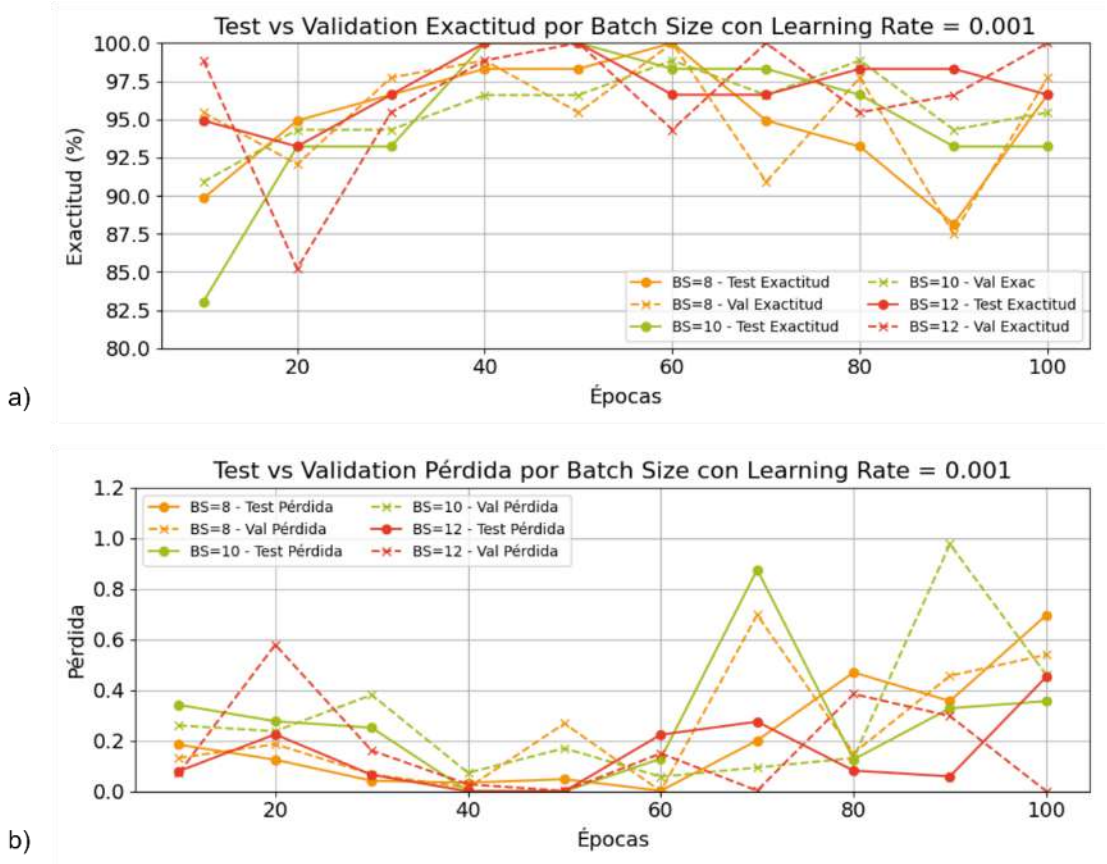

Figura 30. Resultados en términos de perdida y exactitud de los modelos del conjunto B (con learning rate 0.001).

La Figura 31 se muestra los resultados en términos de exactitud y función de perdida de la combinación (tasa de aprendizaje: 0.0001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: $8, 10, 12$; y del conjunto de imágenes aumentas B). En la Figura 31a se muestran los resultados de exactitud, el valor máximo es de 100% aproximadamente, en la época 90 con batch size de 8 y una exactitud de validación de 98%; en la época 100 con batch size de 10 con exactitud de validación de 96%; y en la época 40 con batch size de 12 con exactitud de validación de 97%. En todas, la exactitud de validación disminuye a comparación de la exactitud de prueba. En la

Figura 31b, se muestra la función de pérdida durante cada época antes mencionada, se observa que los valores son cercanos a 0.

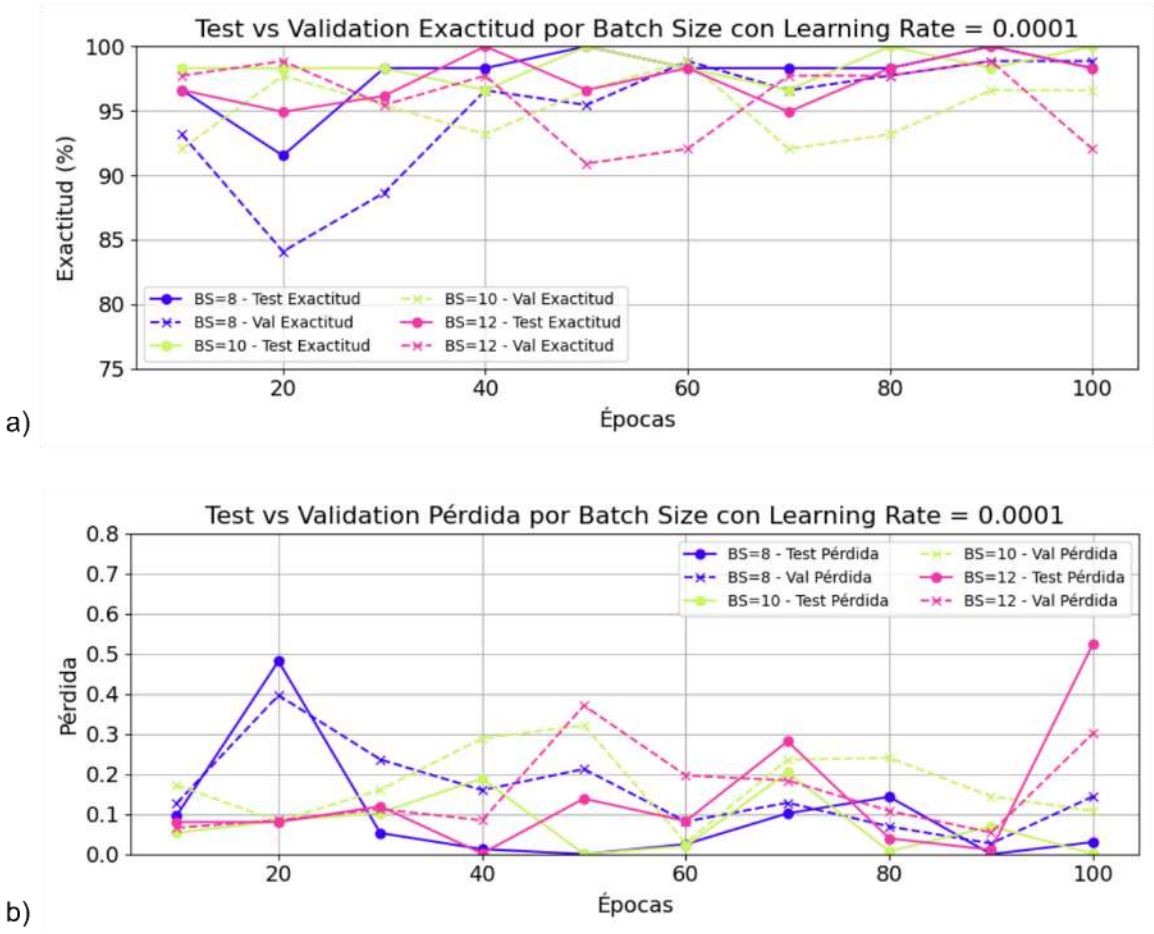

Figura 31. Resultados en términos de perdida y exactitud de los modelos del conjunto B (con learning rate 0.0001).

En la Tabla 16 se muestran los mejores modelos del conjunto de imágenes aumentadas B, el mejor modelo es el subrayado de amarillo, con la configuración de tasa de aprendizaje de 0.001, batch size de 8 y 60 épocas. Al igual que con el conjunto de imágenes aumentada A, los demás modelos presentan una diferencia entre la exactitud del conjunto de validación y del de prueba lo que indica que el modelo no se está ajustando de manera adecuado a la base de datos en general, sino que se adecua mejor al conjunto de prueba que a los demás conjuntos.

Tabla 16. Mejores modelos del conjunto de imágenes aumentadas B.

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	8	60	100	0.0005	100	0.0023
2	0.001	10	40	96.59	0.0732	100	0. 001
3	0.001	12	40	98.86	0.0279	100	0.0002
4	0.0001	8	90	98.86	0.0264	100	0.0001

5	0.0001	10	100	96.59	0.1444	100	0.001
6	0.0001	12	40	97.73	0.0845	100	0.0018

Para el conjunto de imágenes aumentada C, los resultados se presentan en la Figura 32, en términos de exactitud y función de pérdida, considerando la combinación de parámetros (tasa de aprendizaje: 0.001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: $8, 10, 12$; y del conjunto de imágenes aumentas C). En la Figura 32a se muestran los resultados de la exactitud. El valor máximo, cercano al 100% se alcanzó en la época 80 con batch size de 10 y una exactitud de validación de 97%; en la época 30 con batch size de 12 y en la época 100 con batch size de 12 ambos con exactitud de validación de 100%. La única disminución notable en la exactitud se presentó en la primera; en las demás, la exactitud del conjunto de validación se mantuvo estable. En la Figura 32b, se muestra la evolución de la función de pérdida a lo largo de las épocas, observándose valores consistentemente cercanos a cero.

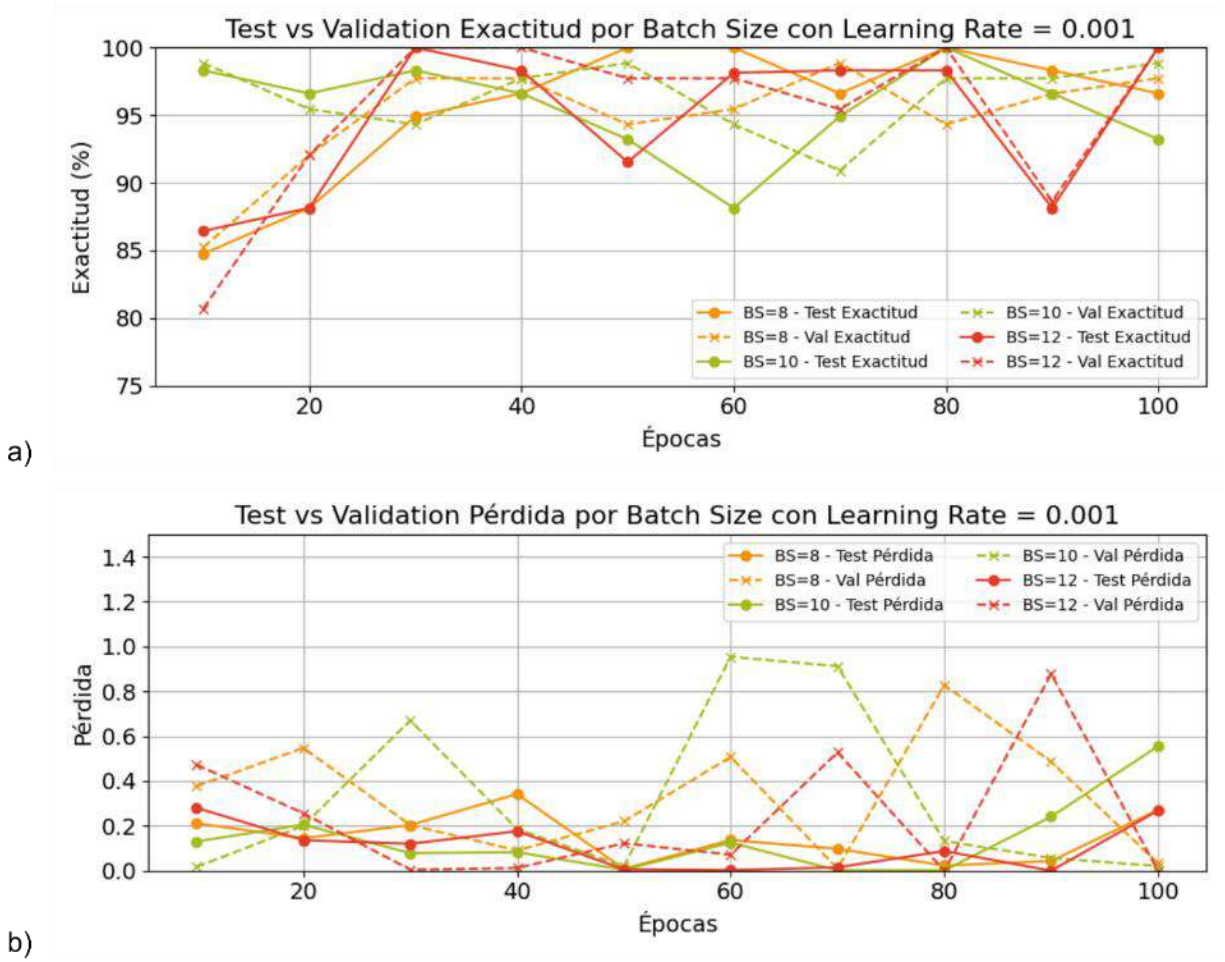

Figura 32. Resultados en términos de perdida y exactitud de los modelos del conjunto C (con learning rate 0.001).

La Figura 33 se muestra los resultados en términos de exactitud y función de perdida de la combinación (tasa de aprendizaje: 0.0001, épocas: $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$; batch size: $8, 10, 12$; y del conjunto de imágenes aumentas C). En la Figura 33a se muestran los resultados de la exactitud el valor máximo es de 100% aproximadamente, en la época 50 con batch size de 8; en la época 70 con batch size de 10; y en la época 90 con batch size de 12. En todas la exactitud de validación se mantiene igual que la exactitud de prueba, es decir, es de 100%. En la Figura 33b, se muestra la función de pérdida durante cada época antes mencionada, se observa que valores son cercanos a 0.

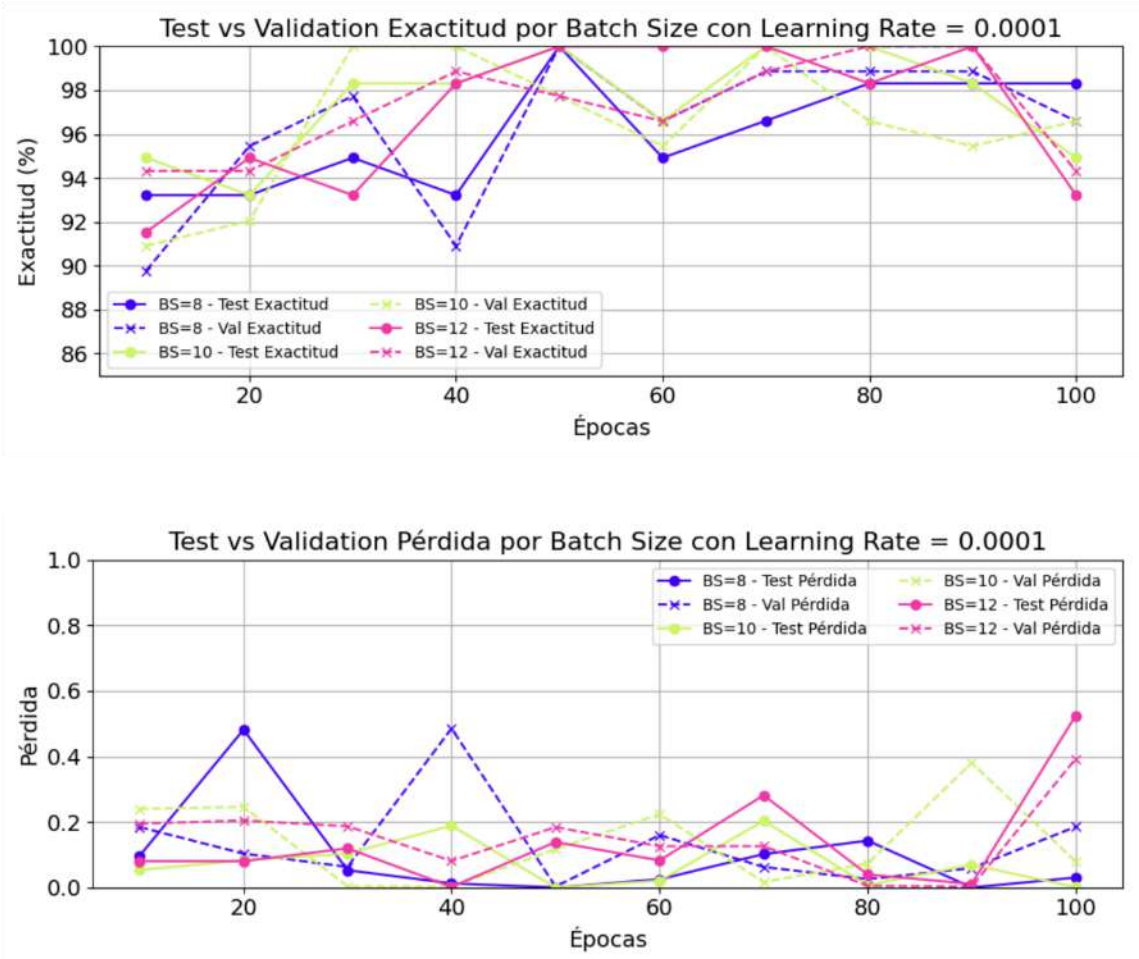

Figura 33. Resultados en términos de perdida y exactitud de los modelos del conjunto C (con learning rate 0.0001)

En la Tabla 17 se muestran los mejores modelos del conjunto de imágenes aumentadas C. A diferencia de los demás modelos de los conjuntos de imágenes aumentadas, estos modelos tuvieron una exactitud igual tanto de la validación como de la prueba, a excepción del experimento 1. Sin

embargo, para la selección del mejor se tomaron en cuenta varios factores, como que la función de pérdida fuera lo más cercana a 0, además; para tomar la decisión se tuvieron en cuenta los mapas de calor obtenidos de cada modelo y obtuvo mejores resultados ante los conjuntos de imágenes aumentadas A y B; por lo que se escogió el modelo subrayado de amarillo, con la configuración de tasa de aprendizaje de 0.0001, batch size de 12 y 90 épocas

Tabla 17. Mejores modelos del conjunto de imágenes aumentadas C.

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	10	80	97	0.1326	100	0.0
2	0.001	12	30	100	0.0048	100	0.002
3	0.001	12	100	100	0.0001	100	0
4	0.0001	8	50	100	0.003	100	0.0086
5	0.0001	10	70	100	0.0164	100	0.0018
6	0.0001	12	90	100	0.0023	100	0.0002

Los mejores modelos de cada conjunto de imágenes aumentadas se muestran en la Tabla 18, de los cuales a primera impresión el modelo del conjunto C es el mejor en función de las métricas de exactitud y precisión.

Tabla 18. Resultados de los mejores modelos de cada conjunto de imágenes aumentadas.

Conjunto de imágenes aumentadas	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
Conjunto de imágenes aumentadas *A*	97.73	0.1769	98.31	0.0249
Conjunto de imágenes aumentadas *B*	100	0.0005	100	0.0023
Conjunto de imágenes aumentadas *C*	100	0.0023	100	0.0002

Sin embargo, para hacer una selección final, cada modelo se comparó con los conjuntos de imágenes aumentadas con los que no fueron entrenados, como se muestra en la Tabla 19. El modelo del conjunto A, se probó con los conjuntos de imágenes aumentadas B y C. El modelo del conjunto B, con los conjuntos de imágenes aumentadas A y C. Y el modelo del conjunto C, con los conjuntos de imágenes aumentadas B y C. El modelo del conjunto de imágenes aumentadas C tiene un mejor desempeño al probarlo con los otros diferentes conjuntos, obteniendo un 96.58 % ante el conjunto A pero con una alta función de perdida, y para el conjunto B obtuvo una exactitud de 98.17% y una función de perdida menor. Mientras que los demás modelos no tuvieron un buen desempeño> el modelo del conjunto A fue el peor con una exactitud menor de 88% y el modelo del conjunto B obtuvo una exactitud de menor del 94%.

Tabla 19. Resultados de probar el mejor modelo de cada conjunto de imágenes con los demás conjuntos.

Conjunto de imágenes aumentadas de prueba	Accuracy (%)	Loss
Conjunto de imágenes aumentadas A		
Conjunto de imágenes aumentadas B	87.68	0.7729
Conjunto de imágenes aumentadas C	79.77	1.2895
Conjunto de imágenes aumentadas B		
Conjunto de imágenes aumentadas A	93.55	0.8430
Conjunto de imágenes aumentadas C	88.27	1.1942
Conjunto de imágenes aumentadas C		
Conjunto de imágenes aumentadas A	96.58	0.3036
Conjunto de imágenes aumentadas B	98.17	0.1259

En general los mejores resultados de los modelos evaluados se obtuvieron con el conjunto de imágenes aumentadas C. La mejora en el desempeño es evidente a medida que el entrenamiento se realiza con un conjunto de imágenes más amplio, como ocurre con el conjunto C. En el caso del primer conjunto de imágenes aumentadas (A) la exactitud máxima alcanzada fue del 98% y el más bajo fue de 66%, con valores de función de pérdida que fluctuaron entre 0.1 a 1. Para el segundo conjunto de imágenes aumentadas (B) la exactitud aumentó tanto en el conjunto de validación como en el de prueba, llegando al 100% en algunos modelos y variando entre $83\% - 100\%$. Sin embargo, al comparar la exactitud entre los conjuntos de prueba y validación, en la mayoría de los casos se observó una diferencia de $2 - 3\%$, lo que sugiere un ligero sobreajuste, donde el modelo se adapta mejor a uno de los conjuntos. La función de pérdida en este caso mostró menor variabilidad, con un mínimo de 0.0002 y el máximo de 0.876. El tercer conjunto (C) se observó una mejora notable en la exactitud: un mayor número de modelos alcanzaron el 100%, siendo el valor más bajo registrado del 84%.

Finalmente, en la Figura 34 se presentan las gráficas de entrenamiento para el modelo con la combinación de parámetros: tasa de aprendizaje de 0.0001, épocas de $10, 30, 50, 70\ y\ 90$, y batch size de 12. Estas muestran la evolución de la función de pérdida y de la exactitud a lo largo de las épocas. En particular, la Figura 34a corresponde a 20 épocas de entrenamiento; la 34b, a 40; la 34c, a 60; la 34d, a 80; y la 34e, a 100.

En las primeras épocas $(10 - 20)$, el modelo presenta un aprendizaje limitado con baja precisión y alta pérdida. A medida que aumenta el número de épocas $(30 - 50)$, se observa una mejora significativa en la precisión y una reducción estable de la pérdida, lo que refleja un buen ajuste del modelo. Se puede ver lo rápido que la exactitud de entrenamiento alcanza el 100% y se mantiene

estable, mientras que la exactitud de validación tiene muchas oscilaciones, donde sube y baja erráticamente, o incluso empieza a bajar, lo que podría indicar que hay un sobreajuste de los datos, que podría estar aprendiendo ciertos patrones del conjunto de datos de entrenamiento que no puede replicar en el conjunto de validación. Otro indicador es cuando la función de pérdida empieza a aumentar después de disminuir, como se ve en épocas $(10 - 30)$.

No obstante, se observan algunas variaciones en las gráficas de validación, especialmente en las primeras épocas del entrenamiento, lo que podría estar relacionado con cambios en los datos de validación o con la sensibilidad del modelo a datos atípicos (outliers). Algunos casos, la pérdida de validación muestra picos, que podrían relacionarse con un leve sobreajuste o con una mala inicialización de los pesos. Sin embargo, a medida que avanzan las épocas, esta variación disminuye y la curva de validación se vuelve más estable, como puede verse en las secciones d) y e). Incluso la diferencia entre las curvas de validación y entrenamiento es menor, lo que sugiere que se ha alcanzado un punto óptimo de entrenamiento sin sobreajuste. Los modelos parecen alcanzar un nivel de sobreajuste controlado, manteniendo una pequeña diferencia entre las métricas de entrenamiento y validación. Este comportamiento sugiere un alto desempeño, rápida convergencia y capacidad de generalización aceptable.

El mejor modelo de acuerdo con las métricas y las comparaciones realizadas pertenece al conjunto de imágenes aumentadas C, sus hiperparametros fueron 0.0001 para la tasa de aprendizaje, 90 épocas, y tamaño de lote 12, la gráfica sobre su desempeño es de la Figura 34e

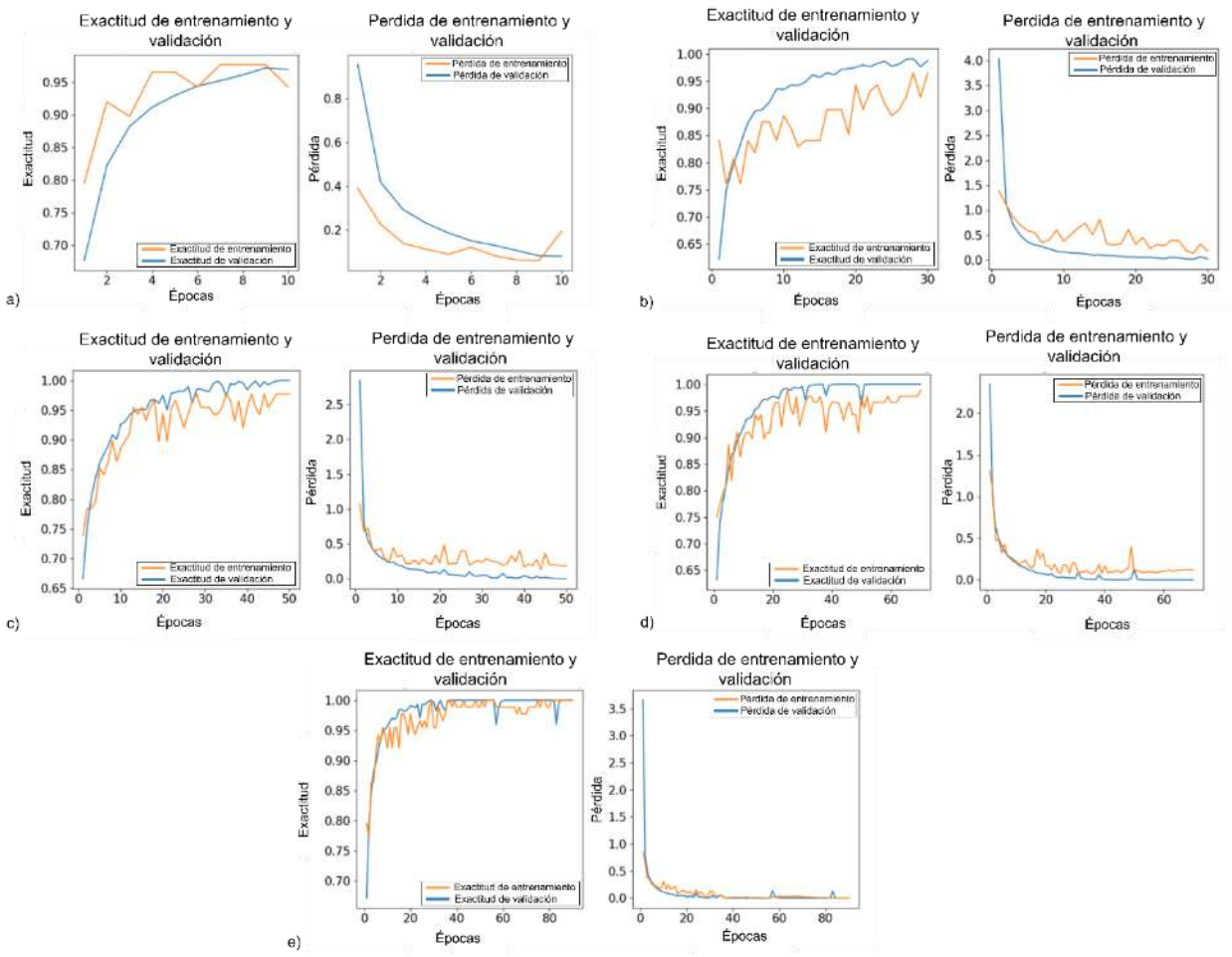

Figura 34. Graficas de exactitud y pérdida del conjunto de entrenamiento y validación

La Tabla 20 muestra el reporte de métricas del modelo CNN seleccionado, con una exactitud y precisión de 100% para cada clase, del mismo modo con recall y f1-score.

Tabla 20. Reporte de clasificación de métricas del modelo CNN

Reporte de clasificación del modelo CNN				
	Accuracy	precisión	Recall	F1-score
Sano o Clase 0	1.00	1.00	1.00	1.00
SP Bilateral o Clase 1	1.00	1.00	1.00	1.00
SP Unilateral o Clase 2	1.00	1.00	1.00	1.00

La matriz de confusión del modelo, Figura 35, muestra la cantidad de imágenes del conjunto de prueba clasificadas de manera correcta. De acuerdo con la Tabla 4, el conjunto de prueba consta de 59 imágenes, (23 clase Sano, 17 clase SP Bilateral, 19 clase SP Unilateral), observando en la matriz de confusión la cantidad corresponde a la cantidad de imágenes predichas correctamente en la clase a la que pertenece. Lo que indica, junto con el reporte de métricas, que el modelo predijo correctamente todas las imágenes del conjunto de prueba. El modelo se utilizará posteriormente

en la adquisición de nuevas imágenes, para obtener un modelo robusto, para fines prácticos se denominará "$Modelo\_1$".

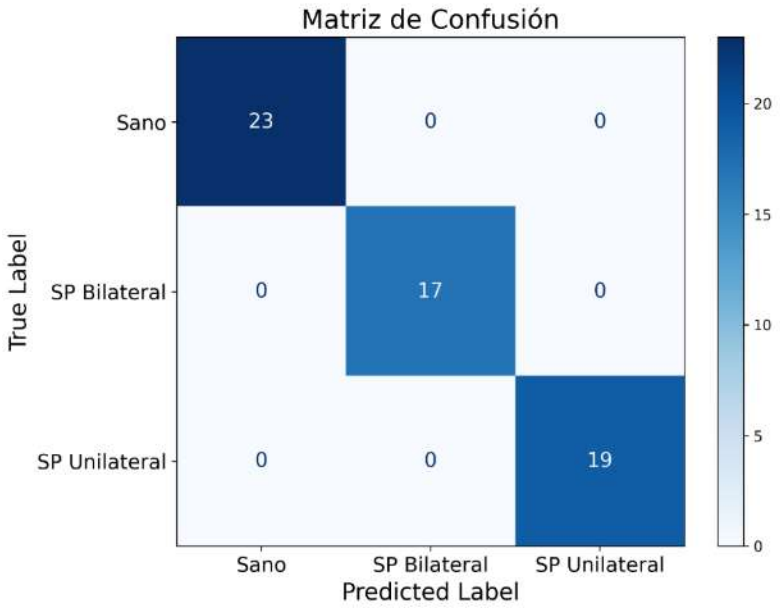

Figura 35. Matriz de confusión del modelo CNN.

4.6. Visualización de características

Para obtener los mapas de calor, se utilizaron las imágenes pertenecientes a los conjuntos de datos de prueba y validación. Con el objetivo de identificar las áreas de cada categoría que influyen más en las decisiones del modelo, se generaron mapas térmicos para cada imagen de los conjuntos de prueba y validación, como se muestra en la Figura 36.

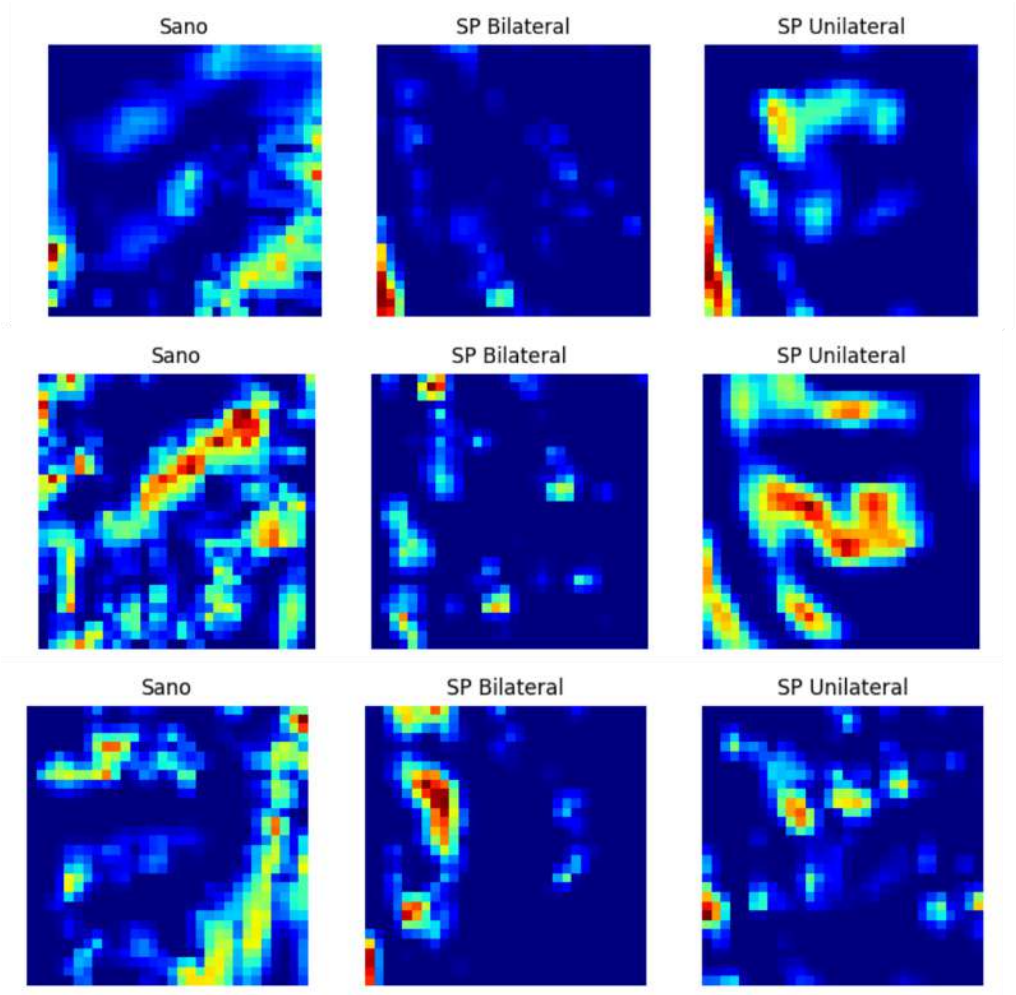

Figura 36. Ejemplos de mapas de calor individuales (por cada imagen)

Posteriormente, se organizaron los mapas de calor según la clase a la que pertenecen (sano, síndrome patelofemoral bilateral, y síndrome patelofemoral unilateral), sumando los mapas correspondientes a cada clase y obteniendo tres mapas térmicos; uno representando todos los mapas térmicos de la clase "Sano", otro para la clase "Bilateral" y el último para la clase "Unilateral", uno para cada clase, y cada uno de ellos se normalizó entre $0\ y\ 255$. Cada uno fue normalizado en un rango de $0\ a\ 255$. Este procedimiento se llevó a cabo en dos ocasiones: la primera utilizando la capa "$Layer\_1\_convolutional$" como capa convolucional (Figura 37a) y la segunda utilizando la capa "$Layer\_2\_convolutional$" como capa convolucional (Figura 37b).

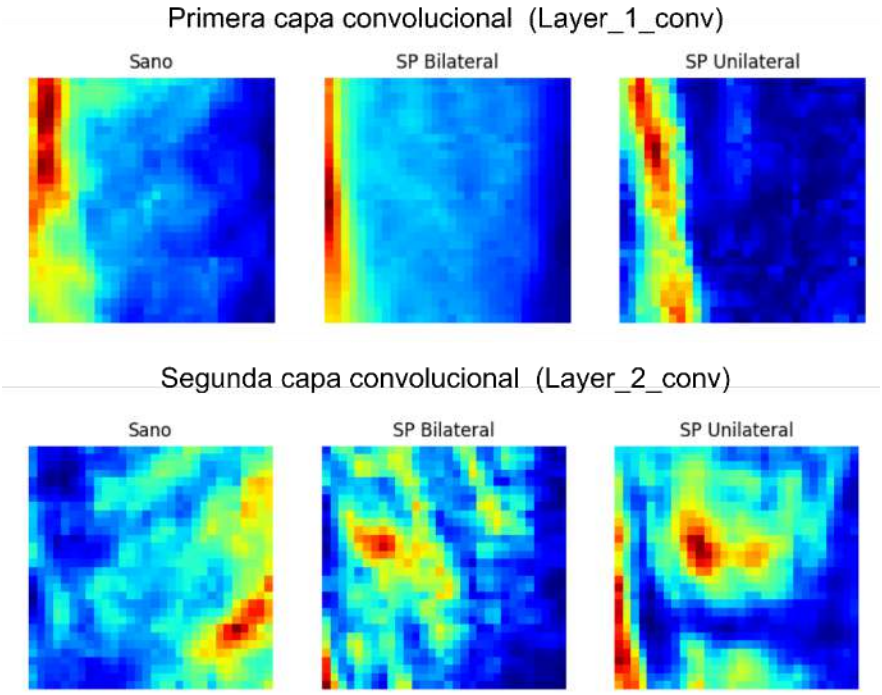

Figura 37. Mapas de calor generales por capa convolucional del modelo

Los resultados fueron mejores en la segunda capa porque a medida que avanzan las capas, el modelo obtiene más características, lo que permite a la red aprender características más complejas o especializadas de cada clase. La Figura 38 ilustra los resultados finales de los mapas de calor, que revelan cuáles aspectos de las imágenes tuvieron un impacto significativo en las decisiones tomadas por la red neuronal CNN. Las áreas resaltadas con tonos fríos, como el azul, indican que esas regiones carecen de información crucial o pertinente que pudiera afectar la decisión de categorización de la imagen. En contraste, las áreas destacadas en tonos cálidos, como el rojo o el amarillo, señalan las partes de la imagen que la CNN ha considerado más significativas para la clasificación. Además, las barras de color representan la escala de los valores, donde los tonos rojos equivalen a niveles de activación elevados cerca de 250, mientras que los tonos azules indican niveles bajos cercanos a 0. Donde el mapa de calor de la clase "Sano" toma en cuentas las áreas inferiores de las imágenes, la clase "SP Bilateral" tiene mayor importancia el lado derecho especialmente en el centro. Y la clase "Unilateral", tiene mayor importancia el centro de la imagen.

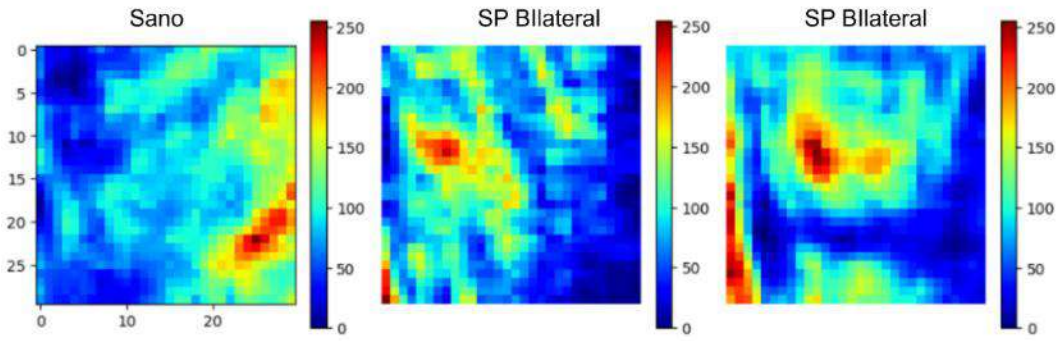

Figura 38. Mapas de calor finales de cada clase.

La Figura 39 presenta algunos ejemplos de la visualización del mapa de calor sobre la imagen original, permitiendo observar cuáles secciones de las imágenes originales el modelo CNN evalúa como más relevantes para una clasificación precisa de cada categoría, resaltando la mayor influencia hacia una categoría específica. Se observan algunos ejemplos, mostrando las imágenes originales (imagen en escala de grises) y debajo el mapa de calor superpuesto sobre las imágenes originales. Las secciones a) y c) muestran imágenes seleccionadas aleatoriamente del conjunto de prueba, mientras que las secciones b) y d) muestran imágenes del conjunto de validación, también seleccionadas aleatoriamente.

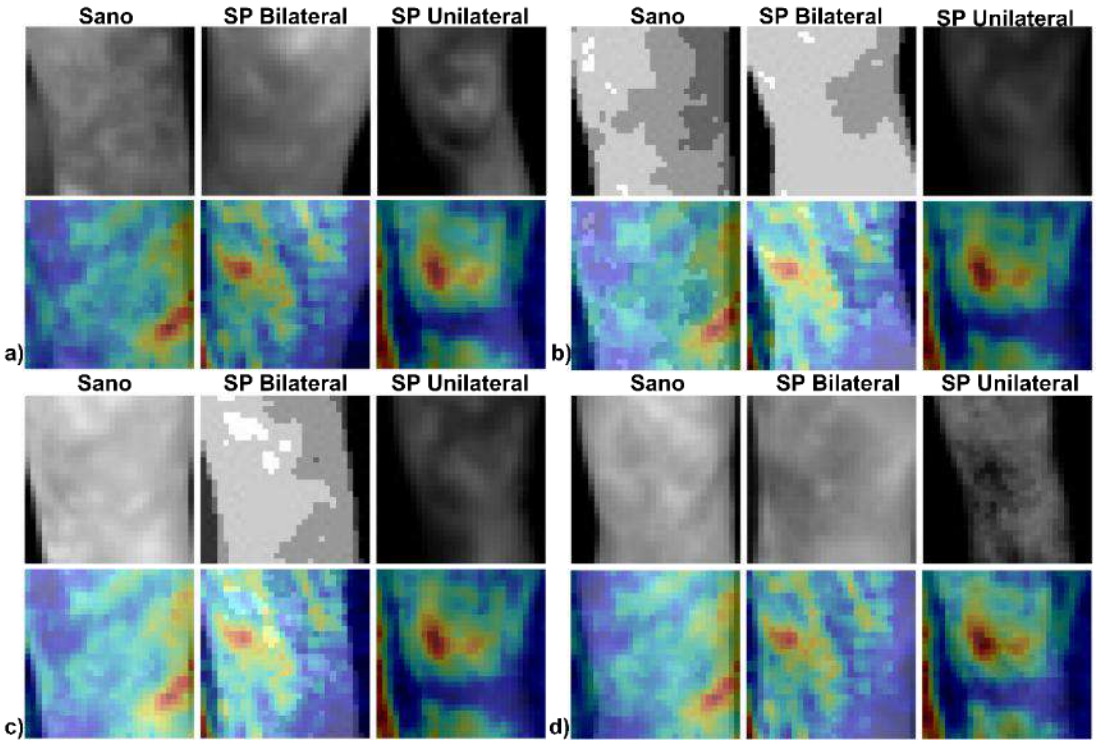

Figura 39. Ejemplos de visualización del mapa de calor por clases

4.7. Componentes del sistema embebido

La metodología utilizada se detalla en el Anexo B, donde se describe en detalle cada etapa desde la instalación de Imagen en la Jetson Nano hasta la interconexión y ajustes de todos los componentes para el funcionamiento integral del sistema embebido, lo que también permite que la interfaz opere para la adquisición de termogramas en tiempo real y su posterior clasificación. Una vez completados los primeros seis pasos del Anexo B, la tarjeta Jetson Nano está preparada para su uso, como se ilustra en la Figura 40 que presenta la pantalla principal (escritorio) de la Jetson Nano. Esta imagen se muestra a través del display HDMI conectada mediante el cable USB-MicroUSB para la alimentación y el cable HDMI para la visualización. Para aprovechar la función táctil del monitor, es necesario conectar un segundo cable USB-MicroUSB, tal como se indica en la Figura 41.

Figura 40. Pantalla principal (escritorio) de la Jetson Nano

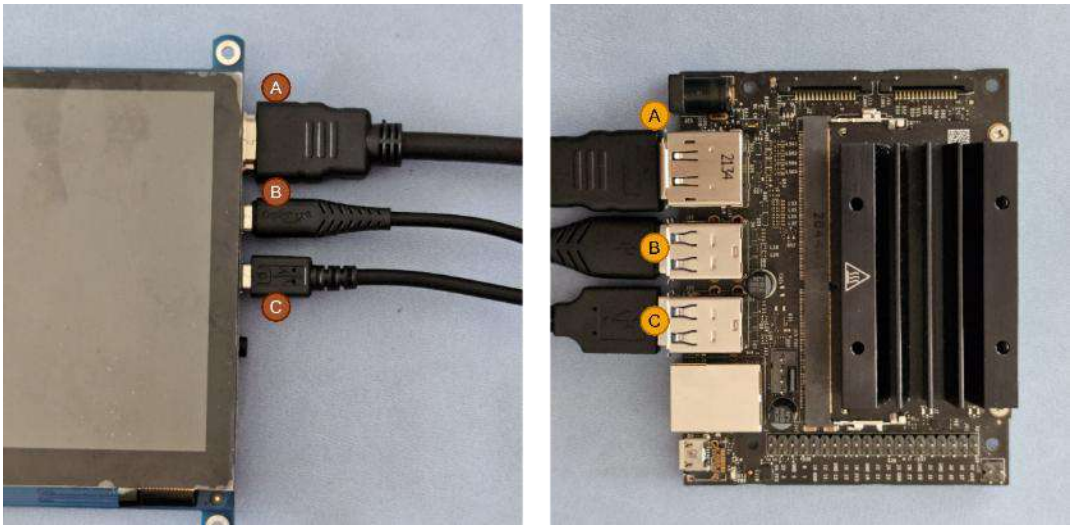

Figura 41. Conexión del display HDMI a la tarjeta Jetson Nano. Elaboración propia

Luego se ejecutaron las instrucciones del Paso 7 del Anexo B, configurando los puertos GPIO para que funcionaran como SPI e I2C mediante la herramienta "jetson-io.py". Asimismo, se llevaron a cabo dos pruebas para verificar la comunicación del protocolo SPI, cuyos resultados en ambos casos fueron "Aprobado".

Continuando con el procedimiento, se llevó a cabo la instalación de OpenCV con soporte para CUDA, con el fin de optimizar el rendimiento de la Jetson Nano en actividades de procesamiento de imágenes. En la Figura 42 se puede apreciar la interfaz presentada por el comando jtop, en la parte "INFO", que proporciona datos sobre la Jetson Nano. En este apartado, se puede verificar la versión de la imagen instalada, que es $L4T: 32.6.1$ y Jetpack: 4.6. Asimismo, se muestra el sistema operativo en el que funciona la tarjeta, y la versión de Python que se tiene instalada (3.6.9). En la sección de bibliotecas se listan las versiones de las bibliotecas más relevantes, donde OpenCV muestra la versión $4.5.1$ y confirma que está configurado de manera adecuada con el soporte para CUDA.

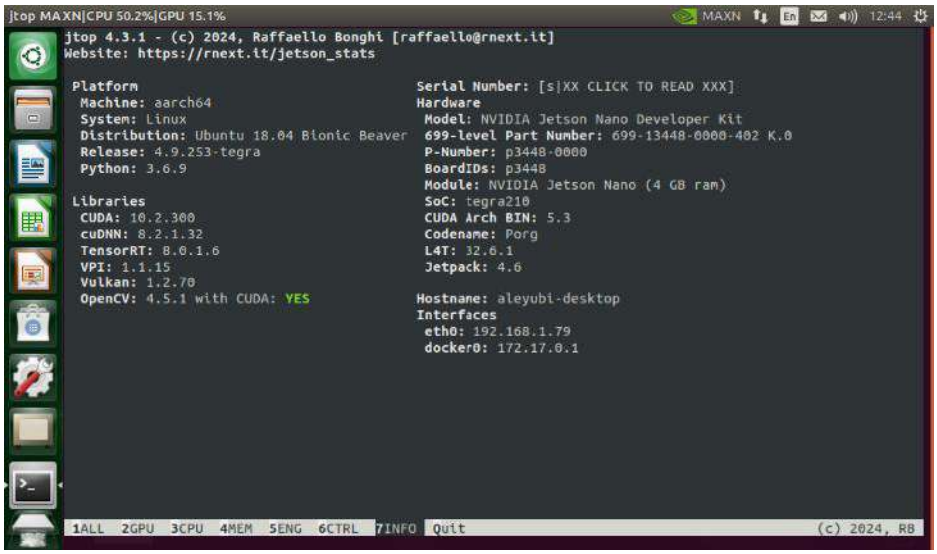

Figura 42. Pantalla de jtop con información acerca de la Jetson Nano

Finalmente, se completaron los pasos 9 a 12. Se llevó a cabo la instalación de Visual Code Studio siguiendo las instrucciones, utilizándolo para programar las cámaras (digital y termográficas), así como para diseñar la interfaz. La conexión de la cámara Raspberry Pi se presenta en la Figura 43, donde se conecta al puerto CSI de la Jetson Nano. Su funcionamiento fue verificado mediante código en Python y comandos ingresados en la terminal.

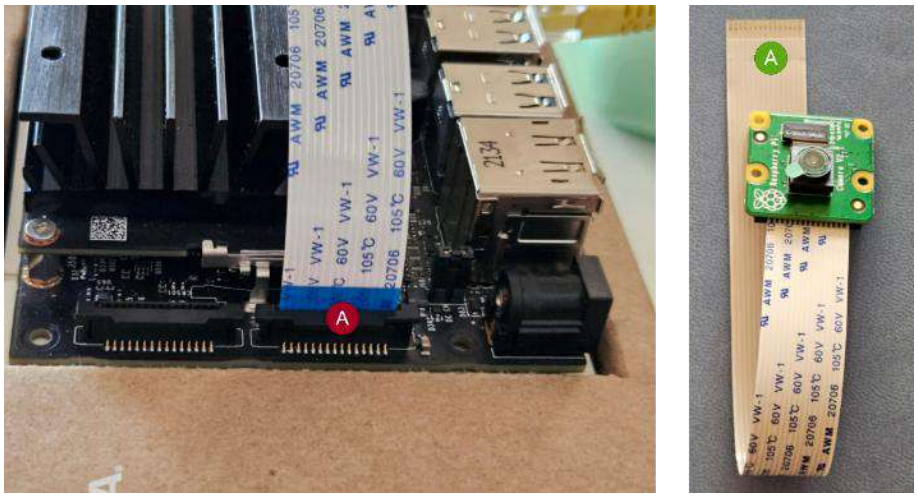

Figura 43. Conexión de la cámara Raspberry Pi a la Jetson Nano. Elaboración propia

En la Figura 44 muestra ejemplos del video registrado por la cámara Raspberry Pi, que fue operada utilizando comandos en la terminal y con los scripts

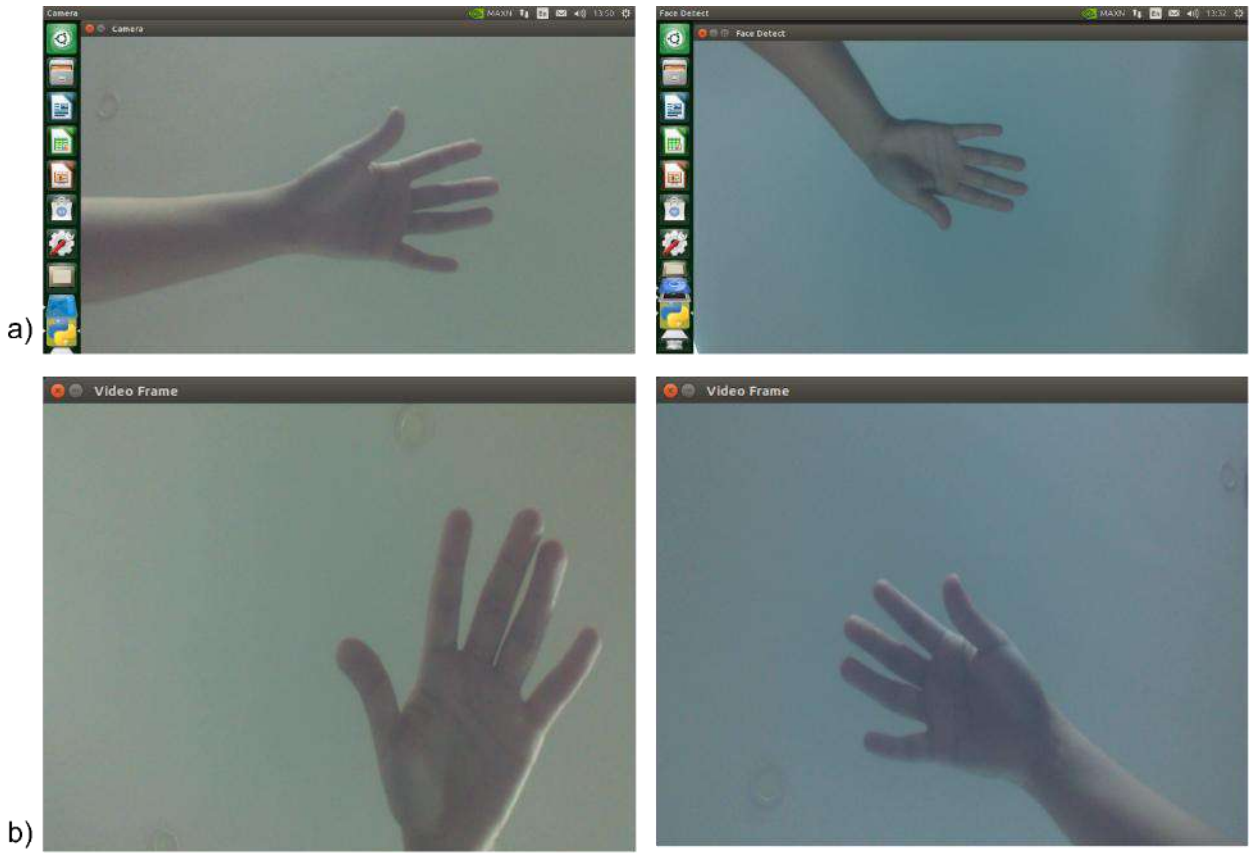

Figura 44. Pruebas realizadas con la cámara Raspberry Pi y la terminal(inciso a) / scripts (inciso b)

La conexión de la cámara termográfica Lepton se muestra en la Figura 45, donde se conectó según el esquema que se encuentra en el Anexo B en el paso 11. Su funcionamiento se comprobó utilizando los repositorios descargados de GitHub; además, se desarrollaron códigos basados en los scripts del repositorio para experimentar aún más con la cámara. Es importante señalar que las librerías Lepton y Lepton3 fueron ajustadas según el puerto de spidev al que se conecta la cámara, considerando si es "$dev/spidev0.0$" o "$dev/spidev0.1$".

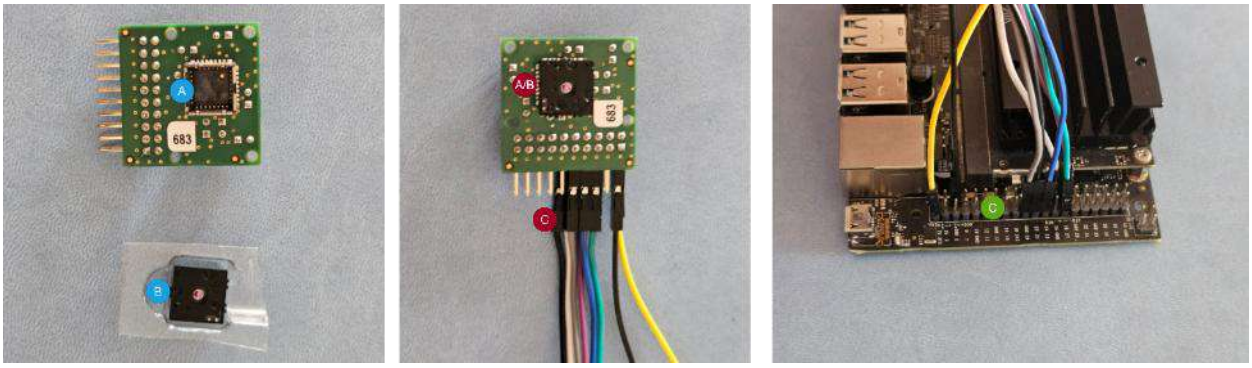

Figura 45. Conexión de la cámara Lepton, Breakout Board Lepton y la Jetson Nano

La Figura 46 presenta ejemplos de las imágenes termográficas tanto en color como en escala de grises, que se lograron al ejecutar los scripts del repositorio de GitHub.

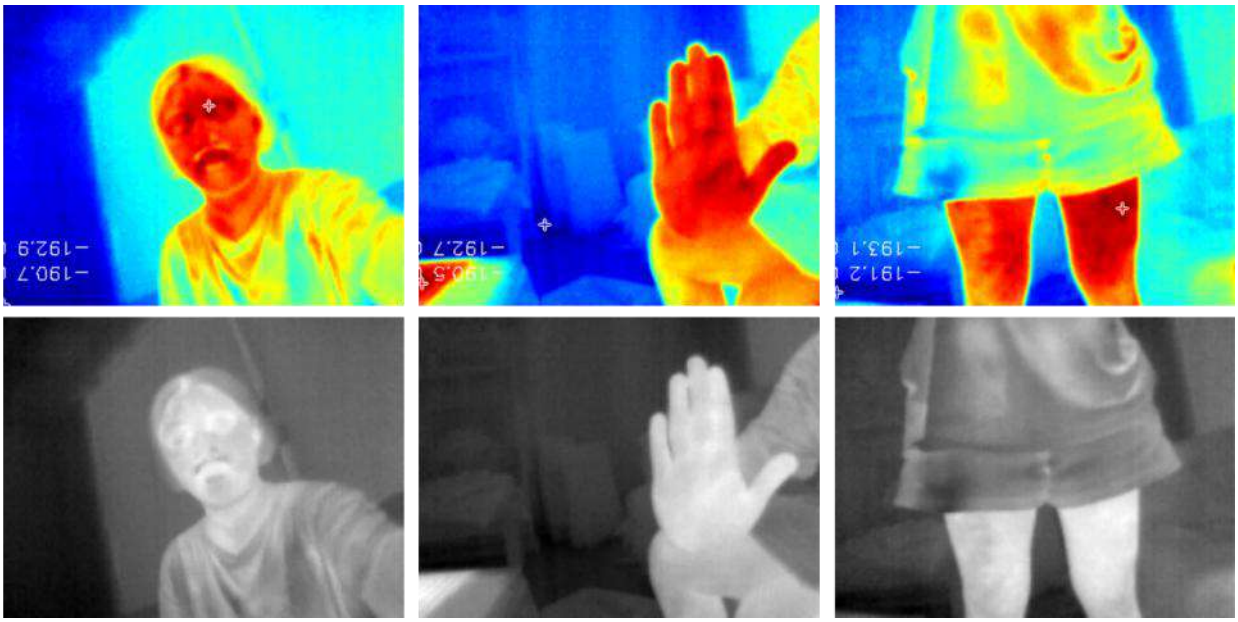

Figura 46. Pruebas realizadas con la cámara termográficas y los repositorios de GitHub

Se realizaron pruebas con diferentes scripts para verificar el funcionamiento de la cámara, en especial en formato de video, en la Figura 47 muestra varios ejemplos de video capturados por la cámara Lepton; a la izquierda se encuentra solo el video termográfico en escala de grises mientras que a la derecha se pueden ver dos videos que se reproducen simultáneamente: uno en color y otro en escala de grises.

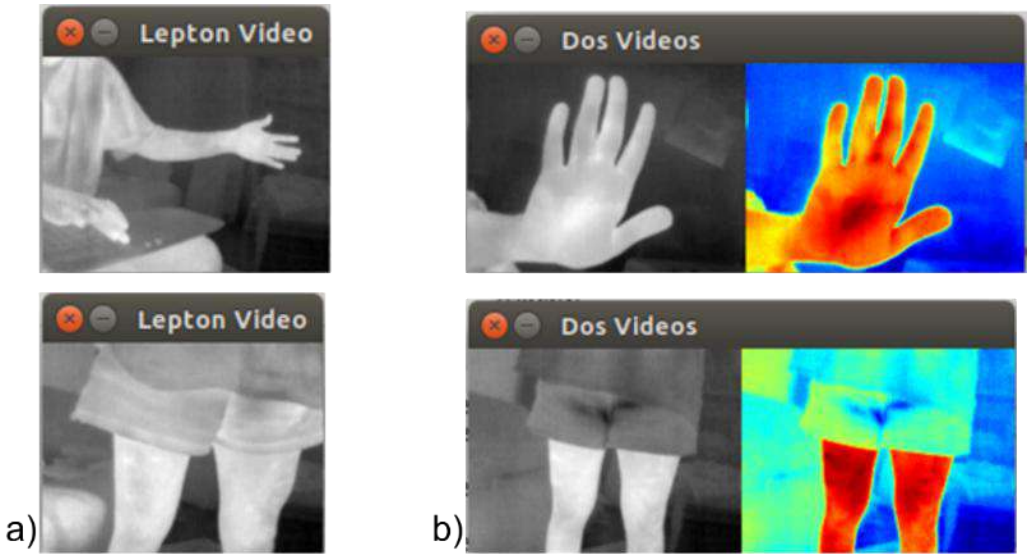

Figura 47. Pruebas realizadas con la cámara termográficas y scripts propios en VS Code

4.8. Modelo Tensorflow Lite

El entrenamiento del modelo CNN fue realizado en Google Colab, usando las librerías de Python: Tensorflow y Keras, con versiones actuales: 2.18.0 para Tensorflow y 3.8.0 para Keras. Así, al guardar el modelo CNN en un archivo ".h5", se guarda en un formato que podría no ser adecuado para versiones previas de estas bibliotecas. En la Figura 48 se puede ver el programa Visual Studio Code presentando un error por problemas de compatibilidad al intentar cargar el modelo en el sistema embebido.

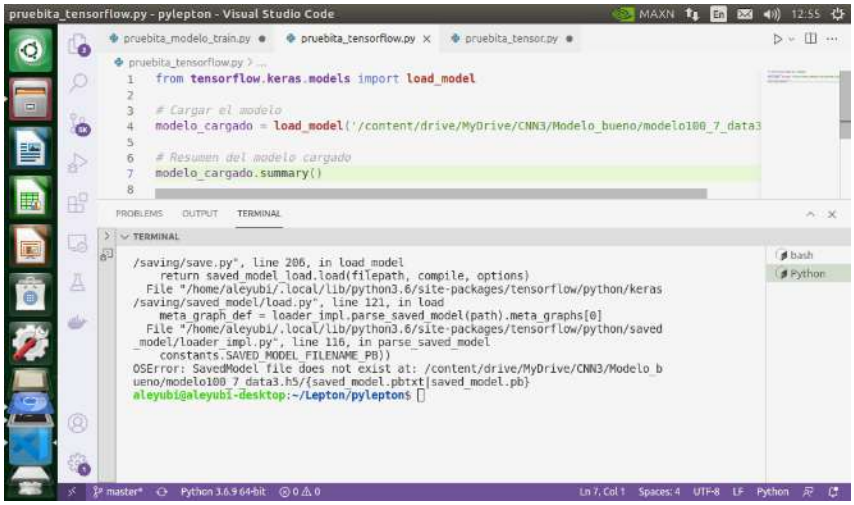

Figura 48. Error al cargar el modelo CNN Tensorflow en el sistema embebido

Las versiones que se han instalado en la tarjeta Jetson Nano, siendo $2.5.0 + nv21.8$ para Tensorflow y $1.0.8$ para Keras, son antiguas y no se pueden actualizar debido al requerimiento de una versión más reciente de Python, la cual no es alternativa para Jetson Nano. Por esta razón, se decidió modificar el modelo a un formato más eficiente, tal como Tensorflow Lite. Se utilizó la librería Tensorflow para crear un convertidor a Tensorflow lite, el cual transforma el modelo y lo almacena en el formato ". tflite".

Para asegurar que no se ha perdido información durante la conversión del modelo, se evaluó el modelo con el conjunto de pruebas, logrando una precisión de 1.0, es decir, 100%. Además, en la Tabla 21 se presenta el informe de clasificación del modelo CNN tflite, comparando con los resultados del modelo CNN, indica que el rendimiento del modelo se mantuvo sin cambios.

Tabla 21. Reporte de clasificación del modelo CNN tflite

	Reporte de clasificación del modelo CNN tflite			
	Accuracy	precisión	Recall	F1-score
Sano o Clase 0	1.00	1.00	1.00	1.00
SP Bilateral o Clase 1	1.00	1.00	1.00	1.00
SP Unilateral o Clase 2	1.00	1.00	1.00	1.00

No obstante, al comparar los tamaños de ambos modelos, se observó que el peso del modelo tflite se redujo de manera significativa. El peso del modelo CNN de Tensorflow es de $2.224\ Megabytes$, mientras que el peso del modelo CNN tflite es de $0.731\ Megabytes$, lo que significa que el tflite representa el 32% del peso del modelo de Tensorflow. Esto es extremadamente beneficioso, ya que en el uso del modelo en un sistema embebido, uno de los principales desafíos es la carga computacional, y al disminuir notablemente el tamaño, esto permite trabajar dentro de esa restricción.

A continuación, se presenta una Figura 49 del programa Visual Code Studio donde se ha cargado el modelo y se ha puesto a prueba, logrando predecir correctamente una imagen seleccionada aleatoriamente del conjunto de prueba.

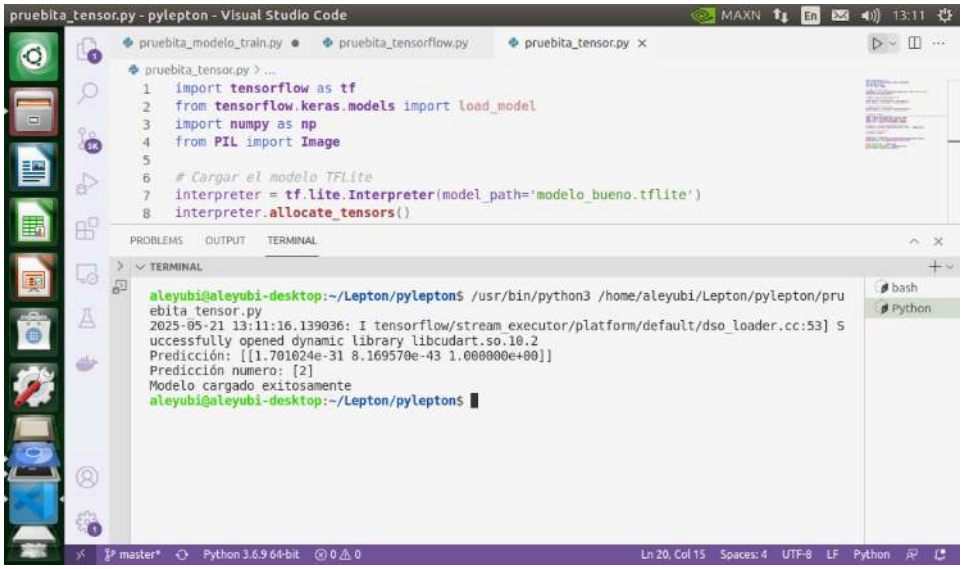

Figura 49. Resultado al predecir una imagen usando el modelo CNN tflite

Al ejecutar los códigos para el entrenamiento del modelo y para la predicción del modelo, se obtuvo un reporte computacional. Al entrenar el modelo se obtuvieron los siguientes datos:

- Tiempo de ejecución: 0.6 segundos
- Memoria usada por el proceso: 5.05 MB

- RAM total usada: 2205 MB de 3964 MB (~55%).
- SWAP: 239 MB de 6078 MB
- GPU: 0% de carga (GR3D_FREQ 0%)
- Núcleos activos con cargas de 65–78% a 1479 MHz
- Esto muestra que el trabajo se ejecutó principalmente en CPU y no GPU.
- Temperaturas:
    - CPU: 43°C
    - GPU: 40.5°C
- Consumo eléctrico:
    - POM_5V_IN: 3880 mW (~3.88 W)
    - CPU consumiendo ~1.6 W, GPU apenas 0.08 W.

Y los datos obtenidos al ejecutar el código de predicción se muestra a continuación:

- Tiempo de ejecución: 238.22 segundos
- Memoria usada por el proceso: 1376.53 MB
- RAM total usada: 2205 MB de 3964 MB (~55%).
- SWAP: 239 MB de 6078 MB

4.9. Interfaz

La interfaz fue desarrollada utilizando diversas librerías de Python como Tkinter, CustomTkinter, docx, PIL, numpy y Tensorflow. Además, se empleó Visual Code Studio para la programación. Se diseñaron múltiples ventanas en función del objetivo de la interfaz, que es capturar en tiempo real imágenes térmicas y clasificarlas en tres categorías, lo cual resulta útil como herramienta para el diagnóstico de enfermedades.

La interfaz está destinada a un sistema embebido. Al arrancar el sistema operativo, es necesario introducir en la terminal el siguiente comando: "*sudo modprobe spidev*". Esto habilitará el uso de la librería spidev, facilitando así la comunicación entre la tarjeta y la cámara termográfica

Lepton. Si este comando no se ejecuta, la tarjeta no será capaz de detectar la cámara. Es importante destacar que para que la interfaz funcione correctamente, es indispensable que tanto la cámara Raspberry como la cámara termográfica estén conectadas.

Como se ha indicado, la interfaz cuenta con diversas pantallas. La primera es la pantalla principal, que presenta dos botones: “Ingresar paciente” y “Salir”, como se muestra en la Figura 50. Al seleccionar el botón “Salir”, la interfaz se cerrará; en cambio, si se elige “Ingresar paciente”, se abrirá una ventana adicional.

Figura 50. Pantalla principal de la interfaz.

La ventana “Ingresar paciente” solicitará al usuario que complete con la información personal del paciente, que incluye: nombre, apellidos paternal y maternal, edad, altura, sexo, peso y diagnóstico (realizado por un médico especialista), tal como se indica en la Figura 51.

Figura 51. Ventana "Ingresar paciente" de la interfaz.

Para avanzar a la siguiente ventana, es necesario completar adecuadamente todos los campos. Se programó de tal forma que si un usuario llena solo algunos campos e intenta hacer clic en "siguiente", aparecerá una ventana que indicará un error. También se implementaron otras limitaciones, por ejemplo; si se intenta ingresar letras en los campos de edad, altura y peso, la interfaz no permitirá la entrada. Por otro lado, si el usuario intenta poner un número en los campos de nombre y apellidos, la interfaz no mostrará lo que se teclea. Además, en los campos de selección múltiple, solo se aceptarán como válidas las opciones proporcionadas en cada lista desplegable.

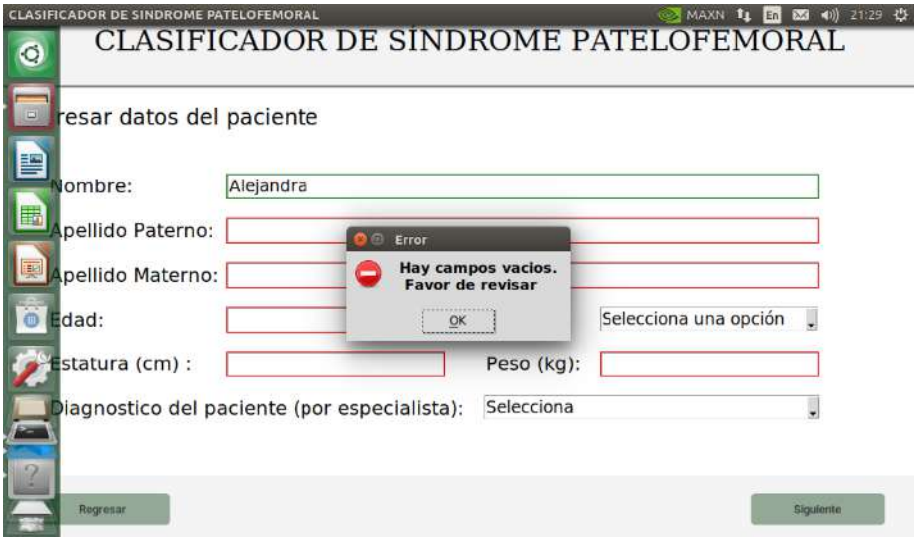

Figura 52. Error al no llenar todos los campos de datos personales del paciente.

Después de registrar todos los datos correctamente, la interfaz presentará una nueva ventana para capturar, como se muestra en la Figura 53. Esta ventana mostrará la imagen proveniente de la

cámara infrarroja; aunque la cámara Raspberry Pi no se muestra también captura imágenes. Se añadieron varios botones, siendo los cuatro principales: “Abrir cámara”, que iniciará la comunicación con las cámaras y mostrará la imagen en la pantalla; “Cerrar cámara”, que finalizará la comunicación con ambas cámaras; “Capturar imagen”, que capturará la imagen mostrada en pantalla y abrirá un cuadro para guardar la imagen con el nombre que se desee; y “Continuar”, que avanzará a la siguiente ventana, pero solo si se ha capturado alguna imagen; de lo contrario, se mostrará un error.

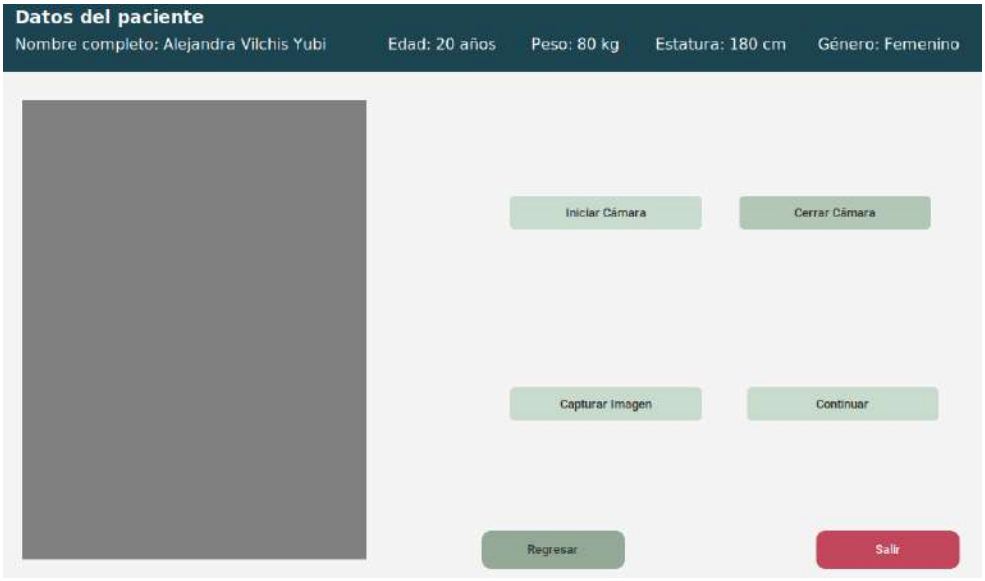

Figura 53. Ventana para capturar imágenes en la interfaz

Un ejemplo de visualización de la imagen está presente en la Figura 54. El cuadrado verde que aparece en la imagen fue agregado porque el modelo fue entrenado con imágenes que tienen dimensiones de $30x30$. Dado que la cámara Lepton presenta una resolución de $160x120$, el cuadrado verde fue añadido para señalar que aquello dentro de él será la imagen de entrada para el modelo CNN entrenado. Este cuadrado mide $90x90$, puesto que el tamaño de $30x30$ resulta muy pequeño en la pantalla y no es práctico al momento de posicionar al paciente. Sin embargo, antes de que la imagen pase por el modelo es redimensionada a $30x30$. Es relevante señalar que únicamente la imagen de termografía será utilizada en el modelo CNN, mientras que la imagen digital tiene el propósito de servir como referencia de lo que se ha capturado, proporcionando información más precisa.

Figura 54. Ejemplo de visualización de las imágenes en la interfaz.

Para guardar la imagen mostrada se realiza presionando el botón “Capturar Imagen”, y aparecerá un nuevo recuadro que muestra la imagen (Figura 55) que se guardará junto con el botón “Guardar imagen”. Al dar clic, aparecerá un cuadro de dialogo, seleccionar el directorio deseado y el nombre de la imagen, y dar clic en “Guardar”

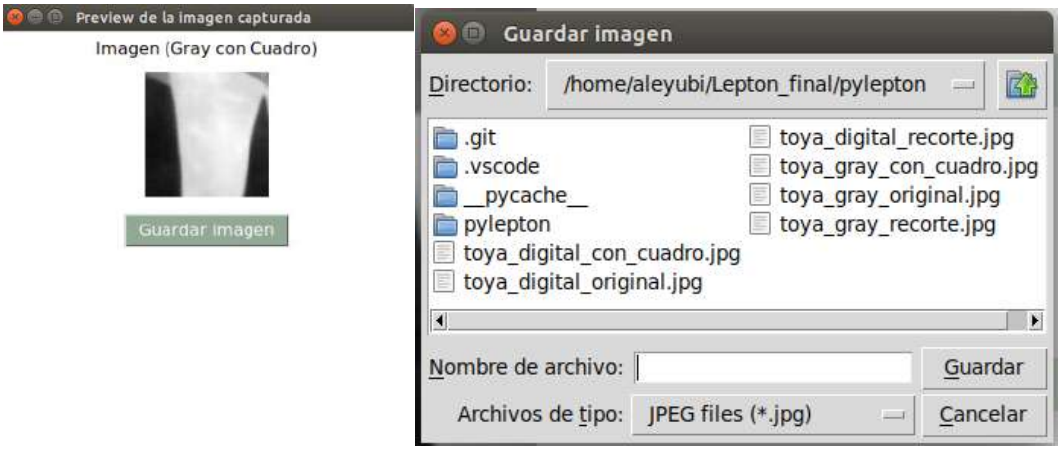

Figura 55. Ejemplo al guardar imagen capturada

En el caso de que la imagen haya sido capturada y se presione "Continuar", el sistema llevará al usuario a la ventana siguiente. Al pasar a la nueva ventana, aparecerá un cuadro de dialogo (Figura 56) para seleccionar la carpeta donde se guarda la base de datos, con el fin de almacenar ahí la nueva imagen capturada, el sistema no dejará avanzar hasta que se seleccione una carpeta.

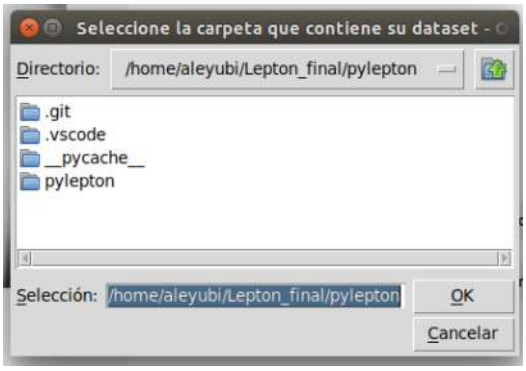

Figura 56. Ejemplo de cuadro de dialogo para selección de carpeta que contiene el dataset

Después de seleccionar la carpeta, en esta nueva ventana se mostrará la imagen térmica capturada y ofrecerá dos opciones: "Predecir" o "Entrenar modelo" (Figura 57), la selección dependerá de lo que el usuario necesite.

Figura 57. Ventana para predecir o reentrenar el modelo.

Si el usuario solo desea realizar predicción sobre la imagen, puede optar por "Predecir" y el modelo efectuará la predicción, mostrando en pantalla el porcentaje de pertenencia a cada clase, y la clase con mayor pertenencia, Figura 58. En este punto de la interfaz, se utiliza el modelo CNN tflite para ejecutarlo y llevar a cabo la predicción. Al final, el usuario podrá obtener el informe del paciente.

Figura 58. Ejemplo de predicción realizada en el sistema embebido.

Si el usuario desea reentrenar el modelo con las imágenes nuevas deberá seleccionar el botón "Entrenar el modelo". Al capturar estas imágenes, se almacenarán en el directorio elegido y la imagen termográfica se guardará con las otras imágenes en la base de datos, tanto las originales como las recientes, y se almacenará en la carpeta de la clase correspondiente según el diagnóstico ingresado. Antes de empezar con el entrenamiento, saldrán 3 cuadros de diálogos, uno después de otro, los cuales pedirán que se seleccione las carpetas donde se contienen las imágenes de prueba, de validación, y las de entrenamiento, como se muestra en la Figura 59.

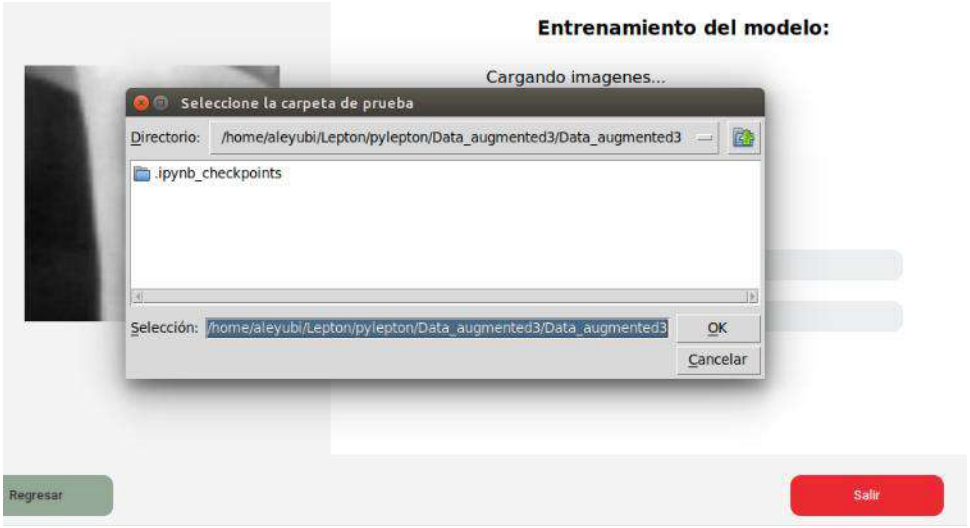

Figura 59. Ejemplo de cuadro de dialogo para la selección de carpeta de imágenes de prueba, validación y entrenamiento.

Así, al iniciar el reentrenamiento, se cargarán las imágenes de la base de datos actualizada (Figura

60a). Al cargar el modelo, se utilizó el modelo CNN dado que el modelo CNN tflite no se puede utilizar para entrenar. Sin embargo, para evitar los errores mencionados en la Sección 4.8, lo que se realizó fue recrear la arquitectura del modelo CNN, y cargar los pesos del modelo CNN, después compilar el modelo (Figura 60b) y empezar el proceso de entrenamiento. Los hiperparametros escogidos para el reentrenamiento son: taza de aprendizaje: 0.0001, épocas: , batch size: 12 (Figura 60c). Al concluir el reentrenamiento, se mostrará en la pantalla la exactitud y pérdida del conjunto de prueba (Figura 60d); además, el modelo se guardará con el mismo nombre, sobrescribiéndolo para actualizarlo, y se convertirá al formato tflite para utilizar la versión más reciente del modelo durante las predicciones.

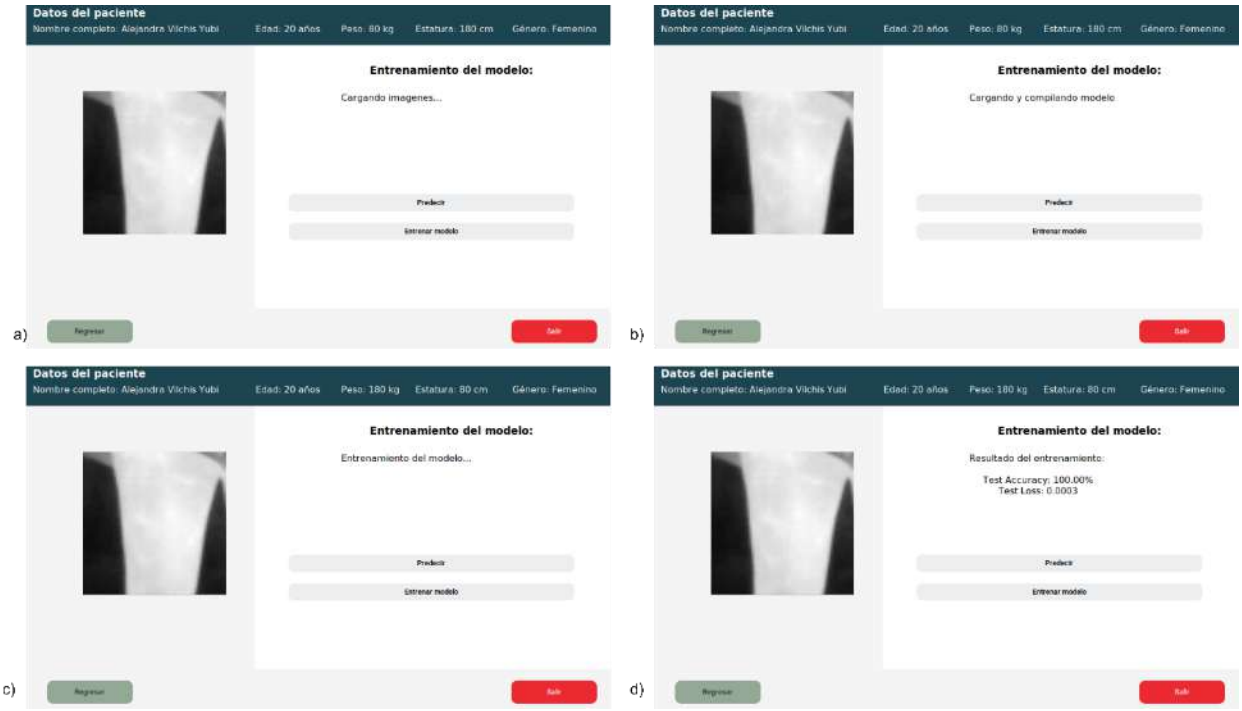

Figura 60. Ejemplos del proceso de reentrenamiento del modelo en el sistema embebido: a) Cargando imágenes, b) Compilando el modelo, c) Entrenamiento del modelo, d) Métricas del modelo reentrenado.

Por último, el usuario podrá realizar predicciones con el modelo CNN que ha sido reentrenado, y como se ve en la Figura 61, este modelo clasificará correctamente la imagen proporcionada. El usuario puede recibir el informe del paciente seleccionando "Obtener reporte".

Figura 61.Ejemplo de predicción después del reentrenamiento en el sistema embebido.

El reporte de resultados incluye la información personal del paciente, las imágenes digitales y termográficas obtenidas, así como los resultados de la predicción.

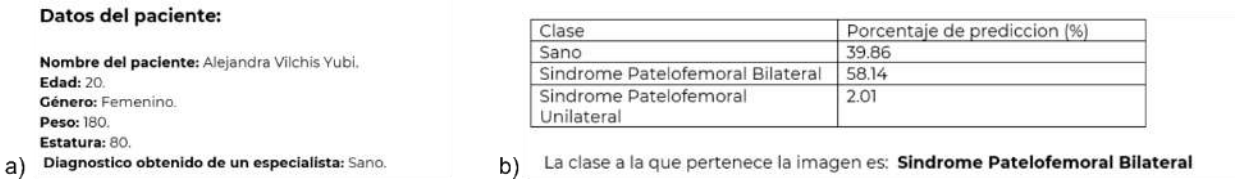

Figura 62. Ejemplo del contenido del reporte generado por la interfaz.

Todas las imágenes capturadas se muestran en la Figura 63, donde en la parte superior se encuentran las imágenes tomadas por la cámara Raspberry pi, mientras que las imágenes en la parte inferior son las imágenes termográficas. La primera columna muestra las imágenes originales de cada cámara; la segunda columna contiene las imágenes originales con un cuadrado verde superpuesto; y la última columna muestra un recorte de la imagen en el cuadrado verde, siendo el recorte de la imagen termográfica lo que pasa por el modelo CNN.

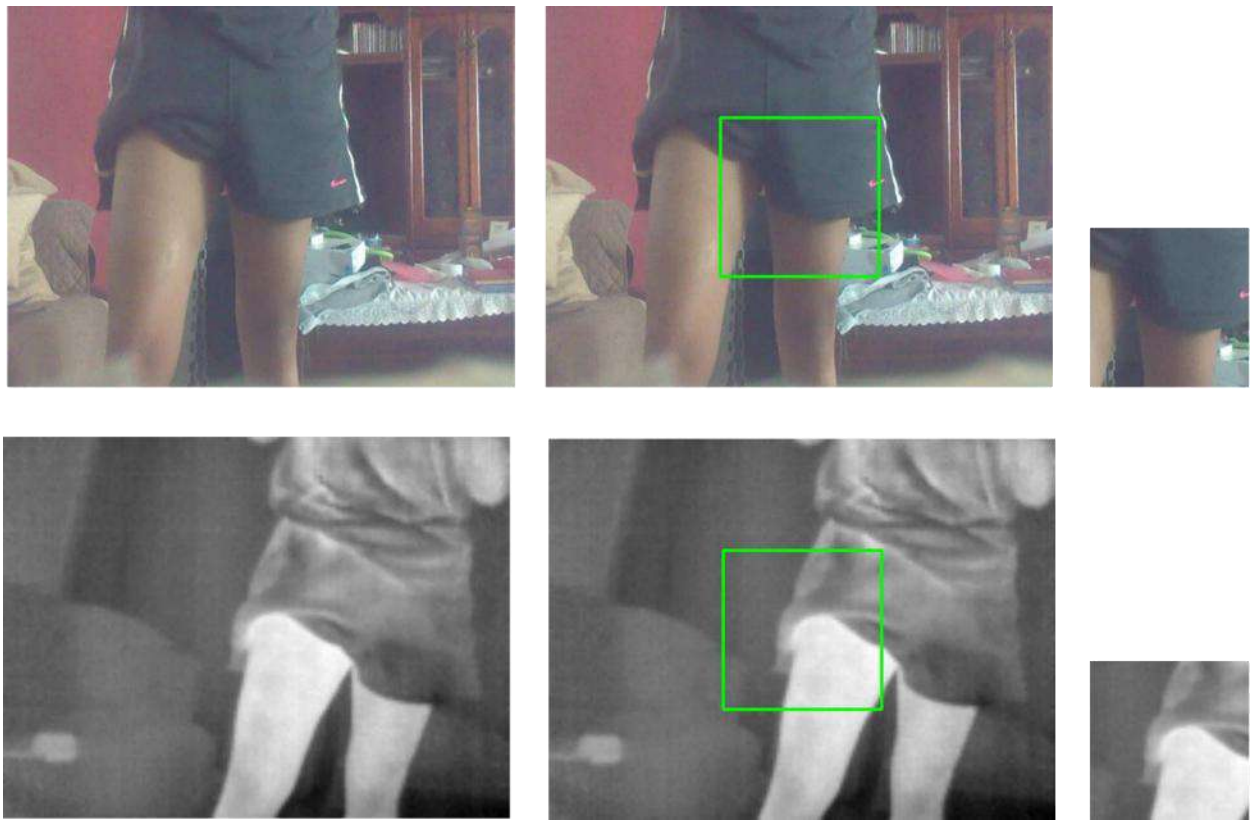

Figura 63. Ejemplo de las imágenes capturadas (diferente al ejemplo anterior) por el sistema embebido mediante la interfaz.

4.10. Carcasa

El diseño de la carcasa fue diseñado por estudiantes de la Universidad Autónoma de Querétaro que pertenecen al grupo de investigación. Fue diseñado mediante SolidWorks. En la Figura 64 se muestra un diseño en 3D de un sistema embebido dentro de una carcasa transparente para mejorar la visualización de los componentes que lo integran como una vista interior. El sistema embebido incluye:

A) Cámara termográfica (Lepton 3.5 con el módulo Breakout Board),

B) Cámara Raspberry Pi v2.0,

C) Jetson Nano Developer Kit y

D) Display HDMI.

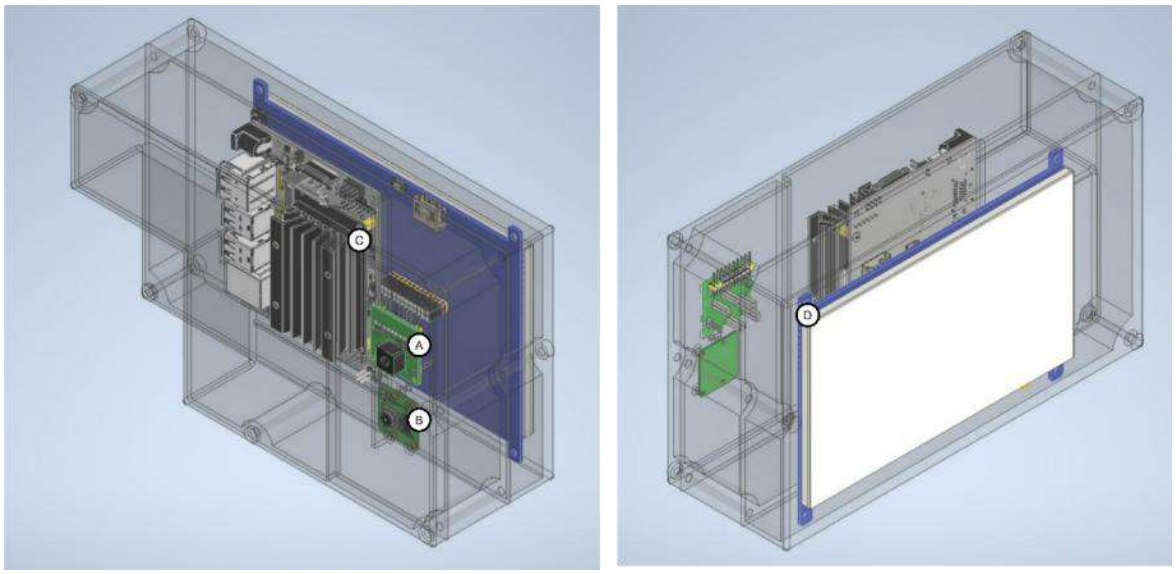

Figura 64. Modelo 3D de la carcasa para el sistema embebido (Vista interna)

La vista exterior de la carcasa se visualiza en la Figura 65a que muestra la parte frontal, es la cara que va hacia el usuario (experto en termografía o especialista clínico), el display HDMI va montado en esta parte. Es el medio de interacción, donde el usuario podrá visualizar datos, resultados o manejar la interfaz del sistema.

La Figura 65b muestra la parte trasera, la cara que va hacia el paciente. Es donde la cámara Raspberry Pi v2.0 y la cámara Lepton 3.5 van puestas, apuntan hacia el exterior de la carcasa para capturar las imágenes del paciente.

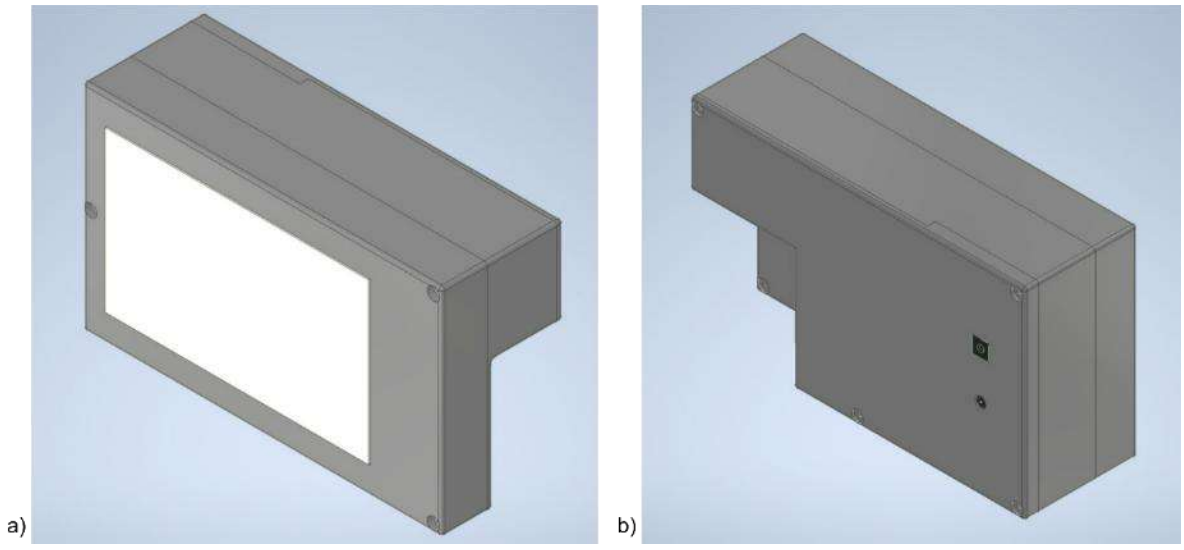

Figura 65. Modelo 3D de la carcasa para el sistema embebido (Vista externa), a) Vista frontal, b) Vista trasera.

La tarjeta Jetson Nano está colocada dentro de la carcasa y funciona como unidad de procesamiento principal. En la Figura 66, se muestra una vista lateral de la carcasa que expone los puertos de la Jetson Nano (USB, HDMI, Ethernet y otros conectores) permitiendo el acceso externo para alimentación, conexión de teclado y mouse.

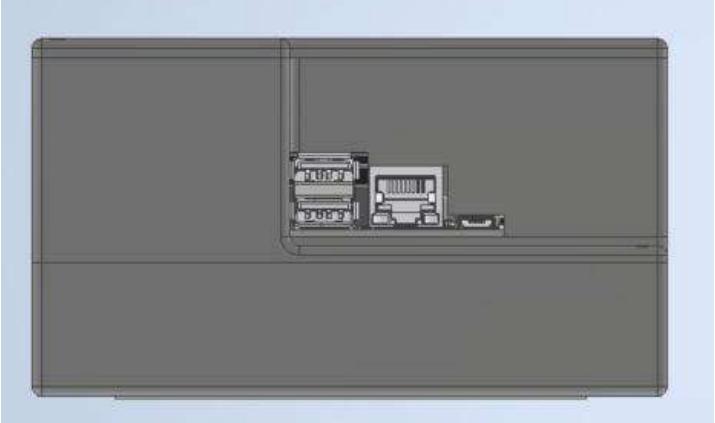

Figura 66. Modelo 3D de la carcasa para el sistema embebido (Vista lateral)

La carcasa se realizó por impresión en 3D. A continuación, en la Figura 67 se muestran la carcasa impresa, donde a) pertenece a la parte de en medio, b) pertenece a la parte trasera y c) a la parte frontal.

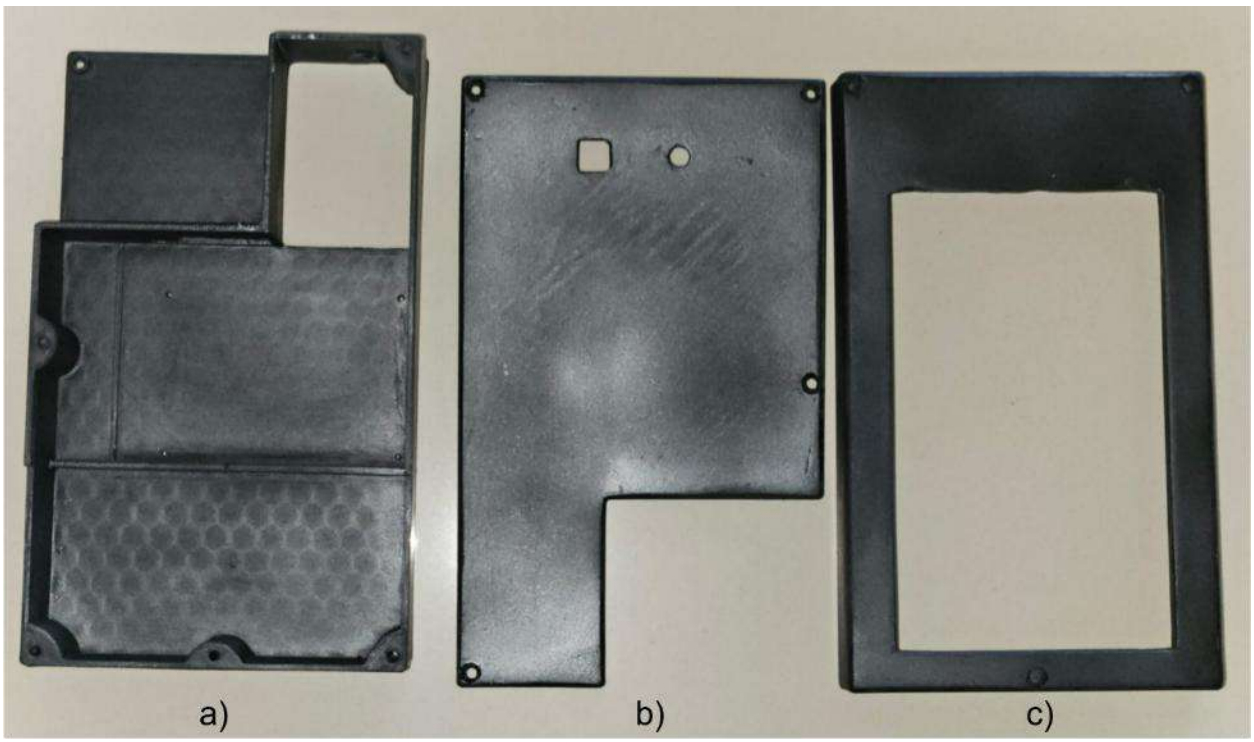

Figura 67. Carcasa impresa en 3D

Se monto todos los dispositivos electrónicos a su parte correspondiente de la carcasa. En la Figura 68 se

muestra el resultado final del sistema embebido montado en la carcasa.

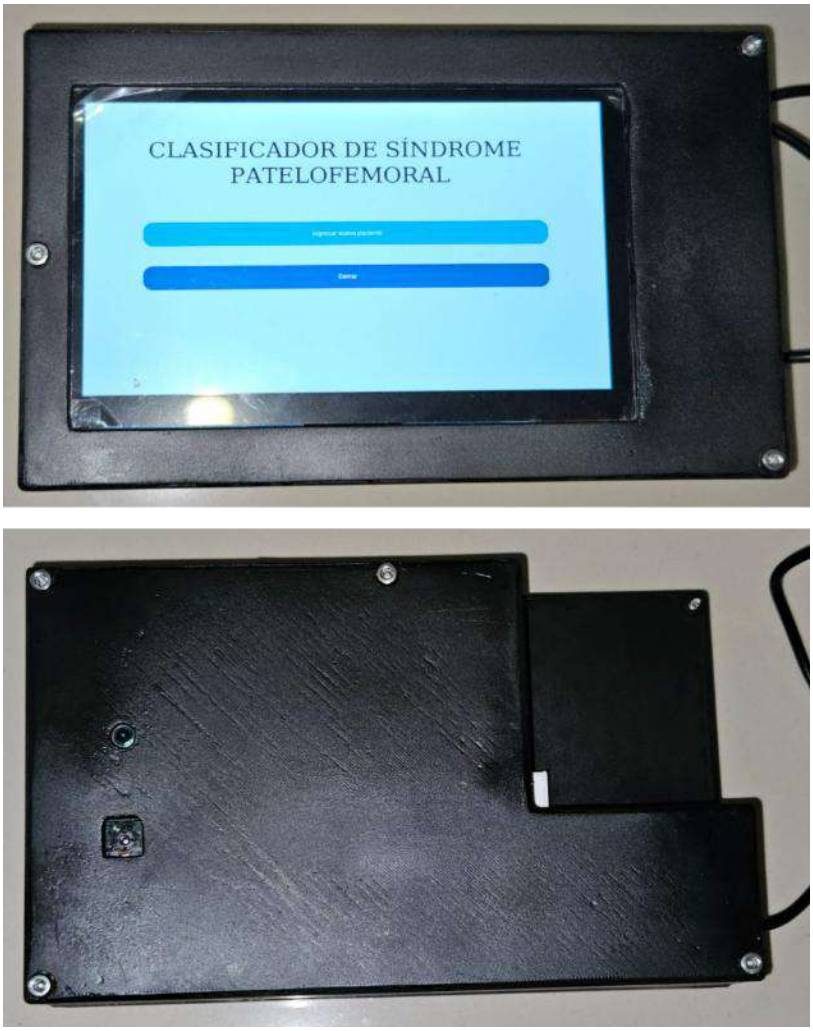

Figura 68. Resultado final del sistema embebido con la carcasa.

4.11. Nuevas imágenes adquiridas

Para cada participante, se adquirieron 7 imágenes termográficas de la región anterior de las rodillas, a excepción de los miembros del grupo bilateral, para quienes se adquirieron 8 imágenes (al inicio, se tomaron imágenes de ambas rodillas). No obstante, solo se consideraron las imágenes que se capturaron 15 min después de que el participante se adaptara, así como las imágenes tomadas después de esos 15 minutos tras la colocación de la compresa caliente. Esto resultó en un total de 23 imágenes termográficas (10 del grupo de control, 11 del grupo bilateral y 2 del grupo unilateral).

Las imágenes termográficas obtenidas se muestran en la Figura 69. Originalmente, las imágenes

tenían unas dimensiones de $180x180$, se ajustaron para enfocar mejor la región de la rodilla y se redimensionaron a $30x30$ píxeles, que es el tamaño de la imagen de entrada del modelo.

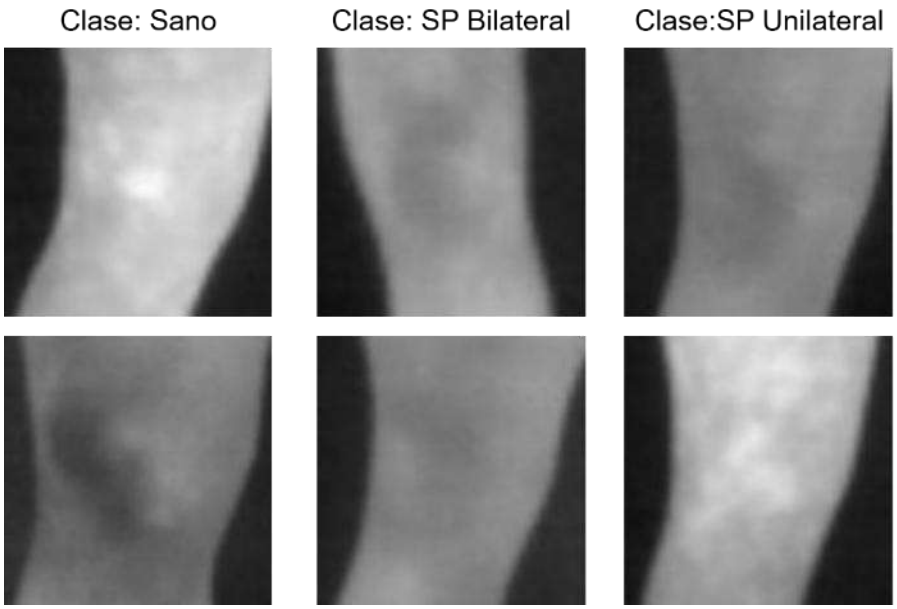

Figura 69. Ejemplo de las imágenes termográficas adquiridas.

Para obtener un modelo robusto, se llevaron a cabo cuatro pruebas utilizando combinaciones diferentes de conjuntos de imágenes. En estos, las nuevas imágenes termográficas se separaron de la misma forma que en la base de datos original, destinando el 75% de los datos al entrenamiento, el 15% a la validación y el 10% a la prueba. La Tabla 22 ilustra cómo se organizó la división del conjunto de imágenes, resultando en 17 para el conjunto de entrenamiento, 3 para validación y 3 para la prueba.

Tabla 22. División del conjunto de imágenes nuevas.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	17 imágenes	Sano: 8 Bilateral: 8 Unilateral: 1
Conjunto de validación	3 imágenes	Sano: 1 Bilateral: 2 Unilateral: 0
Conjunto de prueba	3 imágenes	Sano: 1 Bilateral: 1 Unilateral: 1

Conjunto A. Corresponde al conjunto de entrenamiento conformado por las 23 imágenes obtenidas, a las cuales se les aplicaron las técnicas de aumento de datos empleadas en el Conjunto

de imágenes incrementadas C. Para la validación y prueba, se recurrió al conjunto que fue utilizado para el entrenamiento del "$Modelo\_1$". Así, los grupos de datos se organizaron como se detalla en la tabla.

Tabla 23. División del conjunto de imágenes del conjunto A.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	207 imágenes	Sano: 90 Bilateral: 99 Unilateral: 18
Conjunto de validación	88 imágenes	Sano: 26 Bilateral: 32 Unilateral: 30
Conjunto de prueba	59 imágenes	Sano: 23 Bilateral: 17 Unilateral: 19

Conjunto B. Está compuesto por el conjunto de entrenamiento que incluye 17 imágenes obtenidas del conjunto de entrenamiento. A cada imagen se les aplicaron las técnicas de aumento de datos utilizadas en el Conjunto de imágenes aumentadas C. Las imágenes nuevas se combinaron con las de la base de datos inicial para crear el conjunto de validación y de prueba. Así, los conjuntos de datos se conformaron de la siguiente manera:

Tabla 24. División del conjunto de imágenes del conjunto B.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	153 imágenes	Sano: 72 Bilateral: 72 Unilateral: 9
Conjunto de validación	91 imágenes	Sano: 27 Bilateral: 34 Unilateral: 30
Conjunto de prueba	62 imágenes	Sano: 24 Bilateral: 18 Unilateral: 20

Conjunto C. Incluye el conjunto de entrenamiento, junto con el conjunto de validación y de prueba, que están formados únicamente por las imágenes nuevas. Solo se realizó aumento de datos en el conjunto de entrenamiento. Por lo tanto, los conjuntos de datos se estructuraron de la forma siguiente:

Tabla 25. División del conjunto de imágenes del conjunto C.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	153 imágenes	Sano: 72 Bilateral: 72 Unilateral: 9
Conjunto de validación	3 imágenes	Sano: 1 Bilateral: 2 Unilateral: 0
Conjunto de prueba	3 imágenes	Sano: 1 Bilateral: 1 Unilateral: 1

Conjunto D. Comprende el conjunto de entrenamiento, el conjunto de validación y de prueba, que se formaron mediante la fusión de las imágenes del primer conjunto de datos con las nuevas imágenes. De esta forma, los conjuntos de datos fueron organizados de la forma siguiente:

Tabla 26. División del conjunto de imágenes del conjunto D.

División del conjunto de imágenes	Número de imágenes	Imágenes por clase
Conjunto de entrenamiento	4014 imágenes	Sano: 1359 Bilateral: 1359 Unilateral: 1296
Conjunto de validación	91 imágenes	Sano: 27 Bilateral: 34 Unilateral: 30
Conjunto de prueba	62 imágenes	Sano: 24 Bilateral: 18 Unilateral: 20

4.12. Reentrenamiento del modelo

Para desarrollar un modelo robusto, el "$Modelo\_1$" fue reentrenado utilizando los cuatro conjuntos de datos, y esto se hizo de dos formas:

Modelo sin fine tuning: Se aplicó el "$Modelo\_1$" con una tasa de aprendizaje de 0.0001 y un batch size de 12. Se llevó a cabo pruebas para las épocas en valores de $10, 20, 30, 40, 50, 60, 70, 80, 90\ y\ 100$. Además, se implementó la técnica de "Early stopping" con patience de 5 y min_delta de 0.001.

Modelo con fine tuning: Se empleó el "Modelo_1" con fine-tuning, en el que se congelan (no se entrenan) todas las capas del modelo, excepto las dos finales. Esto significa que esas capas no se

modificarán, manteniendo los pesos ya adquiridos. Los hiperparámetros aplicados fueron $patience$ de 5, y min_delta de 0.001.

Por lo tanto, cada conjunto ($A, B, C\ y\ D$) fue evaluado usando el modelo tanto sin fine-tuning como con fine-tuning. Los resultados más destacados provienen de los conjuntos A y D. Los modelos de los conjuntos B y C mostraron un desempeño deficiente; específicamente, los modelos del grupo B lograron una exactitud que oscila entre el 61% y el 69% para el modelo sin fine-tuning y un 59.84% con la técnica de "Early Stopping". En contraste, para el modelo con fine-tuning, la exactitud varió entre el 75% y $el\ 82\%$, y un 67% utilizando "Early Stopping". En lo que respecta a los modelos del grupo C, se obtuvo una exactitud entre el 33.33% y el 66.67% para el modelo sin fine-tuning y de 66.67% para el "EarlyStopping", a su vez, el modelo fine-tuning mostró una exactitud de entre el 0% y el 33.33%, y un 33.33% con "EarlyStopping".

El bajo rendimiento de los modelos C puede ser atribuido al hecho de que únicamente se emplean imágenes nuevas, y dado que hay muy pocas muestras, el modelo no ha podido generalizar adecuadamente sobre ellas. Además, existe un notable desbalance en las imágenes (solo hay 2 imágenes en la clase unilateral), lo que complica la identificación de patrones para cada categoría. También, no hay suficiente información que permita variaciones tanto en la validación como en la prueba.

Los modelos que pertenecen al conjunto B presentan un desempeño deficiente. Una posible razón podría ser la insuficiencia de imágenes para permitir que el modelo aprenda de manera adecuada los datos nuevos. Esto resulta en que el modelo no logra generalizar bien ni con las imágenes nuevas ni con las originales, a pesar de contar con varias imágenes para la validación y la prueba. Dado que las imágenes de validación y prueba provienen de una combinación de las del conjunto nuevo y las del conjunto original, el modelo se enfrenta a una diversidad más amplia durante la evaluación. Esta falta de representatividad y equilibrio en la etapa de entrenamiento afecta de manera adversa el rendimiento del modelo durante las fases de validación y prueba.

Los resultados de los modelos del conjunto A y D se muestran en las Tablas 27 y 28. Los modelos pertenecientes al conjunto D mostraron un desempeño sólido y consistente, alcanzando exactitudes entre 95% y 98% en los modelos sin fine-tuning y una exactitud de 95.16% al implementar "Early Stopping". Para los modelos con fine-tuning, se logró una exactitud de 96.77%, incluso al

utilizar "Early Stopping".

Tabla 27. Resultados del conjunto D.

Modelo sin fine tuning					
Epocas	*Val accuracy*	*Val loss*	*Test Accuracy*	*Test loss*	*Early stopping*
10	0.9670	1.1818	95.16%	0.5049	
20	0.9670	1.7913	96.77%	0.6181	
30	0.9560	1.281	95.16%	0.6093	
40	0.9670	1.2129	95.16%	0.5949	Épocas = 5 val_accuracy: 0.9560 val_loss: 0.8148 Test Accuracy: 95.16% Test Loss: 0.4787
50	0.978	2.0033	96.77%	0.8184	
60	0.9670	1.4109	96.77%	0.6681	
70	0.9780	2.3975	96.77%	0.8124	
80	0.9670	1.4296	96.77%	0.6232	
90	0.9780	2.1416	96.77%	0.7281	
100	0.9670	2.4150	98.39%	0.5748	
Modelo con fine tuning					
Epocas	*Val accuracy*	*Val loss*	*Test Accuracy*	*Test loss*	*Early stopping*
10	0.9341	1.1305	96.77%	0.8462	
20	0.9341	0.9154	96.77%	0.5247	
30	0.9560	0.8428	96.77%	0.4410	
40	0.9560	0.8058	96.77%	0.4117	Épocas = 42 val_accuracy: 0.9560 val_loss: 0.8088 Test Accuracy: 96.77% Test Loss: 0.4141
50	0.9560	0.7938	96.77%	0.3825	
60	0.9560	0.7593	96.77%	0.3822	
70	0.9560	0.7543	96.77%	0.3955	
80	0.9560	0.7341	96.77%	0.3696	
90	0.9560	0.7319	96.77%	0.3707	
100	0.9560	0.7321	96.77%	0.3672	

De manera similar, los modelos del conjunto A mostrar una mayor variabilidad en su exactitud logrando entre 84% y 96% en los modelos sin fine-tuning y 91.53% con "Early Stopping". En cuanto a los modelos con fine-tuning, se alcanzó una exactitud de entre 89% y 98%, así como una exactitud de 91.53% al usar "Early Stopping".

Tabla 28. Resultados del conjunto A

Modelo sin fine-tuning					
Épocas	*Val accuracy*	*Val loss*	*Test Accuracy*	*Test loss*	*Early stopping*
10	0.8068	0.4319	86.44%	0.2944	
20	0.9205	0.3835	91.53%	0.1485	
30	0.9091	0.3848	89.83%	0.1932	
40	0.8977	0.4037	96.61%	0.1177	Épocas = 21 val_accuracy: 0.9091 val_loss: 0.3638 Test Accuracy: 91.53% Test Loss: 0.1579
50	0.9432	0.4263	86.44%	0.2428	
60	0.8977	0.5041	93.22%	0.1966	
70	0.8523	0.5909	84.75%	0.8028	
80	0.7955	1.4073	84.75%	0.8028	
90	0.8750	0.5555	89.83%	0.2612	
100	0.8523	0.7458	86.44%	0.8124	
Modelo con fine-tuning					
Épocas	*Val accuracy*	*Val loss*	*Test Accuracy*	*Test loss*	*Early stopping*
10	0.9318	0.1829	91.53%	0.5702	
20	0.9318	0.1617	91.53%	0.3415	
30	0.875	0.3650	89.83%	0.2301	
40	0.9545	0.1846	93.22%	0.3051	Épocas = 15 val_accuracy: 0.9205 val_loss: 0.1493 Test Accuracy: 91.53% Test Loss: 0.3452
50	0.9432	0.2305	93.22%	0.2803	
60	0.9659	0.1765	94.92%	0.1767	
70	0.9659	0.1841	94.92%	0.0940	
80	0.9659	0.2153	94.92%	0.0506	
90	0.9659	0.2464	98.31%	0.0416	
100	0.9318	0.2197	98.31%	0.0339	

Los modelos con el mejor desempeño de cada conjunto están marcados en azul. Tanto el Conjunto A como el Conjunto D demostraron un rendimiento superior en comparación con otros enfoques. Esto se puede deber a la forma en que se distribuyen las imágenes; en el conjunto A, se utilizaron todas las nuevas imágenes para el entrenamiento, lo que sugiere que hay un mayor número de imágenes disponibles para generalizar. Sin embargo, los conjuntos de validación y prueba provienen de la base de datos original, lo que contribuye a mejorar las métricas al verificar con datos previamente utilizados. Esto minimiza la diferencia entre el proceso de entrenamiento y la evaluación, facilitando una generalización efectiva del modelo. Por otro lado, el enfoque del conjunto D potencia la variabilidad del modelo durante el entrenamiento al combinar imágenes de la base de datos original con las nuevas, lo que a su vez mejora su capacidad para generalizar.

En la elección del modelo más adecuado, se consideraron las métricas y los gráficos producidos durante el reentrenamiento (exactitud y perdida). Se puede notar que, según la exactitud, los modelos que presentan los valores más altos son los del conjunto A con fine-tuning en las épocas 90 *y* 100; así como del conjunto D en la época 100 con el modelo sin fine-tuning. En la Figura 70 se pueden ver los gráficos generados durante el entrenamiento, donde a) corresponde al modelo con fine-tuning en la época 90 del conjunto A, b) se refiere al modelo con fine-tuning en la época 100 del conjunto A, y c) representa el modelo sin fine-tuning en la época 100 del conjunto D.

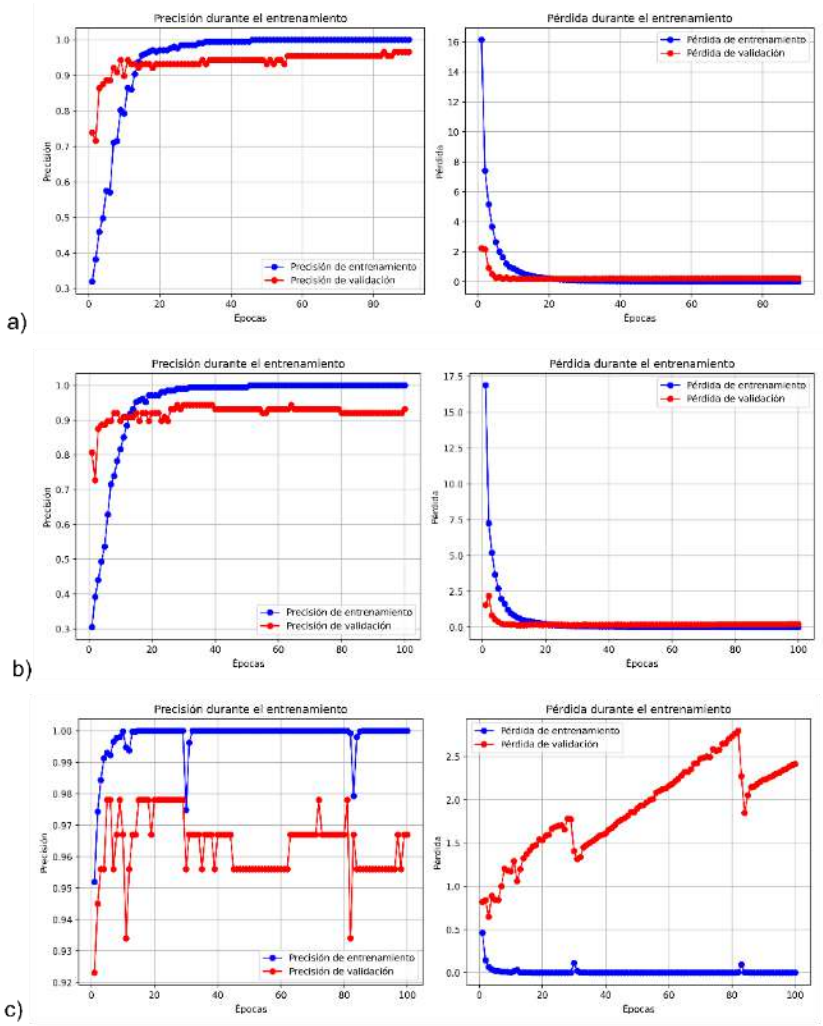

Figura 70. Graficas de exactitud y pérdida del conjunto de entrenamiento y validación del modelo_SP

Al examinar los gráficos, se determina que el modelo escogido corresponde al inciso a). Esto se debe a que el modelo del inciso c) muestra variaciones importantes durante el entrenamiento, especialmente en cuanto a la exactitud de validación. Además, la pérdida de validación va en aumento, mientras que la pérdida de entrenamiento se mantiene baja. Esto sugiere un sobreajuste,

indicando que el modelo ha aprendido los datos de entrenamiento, pero no ha conseguido generalizar bien con los datos de validación. Esto podría ser resultado de la combinación de imágenes viejas y nuevas, lo que probablemente introdujo variaciones en la distribución que impactaron negativamente en el aprendizaje. En cuanto a los incisos a) y b), el rendimiento es estable, con un incremento en la exactitud tanto en el entrenamiento como en la validación, alcanzando valores cercanos a uno. Igualmente, la pérdida disminuye de manera exponencial, estabilizándose en niveles cercanos a cero. Sin embargo, en el inciso a) existe una leve diferencia entre la exactitud de validación y la de entrenamiento (menos del 2%), lo que lo convierte en un modelo bien equilibrado.

Además, el fine tuning mejoró el rendimiento de los modelos, como se observan en las tablas del Anexo B, la mayoría de los modelos con mejor rendimiento son de modelos a los que se les aplico fine-tuning porque permitió reutilizar representaciones generales efectivas de las capas profundas, mientras adapta las capas superiores a las características específicas del nuevo conjunto de datos. Esto facilitó una mejor discriminación entre clases sin necesidad de entrenar el modelo desde cero, logrando mayor precisión y menor pérdida en validación, como se evidenció especialmente en el Conjunto A.

Por lo tanto, el modelo escogido fue el de fine-tuning con 90 épocas del conjunto A, llamado "$Modelo\_SP$". En la Tabla 29 se presenta el informe de clasificación por cada clase y en la Figura 71 se encuentra la matriz de confusión, donde se alcanzó una precisión total del 98%, acertando en 58 de 59 imágenes. Respecto al rendimiento por clase, la clase 2 (SP Unilateral) mostró un desempeño perfecto, logrando el máximo en todas las métricas evaluadas. La clase 0 (Sano) evidenció una leve reducción en el recall (0.96), lo que sugiere que una muestra de esta clase fue erróneamente etiquetada como otra, siendo clasificada como clase 1 (SP Bilateral). En contraste, la clase 1 (SP Bilateral) registró una precisión de 0.94, pero con un recall de 1.00, lo que indica la existencia de falsos positivos en esta categoría. El promedio macro se situó en 0.98 para precisión, 0.99 para recall y 0.98 para f1-score, lo que señala un desempeño equilibrado entre las distintas clases. Las métricas obtenidas demuestran una alta efectividad para clasificar adecuadamente las clases, manteniendo un balance entre precisión y recall. Este resultado sugiere que el modelo es sólido y se desempeña bien en los datos analizados.

Tabla 29 Reporte de clasificación de métricas del modelo SP

Reporte de clasificación			
	Precisión	Recall	F1-score
Sano o Clase 0	1.00	0.96	0.98
SP Bilateral o Clase 1	0.94	1.00	0.97
SP Unilateral o Clase 2	1.00	1.00	1.00
Promedio por clase	0.98	0.99	0.98

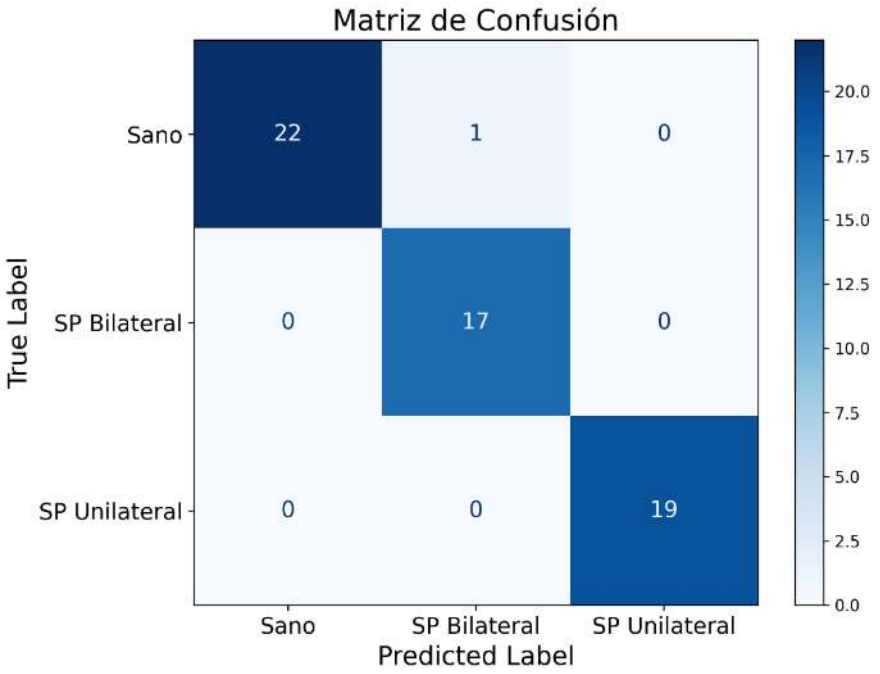

Figura 71. Matriz de confusión del modelo_SP

4.13. Discusión

En cuanto a la selección del modelo de inteligencia artificial, que es nuestro objeto de interés, se optó por una red neuronal convolucional (CNN), decisión respaldada por los resultados obtenidos en pruebas de ablación. Se realizaron múltiples experimentos con diferentes combinaciones de hiperparámetros y se implementaron diferentes técnicas de aumento de datos en el conjunto de imágenes termográficas.

Los mejores resultados se obtuvieron utilizando el conjunto de datos aumentado etiquetado como conjunto C. Se observó una mejora en el desempeño del modelo a medida que se entrenaba con un número mayor de imágenes. En particular, se evidenció un aumento en la exactitud tanto en la fase de validación como en la de prueba, y una disminución constante en la función de pérdida, lo que

sugiere un mejor ajuste del modelo al conjunto de datos. Algunos modelos alcanzaron una exactitud del 100%, con un mínimo de 84%. La selección del modelo óptimo se basó en las métricas de desempeño obtenidas con otros conjuntos de prueba y mediante la visualización de activaciones mediante Grad-CAM, herramienta que permitió verificar que el modelo enfocaba correctamente las regiones relevantes en las imágenes. El modelo final presentó métricas sobresalientes, alcanzando un 100% en precisión, sensibilidad, especificidad y F1-score.

La implementación del sistema embebido se realizó correctamente en la plataforma Jetson Nano, aunque fue necesario resolver varias limitaciones técnicas. Una de las principales dificultades fue la compatibilidad de la cámara con ciertas versiones del sistema JetPack, lo cual requirió ajustar la configuración del kernel para el uso correcto de los GPIO. Además, la Jetson Nano opera con Ubuntu 18.04 y Python 3.6.9, lo que restringió el uso de bibliotecas modernas, generando conflictos de compatibilidad al momento de ejecutar el modelo o diseñar la interfaz. A pesar de estas limitaciones, se logró adaptar el entorno y ejecutar correctamente el sistema, que permitió la identificación y clasificación de imágenes termográficas en tiempo real. Esto fue posible debido a la arquitectura simple del modelo CNN, que favorece la eficiencia computacional. Asimismo, la interfaz fue diseñada de manera ligera para evitar saturar los recursos del sistema, considerando también que la sobrecarga térmica es un riesgo potencial en estos dispositivos.

La funcionalidad del sistema fue validada mediante pruebas clínicas, en las cuales se logró capturar correctamente las imágenes termográficas a través de la interfaz desarrollada. Estas imágenes fueron almacenadas localmente en el sistema, y posteriormente utilizadas para reentrenar el modelo, con el objetivo de robustecer su capacidad de generalización. Se realizaron cuatro pruebas experimentales con distintas bases de datos, siendo la prueba D la que arrojó los mejores resultados, con una exactitud del 98%. No obstante, se observó una ligera disminución del rendimiento (aproximadamente 2%) al incluir nuevas imágenes, debido a las variaciones en las condiciones de captura y a la resolución de las cámaras utilizadas. Estos factores dificultan la generalización del modelo, ya que, aunque las imágenes representan clases similares, presentan diferencias sutiles en su adquisición.

En la comparación con otros modelos de clasificación termográfica, se observó que el modelo propuesto supera en métricas a diversos enfoques previos. Por ejemplo, en el trabajo de Trejo-

Chávez et al. (2022) —en el cual se basa parcialmente este proyecto— se utilizó una base de datos de lesiones en rodilla y se reportó una exactitud del 97.44%, inferior al 98% alcanzado por el modelo desarrollado en esta tesis. Asimismo, en términos de precisión, F1-score y recall, se obtuvo un 100%. En comparación con otros sistemas embebidos similares, el modelo aquí propuesto también mostró un mejor desempeño, superando métricas reportadas de 59%, 94% y 97% (Tabla 30).

Sin embargo, para lograr un modelo verdaderamente robusto, capaz de clasificar imágenes independientemente de la cámara termográfica utilizada, es fundamental incrementar la base de datos. Es necesario adquirir más imágenes y garantizar un balance adecuado entre las clases, lo cual contribuiría a mejorar la capacidad de generalización del modelo y a reducir el riesgo de sobreajuste. De esta manera, se podrían obtener métricas más estables y un desempeño más confiable en diferentes escenarios clínicos o reales.

Tabla 30. Comparación de resultados.

Artículo	Enfermedad	Modelo / Dispositivo	Exactitud
Propio	Síndrome patelofemoral	CNN / Jetson Nano	98%
(Trejo-Chavez et al., 2022)	Lesiones en rodilla	CNN	97.44%
(Millán Duque, 2021)	Plaga en plantas	CNN / Jetson Nano	59.43%
(Saponara et al., 2021).	Detección de personas	CNN / Jetson Nano	94%
(Azariadi et al., 2016)	ECG	SVM / Intel Galileo	97%

CAPITULO 5

CONCLUSIÓN

Se cumplieron los objetivos al desarrollar un modelo de red neuronal convolucional (CNN) que, tras un proceso de selección basado en pruebas de ablación y técnicas de aumento de datos, alcanzó una exactitud del 98% después de la obtención de los termogramas en las pruebas clínicas.

Se desarrolló un sistema embebido en el que se implementó el modelo CNN capaz de obtener termogramas en tiempo real. Para ello, se integró correctamente en el sistema embebido conformado por una tarjeta Jetson Nano, cámara termográfica (Lepton 3.5), cámara digital (Raspberry Pi 2.1V), display HDMI. Asimismo, se logró la adquisición de termogramas en tiempo real y se desarrolló una interfaz funcional. El sistema completo fue montado en una carcasa diseñada e impresa en 3D con apoyo del personal de la carrera de Diseño Industrial.

La validación clínica se realizó en colaboración con personal experto de fisioterapia de la Universidad Autónoma de Querétaro. Durante estas pruebas, se confirmó que el sistema embebido puede no solo capturar imágenes termográficas en tiempo real, sino también ejecutar el modelo CNN e incluso realizar reentrenamiento en campo.

El producto final del sistema embebido ya funcionando se muestra en la Figura 72.

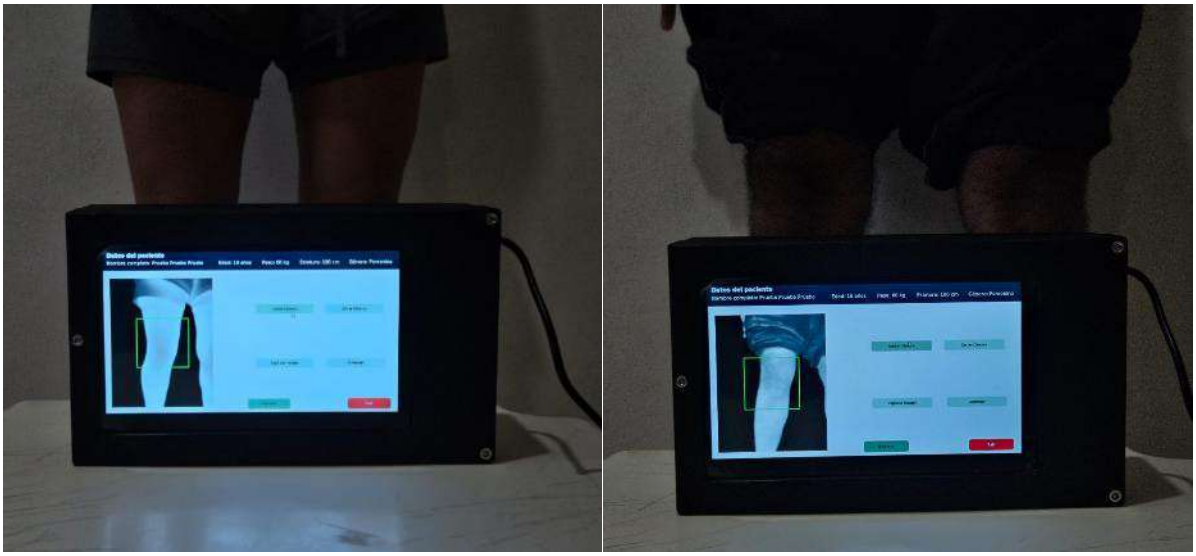

Figura 72. Sistema embebido final

En respuesta a la pregunta realizada al principio: "¿Puede un modelo de aprendizaje profundo aplicado en un sistema embebido lograr la identificación, detección y clasificación de anomalías fisiológicas en miembro inferior?", el sistema embebido desarrollado demuestra que es posible implementar con éxito un modelo (CNN) en un sistema embebido de bajo costo basada en termografía para la detección de anomalías en el miembro inferior.

PROSPECTIVAS

Como trabajo futuro, se plantea ampliar la base de datos termográfica con nuevas adquisiciones, procurando un balance adecuado entre clases. Esto permitirá mejorar la capacidad de generalización del modelo y reducir el riesgo de sobreajuste.

REFERENCIAS

Aized Amin Soofi, & Arshad Awan. (2017a). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*, 459–465. https://doi.org/10.6000/1927-5129.2017.13.76

Aized Amin Soofi, & Arshad Awan. (2017b). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*, 459–465. https://doi.org/10.6000/1927-5129.2017.13.76

Alshehri, A., & AlSaeed, D. (2022). Breast Cancer Detection in Thermography Using Convolutional Neural Networks (CNNs) with Deep Attention Mechanisms. *Applied Sciences*, *12*(24), 12922. https://doi.org/10.3390/app122412922

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

Ammer, K., & Ring, F. (2019). *The Thermal Human Body*. CRC Press.

Azariadi, D., Tsoutsouras, V., Xydis, S., & Soudris, D. (2016). ECG signal analysis and arrhythmia detection on IoT wearable medical devices. *2016 5th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 1–4. https://doi.org/10.1109/MOCAST.2016.7495143

Borrella Petisco, B. (2022). *INTRODUCCIÓN A LA VISIÓN ARTIFICIAL. PROCESOS Y APLICACIONES* [TRABAJO FIN DE GRADO]. UNIVERSIDAD COMPLUTENSE DE MADRID.

Buduma, N., & Papa, J. (2022). *Fundamentals of Deep Learning*. "O'Reilly Media, Inc."

C S Psyco. (2021, September 27). *Install Ubuntu and VMware on windows 11 - how to install ubuntu on windows 11 VMWare workstation - YouTube*. https://www.youtube.com/watch?v=4UuWFNON1M0&t=365s

Canonical Ltd. (2018). *Ubuntu 18.04.6 LTS (Bionic Beaver)*. https://releases.ubuntu.com/bionic/

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.

Code with Aarohi. (2023, January 16). *L-3 Install OpenCV 4.5 on NVIDIA Jetson Nano | Set Up a Camera for NVIDIA Jetson Nano - YouTube*. https://www.youtube.com/watch?v=P-EZr0zy53g&t=231s

Cognex Corporation. (2018). *INTRODUCCIÓN A LA VISIÓN ARTIFICIAL: Una guía para la automatización de procesos y mejorar la calidad*. https://bcnvision.es/uploads/videotutoriales/uploads/guias%20por%20sectores/introduccion%20a%20la%20vision%20artificial_compressed.pdf

Côrte, A. C., Pedrinelli, A., Marttos, A., Souza, I. F. G., Grava, J., & José Hernandez, A. (2019). Infrared thermography study as a complementary method of screening and prevention of muscle injuries: pilot study. *BMJ Open Sport & Exercise Medicine*, *5*(1), e000431. https://doi.org/10.1136/bmjsem-2018-000431

Cruz-Vega, I., Hernandez-Contreras, D., Peregrina-Barreto, H., Rangel-Magdaleno, J. de J., & Ramirez-Cortes, J. M. (2020). Deep Learning Classification for Diabetic Foot Thermograms. *Sensors*, *20*(6), 1762. https://doi.org/10.3390/s20061762

David Y. Gaitonde, M., Alex Ericksen, M., & Rachel C. Robbins, M. (2019). Patellofemoral Pain Syndrome. *American Family Physician*, *99*(2), 88–94.

Davies, E. R. (2017). *Computer Vision*. Academic Press.

de Freitas Barbosa, V. A., de Santana, M. A., Andrade, M. K. S., de Lima, R. de C. F., & dos Santos, W. P. (2020). Deep-wavelet neural networks for breast cancer early diagnosis using mammary termographies. In *Deep Learning for Data Analytics* (pp. 99–124). Elsevier. https://doi.org/10.1016/B978-0-12-819764-6.00007-7

Diakides, M., Bronzino, J. D., & Peterson, D. R. (2012). *Medical Infrared Imaging*. CRC Press.

Dixit, S., DiFiori, J. P., Burton, M., & Mines, B. (2007). Management of Patellofemoral Pain Syndrome.

American Family Physician, *75*(2), 194–202. https://www.aafp.org/pubs/afp/issues/2007/0115/p194.pdf
Dufour, M. (2012). Anatomía del miembro inferior. *EMC - Podología*, *14*(4), 1–12. https://doi.org/10.1016/S1762-827X(12)61929-4
Elgendy, M. (2020). *Deep Learning for Vision Systems*. Simon and Schuster.
Escamilla-Galindo, V. L., Estal-Martínez, A., Adamczyk, J. G., Brito, C. J., Arnaiz-Lastras, J., & Sillero-Quintana, M. (2017). Skin temperature response to unilateral training measured with infrared thermography. *Journal of Exercise Rehabilitation*, *13*(5), 526–534. https://doi.org/10.12965/jer.1735046.523
Farooq, M. A., & Corcoran, P. (2020). Infrared Imaging for Human Thermography and Breast Tumor Classification using Thermal Images. *2020 31st Irish Signals and Systems Conference (ISSC)*, 1–6. https://doi.org/10.1109/ISSC49989.2020.9180164
Filipe, V., Teixeira, P., & Teixeira, A. (2022). Automatic Classification of Foot Thermograms Using Machine Learning Techniques. *Algorithms*, *15*(7), 236. https://doi.org/10.3390/a15070236
García González, C., Albaladejo Vicente, R., Villanueva Orbáiz, R., & Navarro Cabello, E. (2015). Deporte de ocio en España: epidemiología de las lesiones y sus consecuencias. *Apunts Educación Física y Deportes*, *119*, 62–70. https://doi.org/10.5672/apunts.2014-0983.es.(2015/1).119.03
Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). *Fundamental Concepts of Convolutional Neural Network* (pp. 519–567). https://doi.org/10.1007/978-3-030-32644-9_36
Gómez-Carmona, P., Fernández-Cuevas, I., Sillero-Quintana, M., Arnaiz-Lastras, J., & Navandar, A. (2020). Infrared Thermography Protocol on Reducing the Incidence of Soccer Injuries. *Journal of Sport Rehabilitation*, *29*(8), 1222–1227. https://doi.org/10.1123/jsr.2019-0056
González-Acuña, R. G., & Chaparro-Romo, H. A. (2020). *Stigmatic Optics*.
Hall, J. E. (2015). *Guyton & Hall Physiology Review E-Book*. Elsevier Health Sciences.
Hamilton, D., Pacheco, R., Myers, B., & Peltzer, B. (2020). kNN vs. SVM: A comparison of algorithms. In S. M. Hood, S. Drury, T. Steelman, & R. Steffens (Eds.), *Proceedings of the Fire Continuum—Preparing for the Future of Wildland Fire* (pp. 95–109). U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.
Instituto Nacional de Rehabilitación. (2021). *Boletín Médico e Informativo del Instituto Nacional de Rehabilitación*.
jcl9309, & KevinFFF. (2023, April). *Jetson Nano SPI Bus Not Working - Jetson & Embedded Systems / Jetson Nano - NVIDIA Developer Forums*. https://forums.developer.nvidia.com/t/jetson-nano-spi-bus-not-working/249482/9
jetsonhacks. (2019, March 23). *GitHub - JetsonHacksNano/CSI-Camera: Simple example of using a CSI-Camera (like the Raspberry Pi Version 2 camera) with the NVIDIA Jetson Developer Kit*. https://github.com/JetsonHacksNano/CSI-Camera
JetsonHacks. (2023, March 7). *NVIDIA SDK Manager Tutorial: Installing Jetson Software Explained - YouTube*. https://www.youtube.com/watch?v=Ucg5Zqm9ZMk
JetsonHacks. (2024, October 21). *Install VS Code on Jetson Nano in 2024 - YouTube*. https://www.youtube.com/watch?v=epSAtsCOii4
Jiménez Díaz, J. F. (2006). Lesiones musculares en el deporte. *International Journal of Sport Science*, *II*, 55–67.
jvachteren, & SchaneCCC. (2021, November). *Jetson Nano trouble using SPI - Jetson & Embedded Systems / Jetson Nano - NVIDIA Developer Forums*. https://forums.developer.nvidia.com/t/jetson-nano-trouble-using-spi/195510
kangalow. (2019, April 2). *Jetson Nano + Raspberry Pi Camera - JetsonHacks*. https://jetsonhacks.com/2019/04/02/jetson-nano-raspberry-pi-camera/#google_vignette
Katsaragakis, M., Papadopoulos, L., Konijnenburg, M., Catthoor, F., & Soudris, D. (2020). Memory Footprint Optimization Techniques for Machine Learning Applications in Embedded Systems. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–4. https://doi.org/10.1109/ISCAS45731.2020.9181038

Kelleher, J. D. (2019). *Deep Learning*. MIT Press.
Kurniawan, A. (2021). *IoT Projects with NVIDIA Jetson Nano*. Apress. https://doi.org/10.1007/978-1-4842-6452-2
Lankhorst, N. E., Bierma-Zeinstra, S. M. A., & van Middelkoop, M. (2012). Risk Factors for Patellofemoral Pain Syndrome: A Systematic Review. *Journal of Orthopaedic & Sports Physical Therapy*, *42*(2), 81–94. https://doi.org/10.2519/jospt.2012.3803
Latarjet, M., & Liard, A. R. (2004). *Anatomía Humana*. Ed. Médica Panamericana.
Maret, Y., Oberson, D., & Gavrilova, M. (2018). Real-Time Embedded System for Gesture Recognition. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 30–34. https://doi.org/10.1109/SMC.2018.00014
Michael Vernon. (2023, May 29). *GitHub - mishave/pylepton: Quick and dirty pure python library for interfacing with FLIR lepton*. https://github.com/mishave/pylepton
Millán Duque, L. M. (2021). *Sistema embebido para la clasificación de imágenes en tiempo real en aplicaciones agrícolas y/o industriales* [BachelorThesis, Universidad Tecnológica de Pereira]. https://repositorio.utp.edu.co/items/d9fd3c79-779f-4051-9d9d-652ef1346b8e
Mohamed, E. A., Rashed, E. A., Gaber, T., & Karam, O. (2022). Deep learning model for fully automated breast cancer detection system from thermograms. *PLOS ONE*, *17*(1), e0262349. https://doi.org/10.1371/journal.pone.0262349
Moreno Pascual, C., Rodríguez Pérez, V., & Seco Calvo, J. (2008). Epidemiología de las lesiones deportivas. *Fisioterapia*, *30*(1), 40–48. https://doi.org/10.1016/S0211-5638(08)72954-7
Muñoz Ch., S., Astudillo A., C., Miranda V., E., & Albarracin G., J. F. (2018). Lesiones musculares deportivas: Correlación entre anatomía y estudio por imágenes. *Revista Chilena de Radiología*, *24*(1), 22–33. https://doi.org/10.4067/S0717-93082018000100022
Myzhar. (2020, July 12). *Thermal Images on Jetson™ Nano with FLIR Lepton3 – Myzhar's MyzharBot and more…*. https://www.myzhar.com/blog/jetson-nano-with-flir-lepton3/
NVIDIA. (n.d.). *Jetson Nano Developer Kit*. Retrieved June 12, 2025, from https://developer.download.nvidia.com/assets/embedded/secure/jetson/Nano/docs/NV_Jetson_Nano_Developer_Kit_User_Guide.pdf?__token__=exp=1749843610~hmac=722b2d2de7e37aac557835a79454182b893cc4522e1aff61244aa33e12a81a87&t=eyJscyI6ImdzZW8iLCJsc2QiOiJodHRwczovL3d3dy5nb29nbGUuY29tLyJ9
NVIDIA Corporation. (n.d.). *Get Started With Jetson Nano Developer Kit | NVIDIA Developer*. Retrieved April 9, 2025, from https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#prepare
NVIDIA Developer. (n.d.). *SDK Manager | NVIDIA Developer*. Retrieved April 9, 2025, from https://developer.nvidia.com/sdk-manager
Passos, M., & Rocha, A. (2022). Evaluation of infrared thermography with a portable camera as a diagnostic tool for peripheral arterial disease of the lower limbs compared with color Doppler ultrasonography. *Archives of Medical Science – Atherosclerotic Diseases*, *7*(1), 66–72. https://doi.org/10.5114/amsad/150716
Petrou, M., & Petrou, C. (2010). *Image Processing: The Fundamentals*. Wiley. https://doi.org/10.1002/9781119994398
Rafaela Rosas, M. (2011). Lesiones deportivas. Clínica y tratamiento. *Offarm*, *30*(3), 36–42. https://www.elsevier.es/es-revista-offarm-4-articulo-lesiones-deportivas-clinica-tratamiento-X0212047X11205082
Rodríguez-Sanz, D., Losa-Iglesias, M. E., López-López, D., Calvo-Lobo, C., Palomo-López, P., & Becerro-de-Bengoa-Vallejo, R. (2017). Infrared thermography applied to lower limb muscles in elite soccer players with functional ankle equinus and non-equinus condition. *PeerJ*, *5*, e3388. https://doi.org/10.7717/peerj.3388
Sánchez Granja, G., & Asanza, V. (2021). *Implementación de algoritmo de machine learning en un sistema embebido para control de prótesis activa utilizando señales mioeléctricas* [Tesis de Grado, Escuela Superior Politécnica del Litoral].

https://www.dspace.espol.edu.ec/handle/123456789/57230
Saponara, S., Elhanashi, A., & Gagliardi, A. (2021). Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19. *Journal of Real-Time Image Processing*, *18*(6), 1937–1947. https://doi.org/10.1007/s11554-021-01070-6
Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626. https://doi.org/10.1109/ICCV.2017.74
Szeliski, R. (2022). *Computer Vision*. Springer International Publishing. https://doi.org/10.1007/978-3-030-34372-9
T., Tortora, G. J., & Derrickson, B. H. (2011). *Principles of Anatomy and Physiology*.
Teuwen, J., & Moriakov, N. (2020). Convolutional neural networks. In *Handbook of Medical Image Computing and Computer Assisted Intervention* (pp. 481–501). Elsevier. https://doi.org/10.1016/B978-0-12-816176-0.00025-9
Theobald, O. (2017). *Machine Learning for Absolute Beginners: A Plain English Introduction* (2nd ed.). Scatterplot Press.
Trejo-Chavez, O., Amezquita-Sanchez, J. P., Huerta-Rosales, J. R., Morales-Hernandez, L. A., Cruz-Albarran, I. A., & Valtierra-Rodriguez, M. (2022). Automatic Knee Injury Identification through Thermal Image Processing and Convolutional Neural Networks. *Electronics*, *11*(23), 3987. https://doi.org/10.3390/electronics11233987
Trejo-Chavez, O., Priego-Quesada, J. I., Gonzalez-Hernandez, M. P., Morales-Hernandez, L. A., & Cruz-Albarran, I. A. (2023). Knee skin temperature response of patients with bilateral patellofemoral syndrome before and after heat and cold stress. *Journal of Thermal Biology*, *115*, 103601. https://doi.org/10.1016/j.jtherbio.2023.103601
Vardasca, R., Ring, E. F. J., Plassmann, P., & Jones, C. (2012). Termal symmetry of the upper and lower extremities in healthy subjects. *Thermology International*, *22*, 53–60.
Vega Mancilla, S. A. (2022). *Sistema de visión artificial basado en termografía para detección de anomalías musculares en el tren inferior del cuerpo humano*. Universidad Autónoma de Querétaro.
Visual Studio Code. (n.d.). *Visual Studio Code FAQ*. Retrieved April 9, 2025, from https://code.visualstudio.com/docs/supporting/faq#_previous-release-versions
Vollmer, M., & Möllmann, K. P. (2011). *Infrared Thermal Imaging*. John Wiley & Sons.
Waryasz, G. R., & McDermott, A. Y. (2008). Patellofemoral pain syndrome (PFPS): a systematic review of anatomy and potential risk factors. *Dynamic Medicine*, *7*(1), 9. https://doi.org/10.1186/1476-5918-7-9
Williams, T. (2009). *Thermal Imaging Cameras*. CRC Press.

ANEXO A

Tabla 31. Tabla de resultados del conjunto de imágenes aumentadas A (learning rate = 0.0001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.0001	8	10	84.09	0.4938	91.53	0.3086
2	0.0001	8	20	82.95	0.7627	93.22	0.1887
3	0.0001	8	30	90.91	0.3857	91.53	0.3782
4	0.0001	8	40	100	0.0367	93.22	0.2227
5	0.0001	8	50	93.18	0.2571	96.61	0.0757
6	0.0001	8	60	95.45	0.0846	93.22	0.2199
7	0.0001	8	70	97.73	0.0771	94.92	0.0927
8	0.0001	8	80	92.05	0.2789	94.92	0.143
9	0.0001	8	90	93.18	0.2479	98.31	0.0236
10	0.0001	8	100	95.45	0.1427	96.61	0.0403
11	0.0001	10	10	86.36	0.3541	96.44	0.3744
12	0.0001	10	20	87.5	0.3258	91.53	0.2489
13	0.0001	10	30	88.64	0.4822	98.31	0.0613
14	0.0001	10	40	95.23	0.6389	89.83	0.4444
15	0.0001	10	50	87.5	0.2737	89.83	0.2756
16	0.0001	10	60	90.91	0.3536	93.22	0.2336
17	0.0001	10	70	90.91	0.3465	96.61	0.1333
18	0.0001	10	80	95.45	0.2214	94.92	0.167
19	0.0001	10	90	93.18	0.3997	94.92	0.1772
20	0.0001	10	100	94.32	0.268	94.92	0.2299
21	0.0001	12	10	82.95	1.3172	86.44	0.9766
22	0.0001	12	20	88.64	0.229	86.44	0.5763
23	0.0001	12	30	90.91	0.3483	94.92	0.3503
24	0.0001	12	40	93.18	0.273	94.92	0.1642
25	0.0001	12	50	93.18	0.2485	94.92	0.1735
26	0.0001	12	60	92.05	0.152	96.61	0.0801
27	0.0001	12	70	90.91	0.2149	96.61	0.1477
28	0.0001	12	80	93.18	0.5614	94.92	0.2632
29	0.0001	12	90	92.05	0.1948	94.92	0.1264
30	0.0001	12	100	95.45	0.1812	96.61	0.1007

Tabla 32. Tabla de resultados del conjunto de imágenes aumentadas A (learning rate = 0.0001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	8	10	77.27	0.4473	77.97	0.4533
2	0.001	8	20	94.32	0.0776	98.31	0.01
3	0.001	8	30	93.18	0.2221	93.22	0.1592
4	0.001	8	40	82.95	0.6118	83.05	0.2954
5	0.001	8	50	76.24	0.5989	66.1	0.7647
6	0.001	8	60	97.73	0.0501	96.61	0.0565
7	0.001	8	70	84.09	0.4576	81.36	0.4323
8	0.001	8	80	94.32	0.3049	96.61	0.1201
9	0.001	8	90	95.45	0.1725	93.22	0.3893
10	0.001	8	100	93.18	0.7856	88.14	0.6887
11	0.001	10	10	90.91	0.2299	96.61	0.1702
12	0.001	10	20	94.32	0.2129	83.05	0.4392
13	0.001	10	30	96.59	0.1364	94.92	0.1856
14	0.001	10	40	86.36	0.4944	91.53	0.1702
15	0.001	10	50	93.18	0.2433	98.31	0.0334
16	0.001	10	60	94.32	0.2114	94.92	0.3369
17	0.001	10	70	96.59	0.072	96.61	0.0722
18	0.001	10	80	82.95	0.3507	83.05	0.4902
19	0.001	10	90	97.73	0.4713	94.92	0.2949
20	0.001	10	100	85.23	1.2682	91.53	0.3435
21	0.001	12	10	88.64	0.312	88.14	0.2705
22	0.001	12	20	89.77	0.3289	89.83	0.258
23	0.001	12	30	94.32	0.2201	96.61	0.1222
24	0.001	12	40	87.5	0.3192	88.14	0.2259
25	0.001	12	50	97.73	0.1769	98.31	0.0249
26	0.001	12	60	85.23	0.5717	91.53	0.2398
27	0.001	12	70	85.23	0.6419	83.05	0.4463
28	0.001	12	80	98.86	0.0212	93.22	0.1647
29	0.001	12	90	94.32	0.5707	91.53	0.8343
30	0.001	12	100	93.18	0.5831	91.53	0.3036

Tabla 33. Tabla de resultados del conjunto de imágenes aumentadas B (learning rate = 0.0001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.0001	8	10	93.18	0.1268	96.61	0.0964
2	0.0001	8	20	84.09	0.3965	91.53	0.4818
3	0.0001	8	30	88.64	0.2363	98.31	0.0527
4	0.0001	8	40	96.59	0.1598	98.31	0.0126
5	0.0001	8	50	95.45	0.2126	100	0.0001
6	0.0001	8	60	98.86	0.0814	98.31	0.0249
7	0.0001	8	70	96.59	0.1284	98.31	0.1019
8	0.0001	8	80	97.73	0.064	98.31	0.1431
9	0.0001	8	90	98.86	0.0267	100	0
10	0.0001	8	100	98.86	0.1447	98.31	0.0301
11	0.0001	10	10	92.05	0.1723	98.31	0.0535
12	0.0001	10	20	97.73	0.0824	98.31	0.083
13	0.0001	10	30	95.45	0.1604	98.31	0.1034
14	0.0001	10	40	93.18	0.289	96.61	0.1889
15	0.0001	10	50	96.59	0.3414	100	–
16	0.0001	10	60	98.86	0.0283	98.31	0.0204
17	0.0001	10	70	92.05	0.2363	96.61	0.2037
18	0.0001	10	80	93.18	0.2404	100	0.0072
19	0.0001	10	90	96.59	0.1444	98.31	0.0696
20	0.0001	10	100	96.59	0.1081	100	0.001
21	0.0001	12	10	97.73	0.0658	96.61	0.0804
22	0.0001	12	20	98.86	0.0847	94.92	0.0803
23	0.0001	12	30	95.45	0.1125	96.16	0.1195
24	0.0001	12	40	97.73	0.0845	100	0.0018
25	0.0001	12	50	90.91	0.371	96.61	0.1387
26	0.0001	12	60	92.05	0.1964	98.31	0.0826
27	0.0001	12	70	97.73	0.184	94.92	0.2816
28	0.0001	12	80	97.73	0.1078	98.31	0.0394
29	0.0001	12	90	98.86	0.0539	100	0.0109
30	0.0001	12	100	92.05	0.3018	98.31	0.525

Tabla 34. Tabla de resultados del conjunto de imágenes aumentadas A (learning rate = 0.001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	8	10	95.45	0.132	89.83	0.1845
2	0.001	8	20	92.05	0.1861	94.92	0.1249
3	0.001	8	30	97.73	0.0676	96.61	0.0419
4	0.001	8	40	98.86	0.0143	98.31	0.0341
5	0.001	8	50	95.45	0.2681	98.31	0.0483
6	0.001	8	60	100	$5.18e-04$	100	0.0023
7	0.001	8	70	90.91	0.6971	94.92	0.201
8	0.001	8	80	97.73	0.1507	93.22	0.4698
9	0.001	8	90	87.5	0.4565	88.14	0.3568
10	0.001	8	100	97.73	0.5397	96.61	0.6962
11	0.001	10	10	90.91	0.2604	83.05	0.3415
12	0.001	10	20	94.32	0.2375	93.22	0.2771
13	0.001	10	30	94.32	0.3813	93.22	0.2771
14	0.001	10	40	96.59	0.0732	100	0.001
15	0.001	10	50	96.59	0.1704	100	0.0002
16	0.001	10	60	98.86	0.0577	98.31	0.1291
17	0.001	10	70	96.59	0.0943	98.31	0.876
18	0.001	10	80	98.86	0.1316	96.61	0.1259
19	0.001	10	90	94.32	0.9764	93.22	0.3282
20	0.001	10	100	95.45	0.4601	93.22	0.3563
21	0.001	12	10	98.86	0.0723	94.92	0.0797
22	0.001	12	20	85.23	0.5796	93.22	0.2254
23	0.001	12	30	95.45	0.1612	96.61	0.0653
24	0.001	12	40	98.86	0.279	100	0.0002
25	0.001	12	50	100	0.0017	1000	–
26	0.001	12	60	94.32	0.15	96.61	0.2241
27	0.001	12	70	100	0.0028	96.61	0.2753
28	0.001	12	80	95.45	0.3846	98.31	0.0819
29	0.001	12	90	96.59	0.2976	98.31	0.0592
30	0.001	12	100	100	$2.64e-04$	96.61	0.4529

Tabla 35. Tabla de resultados del conjunto de imágenes aumentadas C (learning rate = 0.0001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.001	8	10	85.23	0.3772	84.75	0.2743
2	0.001	8	20	92.05	0.5473	88.14	0.3359
3	0.001	8	30	97.73	0.2036	94.92	0.1678
4	0.001	8	40	97.73	0.0909	96.61	0.5944
5	0.001	8	50	94.32	0.2201	100	0.011
6	0.001	8	60	95.45	0.5075	100	0
7	0.001	8	70	98.86	0.0096	96.61	0.1588
8	0.001	8	80	94.32	0.8281	100	0.0067
9	0.001	8	90	96.59	0.4859	98.31	0.3757
10	0.001	8	100	97.73	0.0334	96.61	0.2096
11	0.001	10	10	98.86	0.0172	98.31	0.0947
12	0.001	10	20	95.45	0.2008	96.61	0.054
13	0.001	10	30	94.32	0.6705	98.31	0.0651
14	0.001	10	40	97.73	0.1865	96.61	0.0396
15	0.001	10	50	98.86	0.0198	93.22	0.6371
16	0.001	10	60	94.32	0.9542	88.14	1.5364
17	0.001	10	70	90.91	0.9118	94.92	0.5109
18	0.001	10	80	97.73	0.1326	100	0
19	0.001	10	90	97.73	0.0569	96.61	0.887
20	0.001	10	100	98.86	0.021	93.22	0.236
21	0.001	12	10	80.68	0.473	86.44	0.3408
22	0.001	12	20	92.05	0.2558	88.14	0.781
23	0.001	12	30	100	0.0048	100	0.002
24	0.001	12	40	100	0.0125	98.31	0.0371
25	0.001	12	50	97.73	0.1226	91.53	0.9736
26	0.001	12	60	97.73	0.0716	98.13	0.1569
27	0.001	12	70	95.45	0.5279	98.31	0.2207
28	0.001	12	80	100	$7.52e-05$	98.31	0.1528
29	0.001	12	90	88.64	0.8774	88.14	1.218
30	0.001	12	100	100	$1.95e-07$	100	0

Tabla 36. Tabla de resultados del conjunto de imágenes aumentadas C (learning rate = 0.001)

Número del experimento	Learning rate	Batch size	Epochs	Val Accuracy (%)	Val loss	Test Accuracy (%)	Test loss
1	0.0001	8	10	0.8977	0.1844	93.22	0.2112
2	0.0001	8	20	0.9545	0.1036	93.22	0.1451
3	0.0001	8	30	0.9773	0.0629	94.92	0.2037
4	0.0001	8	40	0.9091	0.4854	93.22	0.3420
5	0.0001	8	50	1	0.0030	100	0.0086
6	0.0001	8	60	0.9659	0.1599	94.92	0.1369
7	0.0001	8	70	0.9886	0.0626	96.61	0.0979
8	0.0001	8	80	0.9886	0.0253	98.31	0.0231
9	0.0001	8	90	0.9886	0.0590	98.31	0.0435
10	0.0001	8	100	0.9659	0.1853	98.31	0.2706
11	0.0001	10	10	0.9091	0.2401	94.92	0.1305
12	0.0001	10	20	0.9205	0.2459	93.22	0.2053
13	0.0001	10	30	1	0.0038	98.31	0.0784
14	0.0001	10	40	1	0.0020	98.31	0.0827
15	0.0001	10	50	0.9773	0.1175	100	0.0043
16	0.0001	10	60	0.9545	0.2239	96.61	0.1255
17	0.0001	10	70	1	0.0164	100	0.0018
18	0.0001	10	80	0.9659	0.0736	100	0.0016
19	0.0001	10	90	0.9545	0.3805	98.31	0.2411
20	0.0001	10	100	0.9659	0.0790	94.92	0.5548
21	0.0001	12	10	0.9432	0.1951	91.53	0.2806
22	0.0001	12	20	0.9432	0.2046	94.92	0.1362
23	0.0001	12	30	0.9659	0.1869	93.22	0.1201
24	0.0001	12	40	0.9886	0.0811	98.31	0.1765
25	0.0001	12	50	0.9773	0.1843	100	0.0062
26	0.0001	12	60	0.9659	0.1258	100	0.0023
27	0.0001	12	70	0.9886	0.1261	100	0.0152
28	0.0001	12	80	1	0.0052	98.31	0.0877
29	0.0001	12	90	1	0.0023	100	0.0002
30	0.0001	12	100	0.9432	0.3932	93.22	0.2683

ANEXO B

Tabla 37. Tabla de resultados de la Prueba A

Modelo normal						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0.8644	0.3295	88.64%	0.265	
2	20	0.8983	0.2211	90.91%	0.3663	
3	30	0.9153	0.1388	88.64%	0.3873	
4	40	0.8983	0.1908	90.91%	0.3710	Épocas: 21
5	50	0.8814	0.1743	88.64%	0.4582	Val_accuracy: 0.9153
6	60	0.8814	0.2301	90.91%	0.4219	Val_loss: 0.2117
7	70	0.8814	0.2552	88.64%	0.4125	Test Accuracy: 92.05%
8	80	0.8644	0.2607	87.50%	0.5139	Test Loss: 0.3777.
9	90	0.8475	0.3829	82.95%	0.7184	
10	100	0.8136	0.461	88.64%	0.5411	
Fine Tunning (2 capas)						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0.9153	0.1957	90.91%	0.3326	
2	20	0.9153	0.2554	86.36%	0.3715	
3	30	0.8983	0.1703	89.77%	0.3627	
4	40	0.8983	0.183	92.05%	0.4063	Épocas: 28
5	50	0.9322	0.2958	95.45%	0.2073	Val_accuracy: 0.9153
6	60	0.9492	0.16	95.45%	0.1561	Val_loss: 0.2595
7	70	0.9492	0.0628	95.45%	0.1809	Test Accuracy: 95.45%
8	80	0.9831	0.053	95.45%	0.2239	Test Loss: 0.1465
9	90	0.9492	0.1377	92.05%	0.2069	
10	100	0.8644	0.6202	85.23%	0.6092	

Tabla 38. Tabla de resultados de la Prueba B

Modelo normal						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0.75	1.5535	73.02%	1.7062	
2	20	0.7826	1.2891	77.78%	1.2341	
3	30	0.75	1.7084	69.84%	1.6585	
4	40	0.7609	2.0859	74.60%	1.661	Épocas: 21
5	50	0.7174	1.9047	66.67%	1.7833	Val_accuracy: 0.7717
6	60	0.75	2.536	68.25%	2.0491	Val_loss: 1.4805
7	70	0.7283	2.0681	68.25%	1.9204	Test Accuracy: 73.02%
8	80	0.6739	2.509	66.67%	2.3088	Test Loss: 1.3559
9	90	0.6957	2.3828	71.43%	2.0517	
10	100	0.7391	2.2651	68.25%	1.9852	
Fine Tunning (2 capas)						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
11	10	0.7174	2.5311	76.19%	2.0946	
12	20	0.7826	1.5075	76.19%	1.3631	
13	30	0.7935	2.3522	82.54%	1.5811	
14	40	0.7717	2.4931	84.13%	1.6407	Épocas: 6
15	50	0.8043	2.5395	82.54%	1.7618	Val_accuracy: 0.6848
16	60	0.7935	2.607	84.13%	1.6874	Val_loss: 3.3256
17	70	0.7826	2.6821	80.95%	1.75	Test Accuracy: 69.84%
18	80	0.7935	2.6609	82.54%	1.6052	Test Loss: 2.8973
19	90	0.7935	2.7199	82.54%	1.6563	
20	100	0.7935	2.819	82.54%	1.7864	

Tabla 39. Tabla de resultados de la Prueba C

Modelo normal						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0	14.6803	50.00%	5.8348	
2	20	0	13.5725	50.00%	5.6538	
3	30	0	17.4637	50.00%	7.7672	
4	40	0	21.0229	25.00%	9.8181	Épocas: 7
5	50	0	21.5596	25.00%	9.2588	Val_accuracy: 0.0000e+00
6	60	0	19.6716	50.00%	9.2856	Val_loss: 14.7388
7	70	0	23.0272	25.00%	10.5349	Test Accuracy: 50.00%
8	80	0	26.9428	25.00%	12.0649	Test Loss: 7.1866
9	90	0	22.9075	25.00%	11.727	
10	100	0	24.5847	25.00%	12.817	
Fine Tunning (2 capas)						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0	31.5762	25%	16.8343	
2	20	0	33.2138	0%	16.8759	
3	30	0	35.0257	0%	18.2109	
4	40	0	33.7552	0%	17.1283	Épocas: 8
5	50	0	35.2398	0%	17.1478	Val_accuracy: 0.0000e+00
6	60	0	35.797	0%	17.0095	Val_loss: 28.9551
7	70	0	36.6129	0%	17.4383	Test Accuracy: 25.00%
8	80	0	37.1904	0%	18.2038	Test Loss: 17.0988
9	90	0	37.57	0%	17.4091	
10	100	0	37.4503	0%	17.5171	

Tabla 40. Tabla de resultados de la Prueba D

Modelo normal						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0.9565	1.6167	95.24%	0.8611	
2	20	0.9457	1.9434	96.83%	0.9493	
3	30	0.9457	2.2371	96.83%	1.3229	
4	40	0.913	1.8354	93.65%	0.901	Épocas: 6
5	50	0.9565	2.3151	96.83%	1.159	Val_accuracy: 0.9565
6	60	0.9565	2.4144	96.83%	1.2253	Val_loss: 1.2699
7	70	0.9674	2.7371	96.83%	1.3951	Test Accuracy: 96.83%
8	80	0.9674	2.725	95.24%	1.598	Test Loss: 0.6376.
9	90	0.9565	3.6506	95.24%	1.7331	
10	100	0.9565	3.587	96.83%	2.0339	
Fine Tunning (2 capas)						
Número del experimento	***Épocas***	***Val accuracy***	***Val loss***	***Test Accuracy***	***Test loss***	***Early stopping***
1	10	0.9565	2.0149	95.24%	1.2261	
2	20	0.9674	2.2084	95.24%	1.1403	
3	30	0.9674	2.5132	95.24%	1.452	
4	40	0.9674	2.9052	96.83%	1.658	Épocas: 6
5	50	0.9674	3.4084	96.83%	1.9354	Val_accuracy: 0.9565
6	60	0.9565	3.811	96.83%	2.0298	Val_loss: 1.8301
7	70	0.9565	4.0713	96.83%	2.15	Test Accuracy: 95.24%
8	80	0.9674	4.3281	96.83%	2.0407	Test Loss: 0.9872
9	90	0.9674	4.6742	96.83%	2.437	
10	100	0.9674	3.8608	95.24%	2.4236	

Assessment of Range of Motion Before and After Hamstring Percussion Therapy Using Thermography and CNN

Alejandra Vilchis-Yubi[1(✉)], Rogelio Cedeno-Moreno[2], Julio A. Espino-Gonzalez[3], Alberto Mancilla-Morales[3], Luis A. Morales-Hernandez[2], and Irving A. Cruz-Albarran[2]

[1] Engineering Faculty, Campus Aeropuerto, Autonomous University of Queretaro, Ejido Bolaños, 76140 Santiago de Querétaro, Queretaro, Mexico
avilchis16@alumnos.uaq.mx

[2] Laboratory of Artificial Vision and Thermography/Mechatronics, Faculty of Engineering, Autonomous University of Queretaro, Campus San Juan del Rio, 76807 San Juan del Río, Mexico

[3] Physical Therapy, Polytechnic University of Santa Rosa de Jauregui, Santa Rosa Jauregui, 76220 Santiago de Queretaro, Queretaro, Mexico

AQ1

Abstract. Percussion therapy has positive effects on the human body, such as increasing strength, improving range of motion and flexibility. This is due to the vibrations it produces, which when applied to the human body, cause vasodilatory responses. The main contribution of this work is a methodology based on infrared thermography and convolutional neural network to evaluate and classify the range of motion in patients undergoing hamstring percussion therapy. It consists of three steps: (1) data preparation, (2) data augmentation, and (3) convolutional neural network design and validation. In the data preparation step, 50 images acquired in the pre- and post-percussion therapy phases were pooled, and then each image was labeled in one of the two classes: Within and outside range of motion. Data augmentation, including techniques such as flipping and contrast adjustment, was used to expand the dataset. A convolutional neural network model was used to classify the images. When evaluated on a test set, an accuracy of 90.48% was obtained for the classification of 2 classes. These results demonstrate the efficacy and robustness of the approach to assess range of motion in patients undergoing hamstring percussion therapy.

Keywords: Convolutional neural network · infrared thermography · percussion therapy · thermal imaging

1 Introduction

Percussion therapy is defined as the repeated application of pressure perpendicular to the body that produces physiological effects [1]. It can be applied as hand-held percussion therapy, such as a percussion massage gun, which allows the device to "glide" over the

S. T. Kakileti et al. (Eds.): AIIIMA 2024, LNCS 15279, pp. 1–14, 2025.
https://doi.org/10.1007/978-3-031-76584-1_8

ANEXO D

Paso 1. Formateo microSD

Primero se necesita realizar el formateo de la microSD, para ello se siguieron los siguientes pasos:

1. En el (NVIDIA Corporation, n.d.), en la parte titulada *"Write Image to the microSD card"*, en la sección *"Instructions for Windows"*. Hacer clic en *"SD Memory Card Formatter for Windows"*.

   1.1. Aparecerá una nueva página, hacer clic en *"Accept"* y se procederá a descargar el archivo ZIP. Descargar e instalar el software.

2. Colocar la microSD en la computadora usando un adaptador, o si está disponible, insertarla directamente en la ranura correspondiente.

3. Iniciar el programa "SD Card Formatter".

4. Elegir la tarjeta (en este caso, la microSD) y optar por *"Quick Format"*, como se muestra en la Figura 1.

5. Dejar sin completar el campo *"Volume label"*. Proceder con el formateo.

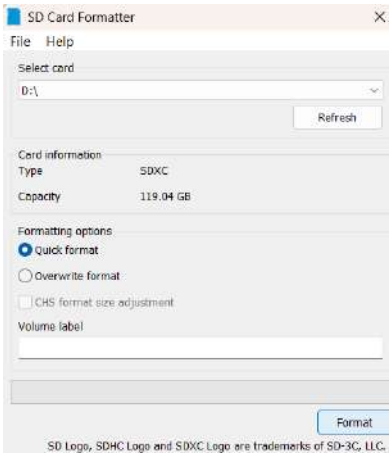

Figura 1. Ventana del software SD Card Formatter

Paso 2. Instalación de virtual machine (máquina virtual)

El procedimiento consiste en implementar el software de máquina virtual con el fin de instalar el programa "SDK Manager" en Ubuntu y cargar la imagen en la microSD. Es necesario contar con una computadora que tenga Linux y conexión a Internet para poder utilizar "SDK Manager" y cargar el kit de desarrollo. Las versiones de sistemas operativos que son compatibles son: Ubuntu Linux x64 Versiones 18.04 *o* 16.04 (JetsonHacks, 2023). Para llevar a cabo la instalación de la máquina virtual, se siguieron las instrucciones del (C S Psyco, 2021)

1. Descargar la imagen ISO de Ubuntu, yo elegí la versión 18. 04. 6 y se obtuvo del (Canonical Ltd., 2018)

2. Conseguir VMware Workstation e instalar (la versión que descargue fue la 17).

3. Iniciar VMware Workstation y seleccionar "*Create a new virtual machine*", como se observa en la Figura 2.

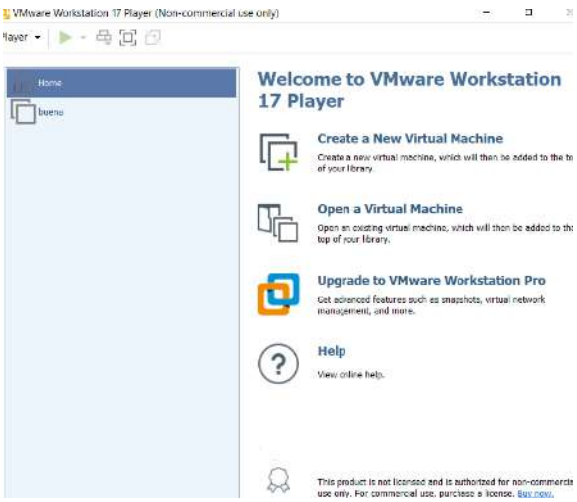

Figura 2. Ventana principal del software VMware Workstation

4. Hacer clic en "*Installer disc image file (iso)*". Localizar el archivo descargado de Ubuntu, seleccionarlo y pulsar en "*Next*".
5. Ingresar el nombre de usuario, un nombre y una contraseña. Hacer clic en "*Next*".

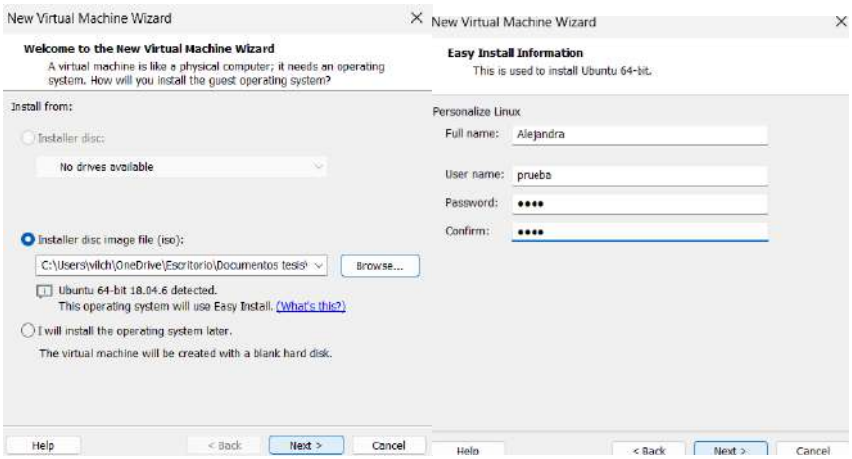

Figura 3. Selección de imagen ISO en el software VMware Workstation . Configuración de usuario en el software VMware Workstation

6. Si se quiere asignar un nombre a la máquina virtual, introducirlo. Luego, hacer clic en "*Next*".
7. En "*Máximum disk size GB*", ingresar la cantidad deseada, se recomienda más de 20 GB, ya que se requieren 17 GB para la descarga de los componentes. Se necesita un mínimo de 29 GB (host) más 17 GB (destino) de espacio libre en disco en el volumen del sistema para cada versión completa (host y destino) del SDK.
8. Y seleccionar "Store virtual disk as a single file" clic en "*Next*", al igual que en la Figura 4.

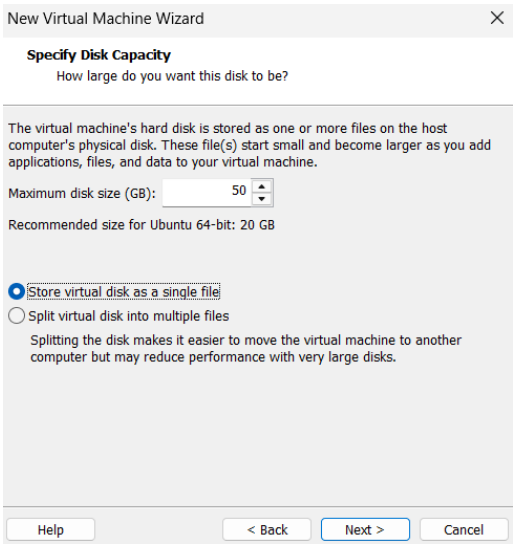

Figura 4. Configuración de disco virtual en el software VMware Workstation

9. Seleccionar "*Customize Hardware*" y fijar la memoria RAM en 16 GB. Dejar las demás configuraciones como están, cerrar y hacer clic en "*Finish*" como se muestra en las imágenes de la Figura 5.

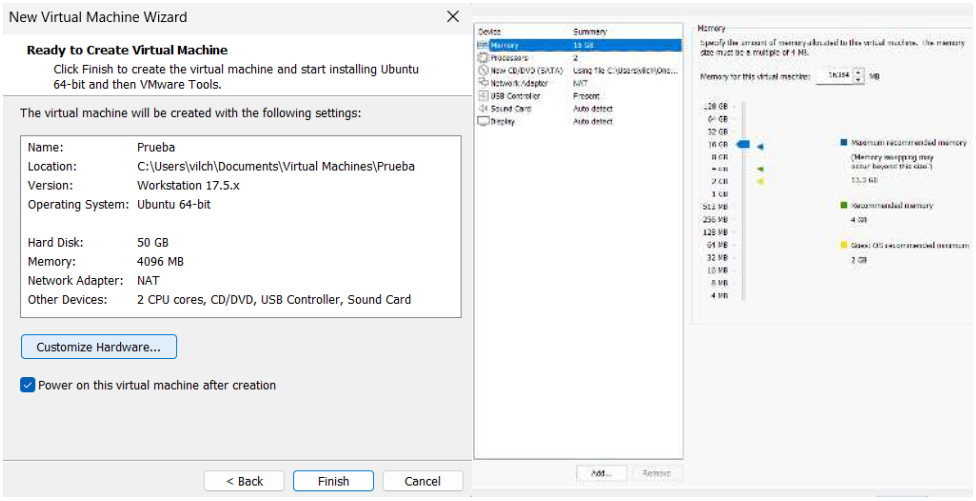

Figura 5. Configuración avanzada de disco virtual en el software VMware Workstation

Parte 3. Instalación de SDK Manager

Una vez que la máquina virtual ha sido configurada, se prosiguió con el proceso de instalar el SDK Manager para poder agregar la imagen en la tarjeta microSD.

1. Realizar la instalación del SDK Manager utilizando el (NVIDIA Developer, n.d.)

2. Hacer clic en .deb Ubuntu. Para obtener el archivo, es necesario tener una cuenta en NVIDIA, por lo que se debe registrar.

Método 1

Cuando se obtenga el SDK Manager, instalar a través de la aplicación “Software” de Ubuntu. Se abrirá automáticamente (Figura 6) y será necesario iniciar sesión con la cuenta.

Figura 6. Ventana principal de SDK Manager

Método 2

1. Abrir la terminal, y entrar a la carpeta donde se descargó el archivo .deb.

2. Introduce el comando necesario para resolver el inconveniente:

```
sudo dpkg --configure -a
```

3. A continuación, introduce el comando que permitirá la instalación del SDK Manager

```
sudo apt install ./sdkmanager_2.2.0-12028_amd64.deb
```

4. Se abrirá en automático y accede a tu cuenta.

Paso 4. Instalación de Imagen en microSD

Se continuó con la instalación de la imagen en la microSD, para ello, se siguieron los pasos del video del (JetsonHacks, 2023)

1. Colocar la microSD en Jetson Nano (Figura 7).

Figura 7. Conexión de microSD a Jetson Nano

2. Conectar los cables restantes, como el de Ethernet, el cable USB a Micro-B desde el Jetson Nano a la computadora, y el cable de alimentación, como se muestra en la Figura 8. Primero, el cable Ethernet, seguido del USB a Micro-B, y, por último, el de alimentación.

Figura 8. Conexión de la tarjeta Jetson Nano a la computadora

3. La sesión virtual identificará la placa detectará la placa y aparecerá un diálogo, como se muestra en la Figura 9, preguntando si se desea conectar al host o a la máquina virtual, selecciona la máquina virtual.

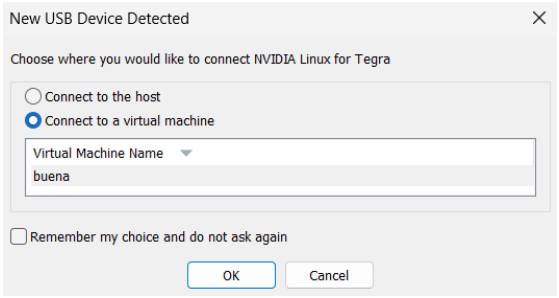

Figura 9. Ventana de selección de conexión para dispositivo USB.

4. SDK manager detectará la tarjeta. En la página inicial, o "*Step* 01", escoge Jetson, Host Machine. En "*Target Hardware*" la placa se identificará al conectar el Jetson Nano. También selecciona "*Deepstream*", como se muestra en la Figura 10.

Figura 10. Ventana de SDK Manager con la selección correspondiente para el flasheo de la tarjeta.

5. Para la versión del se utilizó la versión 4.6, ya que había inconvenientes con el SPI. Un usuario del foro de NVIDIA sugirió esta versión como una alternativa tras intentar múltiples soluciones para el funcionamiento adecuado del SPI. Cuando se selecciona la versión de JetPack en el SDK Manager, solo aparecerán las versiones más nuevas. Para ver versiones anteriores, se realizaron los siguientes pasos:

5.1. Cerrar SDK Manager. Abrir la terminal. Ingresar el siguiente comando:

```
sdkmanager –archived_versions
```

5.2. SDK Manager se abrirá automáticamente.

6. Seleccionar JetPack 4.6, como se muestra en la Figura 11.

Figura 11. Selección de la versión correcta (4.6) de JetPack.

7. Hacer clic en "*Continue*" . Continuará con el "*Step* 02" , hacer clic en "*I accept the terms and conditions of the license agreements*". Como se observa en la Figura 12.

Figura 12. Ventana del software SD Card Formatter

8. Hacer clic en "*Continue*". En el "*Step* 03" comenzará la descarga de los componentes, tanto en el host machine como en la tarjeta. Al concluir, se procederá con la instalación (Figura 13).

Figura 13. Descarga e instalación de componentes.

9. Cuando se va a realizar el flasheo de la Jetson Nano, aparecerá una ventana que ofrecerá dos opciones: "*Automatic Setup*" y "*Manual Setup*". Se siguió el procedimiento del método manual, el cual se utiliza cuando la microSD está vacía.

9.1. Seleccionar la versión de la tarjeta: "*Developer kit versión*".

9.2. Colocar la tarjeta en "*Recovery Mode*".

9.2.1. Desconectar Jetson Nano solo de la alimentación. Conectar con un cable el pin 9 al pin 10 como se muestra en la Figura 14. Estos pines se encuentran en el "*Button Header* [J50]" se puede observar en la Figura 15, donde se muestran las partes de la tarjeta Jetson Nano.

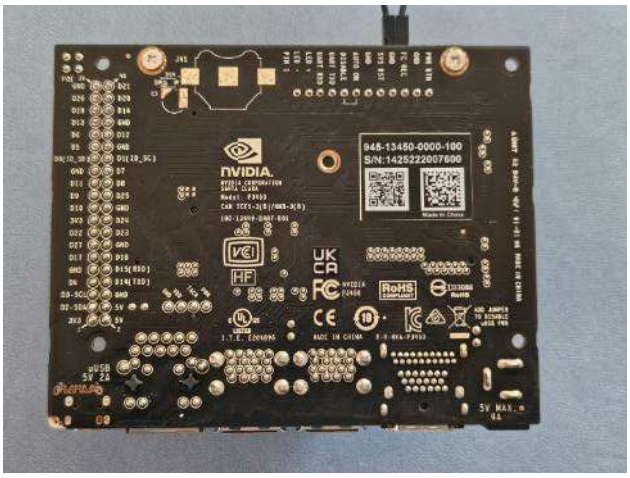

Figura 14. Conexión de pin 9 y pin 10 para entrar al Recovery Mode

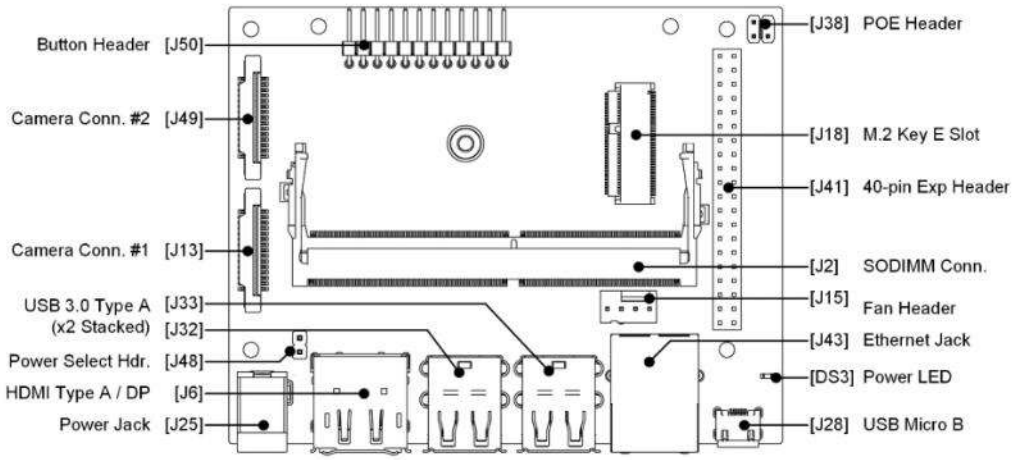

Figura 15. Partes de la tarjeta Jetson Nano

9.2.2. Volver a conectar Jetson Nano a la fuente de alimentación Nota: Este proceso se debe realizar sin cerrar, ni pausar el proceso en el SDK Manager

10. La ventana cambiara nuevamente a "*Automatic Setup*", seleccionar de nuevo: "*Manual Setup*". Proceder al flasheo.

11. Una vez que termine el flasheo, aparecerá una ventana solicitando un nombre de usuario y una contraseña para instalar los demás componentes. Sin embargo, primero se debe configurar la Jetson Nano. Para lograrlo, se llevaron a cabo los siguientes pasos:

11.1. Conectar un display HDMI, teclado y mouse a la tarjeta Jetson Nano, como se muestra en la Figura 16.

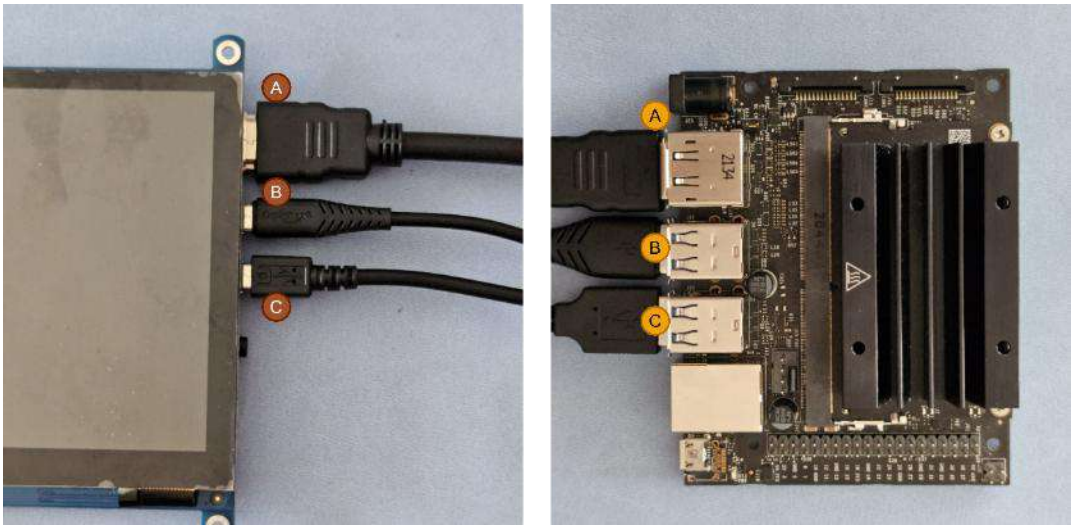

Figura 16. Conexión de display HDMI, teclado y pantalla a la tarjeta Jetson Nano.

11.2. Una vez conectado, en el display HDMI se mostrará una ventana para iniciar la configuración.

11.3. Seguir el procedimiento donde se pedirá ingresar un nombre de usuario, contraseña, idioma, zona horaria, un ejemplo de ello se muestra en la Figura 17.

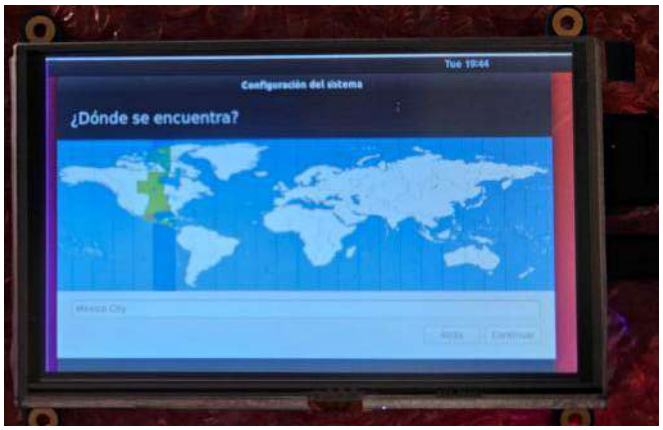

Figura 17. Configuración de la tarjeta Jetson Nano (Zona horaria).

11.4. En la ventana de "*App Partition Size*", dejar todo como esta y hacer clic en "*Continue*".

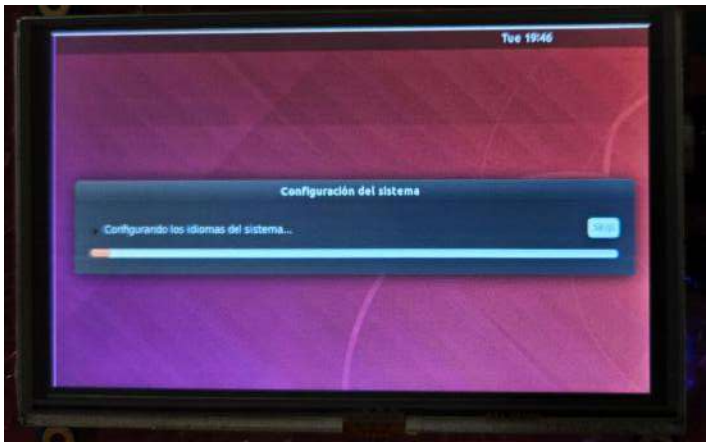

Figura 18. Configuración del sistema Jetson Nano.

12. Una vez terminada la configuración, se continua el proceso en el SDK Manager ingresando el usuario y la contraseña que se introdujeron previamente en la tarjeta Jetson Nano.

13. En "*Connection*" seleccionar: USB Y Ethernet (saber el IP Address)

14. Hacer clic en "*Install*". Esperar a que finalice el proceso. Una vez terminado (Figura 19), ya se podrá utilizar la tarjeta Jetson Nano, donde el display HDMI funcionará como la interfaz visual.

Figura 19. Finalización del flasheo de la tarjeta Jetson Nano.

Paso 5. Modo Headless

Debido a comodidad y por el tamaño del display, se optó por operar la tarjeta Jetson Nano en el Modo Headless. Este modo permite trabajar sin necesidad de que un monitor o pantalla esté enchufada a la unidad. En su lugar, se establece una conexión a través de una sesión remota en una computadora anfitriona, utilizando xrdp.

XRDP es un software que actúa como un servidor de escritorio remoto para Linux, facilitando que los usuarios se conecten a una máquina Linux desde otro dispositivo mediante el protocolo RDP (Protocolo de Escritorio Remoto), el cual es el mismo protocolo utilizado por Windows para sus conexiones remotas.

Para lograr esto, se llevaron a cabo los siguientes pasos:

1. Mientras la Jetson Nano continúa con el display HDMI. Ingresar a la terminal, y escribir "*ifconfig*" para obtener la IP en caso de que no se conozca. Se encuentra en la sección "*eth0*", en "*inet*".

2. Luego, ingresar los siguiente comandos:

```
sudo apt update
sudo apt install xrdp
sudo apt systemctl enable xrdp.
sudo reboot
```

Una vez completados estos pasos, se habrá finalizado la instalación de las librerías necesarias para la conexión remota. Los pasos siguientes se refieren a cómo conectarse de manera remota desde la computadora anfitriona.

1. Primer, se debe desconectar el display, ya que la tarjeta Jetson Nano no permite conexiones simultáneas a través de ambos métodos: pantalla física o conexión remota.

2. En la computadora host, buscar "*Conexión a Escritorio Remoto*". Se abrirá una pequeña ventana,

en el campo "*Equipo*" ingresar la dirección IP de la tarjeta Jetson Nano y hacer clic en "*Conectar*", como se muestre en la Figura 20.

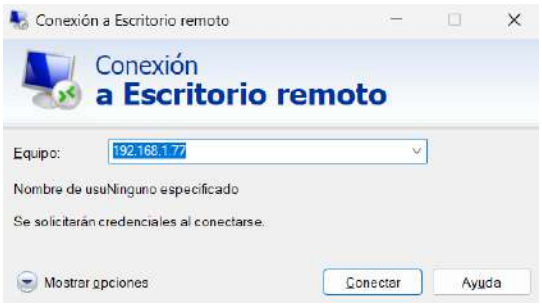

Figura 20. Ventana del software "Conexión a Escritorio Remoto".

3. Se mostrará una ventana de seguridad como la de la Figura 21, hacer clic en "*Si*".

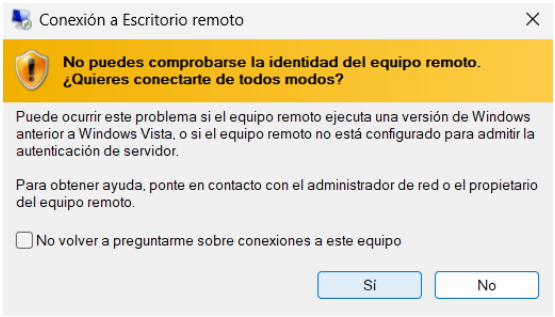

Figura 21. Ventana de seguridad del software "Conexión a Escritorio Remoto".

4. La aplicación se abrirá mostrando una ventana, Figura 22, la cual pedirá datos para iniciar la conexión. En "*Session*", seleccionar "*Xorg*". Luego, ingresar el nombre de usuario y contraseña, que se configuraron en la tarjeta Jetson Nano. Esto permitirá acceder a través de una sesión remota.

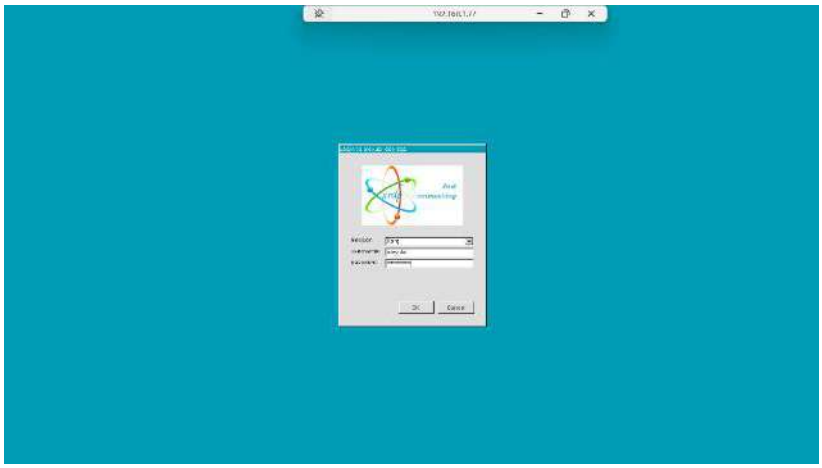

Figura 22. Conexión por medio del protocolo xrdp.

Paso 6. Instalar primeras librerías

Se instalaron 4 librerías: Python; Jetson Stats para obtener información de la tarjeta por medio de

jtop; Nano que es una librería para editar archivos; y Cmake. Se ingresaron los siguiente comandos:

```
sudo apt update
sudo apt upgrade
sudo apt install python3-pip
sudo pip3 install -U jetson-stats.
sudo apt install nano
sudo apt install -y cmake
```

Paso 7. Verificar SPI

Antes de continuar con la instalación de otras librerías para utilizar la cámara Raspberry Pi y la cámara Lepton es fundamental asegurarse de que el protocolo SPI esté operando de manera adecuada. Para lograr esto, se consultaron varias guías en foros de NVIDIA que ofrecen pruebas conocidas como "*SPI test loopback*".

Además, se mantuvo contacto con un usuario regular de los foros de NVIDIA, el cual nos guío para solucionar el problema de SPI que se, y nos brindó un comando que muestra la configuración de los GPIO, nos comentó que la clave está en el GPIO C, donde su configuración (CNF) debe ser igual a 00. Para lograr esto, se llevaron a cabo las siguientes acciones:

1. En la terminal, escribir el siguiente comando:

```
sudo cat /sys/kernel/debug/tegra_gpio
```

2. El resultado debería coincidir con la Figura 23 en el GPIO C:

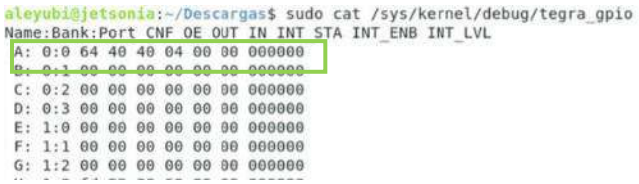

Figura 23. Configuración correcta del GPIO C.

Al verificar que GPIO C se encuentra debidamente configurado, se continuó con los "*SPI test loopback*".

1. En la terminal, escribir el siguiente comando:

```
sudo /opt/nvidia/jetson-io/jetson-io.py
```

2. Aparecerá una ventana llamada "*Jetson Expansion Header Tool*" para la configuración de GPIO, hacer clic en "*Configure Jetson* 40 *pin Header*".

Figura 24. Ventana llamada "Jetson Expansion Header Tool".

3. Luego seleccionar "*Configure header pins manually*", como se muestra en la Figura 25.

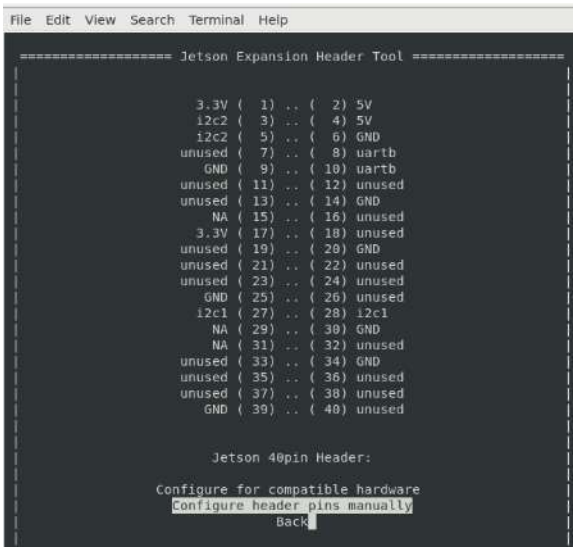

Figura 25. Selección para configurar los pins manualmente.

4. Se mostrarán los GPIO se desean configurar; en este caso, se seleccionará spi1, Figura 26. Hacer clic en "*Back*".

Figura 26. Selección del SPI 1.

5. Para guardar los cambios, hacer clic en "*Save pin changes*" y luego en "*Save and reboot to reconfigure pins*", Figura 27.

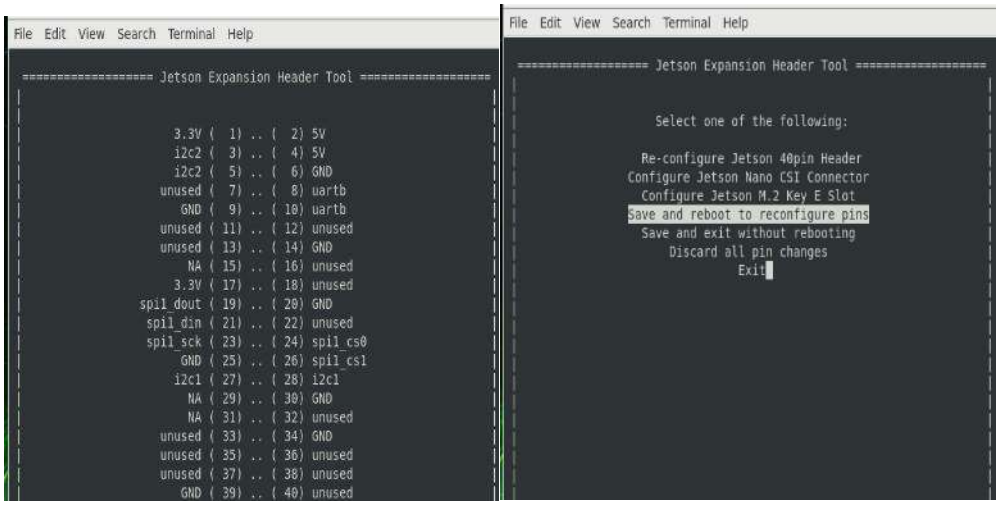

Figura 27. Guardar nueva configuración de los pines.

6. Presionar cualquier tecla. El resultado correcto se muestra en la Figura 28:

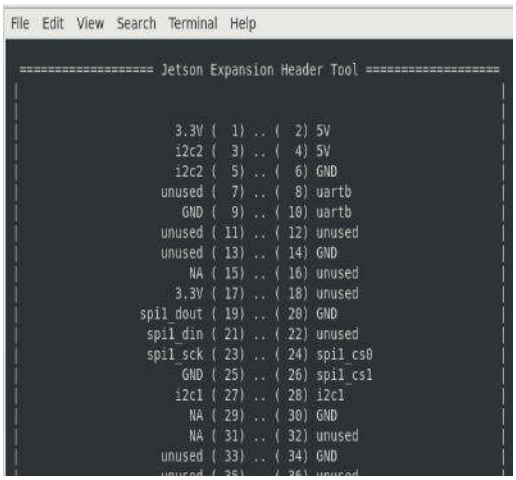

Figura 28. Resultado final de la configuración del SPI 1.

SPI Test Loopback 1

En el (jcl9309 & KevinFFF, 2023)están los pasos para obtener el spidev test. Es un comentario de un foro de NVIDIA y muestra los pasos para modificar el device tree, esta es una solución cuando no está bien configurado los GPIO y fue una de las que se siguieron antes de encontrar solución. Sin embargo, para fines del test, solo se toman en cuenta los pasos a partir del "Step 4", los cuales son los siguientes:

1. En la terminal, escribir el siguiente comando, que sirve para cargar el kernel spidev:

```
sudo modeprobe spidev
```

2. Después descargar y construir el test file con los siguientes comandos:

```
wget https://raw.githubusercontent.com/torvalds/linux/v4.9/tools/spi/spidev_test.c
gcc -o spidev_test spidev_test.c
```

3. Conectar el pin 19 al pin 21 del "40 – *pinExp Header* [*J*41]" según la Figura 29, como si se hiciera cortocircuito, los cuales son los pines de MOSI y MISO, respectivamente, La conexión se muestra en la Figura 29.

Figura 29. Conexión del pin 19 a pin 21, haciendo cortocircuito.

4. En la terminar, correr los siguientes comandos para verificar el funcionamiento de SPI:

```
sudo ./spidev_test -D /dev/spidev0.0 -v -p "HelloWorld123456789abcdef"
sudo ./spidev_test -D /dev/spidev0.0 -s 10000000 -v
```

5. El resultado correcto se muestra en la Figura 30:

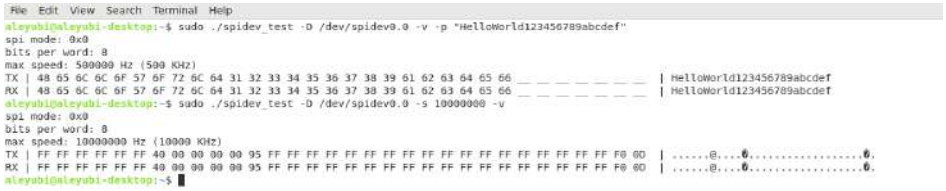

Figura 30. Resultado correcto del SPI Test.

SPI Test Loopback 2

En el (jvachteren & SchaneCCC, 2021) se encuentra el archivo .txt, al igual que el link pasado es un comentario de un foro. Los pasos que se siguieron son los siguientes:

1. Descargar archivo spider_test.txt, se encuentra en el comentario que se muestra en la Figura 31.

Figura 31. Comentario que contiene el archivo .txt a descargar.

2. Abril la terminal, y entrar a la carpeta donde se encuentra la descarga. Ingresar los siguientes comando:

```
sudo mv spidev_test.txt spidev_test
sudo chmod +x spidev_test
```

3. Escribir el siguiente comando:

```
sudo modeprobe spidev
```

4. En la terminar, correr el siguiente comando para verificar el funcionamiento de SPI:

```
sudo ./spidev_test -D /dev/spidev0.0 -g16 -zz
```

5. El resultado deberá ser como en la Figura 32, el cual muestra que se pasó el test:

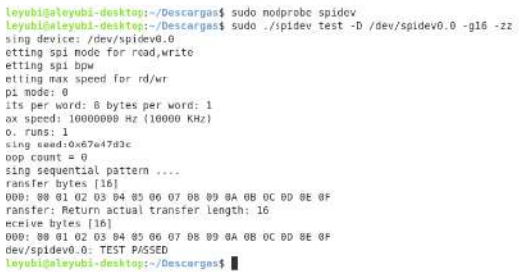

Figura 32. Test SPI aprobado.

Paso 8. Instalar OpenCV con CUDA support

Se instaló OpenCV con CUDA support para mejorar el procesamiento de imágenes o la visión artificial, permitiendo que el ordenador utilice unidades de procesamiento gráfico (GPU) para un procesamiento general más rápido.

Para esto, es necesario encontrar una versión de OpenCV que funcione con CUDA. La versión de CUDA instalada con Jetpack 4. 6 es la 10. 2. 300, así que la versión de OpenCV que es compatible con ella es la 4. 5. 0. Por lo tanto, se siguieron las instrucciones del (Code with Aarohi, 2023):

1. Abrir terminal. Escribir el siguiente comando. Aparecerá una ventana con información acerca de la tarjeta Jetson.

```
jtop
```

Presionar 7 o dar clic en "*INFO*". Y en el apartado "*Libraries*", se muestra OpenCV y su versión. También se indicará si está instalado con soporte de CUDA; si no es así, dirá "NO" en rojo. Si está instalado, dirá "*YES*" en verde. Se mostrará como en la Figura 33.

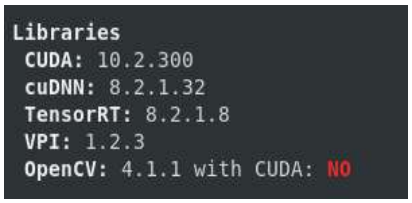

Figura 33. Estado de la configuración del OpenCV con CUDA.

2. En el video, los primeros pasos en son crear un entorno virtual para realizar la instalación ahí. No obstante, se omitió ese paso y se procedió a instalar todo directamente.

3. Primero, es esencial crear un "*Swapfile*" para prevenir errores de "*Out of memory*", ya que esta

función gestiona archivos cuando no hay espacio disponible. Los pasos que se llevaron a cabo son:

3.1. En la terminal, ingresar el siguiente comando:

```
free -h
```

Y se mostrará información acerca del espacio de memoria del swap de aproximadamente de 1.9 Gb.

3.2. Ingresar los siguientes comandos:

```
sudo fallocate -l 4G /var/swapfile
sudo chmod 600 /var/swapfile
sudo mkswap /var/swapfile
sudo swapon /var/swapfile
sudo bash -c 'echo "/var/swapfile swap swap defaults 0 0" >> /etc/fstab'
```

3.3. Reiniciar la tarjeta Jetson Nano usando:

```
sudo reboot
```

3.4. Para verificar que se haya liberado el espacio en swap, ingresar de nuevo este comando; ahora el espacio debería ser de 5 Gb como se muestra en la Figura 34.

```
free -h
```

```
              total        used        free      shared  buff/cache   available
Mem:           3.9G        1.3G        1.3G         48M        1.3G        2.4G
Swap:          5.9G          0B        5.9G
```

Figura 34. Espacio liberado en swap.

4. El siguiente paso que se realizó fue la instalación de las bibliotecas y dependencias requeridas para OpenCV. Es posible que algunas ya estén presentes, aun así, se aconseja introducir todos los siguientes comandos:

```
sudo sh -c "echo '/usr/local/cuda/lib64' >> /etc/ld.so.conf.d/nvidia-tegra.conf"
sudo ldconfig
sudo apt-get install build-essential cmake git unzip pkg-config
sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install libgtk2.0-dev libcanberra-gtk*
sudo apt-get install python3-dev python3-numpy python3-pip
sudo apt-get install libxvidcore-dev libx264-dev libgtk-3-dev
sudo apt-get install libtbb2 libtbb-dev libdc1394-22-dev
sudo apt-get install libv4l-dev v4l-utils
sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install libavresample-dev libvorbis-dev libxine2-dev
sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev
sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev
sudo apt-get install libopenblas-dev libatlas-base-dev libblas-dev
sudo apt-get install liblapack-dev libeigen3-dev gfortran
sudo apt-get install libhdf5-dev protobuf-compiler
sudo apt-get install libprotobuf-dev libgoogle-glog-dev libgflags-dev
```

5. Se continuó con la instalación de OpenCV utilizando repositorios de GitHub:

```
cd ~
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.5.1.zip
wget -O opencv_contrib.zip
https://github.com/opencv/opencv_contrib/archive/4.5.1.zip
```

6. Descomprimir ambos archivos:

```
unzip opencv.zip
unzip opencv_contrib.zip
```

7. Cambiar el nombre de las carpetas y se eliminan los archivos .zip:

```
mv opencv-4.5.1 opencv
mv opencv_contrib-4.5.1 opencv_contrib
rm opencv.zip
rm opencv_contrib.zi
```

8. Compilar OpenCV, para esto se necesita acceder a la carpeta de OpenCV:

```
cd ~/opencv
```

9. Crear un nuevo directorio:

```
mkdir build
```

10. Entrar en el nuevo directorio:

```
cd build
```

11. Ingresar el comando a continuación:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr -D
OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules -D
EIGEN_INCLUDE_PATH=/usr/include/eigen3 -D WITH_OPENCL=OFF -D WITH_CUDA=ON -D
CUDA_ARCH_BIN=5.3 -D CUDA_ARCH_PTX="" -D WITH_CUDNN=ON -D WITH_CUBLAS=ON -D
ENABLE_FAST_MATH=ON -D CUDA_FAST_MATH=ON -D OPENCV_DNN_CUDA=ON -D ENABLE_NEON=ON -D
WITH_QT=OFF -D WITH_OPENMP=ON -D WITH_OPENGL=ON -D BUILD_TIFF=ON -D WITH_FFMPEG=ON -
D WITH_GSTREAMER=ON -D WITH_TBB=ON -D BUILD_TBB=ON -D BUILD_TESTS=OFF -D
WITH_EIGEN=ON -D WITH_V4L=ON -D WITH_LIBV4L=ON -D OPENCV_ENABLE_NONFREE=ON -D
INSTALL_C_EXAMPLES=OFF -D INSTALL_PYTHON_EXAMPLES=OFF -D BUILD_NEW_PYTHON_SUPPORT=ON
-D BUILD_opencv_python3=TRUE -D OPENCV_GENERATE_PKGCONFIG=ON -D
BUILD_EXAMPLES=OFF ..
```

12. Para finalizar la instalación de OpenCV con CUDA, se debe ingresar el siguiente comando; el proceso tomará alrededor de 2 horas en completarse.

```
make -j4
```

13. Finalmente, eliminar todos los archivos sobrantes:

```
cd ~
sudo rm -r /usr/include/opencv4/opencv2
cd ~/opencv/build
sudo make install
sudo ldconfig
```

```
make clean
sudo apt-get update
```

14. Para confirmar que OpenCV se instaló con soporte para CUDA:

14.1. Ingresar en la terminal y se mostrar una ventana con información sobre la tarjeta Jetson Nano, como se muestra en la Figura 35:

```
jtop
```

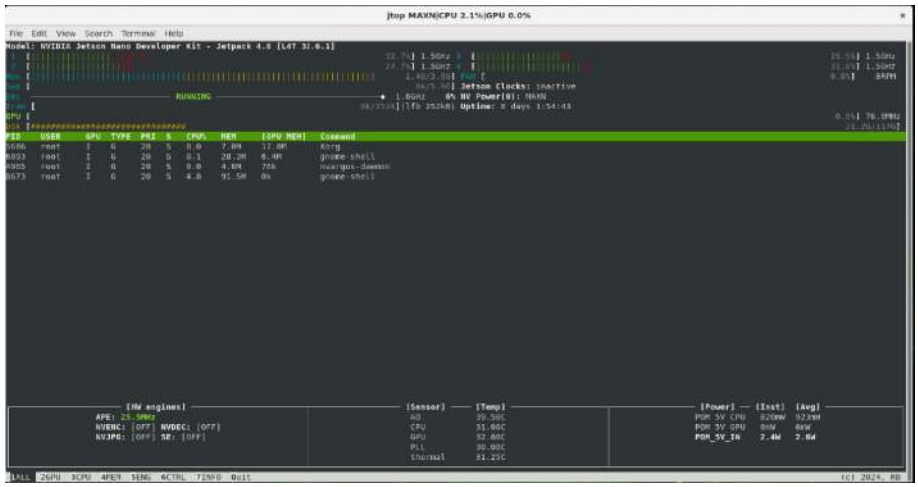

Figura 35. Ventana principal de JTOP.

14.2. Presionar 7. Verificar que en la sección "*OpenCV with Cuda*" esté resaltado en verde: "*YES*", como se muestra en la Figura 42.

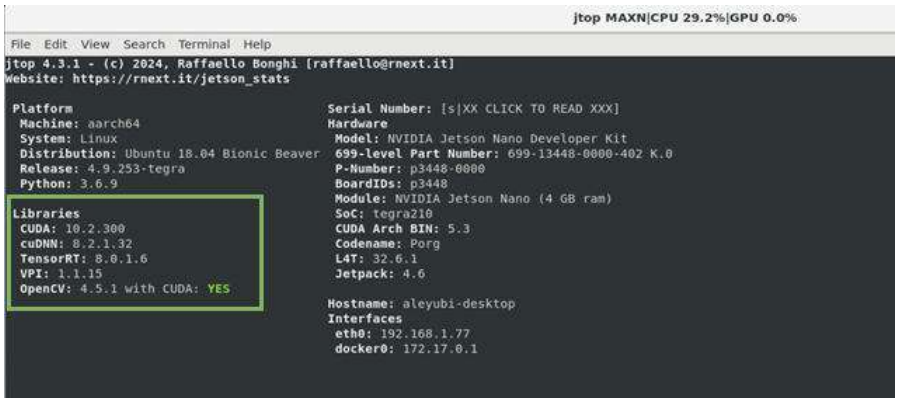

Figura 36. Configuración correcta de OpenCV con CUDA:

15. Debido a que no se ha realizado la instalación en un ambiente virtual, puede haber conflictos con versiones previas de OpenCV que se hayan instalado. Para solucionar esto, escribe lo siguiente en la terminal:

```
python3
import cv2
print(cv2.__version__)
print(cv2.getBuildInformation())
```

Al ejecutar "$print(cv2.\_\_version\_\_)$", debería mostrar la versión de OpenCV en uso, la cual debe ser $4.5.1$, como se observa en la Figura 37. Al utilizar "$print(cv2.getBuildInformation())$", aparecerá información detallada de la versión de OpenCV usada, incluyendo un apartado que indica su compatibilidad con Cuda. Si aparece una versión distinta, se debe eliminar esa versión.

```
aleyubi@aleyubi-desktop:~$ python3
Python 3.6.9 (default, Mar 10 2023, 16:46:00)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.5.1
>>> print(cv2.getBuildInformation())

General configuration for OpenCV 4.5.1 =====================================
  Version control:               unknown

  Extra modules:
    Location (extra):            /home/aleyubi/opencv_contrib/modules
    Version control (extra):     unknown
```

Figura 37. Ventana del software SD Card Formatter

Paso 9. Instalar Visual Studio Code

Es un editor de código que utilizado para modificar archivos de Python necesarios para que la cámara digital Raspberry funcione, así como para la cámara termográfica Lepton 3.5 y para la interfaz gráfica. A diferencia de ciertas bibliotecas enfocadas en la edición de texto como nano, este editor simplifica la modificación de estos archivos. Para ello, se siguieron los pasos del (JetsonHacks, 2024).

1. Al descargar VSCode, es fundamental considerar los requerimientos de la tarjeta Jetson. Visitar el (Visual Studio Code, n.d.), donde se explican las pasos para descargar versiones anteriores. En este caso, se necesita una versión aún más antigua de las que aparecen en esta página. Para obtener la versión correcta, se debe descender hasta la parte inferior del sitio en la sección "$Previous\ release\ versions$". En la parte inferior de este apartado hay una sección llamada "$Linux\ Arm64\ Debian$", donde se encuentra el enlace correcto, como lo muestra la Figura 38. Este enlace se eligió porque la Jetson emplea una arquitectura Arm64 y funciona con Ubuntu, que se basa en Debian.

Linux Arm64	https://update.code.visualstudio.com/{version}/linux-arm64/stable
Linux Arm64 debian	https://update.code.visualstudio.com/{version}/linux-deb-arm64/stable
Linux Arm64 rpm	https://update.code.visualstudio.com/{version}/linux-rpm-arm64/stable
Linux Arm64 CLI	https://update.code.visualstudio.com/{version}/cli-linux-arm64/stable

Figura 38. Link correcto para la descarga de VSCode

2. Copiar el link, y reemplazar "$\{version\}$" en el enlace con la versión deseada. Se probaron varias versiones, y la que funcionó correctamente fue la $1.59.1$. La dirección final se verá así: $https://update.code.visualstudio.com/1.59.1/linux-deb-arm64/stable$.

3. Descargar el archivo, abrir la terminal y ejecutar el siguiente comando:

```
cd ~/Descargas
```

4. Ingresar el siguiente comando para instalar:

```
sudo dpkg -i code_1.59.1-1629374148_arm64.deb
```

5. Para iniciar VSCode, escribir en la terminal:

```
code
```

6. Instalar Python en VSCode y verificar que la versión utilizada sea Python 3.6.9.

Paso 10. Conectar y verificar funcionamiento de la cámara Raspberry

La tarjeta Jetson Nano tiene 2 puertos CSI que están diseñados para conectar cámaras compatibles con la interfaz CS. Los puertos son compatibles con módulos de cámara IMX219, incluido el módulo de cámara Leopard Imaging LI-IMX219-MIPIFF-NANO y el módulo de cámara Raspberry Pi V2 (JetsonHacks, 2023).

1. Para activar los puertos CSI se necesita configurarlo, por lo que se siguieron los siguientes pasos:

```
sudo /opt/nvidia/jetson-io/jetson-io.py
```

1.1. Seleccionar "*Configure Jetson Nano CSI Connector*", como se muestra en la Figura 39.

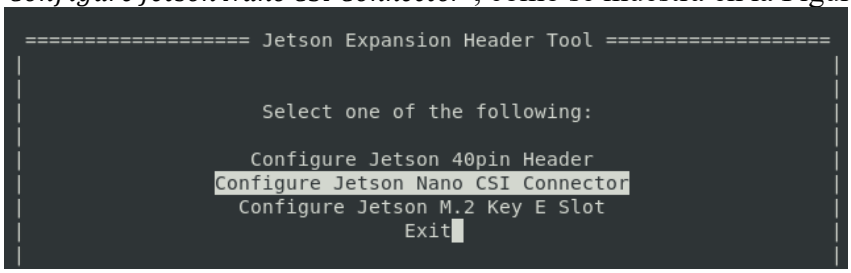

Figura 39. Ventana de configuración de "Jetson Expansión Header Tool".

1.2. Después, elegir "*Configure for compatible hardware*" (Figura 40).

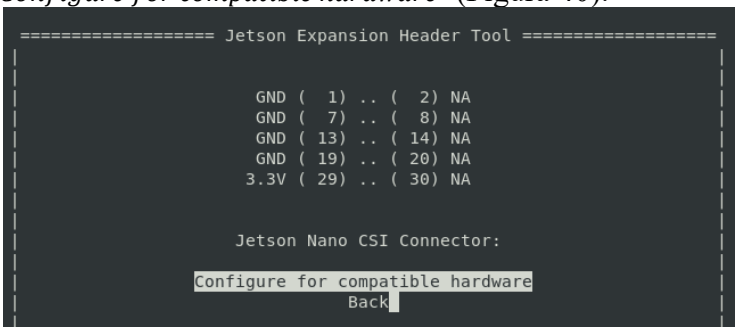

Figura 40. Selección de configuración a un hardware compatible.

1.3. Seleccionar "*Camera IMX219 Dual*", como en la Figura 41.

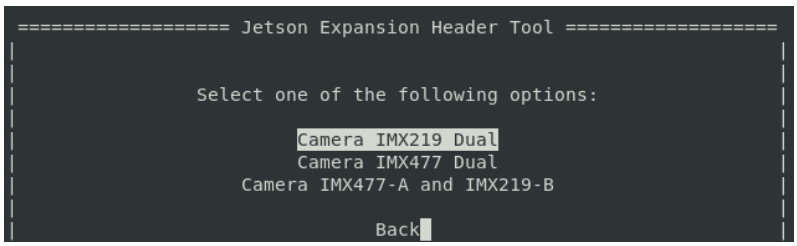

Figura 41. Selección de configuración de la Camera IMX219.

1.4. Hacer clic en "*Save pin changes*". Por último, seleccionar "*Save and reboot to reconfigure*".

2. Conectar la cámara a cualquiera de los dos puertos CSI. Este paso debe realizarse cuando la tarjeta Jetson está apagada; si se intenta mientras esta encendida, la tarjeta no reconocerá la cámara.

3. Para comprobar que la cámara está siendo detectada, abrir la terminal y escribir:

```
ls /dev/video*
```

Este comando muestra los dispositivos de video disponibles en el sistema; si está conectado correctamente a la placa, aparecerá como se muestra en la Figura 42:

*Figura 42. Salida del comando ls /dev/video**

Para probar la funcionalidad de la cámara Raspberry Pi V2, se usaron repositorios sencillos de GitHub y se descargaron librerías como GStreamer, OpenCV y Video4Linux2 (V4L2). Sin embargo, existen dos maneras de llevar a cabo esto, en modo Headless (sin monitor) y en modo Head (con monitor). Los pasos para cada método son los siguientes:

Método modo Headless (sin monitor)

Se utilizó un repositorio de GitHub (jetsonhacks, 2019) junto con su propia página web (kangalow, 2019). Se usó la librería de OpenCV, ya que para ejecutar GStreamer directamente es necesario un monitor o pantalla conectada a la tarjeta. A pesar de esto, es posible llevar a cabo el proceso en modo Head sin problemas, aunque el método Head no se puede hacer de forma Headless. Los pasos que se llevaron a cabo fueron los siguientes:

1. Abrir la terminal e ingresar el siguiente comando que nos devolverá al directorio inicial del usuario:

```
cd ~
```

2. Descargar el repositorio:

```
git clone https://github.com/JetsonHacksNano/CSI-Camera.git
```

3. Navegar a la carpeta que contiene el repositorio:

```
cd CSI-Camera
```

4. Para probar el repositorio, se puede utilizar con el archivo "*simple_camera.py*" el cual solo abre una ventana donde se reproduce el video. Además, se puede usar el archivo "*face_detect.py*"

que, al igual, abrirá una ventana, pero con la capacidad de detectar cualquier rostro que se muestre en el video. Para ejecutar estos archivos se utilizan estos comandos en la terminal:

```
python3 simple_camera.py
python3 face_detect.py
```

5. Para editar estos archivos, se puede utilizar VSCode, lo que permitirá experimentar con varios parámetros como cambiar la resolución, la orientación, etc. O bien, a partir de este código, desarrollar códigos nuevos.

Método modo Head (con monitor)

Para utilizar este método, se requerirá un monitor o pantalla HDMI, sí como un teclado y un mouse. El procedimiento que se realizó fue el siguiente:

1. Cerrar la sesión remota y desconectar la tarjeta Jetson Nano. Esto es imprescindible, ya que no se permite que la tarjeta esté conectada al mismo tiempo tanto a la sesión remota como a la pantalla HDMI.

2. Conectar a los componentes necesarios a la tarjeta.

3. Ingresar los comandos a continuación para comprobar su funcionamiento:

```
nvgstcapture-1.0 --orientation=2

st-launch-1.0 nvarguscamerasrc ! nvoverlaysink

gst-launch-1.0 nvarguscamerasrc sensor_mode=0 ! 'video/x-
raw(memory:NVMM),width=3820, height=2464, framerate=21/1, format=NV12' ! nvvidconv
flip-method=2 ! 'video/x-raw,width=960, height=616' ! nvvidconv ! nvegltransform !
nveglglessink -e
```

Paso 11. Conectar y verificar funcionamiento de la cámara infrarroja Lepton 3.5

A continuación, se describen el procedimiento realizado para la instalación de la librería necesaria, así como conectar la cámara al Breakout Board a la tarjeta Jetson Nano:

1. Antes de proceder a la descarga la librería, se necesita modificar el tamaño del buffer, que actualmente está configurado a 4096 bytes para la comunicación SPI en spidev. Para capturar un conjunto completo de datos que forma la imagen térmica, se requieren 20 KB de búfer. Se utilizó como referencia el (Myzhar, 2020). Existen dos métodos disponibles: uno temporal y otro permanente, se siguieron los pasos del método permanente:

1.1. Abrir la terminal e ingresar el siguiente comando, que permitirá abrir un archivo que se debe modificar para ajustar el tamaño del búfer:

```
sudo gedit /etc/modprobe.d/spidev.conf
```

1.2. Escribir la siguiente línea que define el tamaño del buffer:

```
options spidev bufsiz=20480
```

1.3. Guardar los cambios y salir del archivo. Reiniciar la tarjeta con:

```
sudo reboot
```

1.4. Para confirmar si se ha cambiado el tamaño del búfer, ingresar los siguientes comandos, el resultado debería ser como la Figura 43:

```
sudo modprobe spidev
cat /sys/module/spidev/parameters/bufsiz
```

```
aleyubi@aleyubi-desktop:~$ sudo modprobe spidev
aleyubi@aleyubi-desktop:~$ cat /sys/module/spidev/parameters/bufsiz
20480
aleyubi@aleyubi-desktop:~$
```

Figura 43. Resultado del tamaño del buffer size.

2. La librería que descargará se llama pylepton. Esta librería se encuentra disponible en repositorios de GitHub; sin embargo, hay algunos que solo cuentan con la librería para la versión de la cámara infrarroja Lepton 2.5. Por lo que, el GitHub que contiene una librería tanto para la versión Lepton 2.5 como para la versión Lepton 3.5, puede encontrarse en el (Michael Vernon, 2023). ara llevar a cabo la descarga de la biblioteca, se utiliza el siguiente comando:

```
git clone https://github.com/mishave/pylepton.git
```

3. Continua la conexión de la cámara al Breakout Board y de ahí a la tarjeta Jetson Nano. Los pasos que se siguieron fueron los siguientes:

3.1. Apagar la tarjeta. Conectar cámara Lepton 3.5 al Breakout Board V2, como se muestra en la Figura 44, del lado izquierdo se muestra la cámara Lepton 3.5 y la placa Lepton Camera Breakout Board v2.0:

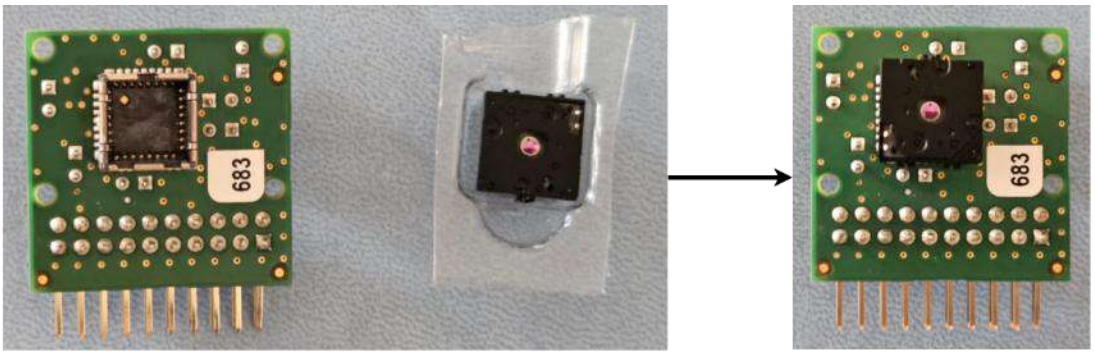

Figura 44. Conexión de la cámara Lepton 3.5 a la placa Breakout Board v2.0.

3.2. Conectar Breakout Board V2 a la tarjeta Jetson Nano. En la Tabla 1 se muestra el número y nombre de los pines de la cámara y de la placa. Del mismo modo en la Figura 45 se muestra de forma visual la conexión de los pines entre ambos componentes cada uno representado por color en específico. La parte de la tarjeta Jetson Nano que se usa para conectar se conoce como "$40 - pin\ Exp\ Header\ [J41]$":

Tabla 1. Conexión de la placa Breakout Board v2.0 a la tarjeta Jetson Nano.

Lepton3 PIN	Lepton3 nombre	Color	Jetson Nano	Jetson Nano PIN

			nombre	
1	GND	Gris	GND	6
2	IN	Rojo	3.3 V	1
5	SDA	Azul	I2C0_SDA	27
7	SPI_CLK	Verde	SPI0_SCK	23
8	SCL	Naranja	I2C0_SCL	28
9	SPI_MOSI	Amarillo	SPI0_MOSI	19
10	SPI_CS	Morado	SPI0_CS0 / SPI0_CS1	24/26
12	SPI_MISO	rosa	SPI0_MISO	21

En la Figura 46 se muestra la conexión real entre la placa Breakout Board v2.0 y la tarjeta Jetson Nano. La conexión se realizó por medio de cables dupont hembra -hembra.

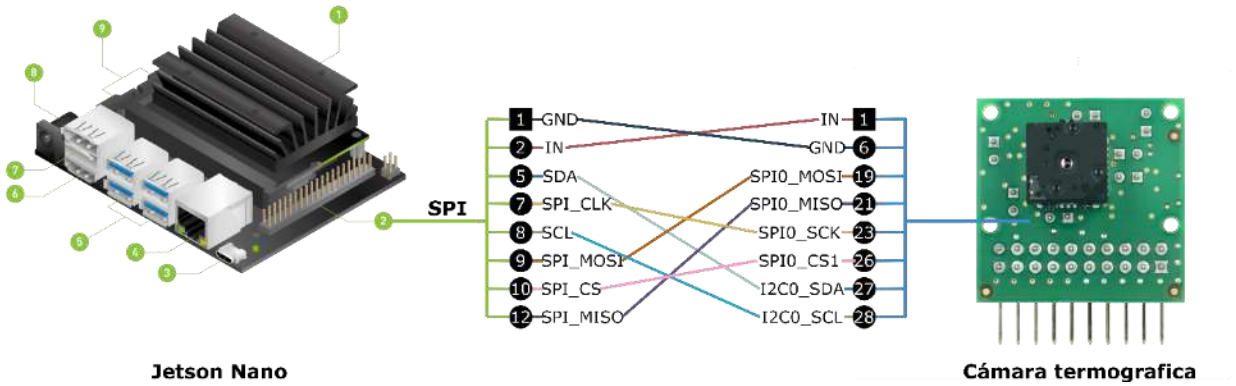

Figura 45. Esquema de conexión de la cámara Lepton 3.5 a la placa Breakout Board v2.0.

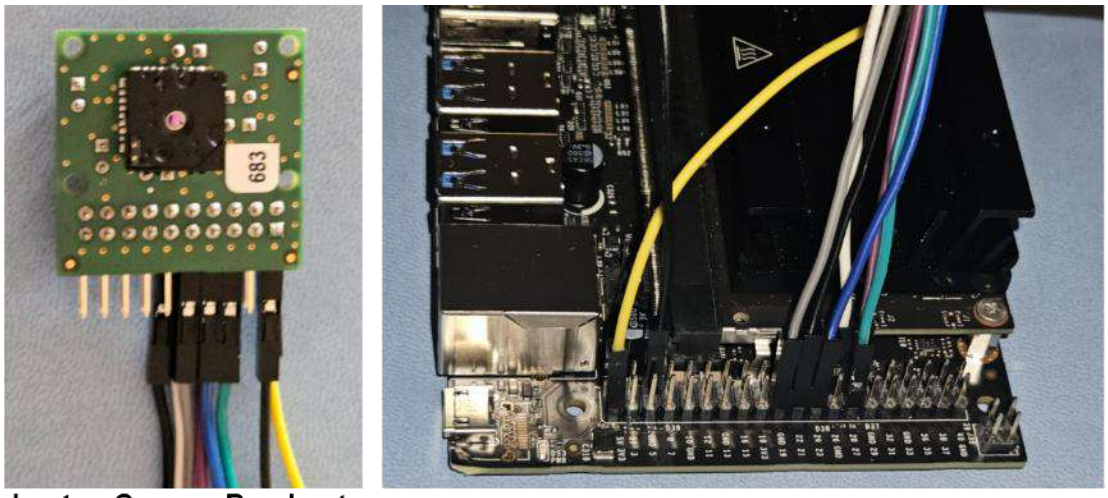

Figura 46. Conexión de la cámara Lepton 3.5 a la placa Breakout Board v2.0.

3.3. Prender tarjeta. Verificar que la comunicación I2C funcione adecuadamente, para ello, abrir la terminal e introducir el siguiente comando, la salida se deberá ver como la Figura 47:

```
sudo i2cdetect -y -r 0
```

El resultado debe mostrarse de la siguiente manera:

```
aleyubi@aleyubi-desktop:~$ sudo i2cdetect -y -r 0
[sudo] password for aleyubi:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- 2a -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figura 47. Salida del comando sudo i2cdetect -y -r 0

3.4. Ingresar en la carpeta pylepton, la cual contiene todo lo descargado del repositorio de GitHub.

3.5. Dentro de la carpeta pylepton, se almacena otra carpeta pylepton la cual contiene las archivos .py de las librerías tanto Lepton como Lepton3. Abrir el archivo "Lepton.py" y modificar en la línea de SPEED a 32100000, como se observa en la Figura 48, se puede editar con VSCode.

```
class Lepton(object):
  """Communication class for FLIR Lepton module on SPI

  Args:
    spi_dev (str): Location of SPI device node. Default '/dev/spidev0.0'.
  """

  ROWS = 60
  COLS = 80
  VOSPI_FRAME_SIZE = COLS + 2
  VOSPI_FRAME_SIZE_BYTES = VOSPI_FRAME_SIZE * 2
  MODE = SPI_MODE_3
  BITS = 8
  SPEED = 18000000
  SPIDEV_MESSAGE_LIMIT = 24

  def __init__(self, spi_dev = "/dev/spidev0.0"):
    self._spi_dev = spi_dev
    self._txbuf = np.zeros(Lepton.VOSPI_FRAME_SIZE, dtype=np.uint16)
```

Figura 48. Modificación del código de la librería Lepton.py

3.6. Lo siguiente a modificar dependerá de la forma en que se conectaron los pines, ya que, como se dijo antes, se puede conectar al pin 24 como "$CS0$" o en el pin 26 como "$CS1$". No obstante, lo que cambiará será la identificación del dispositivo en el kernel spidev. El pin 24 se relaciona con $spidev0.0$ y el pin 26 se asocia con spidev0. 1. Por lo tanto, si se utiliza el pin 24, en ambos archivos, "$Lepton.py$" y "$Lepton3.py$", las líneas que incluyan "$/dev/spidev0.0$" permanecerán así.

3.7. Si se conecta al pin 26, se deberá cambiar las líneas que incluyan "$/dev/spidev0.0$" por "$/dev/spidev0.1$".

Nota: Si está en constante cambio la conexión de estos pines, checar siempre en ambos archivos que corresponda el dispositivo al que se esté llamando con el que esté conectado. Siempre que se modifique guardar los cambios.

4. Cargar al kernel spidev con el siguiente comando:

```
sudo modprobe spidev
```

5. Para verificar el funcionamiento, se puede realizar con los archivos que vienen incluidos en la carpeta del repositorio: *"thermalImage.py"* y *"pylepton_capture.py"*. Se pueden ejecutar directamente desde la terminal con los comandos siguientes:

```
python3 pylepton_capture output.png
python3 thermalImage.py /home/aleyubi/Lepton1/pylepton/ /dev/spidev0.1 "1"
```

6. Se creó el archivo *"capture_simple.py"*, Figura 49, utilizando el código que viene de ejemplo en la página del repositorio del GitHub.

Figura 49. Código obtenido del repositorio de Github.

7. Se elaboraron dos archivos más para experimentar, pero ahora con video, estos archivos se llaman *"pantalla_1.py"* y *"pantalla_2.py"*, un ejemplo se puede observar en la Figura 55, donde se ve el video en escala de grises y del lado derecho a color.

Figura 50. Del lado derecho está el código para obtener video. Del lado izquierdo se muestra la salida a escala de grises y a color del video.

Paso 12. Descargar de librerías para los siguientes pasos

A continuación, se presentan los comandos utilizados para descargar librerías requeridas para diseñar la interfaz y cargar el modelo de CNN en la tarjeta Jetson. Las librerías instalaron fueron las siguientes: Pandas, CustomTkinter, Tkinter, Pillow, Docx, Tensorflow y Keras. Los comando utilizados fueron:

```
pip3 install pandas
pip3 install customtkinter==3.9
sudo apt-get install python3-tk
sudo apt-get install python3-pil
```

```
sudo apt install libjpeg-dev zlib1g-dev
sudo apt install libjpeg8-dev libfreetype6-dev liblcms2-dev
python3 -m pip install --user --upgrade pillow
pip3 install python-docx
pip3 install --pre --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v46 tensorflow==2.5.0+nv21.8
```

Paso 13. Compatibilidad del modelo de CNN con la tarjeta Jetson Nano

El modelo CNN con el que se trabajó durante este proyecto fue entrenado utilizando Google Colab. Tensorflow y Keras fueron las principales librerías que se ocuparon, usando una versión reciente de cada una: 2.18.0 para Tensorflow y 3.8.0 para Keras. Además, se crea un archivo un poco más pesado.

Las versiones instaladas en la tarjeta Jetson nano son: $2.5.0 + nv21.8$ para Tensorflow y 1.0.8 para Keras. Además, al ser un sistema embebido cuenta con menos recurso computacionales para correr un modelo de inteligencia artificial.

Para ejecutar el modelo CNN en la Jetson Nano se convirtió a un formato de Tensorflow Lite. Los pasos que se realizaron fueron los siguientes:

1. Definir las librerías necesarias

```
from tensorflow.keras.models import load_model, Model
from tensorflow.keras.layers import Input
from google.colab import drive
import tensorflow as tf

drive.mount('/content/drive')
```

2. Cargar el modelo llamado “modelo100_4_data3.h5”.

```
# Cargar modelo en TensorFlow 2.x
modelo = load_model("/content/drive/MyDrive/CNN3/Modelo_bueno/modelo100_4_data3.h5")
```

3. Convertir el modelo Tensorflow a TensorFlow Lite

```
converter = tf.lite.TFLiteConverter.from_keras_model(modelo)
tflite_model = converter.convert()
```

4. Guardar el modelo TensorFlow Lite

```
# Guardar el modelo .tflite
with open('modelo.tflite', 'wb') as f:
    f.write(tflite_model)
```

Paso 14. Cargar modelo y probar modelo CNN en la tarjeta Jetson Nano

A continuación, se muestran los comandos utilizados para ejecutar el modelo CNN convertido a Tensorflow Lite

1. Definir las librerías necesarias.

```
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
```

```
from PIL import Image
```

2. Cargar el modelo Tensorflow Lite

```
# Cargar el modelo TFLite
interpreter = tf.lite.Interpreter(model_path='modelo.tflite')
interpreter.allocate_tensors()
```

3. Obtener detalles de entrada y salida

```
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

4. Cargar la imagen que se va a utilizar en el modelo.

```
# Cargar la imagen
image_path = 'clase0_1.jpg'
image = Image.open(image_path).convert('RGB')
image_array = np.expand_dims(image, axis=0)
image_array = image_array.astype(np.float32)
```

5. Predecir

```
# Asignar la imagen preprocesada como entrada
interpreter.set_tensor(input_details[0]['index'], image_array)

# Ejecutar la inferencia
interpreter.invoke()

# Obtener la predicción
output_data = interpreter.get_tensor(output_details[0]['index'])
predicted_class = np.argmax(output_data, axis=1)

print("Predicción:", output_data)
print("Predicción número:", predicted_class)
print("Modelo cargado exitosamente")
```

Paso 15. Conexión final de todos los componentes.

A continuación, se muestran un diagrama con la conexión final de los componentes del sistema embebido.

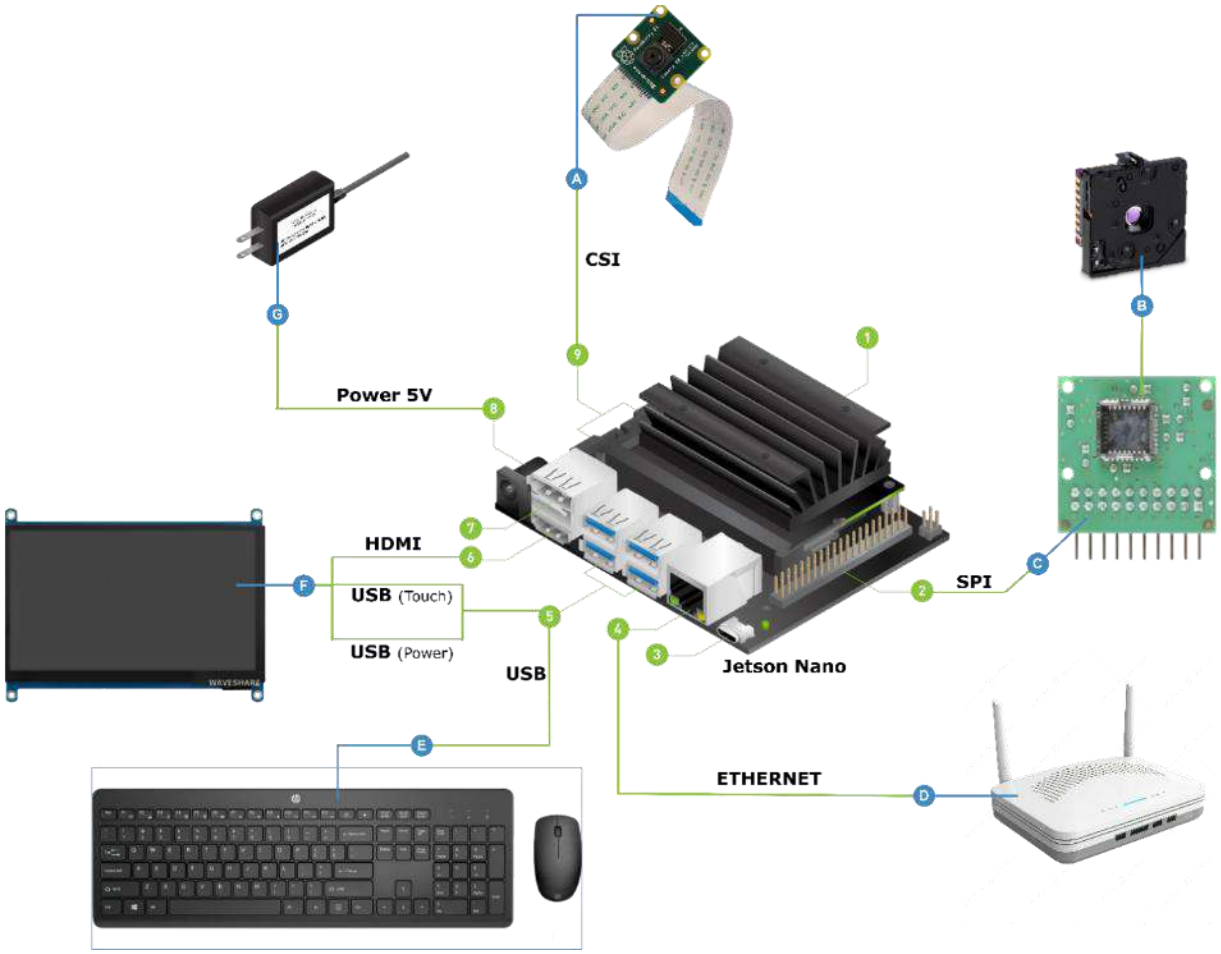

Figura 51. Diagrama sobre la conexión de componentes del sistema embebido, A) Raspberry Pi Modulo cámara, B) Cámara termográfica Lepton 3.5, c) Breakout Board Lepton, D) Ethernet, E) Mouse y teclado, F) HDMI 7¨ Touch Screen LCD, G) Adaptador de corriente.