



**INSTITUTO LATINOAMERICANO DE CIENCIAS
DE LA VIDA Y LA NATURALEZA (ILACVN)**

INGENIERÍA FÍSICA

**TÉCNICAS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN
DE FALLAS EN MÓDULOS FOTOVOLTAICOS**

JHOAN RODRIGO PÉREZ VARGAS

Foz do Iguazú
2022



**INSTITUTO LATINOAMERICANO DE CIENCIAS
DE LA VIDA Y LA NATURALEZA (ILACVN)**

INGENIERÍA FÍSICA

**TÉCNICAS DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN
DE FALLAS EN MÓDULOS FOTOVOLTAICOS**

JHOAN RODRIGO PÉREZ VARGAS

Trabajo de Conclusión del Curso presentado al Instituto Latinoamericano de Ciencias de la Vida y de la Naturaleza de la Universidad Federal de la Integración Latino-Americana, como requisito parcial para obtener la titulación Bacharel en Ingeniería Física.

Orientador: Prof. Dr. Johan Alexander Cortes Suarez

Foz do Iguazú
2022

JHOAN RODRIGO PÉREZ VARGAS

**TÉCNICAS DE APRENDZAJE PROFUNDO PARA LA DETECCIÓN
DE FALLAS EN MÓDULOS FOTOVOLTAICOS**

Trabajo de Conclusión del Curso presentado al Instituto Latinoamericano de Ciencias de la Vida y de la Naturaleza de la Universidad Federal de la integración latinoamericana, como requisito parcial para obtener la titulación Bacharel en Ingeniería Física.

BANCA EXAMINADORA

Orientador: Prof. Dr. Johan Alexander Cortes Suarez
UNILA

Prof. Dr. Marcelo Goncalves Honnicke
UNILA

Prof. Joylan Nunes Maciel
UNILA

Foz do Iguazú, 05 de agosto de 2022.

TERMO DE SUBMISSÃO DE TRABALHOS ACADÊMICOS

Nome completo do autor(a): Jhoan Rodrigo Pérez Vargas

Curso: Engenharia Física

		Tipo de Documento
(.....) graduação	(.....) artigo	
(.....) especialização	(X) trabalho de conclusão de curso	
(.....) mestrado	(.....) monografia	
(.....) doutorado	(.....) dissertação	
	(.....) tese	
	(.....) CD/DVD – obras audiovisuais	
	(.....) _____	

Título do trabalho acadêmico: Técnicas de inteligencia artificial para la detección de fallas en módulos fotovoltaicos.

Nome do orientador(a): Johan Alexander Cortes Suarez

Data da Defesa: 05/08/2022

Licença não-exclusiva de Distribuição

O referido autor(a):

a) Declara que o documento entregue é seu trabalho original, e que o detém o direito de conceder os direitos contidos nesta licença. Declara também que a entrega do documento não infringe, tanto quanto lhe é possível saber, os direitos de qualquer outra pessoa ou entidade.

b) Se o documento entregue contém material do qual não detém os direitos de autor, declara que obteve autorização do detentor dos direitos de autor para conceder à UNILA – Universidade Federal da Integração Latino-Americana os direitos requeridos por esta licença, e que esse material cujos direitos são de terceiros está claramente identificado e reconhecido no texto ou conteúdo do documento entregue.

Se o documento entregue é baseado em trabalho financiado ou apoiado por outra instituição que não a Universidade Federal da Integração Latino-Americana, declara que cumpriu quaisquer obrigações exigidas pelo respectivo contrato ou acordo.

Na qualidade de titular dos direitos do conteúdo supracitado, o autor autoriza a Biblioteca Latino-Americana – BIUNILA a disponibilizar a obra, gratuitamente e de acordo com a licença pública *Creative Commons Licença 3.0 Unported*.

Foz do Iguaçu, 05 de agosto de 2022.

Assinatura do Responsável

Dedico este trabajo a mi familia, quienes siempre han estado ahí para mí y me han apoyado en todo. Sin ellos, no hubiera podido lograr nada de lo que he logrado hasta ahora.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi orientador de tesis, al Dr. Johan Alexander Cortes Suarez, por su guía y apoyo durante todo el proceso de investigación y redacción de esta tesis. También quiero agradecer a los miembros de la banca de tesis, los Dres. Marcelo Goncalves Honnicke y Joylan Nunes Maciel, por su participación en mi proceso formativo.

Un agradecimiento especial a los profesores Dres. Cleilton Aparecido Canal y Eralcilene Moreira Terezio por darme la oportunidad y orientarme durante los proyectos de investigación que desarrolle junto a ellos.

Asimismo, quiero dar las gracias a mi supervisor de prácticas Valentín Silvera Diaz, por sus valiosos aportes, comentarios y sugerencias para el desarrollo de este trabajo, al Parque Tecnológico de ITAIPU y todos los que han participado de una forma u otra en el desarrollo de este trabajo. En particular, quiero agradecer a mi familia y amigos por su apoyo y comprensión durante todo este tiempo.

Finalmente, quiero agradecer a la UNILA, institución que me brinda la oportunidad de estar hoy aquí y fomenta la educación pensando en nuestra América Latina.

*Nuestra misión es alcanzar un mayor nivel de conciencia
y despertar el potencial ilimitado de la humanidad.*

Nikola Tesla

RESUMO

O desempenho dos sistemas fotovoltaicos é afetado por diversos fatores, incluindo falhas ou anomalias que ocorrem nos principais sistemas de geração de energia, os módulos fotovoltaicos (PV). A implantação crescente de plantas fotovoltaicas e a necessidade de aumentar seu desempenho e confiabilidade requer o desenvolvimento de ferramentas de inspeção que permitam identificar anomalias de forma barata, rápida e eficiente. Nesse contexto, vários grupos de pesquisa têm utilizado a termografia infravermelha aérea (aIRT) como ferramenta para identificar falhas em módulos fotovoltaicos por meio de imagens termográficas tiradas por veículos aéreos não tripulados (UAVs). Essas imagens podem ser avaliadas por técnicos experientes para determinar se há uma falha ou não em um determinado módulo PV, porém, essa metodologia se torna complexa e extenuante devido ao grande volume de dados quando se trata de analisar centenas ou milhares de módulos PV presentes em uma grande fazenda solar. O presente trabalho explora a aplicação de técnicas de *Deep Learning* (DL) em um conjunto de imagens obtidas por aIRT para detectar falhas em módulos fotovoltaicos, a fim de avaliar seu desempenho e a capacidade de automatizar o processo de inspeção de falhas. As imagens foram obtidas a partir de conjuntos de dados (*dataset*) de acesso livre disponíveis na web para o desenvolvimento de pesquisas nessa área. Além disso, foi criado um *dataset* sintético para aumentar o volume de dados de entrada para o treinamento de um modelo de DL utilizado para identificação dos módulos PV. As imagens foram pré-processadas para treinar dois modelos de DL: Mask R-CNN, um modelo usado para segmentação, e ResNet, uma rede neural convolucional usada para problemas de classificação. Os resultados obtidos neste trabalho mostraram uma precisão nos conjuntos de teste de até 97,9% e 81,4% para segmentação e classificação, respectivamente. Por fim, foi implementado um modelo conjunto (segmentação e classificação) para a detecção de falhas em imagens e vídeos, cujos resultados demonstram a capacidade dessas ferramentas para a automação de processos com grandes volumes de informação.

Palavras-chave: Energia solar; Deep Learning; Inteligência artificial; Módulos fotovoltaicos; Termografia.

RESUMEN

El desempeño de los sistemas fotovoltaicos se ve afectado por diversos factores, entre ellos, fallas o anomalías que se presentan en los principales sistemas de generación de energía, los módulos fotovoltaicos (PV). El creciente despliegue de plantas fotovoltaicas y la necesidad de aumentar el rendimiento y la fiabilidad de estas, exige el desarrollo de herramientas de inspección que permita identificar anomalías de forma barata, rápida y eficiente. En este contexto, varios grupos de investigación han usado la termografía infrarroja aérea (aIRT) como herramienta para identificar fallas en módulos PV por medio de imágenes termográficas tomadas por vehículos aéreos no tripulados (UAV). Dichas imágenes pueden ser evaluadas por técnicos experimentados para determinar si existe una falla o no en un determinado modulo PV, sin embargo, esta metodología se torna compleja y extenuante debido al gran volumen de información cuando se trata de analizar cientos o miles de módulos PV presentes en una granja solar de grande porte. El presente trabajo explora la aplicación de técnicas de aprendizaje profundo (DL) sobre un conjunto de imágenes obtenidas por aIRT para detectar fallas en módulos PV con el fin de evaluar su desempeño y la capacidad de automatizar el proceso de inspección de fallas. Las imágenes fueron obtenidas a partir de conjuntos de datos (*dataset*) de libre acceso disponible en la web para el desarrollo de investigación en esta área. Adicionalmente se creó un *dataset* sintético para incrementar el volumen de datos de entrada para el entrenamiento de un modelo de DL usado para la identificación de los módulos. Las imágenes se preprocesaron para entrenar dos modelos de DL: Mask R-CNN, un modelo usado para segmentación de instancias, y ResNet, una red neuronal convolucional usada para problemas de clasificación. Los resultados obtenidos en este trabajo mostraron una precisión sobre conjuntos de prueba (*test*) de hasta un 97,9% y 81.4% para la segmentación y la clasificación respectivamente. Finalmente se implementó un modelo conjunto (segmentación y clasificación) para la detección de fallas sobre imágenes y video, cuyos resultados demuestran la capacidad de estas herramientas para la automatización de procesos con grandes volúmenes de información.

Palabras clave: Energía solar; *Deep Learning*; Inteligencia artificial; Módulos fotovoltaicos; Termografía.

LISTA DE ABREVIATURAS Y SIGLAS

ILACVN	Instituto Latino-Americano de Ciências da Vida e da Natureza
UNILA	Universidade Federal da Integração Latino-Americana
IR	Infrared
IRT	Infrared Thermography
aIRT	aereal Infrared Thermography
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
Lr	Learning rate
TL	Transfer Learning
IEC	International Electrotechnical Commission
PV	Photovoltaic
ResNet	Residual Neural Network
ML	Machine Learning
DL	Deep Learning
FC	Fully Connected
AI	Artificial Intelligence
PVM	Photovoltaic Module
UAV	Unmanned Aerial Vehicle

SUMÁRIO

1 INTRODUCCIÓN	12
2 MARCO TEORICO	14
2.1 SOBRE MODULOS FOTOVOLTAICOS.....	14
2.1.1 Módulos Fotovoltaicos.....	14
2.1.2 Fallas.....	14
2.1.3 Termografía infrarroja.....	14
2.2 IMAGENES, PRE-PROCESAMIENTO Y CONVOLUCIÓN.....	16
2.2.1 Imágenes.....	17
2.2.2 Normalización.....	17
2.2.3 Operación De Convolución.....	18
2.2.4 Capa de agrupación.....	20
2.2.5 Segmentación de Imágenes y Visión Por Computadora.....	21
2.3 SOBRE INTELIGENCIA ARTIFICIAL.....	23
2.3.1 Redes Neuronales Artificiales.....	23
2.3.2 Conceptos Relevantes A Optimización De Redes Neuronales.....	26
2.3.4 Redes Neuronales Convolucionales.....	31
2.3.4 Transferencia De Conocimiento.....	36
2.4 ESTADO DEL ARTE.....	37
3 DELINEAMIENTO METODOLOGICO	39
3.1 BASES DE DATOS.....	39
3.1.1 Segmentación.....	39
3.1.2 Clasificación.....	43
3.2 METODOLOGÍA.....	48
3.2.1 Segmentación de instancias.....	48
3.2.2 Clasificador de imágenes.....	49
3.2.3 Identificar Fallas.....	51
3.4 HERRAMIENTAS Y TECNOLOGIAS.....	52
4 ANÁLISIS Y RESULTADOS	53
4.1 SEGMENTACIÓN.....	53
4.2 CLASIFICACIÓN.....	56

4.3 IDENTIFICAR FALLAS.....	63
5 CONSIDERACIONES FINALES	65
REFERENCIAS	67

1 INTRODUCCIÓN

Según el último reporte sobre el estatus global de energías renovables (2022), las energías renovables (ER) representaron el 12,6% de la energía total consumida para el final del año 2020 y las inversiones en las mismas han alcanzado los 336 mil millones de dólares para el año 2021 (REN21, 2022b). Esto, ha permitido que, nuevamente, las ER experimenten un récord de crecimiento anual y que, por primera vez, la energía solar y eólica proporcione más del 10% de la electricidad a nivel mundial.

Respecto a la energía solar fotovoltaica (PV por sus siglas en ingles), esta experimento un aumento del 25% en capacidad instalada para el año 2021, llegando a los 175 gigawatts (GW) y una participación mundial promedió del 5 %, frente al 3,7 % de 2020 (REN21, 2022a). En cuanto a la nueva capacidad instalada, Brasil aparece como líder en la región, añadiendo 5.5GW en el año 2021 (REN21, 2022b), y disputando así el 5° puesto en el ranking a nivel mundial.

A medida que el mundo experimenta una participación cada vez mayor de la energía fotovoltaica en la matriz energética, aumentar el rendimiento y la fiabilidad de las instalaciones fotovoltaicas es de suma importancia. Para ello, un factor clave para reducir los costos de los sistemas PV es aumentar la confiabilidad y el tiempo de vida útil de los módulos PV. Los módulos PV suelen sufrir daños por la temperatura, la lluvia, el viento, etc. y daños mecánicos durante el transporte y la instalación (KÖNTGES *et al.*, 2014). Estos daños acortan la vida útil de los módulos PV, además, podrían afectar a todo el sistema PV, lo que genera problemas de eficiencia económica y pérdida de energía. Es por ello por lo que, se requiere aplicar métodos de prueba no destructivos, rápidos, confiables y automáticos, en la inspección y mantenimiento de los módulos PV con regularidad.

En este contexto, la termografía infrarroja aérea (*aereal Infrared Thermography* - aIRT) se ha convertido en un método de detección de fallos bien establecido y competitivo para el monitoreo y mantenimiento de la condición de los sistemas PV, a través de la combinación de cámaras de termografía infrarroja (*Infrared Thermography* - IRT) con un vehículo aéreo no tripulado (*unmanned aerial vehicle* - UAV) (DE OLIVEIRA *et al.*, 2022). Sin embargo, las actividades manuales realizadas por los especialistas para el análisis de las imágenes obtenidas por aIRT se consideran un problema, pues, además de ser exhaustivas, es necesario tener experiencia profesional y

se recibe una gran cantidad de información, cuando aplicado a parques solares de grande porte.

En este sentido, ya que, el problema principal en el área de visión por computadora y el aprendizaje profundo (Deep Learning - DL) es la detección de objetos y clasificación de estos, e involucra el manejo de grandes volúmenes de datos (LIU *et al.*, 2020), es de gran interés aplicar estas tecnologías para intentar automatizar la detección de fallas en módulos PV.

Durante los últimos años varios grupos de investigación han desarrollado trabajos que implementan tecnologías de aprendizaje de maquina (*Machine Learning - ML*) usando la termografía infrarroja aérea tanto para la detección y segmentación de los módulos PV, como para la clasificación de fallas, consiguiendo precisiones superiores al 90% (DE OLIVEIRA *et al.*, 2022). No obstante, la precisión, robustez y generalización de los algoritmos desarrollados siguen siendo uno de los principales problemas de estos estudios, especialmente cuando se trata de un gran número de clases de fallas y la inspección de plantas fotovoltaicas a gran escala. Por lo tanto, el procedimiento autónomo y la tarea de clasificación aún deben explorarse para mejorar el rendimiento y la aplicabilidad del método aIRT.

En este contexto, el presente trabajo explora la aplicación de técnicas de inteligencia artificial (AI) usando distintos conjuntos de imágenes para detectar fallas en módulos PV, tomando como referencia el trabajo desarrollado por (BOMMES *et al.*, 2021b). Para esto, se utilizaron diferentes conjuntos de imágenes térmicas, etiquetadas y no etiquetadas. Las imágenes se preprocesaron para entrenar dos modelos de AI, Mask R-CNN para la identificación de los paneles PV, por medio del método de segmentación de instancias, y una red neuronal residual (ResNet), con diferentes configuraciones, para la clasificación de las fallas.

2 MARCO TEORICO

En este capítulo se presenta una revisión de la literatura relativa a los principales conceptos sobre módulos fotovoltaicos, procesamiento de imágenes, redes neuronales y trabajos científicos relevantes para el desarrollo de este trabajo.

2.1 SOBRE MODULOS FOTOVOLTAICOS

En esta sección abordaremos, de forma sucinta, definiciones importantes relativas a los módulos fotovoltaicos, fallas e inspección por termografía infrarroja.

2.1.1 Módulos Fotovoltaicos

Según NBR 16.690: un módulo PV es una unidad formada por un conjunto de células fotovoltaicas, interconectadas eléctricamente y encapsuladas, con el objetivo de generar electricidad.

2.1.2 Fallas

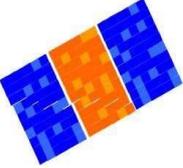
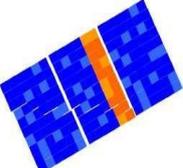
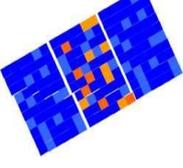
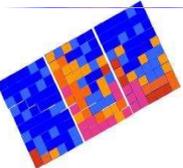
Una falla en un módulo PV es un efecto que degrada la potencia del módulo, que no es reversible por el funcionamiento normal o crea un problema de seguridad (KÖNTGES *et al.*, 2014). Desde el punto de vista de la seguridad, una falla en un módulo PV es una amenaza para el funcionamiento del sistema eléctrico. Las fallas en los módulos PV pueden ser causadas por una variedad de factores, incluyendo el mal funcionamiento de los componentes del módulo, el daño mecánico, la oxidación y la exposición a la radiación ultravioleta (UV) (JAHN; HERZ, 2018).

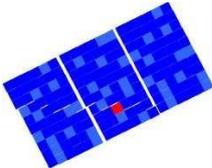
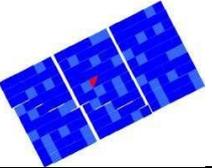
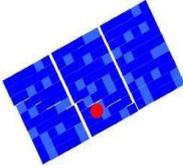
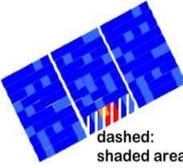
2.1.3 Termografía infrarroja

La termografía infrarroja (IRT) es una técnica no destructiva que permite diagnosticar fallas en módulos PV sin la necesidad de desconectar el módulo del circuito eléctrico. Se basa en la medición de la radiación infrarroja emitida por el módulo la cual está relacionada con la temperatura del módulo, de modo que las áreas más calientes del módulo indican fallas en el mismo.

El método de termografía en condiciones estables para la detección de fallas en módulos PV es una técnica común y fácil de aplicar (DENIO, 2012). Este método permite el análisis de módulos PV en campo en condiciones de trabajo. El Cuadro 1 muestra un resumen de los patrones de imagen infrarroja que pueden ser observados en un módulo PV mediante el uso de termografía en mediciones al aire libre, una breve descripción, posibles causas de falla y su influencia en la salida eléctrica.

Cuadro 1 – Resumen de patrones observados en imágenes IR sobre módulos PV

Patrón	Descripción	Posible razón de falla	Mediciones eléctricas
	Un módulo más caliente que los otros	Modulo en circuito abierto o no conectado al sistema	Módulo normalmente funcional
	Una línea (sub-string) más calientes que las otras líneas en el módulo	Corto circuito (CC) o sub-string abierto, CC en el diodo bypass o CC interno	Pérdida de potencia de los sub-strings
	Celdas individuales más calientes, no se reconoce ningún patrón (patrón de mosaico).	Módulo entero está en CC, todos los diodos de bypass en CC o conexión errada	Potencia del módulo drásticamente reducida
	Celdas individuales más calientes, las partes inferiores y cercanas al marco están más calientes que las partes superior y media.	"Shunts" masivos causados por degradación inducida del potencial (PID) y/o polarización	Potencia del módulo reducida.

	Una célula claramente más caliente que las otras	Efectos de sombra, defecto de célula o célula delaminada	Reducción de potencia, no necesariamente permanente.
	Parte de una célula más caliente	Célula quebrada o interconexión de string desconectada	Reducción drástica de potencia.
	Punto caliente (Hot Spot)	Artefacto sobre el módulo, parcialmente sombreado.	Reducción de potencia, dependiendo de la forma y tamaño de la parte afectada
	Sub-string notablemente más caliente que los otros cuando están igualmente sombreados.	Sub-string con diodo de bypass en circuito abierto o circuito perdido.	Reducción de corriente e de potencia cuando parte del sub-string esta sombreado

Fuente: Adaptado de (KÖNTGES *et al.*, 2014)

Una característica importante de la técnica IRT es la posibilidad de aplicarla a gran escala, a través de la combinación de cámaras IR con UAV, configurando así la termografía aérea infrarroja (aIRT) (DENIO, 2012). Múltiples publicaciones han demostrado la capacidad de esta técnica para detectar fallas en módulos fotovoltaicos de forma rápida y eficiente (BOMMES *et al.*, 2021a; DE OLIVEIRA *et al.*, 2022; RICHARDSON *et al.*, 2017). Sin embargo, la tarea de detectar las fallas se torna compleja cuando aplicada a plantas de grande porte, con cientos o miles de módulos PV.

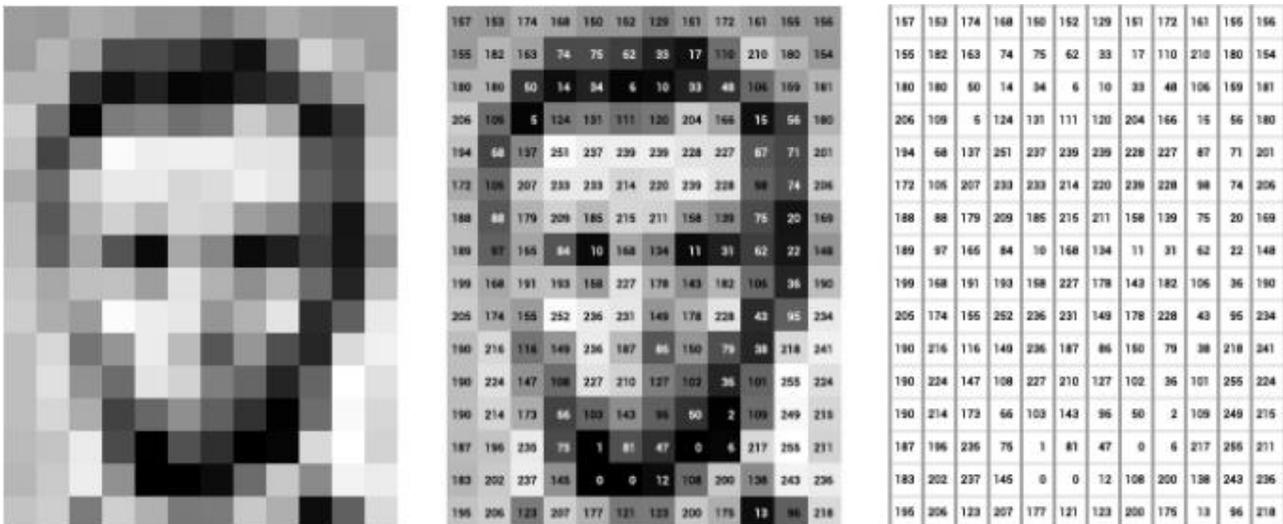
2.2 IMAGENES, PRE-PROCESAMIENTO Y CONVOLUCIÓN

En esta sección se define conceptos importantes relativos al procesamiento de imágenes, operaciones de convolución y segmentación de imágenes que serán utilizados en las técnicas de aprendizaje de máquina.

2.2.1 Imágenes

Cuando se aborda el uso de “imágenes” en el contexto de aprendizaje de maquina (ML), las mismas deben entenderse como una matriz de pixeles que representan gráficamente una imagen. Si se toma una imagen en escala de grises como ejemplo, cada píxel puede representarse como un número entre 0 y 255, donde 0 es completamente negro y 255 es completamente blanco. Si la imagen es a color, cada píxel se puede representar como un vector de tres números, uno para cada componente: color rojo, verde y azul (*RGB* por sus siglas en inglés). En general, la representación de una imagen como una matriz de pixeles (Figura 1) es la forma en la cual se procesan las imágenes en la mayoría de los algoritmos de aprendizaje automático.

Figura 1 – Representación matricial de una imagen a escala de grises



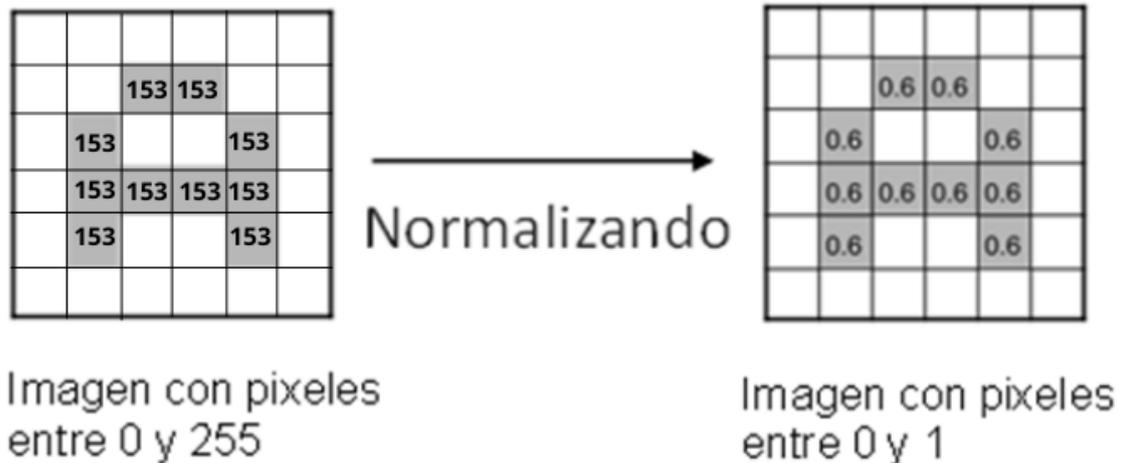
Fuente: obtenida de (MARTÍNEZ, 2018)

2.2.2 Normalización

Normalizar una imagen es un proceso por el cual se convierte una imagen de un formato a otro. El objetivo de la normalización es ajustar la escala de los valores de la imagen para que estén dentro de un rango específico. Esto es importante porque muchos algoritmos de aprendizaje automático no funcionan bien si los valores de entrada están fuera de un rango específico. Dado que los colores representados por los pixeles asumen

valores que van de 0 a 255, se suele aplicar una transformación de cada píxel dividiendo su valor entre 255, tal que, los valores de la matriz siempre poseen un valor entre 0 y 1 (Figura 2).

Figura 2 – Representación de la operación de normalización para una imagen a blanco y negro

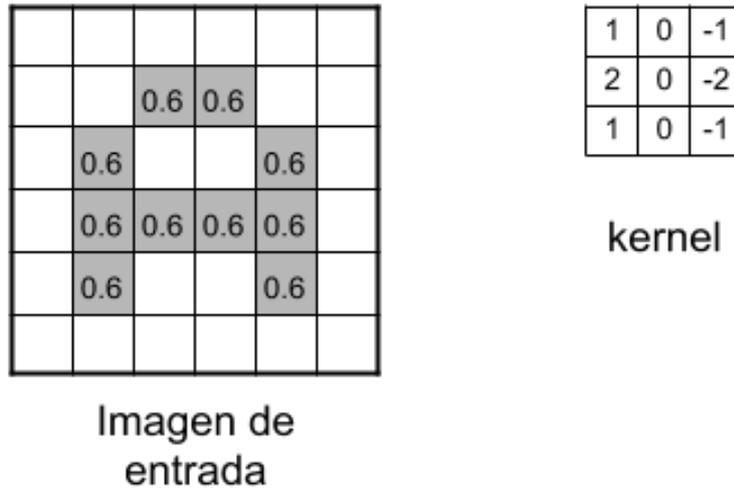


Fuente: Adaptada de (BAGNATO, 2022)

2.2.3 Operación De Convolución

En el contexto de la visión por computadora y el procesamiento de imágenes, la convolución se refiere a una técnica de procesamiento de señales y de imagen que se utiliza a menudo para realzar o detectar ciertas características en una imagen. Esta operación se realiza multiplicando pequeñas submatrices de una imagen por un kernel del mismo tamaño, que es una pequeña matriz de números (Figura 3) cuyos valores y dimensión son establecidos dependiendo de la características que se buscan resaltar (WANG *et al.*, 2021). El kernel se desliza a lo largo de la imagen, de izquierda a derecha y de arriba-abajo, y en cada paso se aplica la operación de convolución. Esto produce una nueva imagen que contiene información sobre las características de la imagen original (Figura 4).

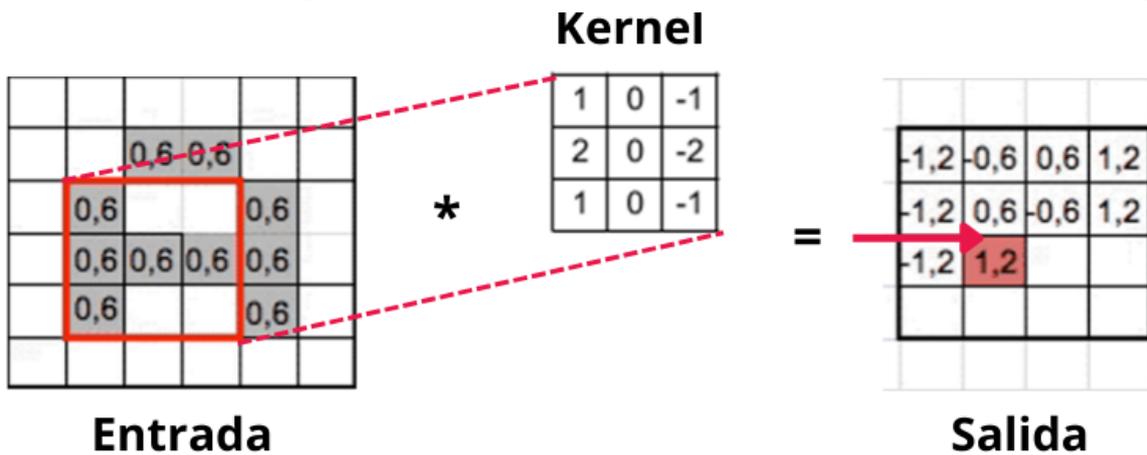
Figura 3 – Elementos de la operación de convolución



Fuente: obtenida de (BAGNATO, 2022)

En aplicaciones de ML no se suele aplicar un único kernel, por el contrario, se aplican varios. A ese conjunto se les suele llamar filtros (BAGNATO, 2022). Por ejemplo, en una convolución sobre una imagen 28x28x1 se podría aplicar 32 filtros, con lo cual se obtendría 32 matrices de salida. A este conjunto de salida se le conoce como mapa de características o *feature mapping*.

Figura 4– Operación de convolución sobre la representación matricial de la imagen de entrada, generando así, una matriz que compone el *feature mapping*.



Fuente: Obtenida de (BAGNATO, 2022)

Es importante notar que, luego de aplicada una operación de convolución, la matriz de salida tendrá dimensión menor que la matriz que representa la imagen de entrada. La reducción de la dimensión depende del tamaño del Kernel. En el caso de

imágenes con múltiples canales (por ejemplo, RGB), el Kernel debe tener la misma profundidad, es decir, el mismo número de canales (x3) que la de la imagen de entrada. (SAHA, 2018)

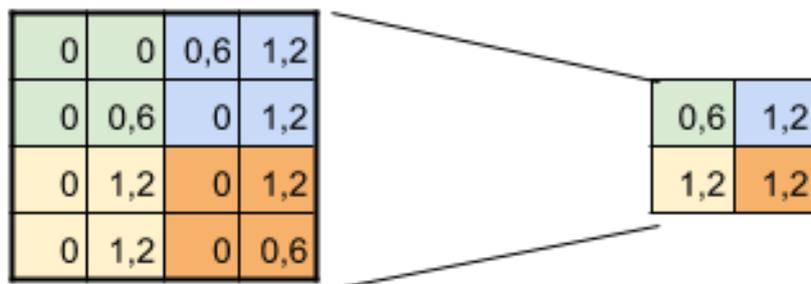
2.2.4 Capa de agrupación

La capa de agrupación o *pooling* es un proceso mediante el cual se selecciona una pequeña cantidad de datos de un grupo mayor. El objetivo de este procedimiento es disminuir la potencia computacional requerida para procesar los datos a través de la reducción de la dimensionalidad. Además, es útil para extraer características dominantes que son invariantes rotacionales y posicionales, manteniendo así el proceso de entrenamiento efectivo del modelo. La capa convolucional y la capa de agrupación forman la capa *i*-ésima de una CNN.

Para el ejemplo citado anteriormente, observe que al aplicar la convolución con 32 filtros se generaban 32 imágenes de salida, por lo que, al aplicar operaciones de convolución consecutivamente los datos aumentarían en profundidad de forma exponencial, haciendo inviable una red que aplique varias operaciones de convolución.

El *Max Pooling* es una técnica de *Pooling* que se utiliza para seleccionar el valor máximo de una matriz de datos. Se utiliza comúnmente en redes neuronales convolucionales para reducir el tamaño de los datos y reducir el tiempo de entrenamiento, además que, también funciona como un supresor de ruido (SAHA, 2018). Se puede observar un ejemplo de esta operación en La Figura 5.

Figura 5– Aplicación de *Max-Pooling 2x2*, reduciendo la salida a la mitad de tamaño.



Fuente: Obtenida de (BAGNATO, 2022)

Aplicado al ejemplo anterior, suponga que se aplica *Max-Pooling* de tamaño 2x2. Esto quiere decir que se recorrerá cada una de las 32 imágenes de características obtenidas anteriormente de 28x28px de izquierda-derecha, de arriba-abajo, pero, en vez de tomar de a 1 píxel, se tomaran pequeñas matrices de 2x2px (2 de alto por 2 de ancho = 4 píxeles) y se preservara el valor “más alto” de entre esos 4 píxeles. En este caso, usando 2x2, la imagen resultante es reducida “a la mitad” y quedará de 14x14 píxeles. Luego de este proceso de *Pooling* se obtendrán 32 imágenes de 14x14, esto es, menos información, pero aun así almacenando los datos más importantes para detectar las características deseadas.

2.2.5 Segmentación de Imágenes y Visión Por Computadora

La segmentación de imágenes es el proceso de partición de una imagen en regiones significativas con respecto a una aplicación en particular. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen para que sea más fácil de analizar. La segmentación de imágenes suele usarse para localizar objetos y límites (líneas, curvas, etc.) en imágenes. Más precisamente, la segmentación de imágenes es el proceso de asignar una etiqueta a cada píxel en una imagen de tal manera que los píxeles con la misma etiqueta compartan ciertas características visuales.

En este contexto, la visión por computadora tiene cuatro tareas básicas: clasificación de imágenes, detección de objetos, segmentación semántica y segmentación de instancias (Tian et al., 2021). El objetivo principal de la clasificación de imágenes es predecir un conjunto de etiquetas para caracterizar el contenido de una imagen de entrada. La detección de objetos se basa en la clasificación de imágenes y permite localizar los objetos en una imagen por medio de un cuadro delimitador de coordenadas (x, y) para cada objeto y una etiqueta de clase para cada cuadro (Figura 6).

Figura 6 – Ejemplo de detección de objetos.



Fuente: Obtenida de (EVERINGHAM; WINN, 2012)

Los algoritmos de segmentación semántica asocian cada píxel de una imagen de entrada a una etiqueta de clase, incluyendo una etiqueta de clase para el fondo o *background*. A pesar de que los algoritmos de segmentación semántica son capaces de etiquetar cada objeto en una imagen, estos algoritmos no pueden diferenciar objetos de una misma clase, esto representa una problemática, ya que, cuando dos objetos de una misma clase se superponen en una imagen no se consigue definir los límites entre un objeto y otro (TIAN *et al.*, 2021). La Figura 7 ejemplifica este hecho, pues observe que, aunque hay dos objetos en la imagen, ambos pertenecientes a la etiqueta bus. La segmentación semántica no consigue diferenciar entre ellos.

Figura 7 – Segmentación semántica



Fuente: Adaptada de (EVERINGHAM; WINN, 2012)

Por otro lado, la segmentación de instancias determina una máscara (Mask) para cada objeto en una imagen, incluso si los objetos son de la misma etiqueta de

clase, de esta forma consigue definir adecuadamente los contornos, solucionando así el problema de la segmentación semántica (Figura 8). Se puede considerar que realiza las tareas de detección de objetos y segmentación semántica al mismo tiempo, lo que le otorga un importante valor, pues el problema de lidiar con múltiples objetos superpuestos y fondos complejos es determinante para el desarrollo de la conducción inteligente, detección por teledetección, y salud médica. (TIAN *et al.*, 2021)

Figura 8 – Segmentación de instancias



Fuente: Adaptada de (EVERINGHAM; WINN, 2012)

2.3 SOBRE INTELIGENCIA ARTIFICIAL

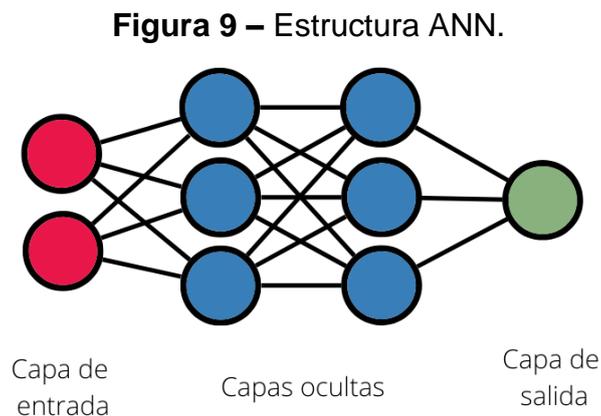
En este tópico describimos los fundamentos teóricos sobre redes neuronales artificiales y técnicas de ML que serán utilizados durante la metodología de este trabajo para la segmentación y clasificación de las imágenes IR con el objetivo de detectar fallas en módulos PV. Para ello, haremos una descripción sencilla y ejemplificada sobre las operaciones matemáticas involucradas en el proceso de aprendizaje de una inteligencia artificial.

2.3.1 Redes Neuronales Artificiales

Una red neuronal artificial (*Artificial Neural Networks* - ANN) es un modelo computacional inspirado en el funcionamiento de las neuronas del cerebro (RUSSELL S. J.; NORVIG P., 2015). Una ANN está compuesta por una serie de neuronas o nodos conectadas entre sí y cada neurona se activa o se inhibe en función de la señal que recibe de las otras neuronas. Las neuronas se agrupan en capas, y cada capa está dedicada a

una tarea específica de procesamiento de la información. Por ejemplo, la capa más simple en cuanto estructura es la capa *Fully Connected (FC)*. Estas capas conectan cada uno de los nodos de la entrada de la capa, con cada uno de los nodos de la salida de la capa.

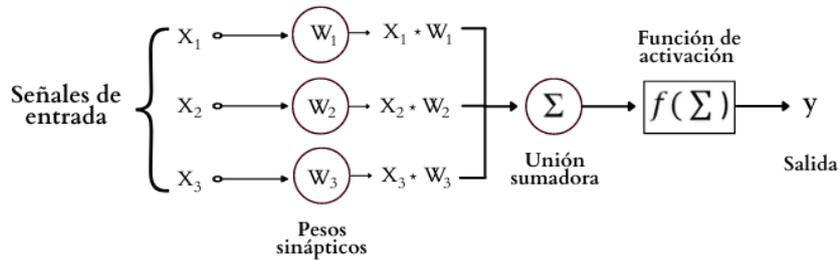
Una ANN se compone de tres tipos de capas (Figura 9): una capa de entrada, que toma alguna representación numérica de los datos; capas ocultas que realizan transformaciones en los datos que normalmente no son lineales; y una capa de salida, que produce una predicción (WARING *et al.*, 2020).



Fuente: El autor, 2022.

Existen varios tipos de redes neuronales, entre ellas las redes *feed-forward*, una red progresiva tal que la salida de una neurona se conecta con otra neurona de la siguiente capa siempre en la dirección izquierda/derecha (como la ANN de la Figura 9). La Figura 10, muestra una red neuronal artificial simple conocida como perceptrón, el cual toma un vector de datos como entrada y lo multiplica por un vector de pesos. Posteriormente, los valores ponderados por los pesos son sumados y sobre este valor es aplicada una función de activación. La función de activación determina si el perceptrón "se activa" o no. Si el perceptrón se activa, devuelve una salida con valor 1, caso contrario, devuelve una salida con valor 0. Los valores del vector de pesos pueden ser ajustado para que el perceptrón aprenda cómo clasificar los datos de entrada. Para hacer esto, se proporciona un conjunto de datos de entrenamiento etiquetado. Dichas etiquetas pueden ser valores 0 o 1. El perceptrón usa estos datos para evaluar su capacidad de clasificar correctamente la entrada y posteriormente, con la ayuda de algunas métricas de error, ajustar sus pesos, de modo que, pueda mejorar su precisión. Una ANN formada por varios unidades que replican el comportamiento del perceptrón forman una red llamada perceptrón multicapa.

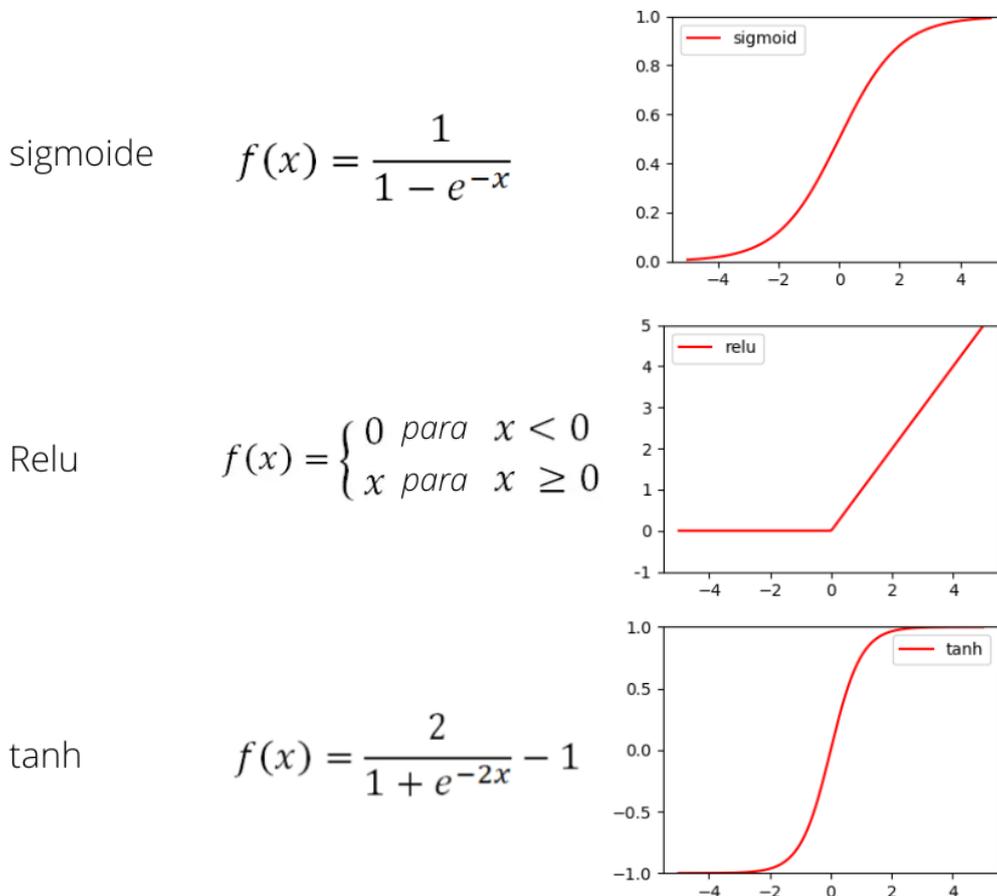
Figura 10– Ejemplo de un perceptrón.



Fuente: El autor, 2022.

Es importante notar que, la función de activación cumple un papel fundamental en la red, ya que determina si la información recibida por una neurona es propagada para la siguiente capa de la red (JAIN, 2019). Existen varios tipos de función de activación, dentro de las más comunes son: sigmoide, ReLu (*Rectifier Linear Unit*) y tanh (SHARMA, 2017) (Figura 11).

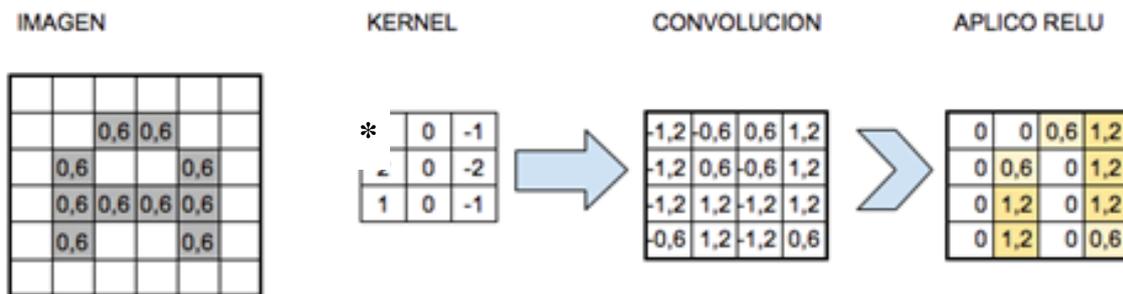
Figura 11 – Principales funciones de activación.



Fuente: El autor, 2022.

Para dar mayor claridad de lo que hace la función de activación, observe la Figura 12, que ejemplifica la aplicación de una función ReLu después de aplicada una operación de convolución. Note que, la aplicación de la función de activación no afecta la dimensión de la matriz obtenida por la convolución.

Figura 12 – Aplicación de la función de activación.



Fuente: Obtenida de (aprendemachinelearning, 2018).

Otra función de activación de gran interés es la función de activación Softmax, también conocida como Función Exponencial Normalizada.

$$\text{softmax}(x_j) = \frac{e^{x_j}}{\sum_{k=1}^k e^{x_k}} \quad (1)$$

Esta función toma vectores de números reales como entradas, y los normaliza en una distribución de probabilidad proporcional a los exponentes de los números de entrada. Así, la aplicación de Softmax, asegura que la suma de todas las probabilidades asignadas sea igual a 1 (TURING, 2022). Esta función de activación es de gran importancia para en la resolución de problemas de clasificación multiclase porque entrega una distribución de probabilidad, la cual es necesaria para estimar la clase a la que pertenece un nuevo dato.

2.3.2 Conceptos Relevantes A Optimización De Redes Neuronales

2.3.2.1 Descenso del gradiente

El descenso del gradiente es un algoritmo de optimización utilizado para minimizar una función de costo. Se basa en el cálculo de la derivada de la función de costo con respecto a los parámetros del modelo. Los parámetros se actualizan en la dirección del gradiente descendente para minimizar la función de costo.

2.3.2.2 Función de costo

La función de costo o *Loss Function* es una medida de qué tan bien se está ajustando el modelo a los datos. Se trata de una función de error que se minimiza durante el entrenamiento del modelo. En este contexto, *Cross Entropy Loss*, que matemáticamente se expresa en (2), es la función de pérdida más comúnmente utilizada en problemas de clasificación (BROWNLEE, 2020). Esta función disminuye a medida que la probabilidad predicha converge a la etiqueta real. Por lo tanto, mide el rendimiento de un modelo de clasificación cuya salida prevista es un valor de probabilidad entre 0 y 1.

$$L = -\frac{1}{m} \sum_{i=1}^m y_i * \log(\hat{y}_i) \quad (2)$$

Es importante notar que, para modelos de clasificación, el vector y_i es de tipo *one-hot*, un vector que contiene un solo elemento igual a 1 y todos los demás elementos iguales a 0. Los vectores *one-hot* a menudo se usan para representar datos categóricos, como etiquetas. Por ejemplo, si tenemos un conjunto de datos con tres etiquetas, A, B y C, podemos representar cada etiqueta usando un vector *one-hot*, tal que, la etiqueta A se representaría como [1, 0, 0], la etiqueta B se representaría como [0, 1, 0] y la etiqueta C se representaría como [0, 0, 1]. De esta forma y_i actúa como una delta de Kronecker en (2), en el sentido que, el sumatorio solo sumará las probabilidades asignadas a la etiqueta real del objeto a clasificar. Así, *Loss* retorna una media de las predicciones realizadas para el conjunto de datos evaluadas por el modelo.

2.3.2.3 Iteraciones, tamaño de lote y épocas

Una iteración es un ciclo de aprendizaje que incluye una pasada de todos los datos de entrada y una actualización de los pesos. El tamaño del lote o *batch size* es el número de ejemplos de entrenamiento que se usan en cada iteración, generalmente se establece como potencias de 2 (BROWNLEE, 2018), esto debido a la relación que existe con la cantidad de memoria asignada para almacenar los datos. Las épocas son el número de iteraciones completadas y necesarias para pasar por todo el conjunto de datos. Es decir, si tenemos 1000 datos y establecemos un *batch size* de 200, se realizarán 5 iteraciones por cada época de entrenamiento.

2.3.2.4 Optimizadores

Los optimizadores se utilizan para actualizar los parámetros del modelo en cada iteración del algoritmo de descenso del gradiente. Para esto se selecciona una tasa de aprendizaje que controla qué tanto cambian los pesos en cada iteración. Los optimizadores más comunes son el descenso de gradiente estocástico (SGD) y el descenso de gradiente de momento (SGDM). SGD se basa en el cálculo de la derivada de la función de costo con respecto a cada parámetro del modelo. Los pesos se actualizan en la dirección opuesta al gradiente con el fin de minimizar la función.

2.3.2.5. Learning Rate

La tasa de aprendizaje o *Learning Rate* es un hiperparámetro importante en el aprendizaje profundo. Controla cuánto se debe cambiar el modelo en respuesta al error estimado cada vez que se actualizan los pesos del modelo. Si la tasa de aprendizaje es demasiado alta, el modelo podría sobrepasar el objetivo y no converger. Si la tasa de aprendizaje es demasiado baja, el modelo puede tardar demasiado en converger (ZULKIFLI, 2018).

Existen diferentes estrategias para adaptar la tasa de aprendizaje durante el entrenamiento. Un enfoque popular es utilizar una tasa de aprendizaje más pequeña para las primeras épocas y luego aumentarla gradualmente a medida que avanza el entrenamiento. En general, una buena regla general es comenzar con una pequeña tasa de aprendizaje y luego aumentarla si el entrenamiento no está convergiendo.

2.3.2.6 Backpropagation

Propagación hacia atrás de errores o *Backward propagation of errors* (*Backpropagation*) es un método de aprendizaje automático para ANN. Se llama así porque el algoritmo “propaga” el error de la salida de la red hacia atrás a través de las capas de la red, de tal forma que los pesos de las neuronas se puedan ajustar para minimizar el error. El proceso, de forma resumida, se ejecuta en dos etapas. El primer proceso *Feed-Forward*, en el cual se establece un valor en la capa de entrada, luego, es ejecutado el proceso de cálculo entre las capas ocultas de la red hasta que la respuesta se reproduce en la capa de salida (GRÜBLER, 2018). Después de finalizado este proceso, el valor obtenido en la salida se compara con el valor deseado y es calculada la diferencia entre estos y definida como el error de la red. Este error será propagado hacia atrás en la red, a través de cálculos de diferenciación parcial (NIELSEN, 2019), lo que causará un cambio en los pesos sinápticos de la red, cuya variación dependerá del *Learning Rate* establecido, y es a través de este proceso que ocurre el aprendizaje.

2.3.2.7. Media de la precisión promedio

La media de la precisión promedio (mAP) es una métrica que mide el rendimiento de un algoritmo de detección de objetos en un set de datos de imagen. Esta métrica se calcula comparando la precisión y el rendimiento de un algoritmo de detección de objetos contra un conjunto de datos de referencia. El mAP se calcula promediando la precisión de detección para cada imagen en el conjunto de datos (YOHANANDAN, 2020).

2.3.2.8 Sobreajuste

En general, el sobreajuste o *overfitting* es un problema común en el aprendizaje automático. Se produce cuando un modelo aprende los detalles y el ruido de los datos de entrenamiento de tal manera que afecta negativamente el rendimiento del modelo en nuevos datos (IBM, 2021). Esto significa que el modelo se comporta bien en los datos de entrenamiento, pero no generaliza bien a los nuevos datos. Existen varias estrategias para evitar el sobreajuste, entre ellas:

- Parada temprana: Detener el entrenamiento del modelo antes de que este aprenda el ruido dentro de los datos.
- Aumento de datos: Inyectar más datos al modelo para hacer que este sea más estable.
- Selección de características: Identificar las características más importantes dentro de los datos de entrenamiento y luego eliminar las irrelevantes o redundantes.
- Regularización: Aplicar una "penalización" a los parámetros de entrada con los coeficientes más grandes, lo que posteriormente limita la cantidad de varianza en el modelo.
- Métodos de conjunto: Los métodos de aprendizaje conjunto se componen de un conjunto de clasificadores, por ejemplo, árboles de decisión, y sus predicciones se agregan para identificar el resultado más popular.

2.3.2.9. Matriz de confusión

La matriz de confusión es una herramienta de evaluación de rendimiento de modelos de aprendizaje automático que se puede usar para ver qué tan bien se está realizando un modelo. Se puede usar para evaluar cualquier tipo de modelo de aprendizaje, pero es más comúnmente usada para evaluar modelos de clasificación (NARKHEDE, 2018). La matriz de confusión contiene información sobre los verdaderos positivos (TP), los verdaderos negativos (TN), los falsos positivos (FP) y los falsos negativos (FN). Esta información se puede usar para calcular la precisión, la exhaustividad, el F1-score y el soporte.

- Precisión: La precisión es el número de veces que el modelo se ha clasificado correctamente dividido por el número total de veces que se ha utilizado el modelo.
- Exhaustividad: La exhaustividad es el número de veces que el modelo ha clasificado correctamente un ejemplo dividido por el número total de ejemplos que podrían haber sido clasificados correctamente.

- F1-score: El F1-score es una medida de precisión y exhaustividad. Se calcula dividiendo la precisión por la exhaustividad.
- Soporte: El soporte es el número de ejemplos en el conjunto de datos que se clasifican como positivos o negativos.

2.3.4 Redes Neuronales Convolucionales

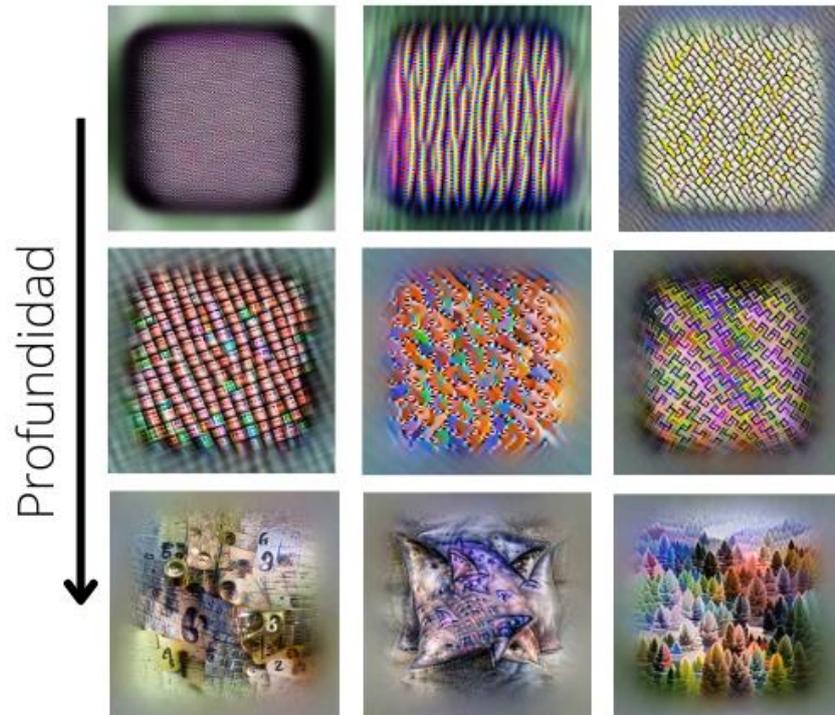
Una red neuronal convolucional (*Convolutional Neural Network* - CNN) es un tipo de ANN utilizada en tareas de ML. Las CNN fueron introducidas por primera vez por Fu Jie Zhu y Lecun Yam en los años 80 y están inspiradas en la forma en que el cerebro humano procesa la información visual. Las CNN son similares a las redes neuronales tradicionales, pues, están compuestas por una serie de capas ocultas, pero también tienen una serie de capas convolucionales que le permite extraer características de la imagen de entrada. Por esta razón se utilizan a menudo para tareas de segmentación de imágenes, como identificar objetos en una imagen o clasificar los píxeles de una imagen en diferentes clases. La función de activación más utilizada para este tipo de redes neuronales es la función ReLu.

Las CNN utilizan técnicas de DL para aprender los detalles de una imagen a partir de una representación más abstracta. Esto se parece mucho a cómo funciona el cerebro humano, ya que, este no procesa las imágenes de forma lineal. En su lugar, utiliza varias capas de neuronas para extraer características de la imagen a medida que avanza hacia capas más abstractas. Las CNN utilizan una técnica similar para aprender las características de una imagen. Cada capa de la CNN aprende un conjunto de características más abstractas de la imagen a medida que avanza hacia la capa siguiente, esto es, aumenta la profundidad. Esto puede ser fácilmente comprendido al examinar el trabajo realizado por (OLAH *et al.*, 2017) accesible en [distill.pub](https://distill.pub/2017/feature-visualization/)¹ y la herramienta Microscope² de OpenIA, en donde se puede explorar la visualización de características neuronales y cómo las CNN construyen su comprensión de las imágenes de una forma interactiva. En la Figura 13 se pueden observar algunos ejemplos de dicho comportamiento.

¹ <https://distill.pub/2017/feature-visualization/>

² <https://microscope.openai.com/models>

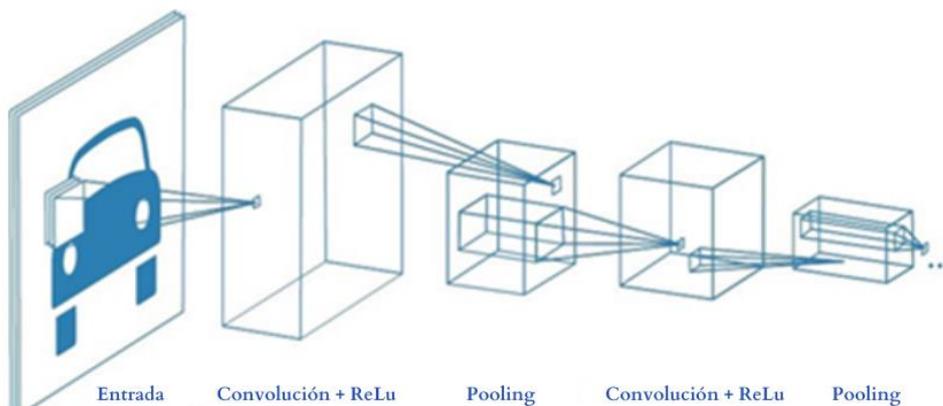
Figura 13 – Interpretabilidad de las imágenes por una CNN en función de la profundidad.



Fuente: El autor, 2022.

La Figura 14 permite obtener una visión de lo que sucede cuando una CNN es aplicada sobre una imagen de entrada. Observe como al aplicar estas operaciones de convolución de forma consecutiva, se puede notar como las dimensiones de la matriz irán disminuyendo, al mismo tiempo que, la cantidad canales irán aumentando, según la cantidad de filtros usados para cada convolución.

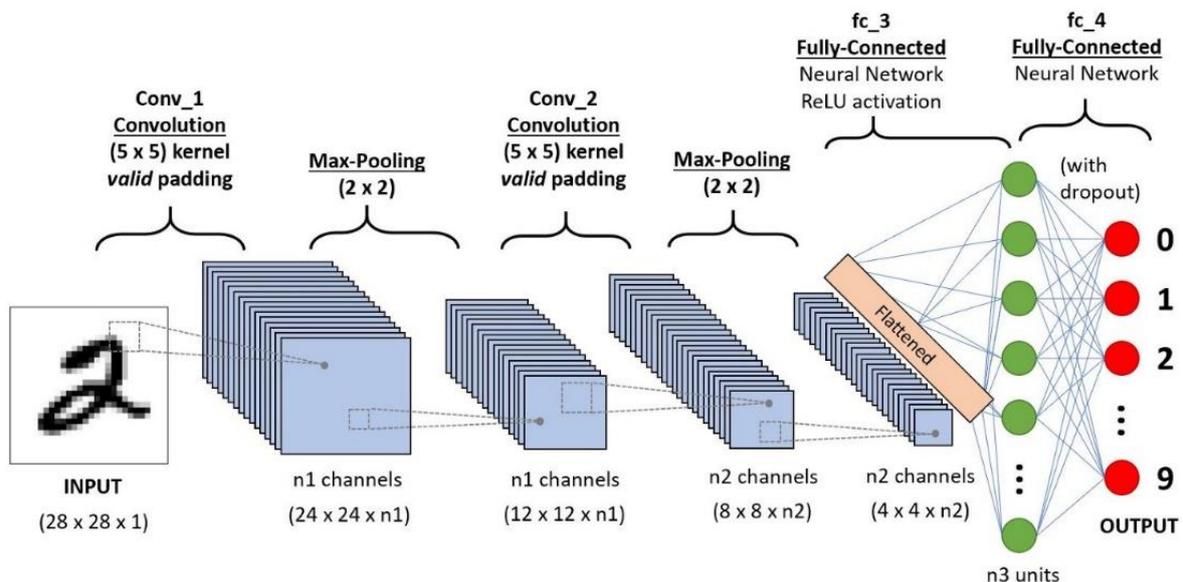
Figura 14 – Comportamiento de las dimensiones y los canales durante la extracción de características por una CNN.



Fuente: Adaptada de (Saha, 2018).

Ahora, para que esta información pueda usada para una tarea de ML, es común agregar una capa FC, como una forma “barata” de aprender combinaciones no lineales de las características de alto nivel representadas por la salida de la capa convolucional. La Figura 15 muestra un ejemplo de una arquitectura CNN para la clasificación de números. Para alimentar la capa FC es necesario hacer una operación de aplanamiento o *flattening*. La operación de aplanar una matriz en ML consiste en transformar una matriz en un vector, es decir, en una matriz de una sola columna o una matriz de una sola fila. Posteriormente a la salida de la capa FC se le suele aplicar la función de activación Softmax para realizar la clasificación de imágenes.

Figura 15 – Secuencia de una CNN para clasificar dígitos.



Fuente: Obtenida de (SAHA, 2018)

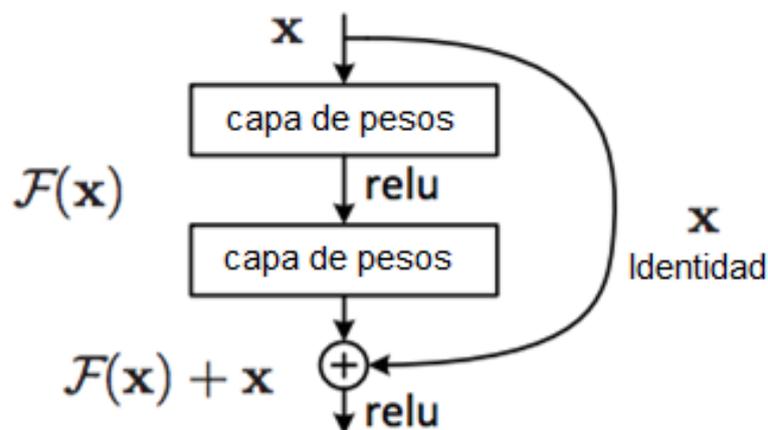
2.3.4.1 ResNet

Residual neural network (ResNet) es una CNN desarrollada por Microsoft Research e introducido por primera vez por (HE *et al.*, 2015) con el objetivo de mejorar el rendimiento de las redes neuronales profundas. Se diseñó para superar el problema de la pérdida de rendimiento en las redes neuronales profundas, lo que se conoce como el problema de la pérdida de información o desvanecimiento del gradiente.

Las ResNet son similares a las CNN, pues utilizan una serie de capas para extraer características de los datos de entrada. Sin embargo, las ResNet introducen una

novedosa forma de conectar estas capas llamada *residual learning*. La Figura 16 muestra la idea básica detrás de este abordaje llamado “bloque residual”. Un bloque residual consta de dos partes: una vía principal y una vía de acceso directo. La ruta principal contiene varias capas, mientras que la ruta de acceso directo contiene una sola capa, la identidad. Las capas en la ruta principal están conectadas en serie. La vía de acceso directo está conectada al camino principal a través de una conexión de salto. La conexión de salto permite que el gradiente fluya directamente desde la vía de acceso directo a la vía principal, sin pasar por las capas de la vía principal. El término "residual" se refiere al hecho de que la entrada al bloque es también la salida del bloque. Las capas convolucionales en un bloque residual suelen ir seguidas de una capa de normalización y una capa de activación ReLU. Esto permite que la red aprenda mejor los patrones de los datos de entrada y, como resultado, mejora el rendimiento de la red (HE *et al.*, 2015).

Figura 16 – *Residual learning*: Bloque residual.

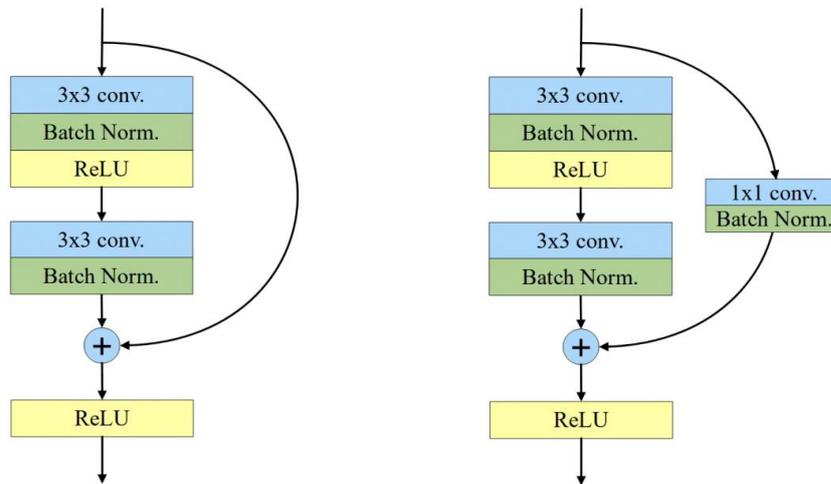


Fuente: Adaptada de (HE *et al.*, 2015)

Por ejemplo, note que, si la salida de las capas de la ruta principal fuese cero, esto es $F(x) \rightarrow 0$, la información que fluye a través del bloque no se pierde, puesto que es conservada por la ruta de atajo.

Las principales variantes de ResNet, son ResNet18, ResNet50 y ResNet101, en donde el número que le acompaña se refiere a la cantidad de capas que contiene la red. La Figura 17 permite observar dos estructuras de bloque residual implementadas en ResNet 18 y 34.

Figura 17 – Residual learning: Bloques residuales ResNet18 - ResNet34.



Fuente: Adaptada de (ÖZDEMİR, 2022)

Las redes ResNet han demostrado ser más precisas que las redes tradicionales en la detección de objetos (DAI *et al.*, 2016). Esto se debe a que las redes ResNet pueden aprender características más complejas de los objetos.

2.3.4.2 Mask R-CNN

Mask R-CNN es una red neuronal profunda para segmentación de instancias introducida por el equipo de inteligencia artificial de Facebook inc. en el 2017. Este algoritmo se basa en algoritmos de detección de objetos previamente desarrollados como R-CNN (GIRSHICK *et al.*, 2013), Fast R-CNN (GIRSHICK, 2015) y Faster R-CNN (REN *et al.*, 2015).

Mask R-CNN utiliza la arquitectura ResNet101 para extraer características de la imagen y una *Region proposal network* (RPN) (LIN *et al.*, 2016), una red neuronal que se utiliza en algoritmos de detección de objetos para generar regiones candidatas o de interés (*Region of interés – RoI*), con el fin de proponer cuadros delimitadores (*bounding boxes*) a los objetos candidatos (HE *et al.*, 2017). La salida de Mask R-CNN es una pequeña red totalmente conectada aplicada a cada RoI, que predice una máscara de segmentación para cada píxel (WENG, 2017).

La principal diferencia de Mask R-CNN con respecto a otras metodologías de detección y segmentación de objetos es que Mask R-CNN predice máscaras, un tipo de

datos de imagen que se utilizan para indicar la pertenencia de píxeles a objetos en una imagen, además de las propias *bounding boxes* (Figura 18). Esto permite una segmentación más precisa de los objetos, ya que se pueden identificar con mayor facilidad los bordes de los objetos.

Figura 18 – Ejemplo de detección con *Mask*: cuadros delimitadores y mascarar.



Fuente: Obtenida de (WALEED ABDULLA, 2017)

Mask R-CNN se ha utilizado con éxito para la segmentación de imágenes en aplicaciones como la detección de objetos en imágenes de satélite (ZHAO *et al.*, 2018), la detección de objetos en imágenes médicas (SHU *et al.*, 2020) y la detección de objetos en imágenes de video (CHEN *et al.*, 2020).

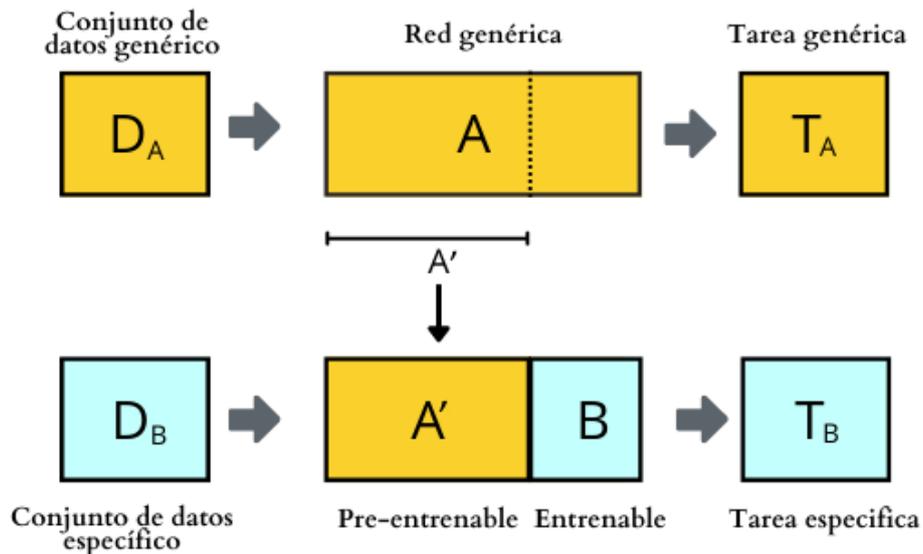
2.3.4 Transferencia De Conocimiento

La transferencia de conocimiento es un campo de la inteligencia artificial que se preocupa por el reusó de conocimiento aprendido por una ANN para solucionar nuevos problemas. El objetivo de la transferencia de conocimiento es aumentar la eficiencia del aprendizaje de la máquina al permitirle reutilizar el conocimiento aprendido previamente para resolver nuevos problemas (ZHU *et al.*, 2020). La Figura 19 permite visualizar el esquema de este procedimiento.

Por ejemplo, supongamos que una modelo de AI ha aprendido a reconocer imágenes de gatos. Si ahora se desea entrenar un modelo para identificar perros, la AI

puede utilizar el conocimiento aprendido sobre gatos para ayudarla a identificar si la nueva imagen contiene un gato o no. De esta manera, la AI está utilizando el conocimiento aprendido sobre gatos para solucionar un nuevo problema, lo que se denomina transferencia de conocimiento.

Figura 19 – Esquema general de la transferencia de conocimiento.



Fuente: El autor, 2022.

2.4 ESTADO DEL ARTE

Como mencionado anteriormente, la técnica aIRT ha sido usada por múltiples grupos de investigación para la inspección de los módulos PV, sin embargo, hay una serie de limitaciones que deben tenerse en cuenta al utilizar esta técnica. Una de ellas, la complejidad de analizar un gran volumen de datos cuando aplicada en granjas solares de grande porte, puesto que, los datos aumentan en gran proporción y tener un personal especializado para analizar esta información demanda recursos económicos, humanos y tiempo.

En este contexto, estudios recientes han demostrado que el uso de técnicas de aprendizaje profundo puede ser útil para el análisis de datos, no solo provenientes de la técnica aIRT, sino también de imágenes RGB de módulos o de plantas fotovoltaicas obtenidas por UAV o por satélite. El uso de estas técnicas ha sido explorado tanto para la detección de los módulos PV como para la identificación y clasificación de fallas presentes en los mismos. Ejemplo de ello es el trabajo desarrollado por (RICO ESPINOSA *et al.*, 2020), en el cual se hace uso de una CNN para segmentación semántica junto con un

clasificador para catalogar fallas en imágenes RGB de módulos fotovoltaicos y en cuyos resultado se obtuvo una precisión promedio del 75% cuando se clasifican 2 clases (con y sin anomalía) y 70% para 4 clases de estados.

Por otro lado, un trabajo desarrollado por (VEGA DÍAZ *et al.*, 2020) usa aprendizaje profundo para la detección de los módulos PV a partir de imágenes térmicas obtenidas por aIRT consiguiendo precisiones de hasta 99,6%, una efectividad tan alta como la obtenida con métodos clásicos para procesamiento de imágenes. Ya la solución presentada por (MORADI SIZKOUHI *et al.*, 2022) para la inspección de plantas fotovoltaicas a partir de un software que realiza automáticamente el monitoreo aéreo de las plantas fotovoltaicas, la planificación óptima de la trayectoria, el procesamiento de imágenes, el reconocimiento de patrones para la detección y el análisis de fallas en tiempo real consigue hasta un 93% de precisión.

En cuanto a detección y clasificación de fallas a partir de imágenes térmicas (DUNDERDALE *et al.*, 2020) aplica varios métodos de ML sobre un conjunto de alrededor de 800 imágenes y 5 clases para esta tarea, demostrando el buen rendimiento de VGG-16 y MobileNet, y alcanzando hasta un 89,5% de precisión.

Un trabajo más completo y robusto, es el presentado por (BOMMES *et al.*, 2021b) el cual usa la arquitectura ResNet50 para clasificar 10 tipos de anomalías obteniendo hasta un 90% de precisión. La clasificación se realiza sobre módulos previamente extraídos de imágenes térmicas obtenidas por aIRT, esto gracias a la detección de los módulos realizada con la implementación del algoritmo Mask R-CNN, para el cual se obtuvo una precisión promedio de hasta 90,01%.

3 DELINEAMIENTO METODOLOGICO

En esta sección se describe a detalle las bases de datos usadas, el preprocesamiento de imágenes realizado y las técnicas implementadas para el entrenamiento de los algoritmos de segmentación, basado en Mask R-CNN, y de clasificación de imágenes, basado en ResNet. Posteriormente se describen las configuraciones experimentales y las herramientas tecnológicas usadas.

3.1 BASES DE DATOS

En esta sección se presenta un análisis exploratorio de las bases de datos usadas para el entrenamiento de los algoritmos de segmentación y clasificación. También se describen las modificaciones que fueron necesarias para su uso como entrada de datos en los diferentes algoritmos.

3.1.1 Segmentación

Es importante resaltar que, para entrenar un algoritmo de segmentación como Mask R-CNN se debe suministrar un gran volumen de datos etiquetados. En el contexto de este trabajo, esto implica suministrar un conjunto de datos de imágenes en donde se identifica uno a uno cada panel PV en las imágenes de entrada como mostrado en la Figura 20.

Estas etiquetas deben estar acompañadas de un archivo de datos, generalmente un archivo JSON, un formato de texto sencillo para almacenar y transmitir datos estructurados, que contenga las informaciones relativas a la ubicación de cada punto que forma el contorno de la imagen. El etiquetado de imágenes se puede hacer de forma manual en plataformas de uso gratuito como MakeSense o Cvat.

Figura 20 – Etiketando módulos PV.



Fuente: El autor, 2022.

Debido a la complejidad de etiquetar una gran cantidad de imágenes, en este trabajo se optó por seguir la metodología propuesta por (AZEVEDO, 2021). Esta metodología consiste en usar dos bases de datos, llamadas *Background* y *Foreground*, para la construcción de un tercer conjunto de imágenes etiquetado, llamado “dataset sintético”. El procedimiento consiste en tomar una imagen del *Background* y “pegarle” de forma aleatoria varias imágenes contenidas en el *Foreground*, al mismo tiempo que son almacenadas las informaciones de la ubicación espacial relativa a las dimensiones de la imagen y una etiqueta de clase a la cual pertenece el objeto en un archivo JSON. Para esta etapa solo se usó la etiqueta “PVM” como referencia a *photovoltaic module*. De esta forma es posible obtener cientos de imágenes etiquetadas al ejecutar un Script de Python que tome una muestra aleatoria del *Foreground* y ejecute este procedimiento sobre imágenes aleatorias del *Background*, que servirán como entrada al algoritmo Mask, lo que le permitirá aprender a reconocer qué es un panel PV.

Las imágenes del *Background* fueron obtenidas de sitio web FLIR³, empresa líder en el desarrollo de sistema de generación de imágenes termográficas, que proporciona un conjunto de datos gratuito de espectro térmico y visible totalmente etiquetadas para el desarrollo de sistemas de detección de objetos utilizando redes neuronales convolucionales (CNN). Para el objetivo de este trabajo solo se usó una parte del conjunto de datos de imágenes térmicas contenidas en la carpeta ‘train’,

³ <https://www.flir.com/oem/adas/adas-dataset-form/>

específicamente 8.862 imágenes en formato JPEG con profundidad de 8 bits de 640 x 512 píxeles, tomadas con una cámara térmica Teledyne FLIR Tau 2 640x512, 13mm f/1.0 (HFOV 45°, VFOV 37°). La Figura 21 presenta una muestra de las imágenes de este conjunto de datos.

Figura 21 – Muestra de imágenes contenidas en el *Background*.

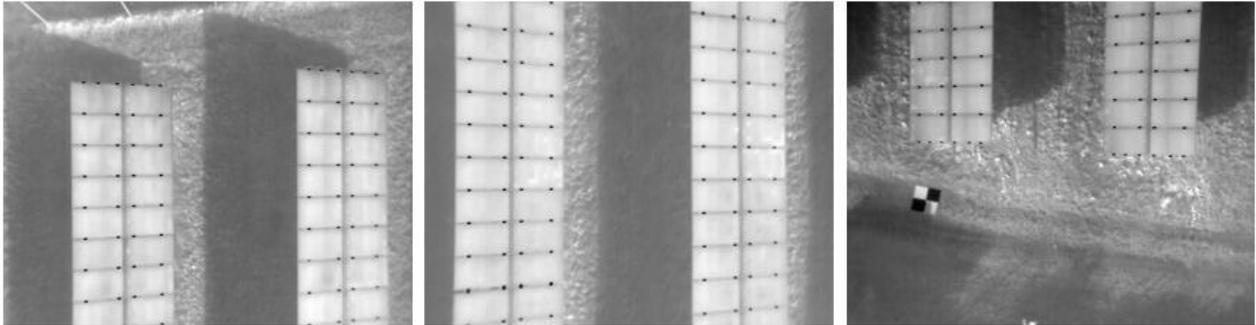


Fuente: El autor, 2022.

El *Foreground* consiste en un conjunto de imágenes de módulos PV individuales en formato PNG de dimensiones variables, obtenido a través de la edición de imágenes de otro conjunto de datos de imágenes aéreas de granjas solares usando el editor GIMP, a partir de las cuales se realizaron recortes de 300 módulos PV, añadiéndole un fondo transparente y guardándolo en formato PNG.

Las imágenes fuente del *Foreground* fueron obtenidas de la página web de documentación del programa PV HAWK (BOMMES, 2021), el cual hace parte de un proyecto de investigación que plantea un programa para la inspección automatizada de plantas fotovoltaicas a gran escala mediante videos IRT adquiridos por un dron y que se encuentra disponible para la investigación en el desarrollo de algoritmos de aprendizaje automático para la detección de defectos o la predicción de potencia. Este *dataset* cuenta con 2.541 imágenes en formato TIFF, para las cuales fue necesario realizar un preprocesamiento usando la función Autotune de Adobe PhotoShop siendo posteriormente guardadas en formato jpeg. La Figura 22 presenta una muestra de las imágenes del conjunto de datos fuente después de su preprocesamiento y la Figura 23 las imágenes extraídas para la construcción del *Foreground*. Adicionalmente se extrajeron algunos frames de videos IR disponibles en el mismo *site*. Estos videos contienen tomas más cercanas y realizan un recorrido tanto horizontal como vertical.

Figura 22 – Muestra de imágenes del conjunto de datos PV HAWK.



Fuente: El autor, 2022.

Figura 23 – Muestra de imágenes contenidas en el *Foreground*.



Fuente: El autor, 2022.

Posteriormente, se procedió a realizar la “combinación” de estos *dataset* para la obtención del *dataset* sintético, como descrito anteriormente, un conjunto de datos de gran utilidad, construido a partir de pequeños conjuntos de datos, que permite crear miles de ejemplos que servirán como *input* para el entrenamiento de modelos de DL. La Figura 24 muestra algunos ejemplos de las imágenes obtenidas en el *dataset* sintético.

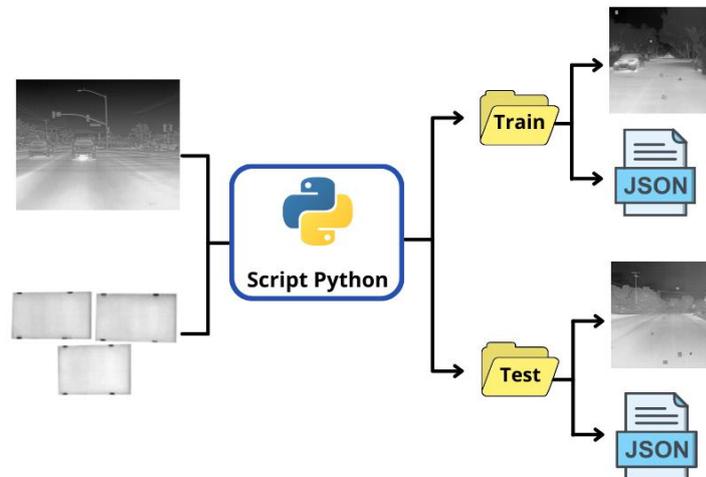
Figura 24 – Muestra de imágenes del dataset sintético.



Fuente: El autor, 2022.

El script de Python ejecutado para esta tarea fue configurado para generar dos carpetas, una de entrenamiento (*Train*) y otra de prueba (*Test*). Cada carpeta conteniendo el conjunto de imágenes sintético y un archivo JSON con los metadatos de las etiquetas. A partir de las imágenes de entrada se establecieron los parámetros para generar 40.000 y 10.000 imágenes en el dataset de *Train* y *Test* respectivamente. la Figura 25 resume el procedimiento realizado en esta etapa del trabajo.

Figura 25 – Esquema del proceso de obtención del dataset sintético.



Fuente: El autor, 2022.

3.1.2 Clasificación

Para el entrenamiento del modelo de clasificación ResNet se usó un conjunto de imágenes de módulos PV obtenido del repositorio de GitHub de Raptor Maps Inc. (RAPTOR MAPS, 2020) el cual incluye 20.000 imágenes con dos clases de subconjuntos en proporciones iguales: con anomalía y sin anomalía. Este conjunto de datos fue publicado por (MILLENDORF *et al.*, 2020) de Raptor Maps Inc. para la comunidad de investigación y fue obtenido por drones pilotados y sistemas aéreos no tripulados utilizando cámaras IR de onda media y onda larga (3-13.5 μm). La obtención de las imágenes se realizó de acuerdo con la norma IEC TS 62446–3:2017. Cada imagen IR tiene un tamaño de $40 \times 24 \times 1$, en donde la escala de grises representa los valores de temperatura, y la resolución espacial varía de 3.0 a 15.0 cm / píxel, debido a las diferentes distancias de la cámara IR a los módulos solares. Dentro del subconjunto con anomalías existen 11 clases de anomalías con un desequilibrio entre las clases. Por lo tanto, en total hay 12 clases en

el conjunto total. El número de muestras de cada clase y su descripción se muestra en el Cuadro 2.

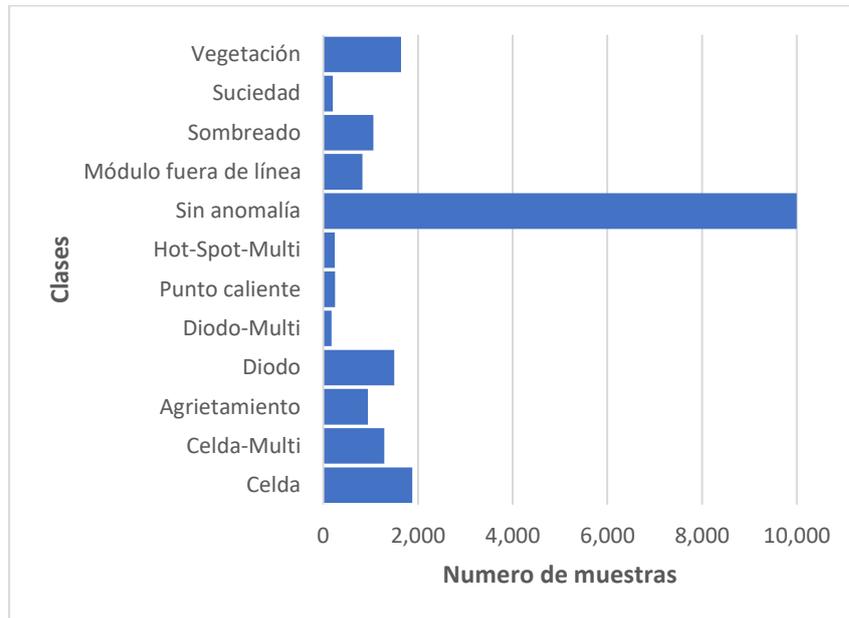
Cuadro 2 – Descripción de los módulos solares.

Módulos	No. Imágenes	Porcentaje del total [%]	Descripción
Celda	1,877	9.4	El punto caliente ocurrió en una sola celda con geometría cuadrada
Celda-Multi	1,288	6.4	El punto caliente se produjo en varias celdas con una geometría cuadrada en cada celda
Agrietamiento	940	4.7	Agrietamiento en la superficie del módulo
Diodo	1,499	7.5	Diodo de derivación activado, normalmente 1/3 del módulo
Diodo-Multi	175	0.9	Múltiples diodos de derivación activados, típicamente 2/3 del módulo
Punto caliente	249	1.2	El punto caliente en un módulo de película delgada.
Hot-Spot-Multi	246	1.2	Múltiples puntos calientes en un módulo de película delgada.
Sin anomalía	10,000	50	Módulo solar nominal
Módulo fuera de línea	827	4.1	Todo el módulo se calienta
Sombreado	1,056	5.3	Luz solar obstruida por vegetación, estructuras hechas por el hombre o filas adyacentes.
Suciedad	204	1	Suciedad, polvo u otros desechos en la superficie del módulo.
Vegetación	1,639	8.2	Paneles bloqueados por la vegetación.

Fuente: El autor, 2022.

El número de imágenes por clase varía de 175 a 10.000, y como se puede observar en la Figura 26 existe un desequilibrio de clases en el conjunto de datos lo cual representa un gran desafío para la técnica de aprendizaje profundo, pues lo ideal sería tener una representatividad igualitaria de todas las clases para evitar sesgos en el entrenamiento.

Figura 26 – Número de imágenes de cada clase en el conjunto de datos de módulos solares infrarrojos.



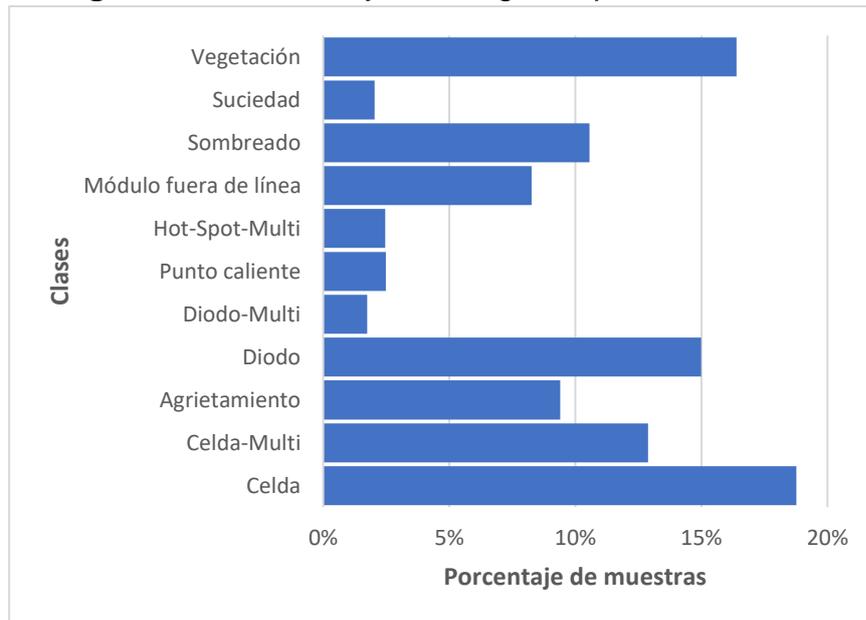
Fuente: El autor, 2022.

La Tabla 1 y la Figura 27 permiten explorar mejor las subclases dentro del conjunto de anomalías, destacándose las que poseen mayor representatividad dentro del conjunto.

Tabla 1 – Nuero de imágenes en las subclases de anomalías.

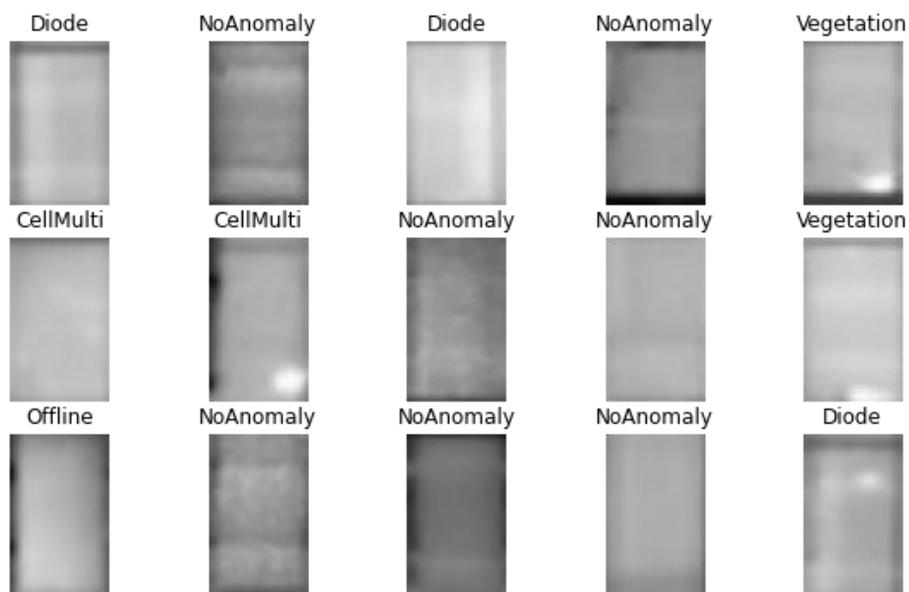
Módulos	No. Imágenes	Porcentaje del total [%]
Celda	1,877	19%
Celda-Multi	1,288	13%
Agrietamiento	940	9%
Diodo	1,499	15%
Diodo-Multi	175	2%
Punto caliente	249	2%
Hot-Spot-Multi	246	2%
Módulo fuera de línea	827	8%
Sombreado	1,056	11%
Suciedad	204	2%
Vegetación	1,639	16%

Fuente: El autor, 2022

Figura 27 – Porcentaje de imágenes por clase.

Fuente: El autor, 2022.

La Figura 28 permita visualizar el tipo de imágenes contenido en el dataset con su respectiva etiqueta de clase.

Figura 28 – Muestra del dataset para clasificación.

Fuente: El autor, 2022.

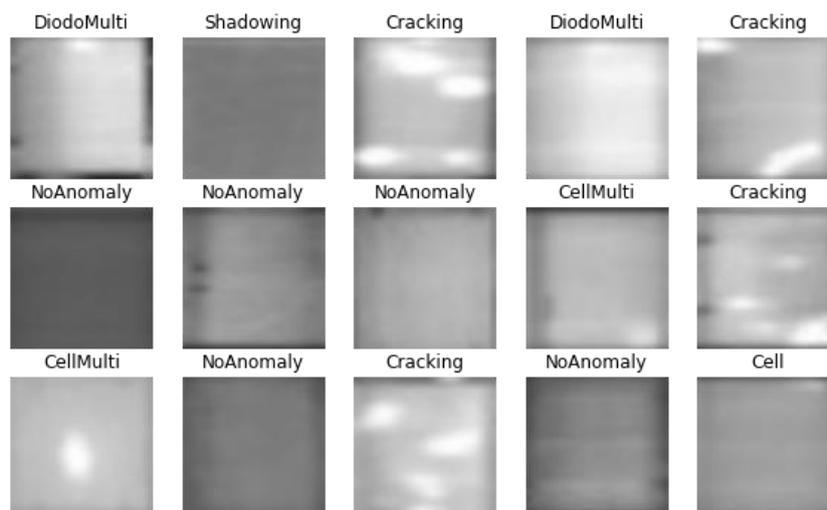
El dataset fue dividido en dos subconjuntos, entrenamiento (*Train*) y validación (*Test*), con una proporción de 80:20 para cada etiqueta de clase como mostrado en la tabla 2.

Tabla 2 – Numero de imágenes de Test y Train por subclase.

Módulos	No. Imágenes	Train	Test
Celda	1,877	1,502	375
Celda-Multi	1,288	1,030	258
Agrietamiento	940	752	188
Diodo	1,499	1,199	300
Diodo-Multi	175	140	35
Punto caliente	249	199	50
Hot-Spot-Multi	246	197	49
Sin anomalía	10,000	8,000	2,000
Módulo fuera de línea	827	662	165
Sombreado	1,056	845	211
Suciedad	204	163	41
Vegetación	1,639	1,311	328

Fuente: El autor, 2022.

Por ultimo las imágenes debieron ser normalizadas, dividiendo cada valor de su representación matricial por 255, y redimensionadas a 244x244 pixeles (Figura 29). El cambio de dimensiones es necesario una vez que se pretende usar estructuras ResNet con la técnica de transferencia de conocimiento.

Figura 29 – Imágenes normalizadas y redimensionadas

Fuente: El autor, 2022.

3.2 METODOLOGÍA

3.2.1 Segmentación de instancias

Para la identificación individual de un módulo PV se implementó la técnica de *ML* de segmentación de instancias, haciendo uso del algoritmo Mask R-CNN. Para el entrenamiento del algoritmo se usaron los scripts de código disponibles en el repositorio GitHub de Matterport⁴, en donde se encuentra una implementación de Mask R-CNN desarrollada en Python3 usando bibliotecas como Keras y TensorFlow. Adicionalmente se desarrolló la metodología propuesta por (CANU, 2021).

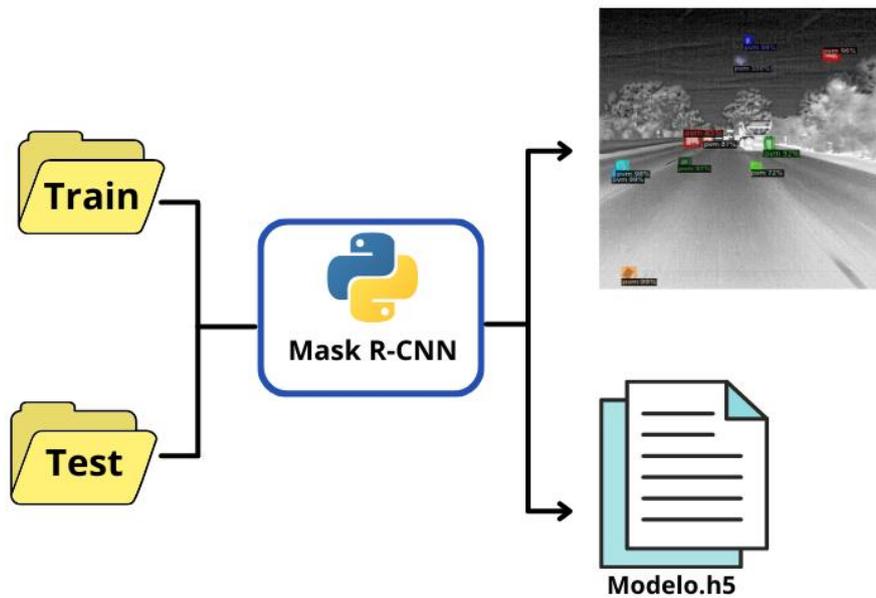
El modelo recibió como datos de entrada los conjuntos de datos provenientes del preprocesamiento de imágenes de segmentación como mostrado en la Figura 30. En este punto se presentó una gran dificultad en realizar un entrenamiento continuo, pues los tiempos de entrenamiento eran largos y el entorno de ejecución alojado de Colab tiene una limitación de tiempo de vida de 12 horas continuas, siempre y cuando el equipo este en línea.

Por esta razón se decidió ejecutar entrenamientos con una limitación de 10 épocas por sesión con 10 sesiones en total. Una vez terminado este proceso se guardaban los pesos para luego ser usados en la próxima ejecución, sin embargo, los resultados de precisión no fueron almacenados para estos procesos. Al finalizar el entrenamiento, los datos que almacenan los pesos de la red fueron almacenados en un archivo llamado “modelo.h5”. Los archivos h5 son un tipo de archivo de datos en Formato de Datos Jerárquicos (HDF) utilizado para almacenar grandes cantidades de datos numéricos, gráficos y de texto.

Una vez almacenadas las informaciones de aprendizaje, estas pueden ser cargadas para poner a prueba el modelo en datos no conocidos por la red.

⁴ https://github.com/matterport/Mask_RCNN

Figura 30 – Esquema del proceso de segmentación de paneles.



Fuente: El autor, 2022.

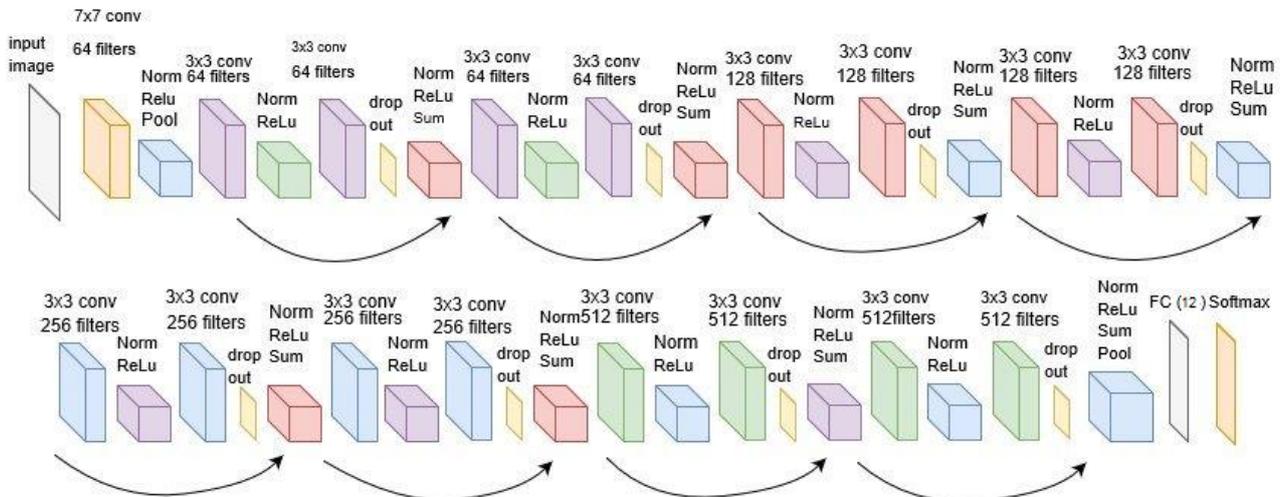
3.2.2 Clasificador de imágenes

Para la clasificación de imágenes este trabajo propone el entrenamiento de una CNN basada en la arquitectura ResNet, aplicando diferentes configuraciones y evaluando su impacto en cuanto al desempeño del modelo al predecir las clases para el conjunto de datos de entrenamiento.

Las arquitecturas evaluadas, ResNet18, ResNet34 y ResNet50, fueron importadas de la biblioteca PyTorch, y fueron modificadas sustituyendo la última capa del modelo (la cual originalmente está diseñada para clasificar 1000 clases) por una capa lineal FC y una SoftMax con 12 salidas, una para cada clase de nuestro conjunto de datos. El esquema de la Figura 31 muestra la arquitectura de ResNet18 modificada.

La generalización del modelo se evaluó en un conjunto de datos de validación, el cual es “invisible” al proceso de entrenamiento. Inicialmente se evaluó el impacto del aumento de las épocas de entrenamiento sobre la arquitectura más pequeña disponible, ResNet18, y posteriormente se compararon los resultados al aplicar la técnica de *Transfer Learning*.

Figura 31 – Arquitectura ResNet18 modificada.



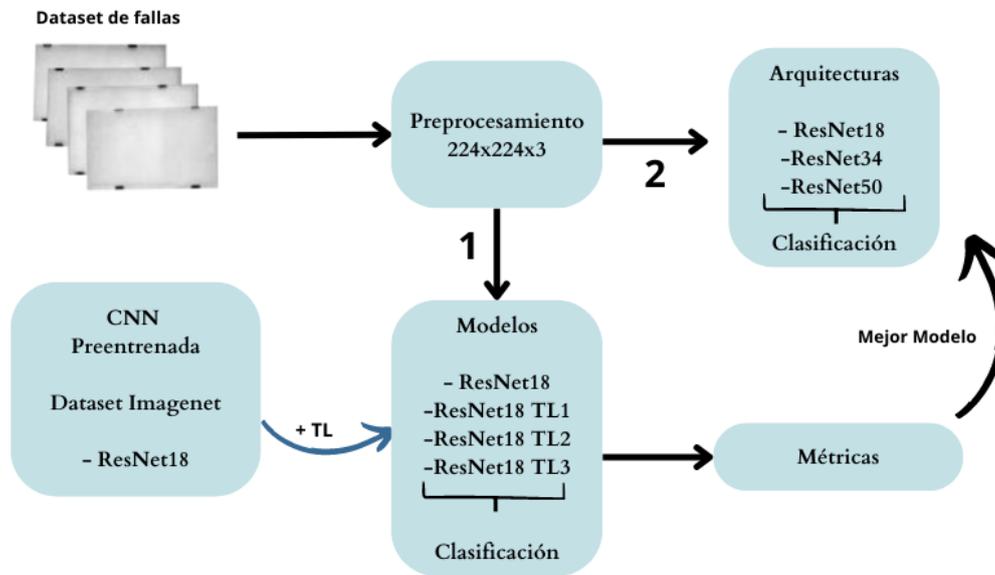
Fuente: Adaptada de (AL RABBANI ALIF *et al.*, 2017)

En la aplicación del método de *Transfer Learning*, se siguieron tres metodologías diferentes. La primera de ellas, TL1, consistía en cargar los pesos del modelo pre entrenado sobre el *database* ImageNet⁵, congelar los valores de los pesos de la red y permitir que la red entrenara solo la capa lineal. El segundo método, TL2, llamado *Fine Tuning*, consiste en permitir que, durante el entrenamiento, los pesos cargados sean alterados por el *Backpropagation*, de esta forma, el entrenamiento adaptaría los pesos del modelo para aprender a clasificar el conjunto de datos suministrado. El último método se puede entender como una mezcla de los dos métodos anteriormente descritos, en donde, el modelo recibe los pesos pre entrenados, estos son “congelados”, se entrena la última capa y posteriormente, después de un determinado número de épocas, se le permite al modelo reajustar los pesos cargados en el resto de la red.

Posteriormente, una vez evaluado el desempeño de estas metodologías usando ResNet18, se seleccionó la que obtuvo mejor desempeño y se procedió a entrenar arquitecturas mayores, como ResNet34 y ResNet50 (Figura 32). Por último, se seleccionó la arquitectura más robusta y se alteraron los valores del *LR* para determinar si influenciaba en el desempeño del modelo.

⁵ <https://www.image-net.org/>

Figura 32 –Procedimiento para selección del mejor modelo/Arquitectura.

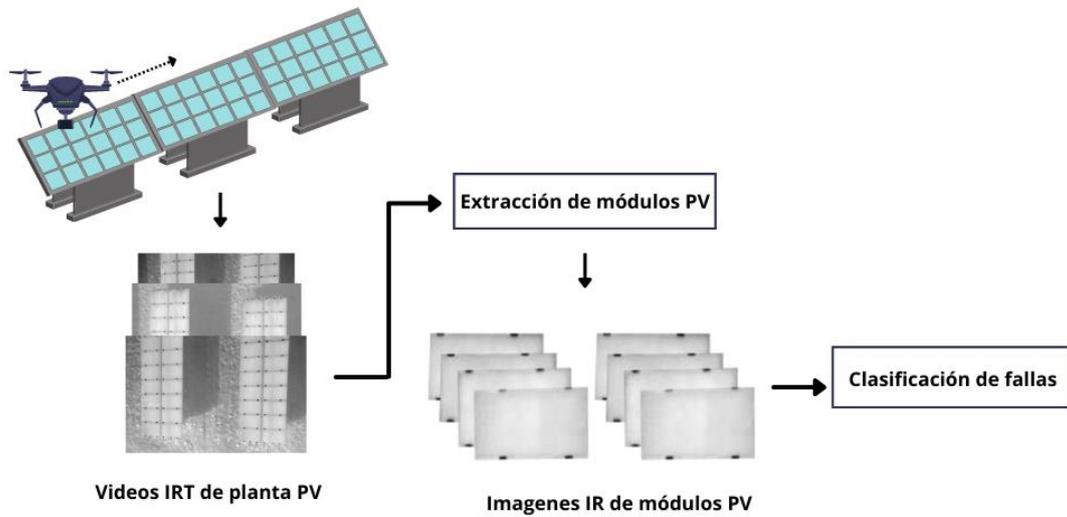


Fuente: El autor, 2022.

3.2.3 Identificar Fallas

Como procedimiento final, y aprovechando los resultados de los modelos anteriormente descritos, se desarrolló un algoritmo conjunto que permite segmentar y al mismo tiempo identificar las fallas sobre cada instancia hallada en un video de inspección de una granja fotovoltaica, así, la tarea del modelo entrenado por Mask R-CNN es identificar cada módulo en una *frame* del video de entrada, tal que, cada segmento identificado es recortado y el grupo de módulos extraídos del *frame* conformarán un *batch* que será la entrada de datos del modelo de clasificación, que a su vez retornará un vector con las respectivas clasificaciones predichas para este conjunto de imágenes. La Figura 33 resume el flujo de este proceso.

Figura 33 – Esquema del proceso identificación de fallas sobre un video.



Fuente: El autor, 2022.

3.4 HERRAMIENTAS Y TECNOLOGIAS

Para obtener mejores resultados de entrenamiento se usó el ambiente de programación Google Colab en su versión de paga Google Colab Pro, el cual permite cambiar el entorno de ejecución e implementar aceleración por hardware, como GPU o TPU, indispensables para mejorar el rendimiento al ejecutar operaciones en ML.

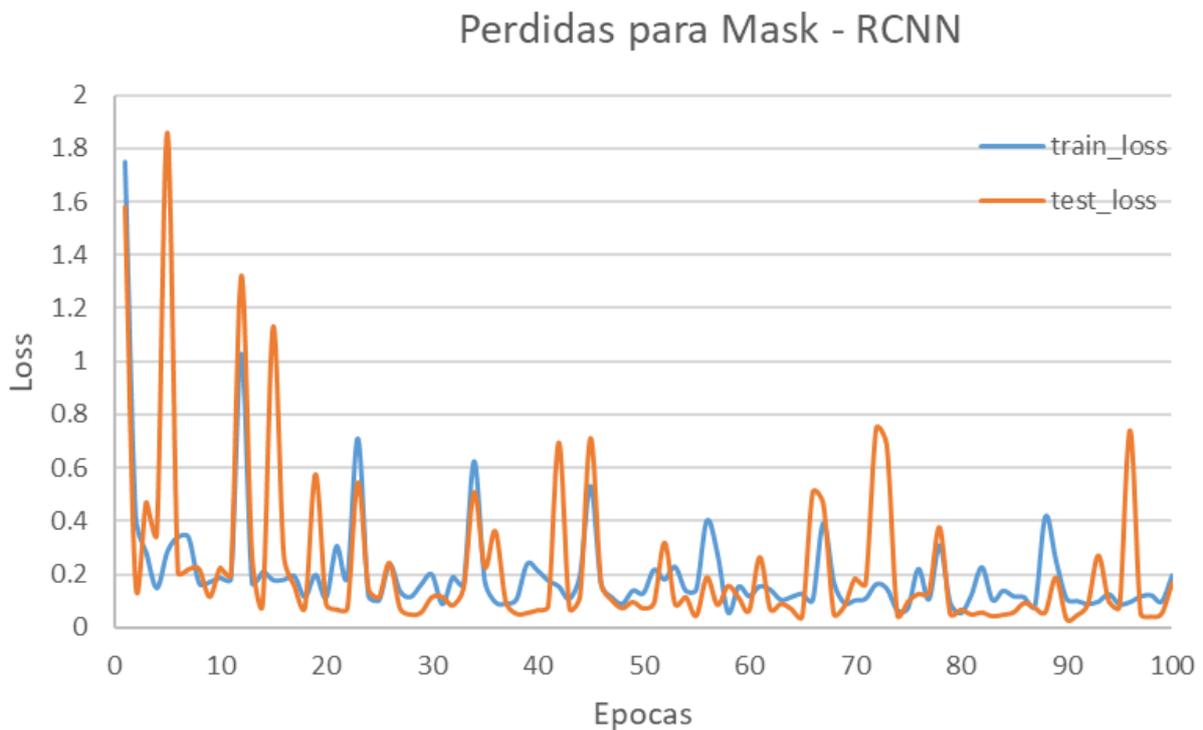
Los resultados, análisis y gráficos fueron elaborados con el auxilio de planillas electrónicas y lenguaje de programación Python.

4 ANÁLISIS Y RESULTADOS

4.1 SEGMENTACIÓN

Como descrito en el capítulo anterior, el algoritmo Mask R-CNN fue entrenado tomando como entrada las carpetas *Train* y *Test* obtenidas en el preprocesamiento de imágenes de segmentación. Se ejecuto un proceso de entrenamiento que consto de un total de 100 épocas. La Figura 34 muestra como la métrica de perdida se comporta con el aumento de las épocas.

Figura 34 – Perdidas del modelo.

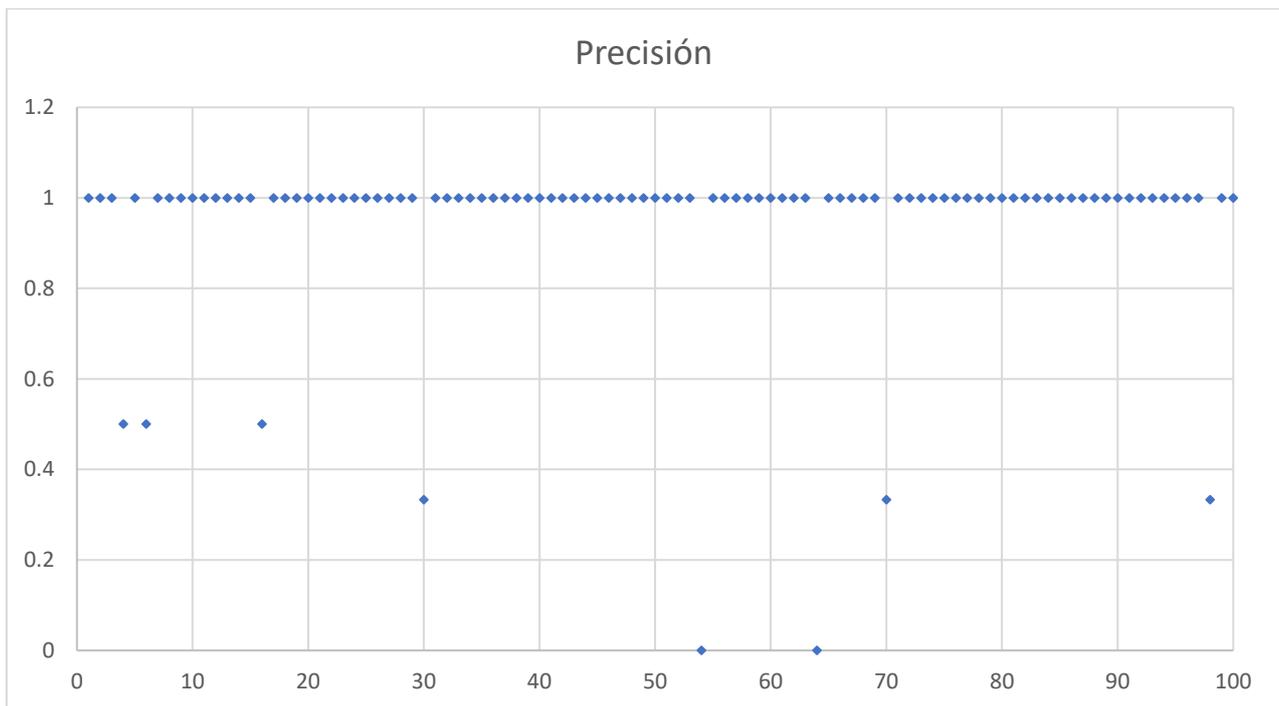


Fuente: El autor, 2022.

Como se puede apreciar, este modelo reduce el error rápidamente en las primeras épocas de entrenamiento, tanto para el conjunto de entrenamiento como para el de prueba, sin embargo, los mejores resultados para ambas métricas de perdida se observan alrededor de la época 60, a partir de la cual ya no se percibe un decaimiento en el valor de perdida.

Una evaluación del modelo sobre un subconjunto del conjunto *test* que contenía un total de 500 imágenes, mostro una media de la precisión promedio (mAP) de hasta un 96% en la tarea de segmentación de los módulos. La Figura 35 muestra los primeros 100 valores de mAP obtenidas sobre este subconjunto. Como se puede notar, solo en algunos pocos casos se presentaron fallas graves en la precisión al segmentar los módulos.

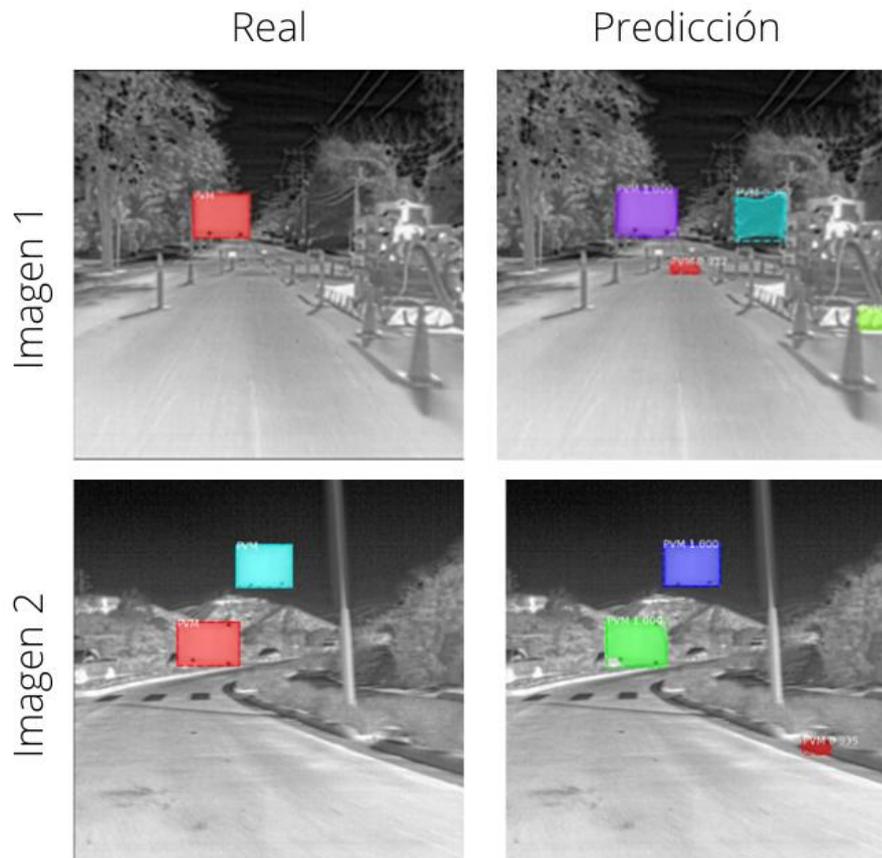
Figura 35 – Precisión del modelo sobre subconjunto de validación.



Fuente: El autor, 2022.

Las fallas en la segmentación se pueden interpretar mejor al observar la Figura 36, observe por ejemplo como en la imagen 1 solo existe un panel real que pudo ser identificado en su totalidad con una alta precisión (suponga 100%), sin embargo, en la imagen de detección se presentaron 3 falsas detecciones, cuyo valor de precisión es 0% para cada uno, podríamos entonces calcular el mPA como 1 entre 4 y nos daría un aproximado de 0.25. Esta métrica también arrojo una precisión 97,9% media sobre el conjunto *Test*.

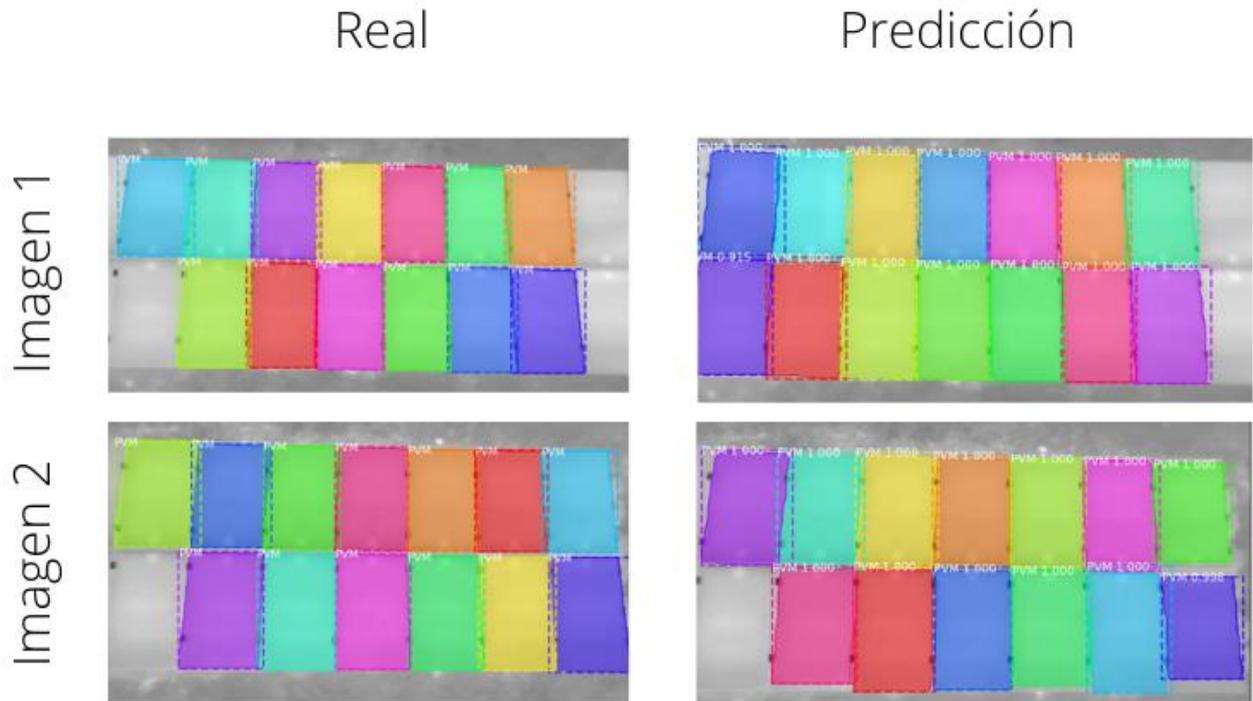
Figura 36 – Resultados comparativo de la segmentación.



Fuente: El autor, 2022.

Posteriormente, una vez que el entrenamiento ha finalizado y los pesos de la red han sido guardados en formato .h5, se procedió a evaluar el modelo en algunas pocas imágenes originales del *dataset* (de donde se obtuvo el *Foreground*), en concreto, las provenientes de frames de tomas más cercanas las cuales permiten una mejor visualización de los resultados. Para evaluar la eficiencia del modelo sobre este conjunto de datos se etiquetaron unas 40 imágenes y se calculó el mPA, arrojando un resultado de 99,7%. Los resultados obtenidos se pueden evidenciar en la Figura 37.

Se evidencia así una precisión mayor en este último conjunto, esto debido quizá al hecho de que las imágenes no contienen fondos con objetos y escenas tan diversas como las de *dataset* sintético. Es importante resaltar que este conjunto de validación fue bastante pequeño debido a la complejidad de la tarea de etiquetado, sin embargo, esta prueba es interesante para tener una medida de cómo se comporta el mAP en ambientes más semejantes a los de operación.

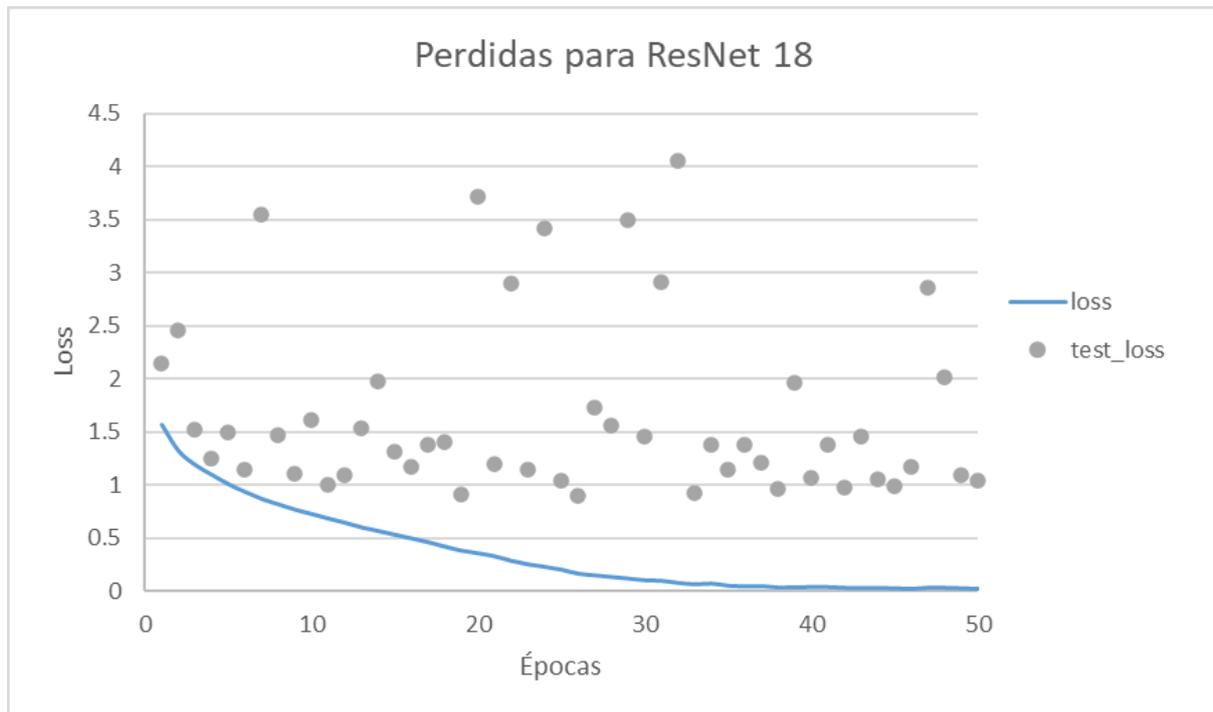
Figura 37 – Resultados de la segmentación de imágenes.

Fuente: El autor, 2022.

Este método resulto muy eficaz para la segmentación de imágenes y varias informaciones pueden ser extraídas a partir los datos de salida del modelo. Para el contexto de este trabajo dos funcionalidades fueron implementadas. La primera consiste en imprimir un identificador para cada panel y un contador del total de paneles hallados por cada imagen. La segunda funcionalidad consiste en la extracción de paneles individuales, en simples palabras, un recorte de imagen para cada panel individual.

4.2 CLASIFICACIÓN

De acuerdo con la metodología descrita en el capítulo anterior, fue entrenado un clasificador con la arquitectura ResNet18 durante 50 épocas. La Figura 38 compara los resultados de perdida para el entrenamiento y test.

Figura 38 – Pérdidas del modelo ResNet18 para 50 épocas

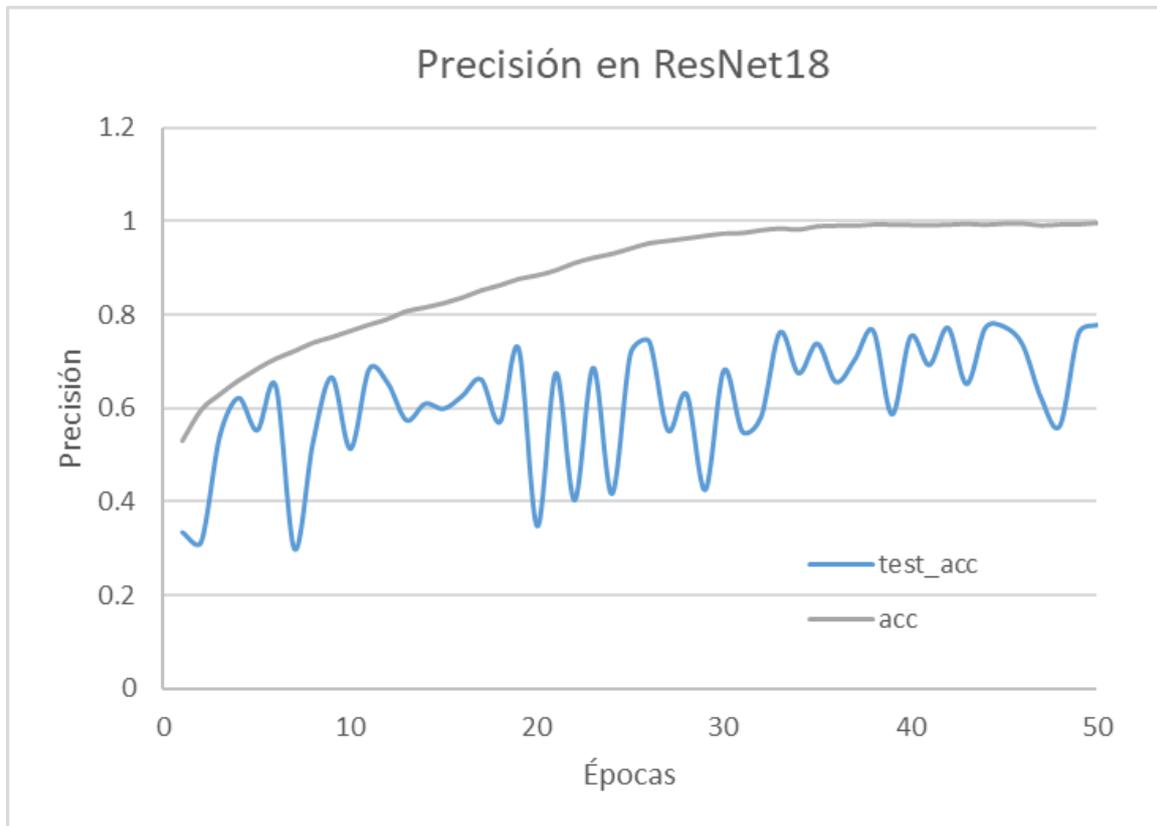
Fuente: El autor, 2022.

Se observa que, con el aumento de las épocas, el modelo reduce las fallas en cuanto a la predicción de las clases sobre el conjunto de datos de entrenamiento, y alrededor de la época 40 ya no se logra percibir una reducción significativa en la función de costo. Sin embargo, en los datos de *test* la pérdida se mantiene, por lo general, en un rango entre 2 y 1, lo que se traduce en una falla de clasificación para la etiqueta real, esto es, se asigna una probabilidad baja de pertenecer a la clase, entre 1% y 10%. Esto se deduce al ejecutar una operación inversa en la ecuación (2). Los puntos fuera de este rango se pueden interpretar como fallas graves en la predicción, esto puede ocurrir, por ejemplo, debido a que en el *batch* de *test* se presentaron módulos pertenecientes a las clases con baja representatividad en el conjunto de datos, los cuales son difíciles de clasificar, pues el modelo no cuenta con suficientes datos para aprender a clasificar correctamente estas clases.

En cuanto a la precisión, en la Figura 39 se observa como el modelo a aprendido a clasificar de forma exitosa el conjunto de datos de entrenamiento, llegando a una precisión cercana al 100%. Ya sobre el conjunto de test, esto es, respecto a los nuevos datos, se puede observar un aumento en la precisión con el paso de las épocas, pero nunca superando el 80%, indicando que el modelo aún podría mejorar. Como se puede observar

el aumento de la precisión ocurre con un poco de “ruido”, esto quizá, generado nuevamente por la aparición de datos difíciles de clasificar.

Figura 39 – Precisión del modelo ResNet18 para 50 épocas



Fuente: El autor, 2022.

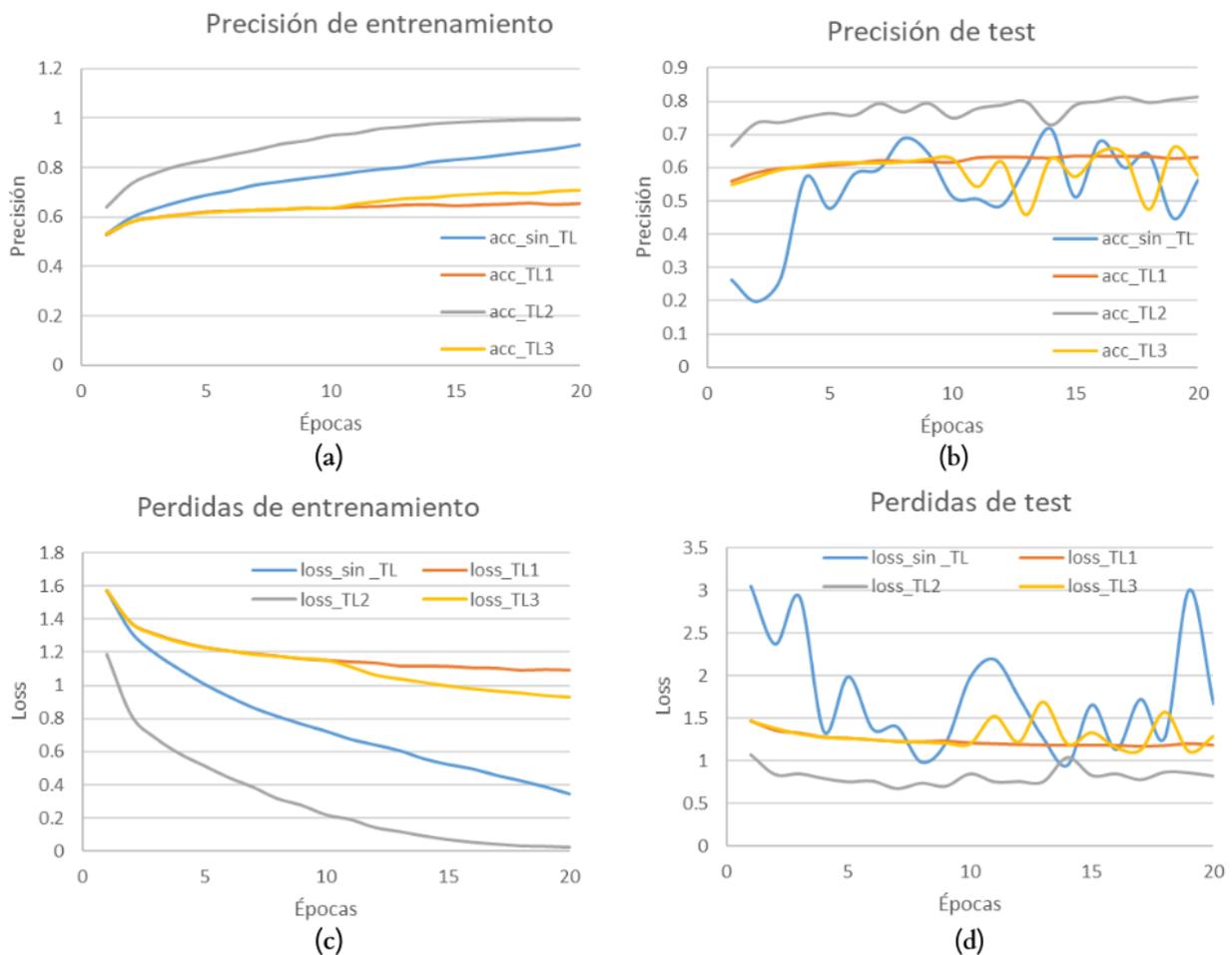
En cuanto a la comparación del método anterior y los métodos definidos para *Transfer Learning* TL1, TL2 y TL3 en la Figura 40 se puede observar un mayor rendimiento en el método *Fine-Tuning* (TL2), convergiendo rápidamente en cuanto a las métricas de precisión (Figura 40 a) y pérdida (Figura 40 b) durante el entrenamiento, y obteniendo mejores resultados cuando evaluado sobre el conjunto de test (Figura 40 c y d). Aquí se puede notar una de las grandes ventajas del método *Fine-Tuning*, ya que, como se puede notar, el modelo aprende rápidamente como ejecutar la tarea específica de clasificar las fallas al adaptar el conocimiento previo en unas cuantas épocas de entrenamiento.

Se nota también como los métodos TL1 y TL3 no logran mejorar en gran medida el rendimiento del modelo, incluso consiguiendo peores métricas que el modelo que aprende desde cero. Esto indica que, el aprendizaje previo no es capaz de ejecutar esta nueva tarea, lo cual es esperado, ya que, el modelo importado para clasificar el *dataset*

Imagenet clasifica una gran cantidad de clases de elementos, pero no está diseñado para la tarea específica que este trabajo aborda.

Cabe resaltar que, como TL3 es una combinación de TL1 y TL2, lo más probable es que, con un mayor número de épocas, TL3 consiga igualar las métricas de TL2 una vez que, a partir de determinada época, se le da la libertad de adaptar el conocimiento. Esto se puede notar en las métricas de precisión y pérdida para entrenamiento (Figura 40 a y c) pues, a partir de la época 10, época en la cual el modelo se encuentra libre para ajustar los pesos de la red, las métricas de TL3 comienzan a tornarse mejores que las de TL1.

Figura 40– Comparación sin y con *Transfer Learning* para ResNet18

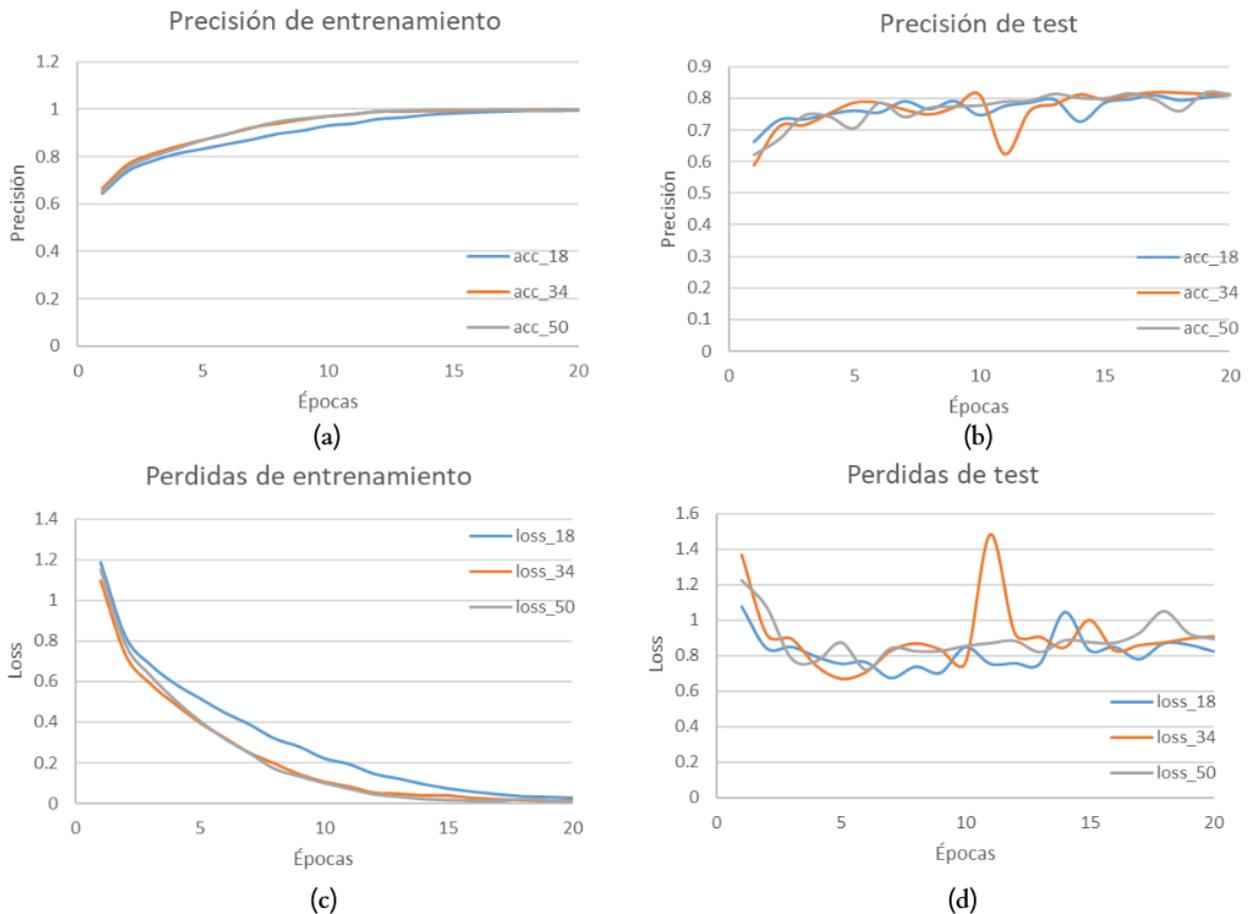


Fuente: El autor, 2022.

Una vez determinado que el método *Fine-Tuning* tiene mejor desempeño, se procedió a comparar las arquitecturas ResNet18, ResNet34 y ResNet50 con este método, fijando un máximo de 20 épocas de entrenamiento. La Figura 41 muestra los resultados de precisión y pérdida sobre los conjuntos de entrenamiento y *test*.

Es posible observar que, en cuanto al entrenamiento, ResNet 34 y 50 tienen mejor desempeño en la precisión y pérdida (Figura 41 a y c) que ResNet 18. Esto puede estar relacionado a la robustez de los modelos, pues, al poseer más capas, son capaces de aprender más y mejor las diferentes características que serán de utilidad para realizar una correcta clasificación.

Figura 41 – Comparación de arquitecturas ResNet.



Fuente: El autor, 2022.

En la Figura 41(d) se puede observar que conforme avanzan las épocas de entrenamiento las diferentes arquitecturas logran un mejor desempeño en cuanto a la métrica de pérdidas, sin embargo, esta métrica alcanza un mínimo y se puede percibir una tendencia creciente a partir de la época 10, lo que se puede interpretar como el inicio del *overfitting*. Es por esto que, se decide usar la estrategia de parada temprana y de esta forma seguir entrenando solo hasta la época 20, época para la cual la mayoría de las métricas consiguen su mejor desempeño.

Por último, debido a que ResNet50 presenta mejor desempeño en la mayoría de las métricas (incluyendo Figura 41b), se procedió a usar esta arquitectura para entrenar el modelo realizando modificaciones en el *Learning Rate*. Los resultados de las diferentes arquitecturas y métodos implementados se resumen en la tabla 3.

Tabla 3 – Resumen de desempeño de las arquitecturas ResNet.

Arquitectura	Epochs	L. Rate	Método	Acc	Loss	Test_Acc	Test_Loss
ResNet18	20	0.01	Sin TL	0.89097	0.34582	0.56343	1.66375
	50	0.01	Sin TL	0.99787	0.01951	0.7802	1.04024
	20	0.01	TL 1	0.65341	1.09416	0.62986	1.17908
	20	0.01	TL 2	0.99662	0.02604	0.81223	0.82166
	20	0.01	TL 3	0.70873	0.93292	0.57525	1.29015
ResNet34	20	0.01	TL 2	0.99713	0.01259	0.81215	0.90779
ResNet50	20	0.01	TL 2	0.99887	0.00791	0.81407	0.89313
	20	0.10	TL 2	0.9771	0.06817	0.44313	3.74978
	20	0.05	TL 2	0.99625	0.01355	0.76418	1.13599

Fuente: El autor, 2022.

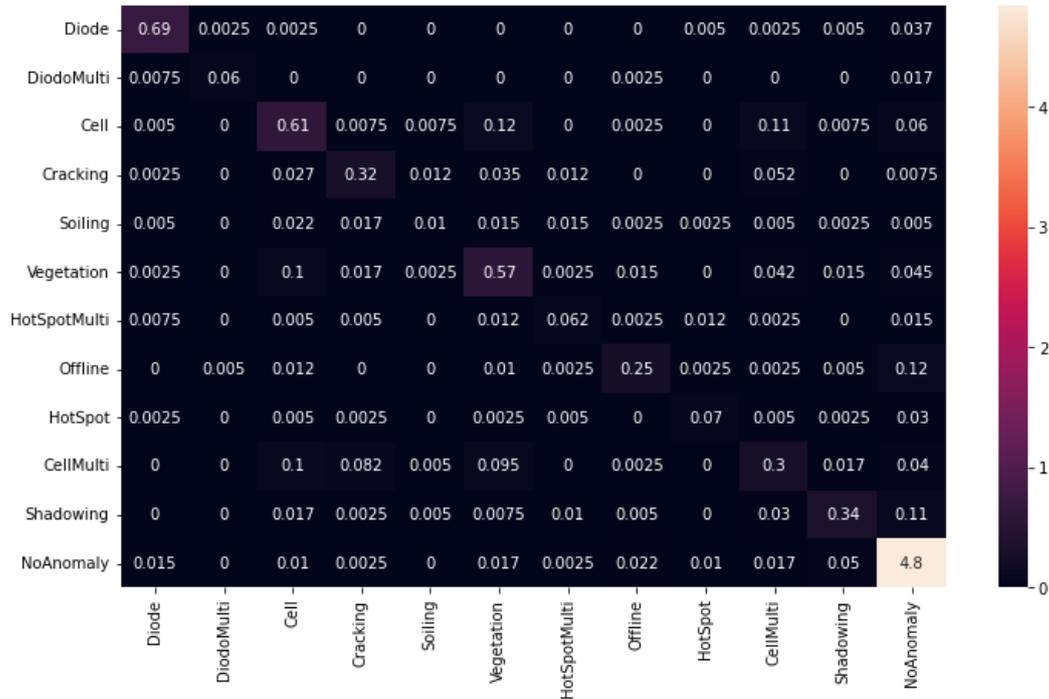
Como se puede observar, el modelo con mayor desempeño es ResNet50 para el método *Fine-Tuning* con LR de 0.01, consiguiendo una precisión de hasta 81,4% en el conjunto de datos de prueba. Este resultado es mejor que el obtenido por (LE *et al.*, 2021), en el que se evalúa este mismo conjunto de datos con el mismo modelo, pero a la vez inferior a un modelo más robusto presentado en el mismo trabajo, en donde se propuso un modelo ensamblado donde se obtuvo hasta un 85,9% de precisión.

Aunque la biblioteca PyTorch contiene 2 estructuras más, ResNet101 y ResNet 152, no sé procedió a entrenarlas debido a que los costos computacionales son mayores y no se percibieron cambios significativos en las métricas obtenidas por los modelos entrenados a medida que la arquitectura iba incrementando en cuanto al número de capas.

Por último, con el objetivo de evaluar el modelo obtenido por Resnet50 la Figura 43 presentar la matriz de confusión. En la diagonal los valores corresponden a las clasificaciones hechas de forma correcta para cada clase (TP), por otro lado, los valores fuera de la diagonal representan las clasificaciones fallidas (TN, FP, FN), sin embargo, este grafico no resulta ser tan útil debido al desequilibrio en el dataset, pues, como se puede notar, la precisión de la clases “NoAnomaly” asume el valor máximo y la escala que permite comparar los valores de precisión pierde relevancia. Aunque sé puede notar que algunos

de los mayores valores se observan sobre algunas de las clases con mayor representatividad en el *dataset*.

Figura 43 – Matriz de confusión ResNet50.



Fuente: El autor, 2022.

Otras informaciones que sí podrían ser útiles para analizar el desempeño del modelo y derivan del cálculo de la matriz de confusión, como explicado en la sección 2.3.2.9, son mostrados en la Tabla 4.

Tabla 4 – Resumen de desempeño de la arquitectura ResNet50.

	Precisión	Exhaustividad	f1-score	Soporte
Diodo	0.93	0.94	0.93	297
Diodo-Multi	0.69	0.89	0.77	27
Celda	0.65	0.67	0.66	367
Agrietamiento	0.68	0.7	0.69	183
Suciedad	0.1	0.24	0.14	17
Vegetación	0.7	0.64	0.67	358
Hot-Spot-Multi	0.5	0.56	0.53	45
Módulo fuera de línea	0.61	0.82	0.7	124
Punto caliente	0.56	0.68	0.62	41
Celda-Multi	0.47	0.53	0.5	230
Sombreado	0.65	0.77	0.7	180
Sin anomalía	0.97	0.91	0.94	2135

Fuente: El autor, 2022.

Como se puede observar, nuevamente, las mejores métricas se presentan en la mayoría de las clases con mayor representación dentro del conjunto (mayor soporte), como por ejemplo las clases sin anomalía, los fuera de línea y los de falla relacionada con el Diodo. Por tanto, aquí es posible notar como el desequilibrio entre clases afecta el rendimiento del modelo para una correcta clasificación de los diferentes tipos de fallas.

4.3 IDENTIFICAR FALLAS

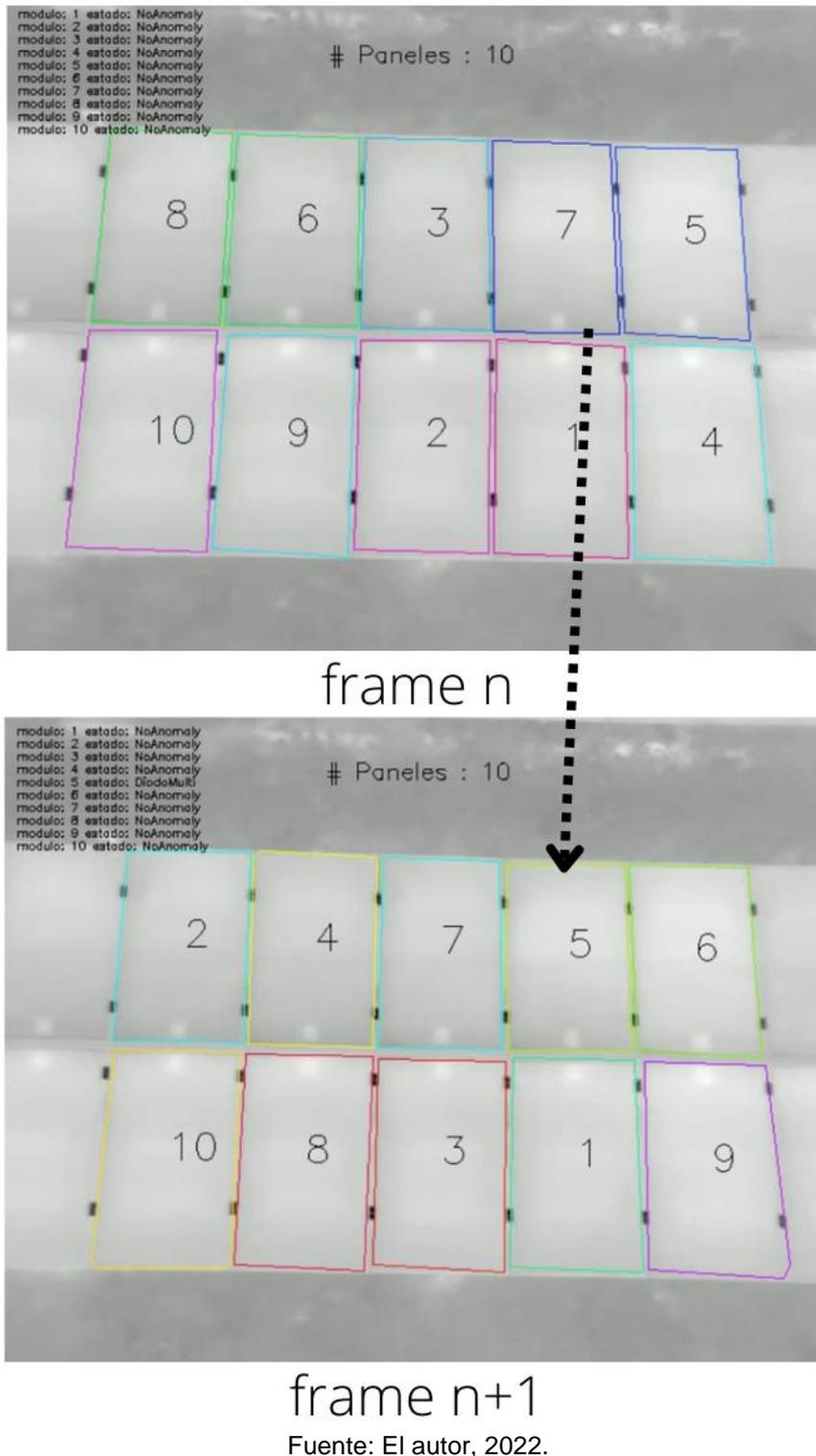
De acuerdo con la metodología planteada, se realizó la implementación de los modelos previamente entrenados de segmentación y clasificación en un único script de Python, tal que, fuera posible hacer la identificación de cada panel y a su vez fuese posible agregar una descripción del estado del mismo. Este resultado también se consiguió implementar en un video disponible en Youtube⁶, en donde la clasificación era ejecutada por el modelo ResNet50 para los diferentes paneles hallados por el modelo Mask en cada *frame*.

La Figura 44 muestra dos *frames* consecutivos, *frame n* y *frame n+1*. Como se puede notar, las etiquetas asignadas al estado de cada panel hallados en el *frame n* (parte superior izquierda) parecen etiquetar de forma adecuada al estado real de estos. En algunos casos se notó una variación en la clasificación para frames consecutivos, por ejemplo, en la Figura 44 para el panel 7 en el *frame n*, se presenta su estado como sin anomalía, sin embargo, en el *frame n+1* el mismo panel, identificado como panel 5, presenta anomalía de Diodo. Una hipótesis que podría explicar este resultado puede ser planteada en torno a que la clasificación es hecha con mayor precisión según el ángulo al que este respecto a la cámara y la variación del mismo puede generar estas incongruencias con el paso de los *frames*.

Como observado, una mejora a implementar a la metodología usada hasta el momento, es la necesidad de una “memoria” que le permita al sistema hacer un seguimiento o *tracking* a un panel ya identificado, esto es, mantener un único identificador único para cada panel para los diferentes *frames* de un video.

⁶ <https://youtu.be/10s7QRgeEhc>

Figura 44 – Ejemplo de identificación de fallas.



Es importante resaltar que uno de los grandes desafíos al implementar una técnica como la presentada en este trabajo es la presencia de un desequilibrio en las muestras para cada clase en el *dataset*. También es importante que las imágenes sean tomadas bajo unos parámetros estándar a fin de garantizar unos buenos resultados.

5 CONSIDERACIONES FINALES

Se ha de resaltar que, para el desarrollo de este trabajo, se usó una metodología para crear un *dataset* sintético, ideal para cuando se trabaja a partir de una cantidad limitada de datos. Esto es de suma importancia, pues, comúnmente, es difícil disponer de bases de datos suficientemente grandes para entrenar modelos de DL para tareas específicas, en este contexto, este *dataset* puede ser de utilidad para investigadores que desarrollen trabajos relacionados y que tengan como objetivo la identificación de módulos PV en imágenes obtenidas por aIRT.

Con el objetivo de segmentar los módulos PV en una imagen, se entrenó el modelo Mask R-CNN alcanzando precisiones superiores al 95%, demostrando así, la eficacia de esta técnica y la capacidad de estas herramientas de IA para resolver tareas de detección de objetos.

Una vez que el objetivo final de este trabajo consistía en detectar las fallas en los módulos PV, se entrenó un clasificador usando la estructura ResNet, probando diferentes configuraciones, con el fin de clasificar un determinado modulo PV y consiguiéndose resultados de precisión superiores al 81%, esto a pesar del desequilibrio en los datos de entrada para cada clase de anomalía.

También se logró implementar conjuntamente las técnicas de segmentación y clasificación, y se mostró que, a pesar de algunos errores observados en la clasificación de fallas, es posible integrarlas con el objetivo de usarse como herramienta en la inspección de fallas de los módulos PV en granjas solares. La metodología seguida en este trabajo se puede adaptar para aplicarse en otras tareas de inspección como el control de la contaminación de aguas superficiales (LEGA; NAPOLI, 2010), delaminaciones subsuperficiales en cubiertas de puentes de hormigón (OMAR; NEHDI, 2017) y detección anomalías térmicas en edificios(ORTIZ-SANZ *et al.*, 2019), entre otros.

Como trabajo futuro se desea crear conjuntos de datos de anomalías, bajo unas mismas normas técnicas, con el objetivo de validar la precisión del modelo desarrollado en este trabajo para la detección de anomalías. También es de gran interés implementar una función de *Tracking*, tal que, el sistema identifique como único cada panel hallado en la secuencia de *frames* de un video, así, evitar más de una identificación y clasificación para cada panel. De igual forma se espera limitar la detección y clasificación a una pequeña región central para evitar la multiplicidad en la clasificación a medida que el

ángulo entre la cámara y el panel varia con el pasar de los *frames* de video. Adicionalmente, se desea desarrollar una interfaz de usuario que ofrezca un informe de las anomalías encontradas, brindando información clara y puntual al personal técnico de mantenimiento. Por último, se desea probar otros algoritmos para la segmentación de instancias, como Detectron2, un modelo mejorado de Mask R-CNN desarrollado por Facebook, o YOLO (MOHAMED *et al.*, 2021), y otros métodos para la clasificación como Meta Pseudo Labels (PHAM *et al.*, 2020).

REFERENCIAS

- AL RABBANI ALIF, M.; AHMED, S.; HASAN, M. A. Isolated Bangla handwritten character recognition with convolutional neural network. *In: 2017 20TH INTERNATIONAL CONFERENCE OF COMPUTER AND INFORMATION TECHNOLOGY (ICCIT)*, 2017. **Anais: IEEE**, 2017. p. 1–6.
- AZEVEDO, A. **Banco de dados sintético**. 2021. Disponível em: <https://github.com/AlcineiAzevedo/BancoDadosSinteticoMaracuja/blob/main/CriaBancoDados%20Sintetico%201.0.ipynb>. Acesso at: 6 Aug. 2022.
- BAGNATO, J. I. Aprende Machine Learning en Español. *In: Leanpubed*. 2022. v. 1, p. 143–155.
- BOMMES, L. **PV Hawk**. 2021. Disponível em: <https://lukasbommes.github.io/PV-Hawk/tutorial.html#step-2-download-the-example-dataset>. Acesso at: 7 Aug. 2022.
- BOMMES, L.; HOFFMANN, M.; BUERHOP-LUTZ, C.; PICKEL, T.; HAUCH, J.; BRABEC, C.; MAIER, A.; PETERS, I. M. Anomaly Detection in IR Images of PV Modules using Supervised Contrastive Learning. 2021a. Disponível em: <http://arxiv.org/abs/2112.02922>.
- BOMMES, L.; PICKEL, T.; BUERHOP-LUTZ, C.; HAUCH, J.; BRABEC, C.; PETERS, I. M. Computer Vision Tool for Detection, Mapping and Fault Classification of PV Modules in Aerial IR Videos. 2021b. Disponível em: <http://arxiv.org/abs/2106.07314>.
- BROWNLEE, J. **Difference Between a Batch and an Epoch in a Neural Network**. 2018. Disponível em: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=batches%20and%20epochs,-,What%20Is%20the%20Difference%20Between%20Batch%20and%20Epoch%3F,passes%20through%20the%20training%20dataset>. Acesso at: 1 Jul. 2022.
- BROWNLEE, Jason. **How to Choose Loss Functions When Training Deep Learning Neural Networks**. 2020. Disponível em: <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>. Acesso at: 17 May 2022.
- CANU, S. **Train Mask R-CNN for Image Segmentation**. 2021. Disponível em: <https://pysource.com/2021/08/10/train-mask-r-cnn-for-image-segmentation-online-free-gpu/>. Acesso at: 7 Aug. 2022.
- CHEN, T.; JIANG, Y.; JIAN, W.; QIU, L.; LIU, H.; XIAO, Z. Maintenance Personnel Detection and Analysis Using Mask-RCNN Optimization on Power Grid Monitoring Video. **Neural Processing Letters**, v. 51, n. 2, p. 1599–1610, 2020.
- DAI, J.; LI, Y.; HE, K.; SUN, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. 2016.
- DE OLIVEIRA, A. K. V.; AGHAEI, M.; RÜTHER, R. **Automatic Inspection of Photovoltaic Power Plants Using Aerial Infrared Thermography: A Review**. MDPI, 2022.

DENIO, H. Aerial solar Thermography and condition monitoring of photovoltaic systems. *In: 2012 38TH IEEE PHOTOVOLTAIC SPECIALISTS CONFERENCE, 2012. Anais IEEE, 2012. p. 000613–000618.*

DUNDERDALE, C.; BRETTEENY, W.; CLOHESSY, C.; DYK, E. E. Photovoltaic defect classification through thermal infrared imaging using a machine learning approach. **Progress in Photovoltaics: Research and Applications**, v. 28, n. 3, p. 177–188, 2020.

EVERINGHAM, M.; WINN, J. **PASCAL VOC2011 Example Images**. 2012.

GIRSHICK, Ross. Fast R-CNN. 2015. Disponível em: <http://arxiv.org/abs/1504.08083>.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. 2013. Disponível em: <http://arxiv.org/abs/1311.2524>.

GRÜBLER, M. **Entendendo o funcionamento de uma rede neuronal artificial**. 2018.

HE, Kaiming.; GKIOXARI, Georgia.; DOLLÁR, Piotr.; GIRSHICK, Ross. Mask R-CNN. 2017. Disponível em: <http://arxiv.org/abs/1703.06870>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep Residual Learning for Image Recognition. 2015. Disponível em: <http://arxiv.org/abs/1512.03385>.

IBM. **Overfitting**. 2021. Disponível em: <https://www.ibm.com/cloud/learn/overfitting>. Acesso at: 8 Aug. 2022.

JAHN, U.; HERZ, M. **Review on Infrared and Electroluminescence Imaging for PV Field Applications**. 2018.

JAIN, V. **Everything you need to know about “Activation Functions” in Deep learning models**. 2019. Disponível em: <https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253>. Acesso at: 6 Aug. 2022.

KÖNTGES, M. 1972-; KURTZ, S.; PACKARD, C.; JAHN, U.; BERGER, K. A.; KATO, K.; INTERNATIONAL ENERGY AGENCY PHOTOVOLTAIC POWER SYSTEMS PROGRAMME. **Review of failures of photovoltaic modules**. 2014.

LE, M.; VAN SU, L.; DANG KHOA, N.; DAO, V. D.; NGOC HUNG, V.; HONG HA THI, V. Remote anomaly detection and classification of solar photovoltaic modules based on deep neural network. **Sustainable Energy Technologies and Assessments**, v. 48, 2021.

LEGA, M.; NAPOLI, R. M. A. Aerial infrared thermography in the surface waters contamination monitoring. **Desalination and Water Treatment**, v. 23, n. 1–3, p. 141–151, 2010.

LIN, T.-Y.; DOLLÁR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature Pyramid Networks for Object Detection. 2016.

LIU, L.; OUYANG, W.; WANG, X.; FIEGUTH, P.; CHEN, J.; LIU, X.; PIETIKÄINEN, M. Deep Learning for Generic Object Detection: A Survey. **International Journal of Computer Vision**, v. 128, n. 2, p. 261–318, 2020.

MARTÍNEZ, J. **¿Cómo Se Representan las Imágenes en una Computadora?**. 2018. Disponível em: <https://datasmarts.net/es/como-se-representan-las-imagenes-en-una-computadora/>. Acesso at: 15 Jun. 2022.

MILLENDORF, M.; OBROPTA, E.; VADHAVKAR, N. INFRARED SOLAR MODULE DATASET FOR ANOMALY DETECTION. 2020. Disponível em: <https://github.com/RaptorMaps/>.

MOHAMED, E.; SHAKER, A.; EL-SALLAB, A.; HADHOUD, M. INSTA-YOLO: Real-Time Instance Segmentation. 2021.

MORADI SIZKOUHI, A. M.; ESMAILIFAR, S. M.; AGHAEI, M.; KARIMKHANI, M. RoboPV: An integrated software package for autonomous aerial monitoring of large scale PV plants. **Energy Conversion and Management**, v. 254, p. 115217, 2022.

NARKHEDE, S. **Understanding Confusion Matrix**. 2018. Disponível em: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. Acesso at: 9 Aug. 2022.

NIELSEN, M. **How the backpropagation algorithm works**. 2019. Disponível em: <http://neuralnetworksanddeeplearning.com/chap2.html>. Acesso at: 15 Jul. 2022.

OLAH, C.; MORDVINTSEV, A.; SCHUBERT, L. Feature Visualization. **Distill**, v. 2, n. 11, 2017.

OMAR, T.; NEHDI, M. L. Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography. **Automation in Construction**, v. 83, p. 360–371, 2017.

ORTIZ-SANZ, J.; GIL-DOCAMPO, M.; ARZA-GARCÍA, M.; CAÑAS-GUERRERO, I. IR Thermography from UAVs to Monitor Thermal Anomalies in the Envelopes of Traditional Wine Cellars: Field Test. **Remote Sensing**, v. 11, n. 12, p. 1424, 2019.

ÖZDEMİR, H. **Residual Block**. 2022. Disponível em: https://www.youtube.com/watch?v=r0HvOljziw4&ab_channel=ComputerVisionwithH%C3%BCseyin%C3%96zdemir. Acesso at: 14 Jun. 2022.

PHAM, H.; DAI, Z.; XIE, Q.; LUONG, M.-T.; LE, Q. v. Meta Pseudo Labels. 2020.

RAPTOR MAPS. **Infrared solar modules**. 2020. Disponível em: <https://github.com/RaptorMaps/InfraredSolarModules>. Acesso at: 7 Aug. 2022.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2015. Disponível em: <http://arxiv.org/abs/1506.01497>.

REN21. **Renewable energy data in perspective**. Paris: 2022a.

REN21. **Renewables 2022 Global Status Report**. 2022b.

RICHARDSON, W.; KRISHNASWAMI, H.; VEGA, R.; CERVANTES, M. A low cost, edge computing, all-sky imager for cloud tracking and intra-hour irradiance forecasting. **Sustainability (Switzerland)**, v. 9, n. 4, p. 1–17, 2017.

RICO ESPINOSA, A.; BRESSAN, M.; GIRALDO, L. F. Failure signature classification in solar photovoltaic plants using RGB images and convolutional neural networks. **Renewable Energy**, v. 162, p. 249–256, 2020.

RUSSELL S. J.; NORVIG P. **Artificial Intelligence: A Modern Approach**. 3. Ed. 2015. 2015.v. 1.

SAHA, S. **A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way**. 2018.

SHARMA, S. **Activation Functions in Neural Networks**. 2017. Disponível em: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>. Acesso at: 6 Aug. 2022.

SHU, J.-H.; NIAN, F.-D.; YU, M.-H.; LI, X. An Improved Mask R-CNN Model for Multiorgan Segmentation. **Mathematical Problems in Engineering**, v. 2020, p. 1–11, 2020.

TIAN, D.; HAN, Y.; WANG, B.; GUAN, T.; GU, H.; WEI, W. Review of object instance segmentation based on deep learning. **Journal of Electronic Imaging**, v. 31, n. 04, 2021.

TURING. **Softmax: Multiclass Neural Networks**. 2022. Disponível em: <https://www.turing.com/kb/softmax-multiclass-neural-networks>. Acesso at: 6 Aug. 2022.

VEGA DÍAZ, J. J.; VLAMINCK, M.; LEFKADITIS, D.; ORJUELA VARGAS, S. A.; LUONG, H. Solar Panel Detection within Complex Backgrounds Using Thermal Images Acquired by UAVs. **Sensors**, v. 20, n. 21, p. 6219, 2020.

WALEED ABDULLA. **Mask R-CNN for Object Detection and Segmentation**. 2017. Disponível em: https://github.com/matterport/Mask_RCNN. Acesso at: 26 Jun. 2022.

WANG, H.; MA, J.; LI, D. Two-Dimensional Hybrid Perovskite-Based van der Waals Heterostructures. **The Journal of Physical Chemistry Letters**, v. 12, n. 34, p. 8178–8187, 2021.

WARING, J.; LINDVALL, C.; UMETON, R. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. **Artificial Intelligence in Medicine**, v. 104, p. 101822, 2020. Disponível em: Acesso at: 24 Jul. 2022.

WENG, L. Object Detection for Dummies Part 3: R-CNN Family. **lilianweng.github.io** 2017. Disponível em: <https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/>. Acesso at: 2 Jul. 2022.

YOHANANDAN, S. **MAP (mean Average Precision) might confuse you!**. 2020. Disponível em: <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2>. Acesso at: 8 Aug. 2022.

ZHAO, K.; KANG, J.; JUNG, J.; SOHN, G. Building Extraction From Satellite Images Using Mask R-CNN With Building Boundary Regularization. *In*: PROCEEDINGS OF THE IEEE

CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR) WORKSHOPS. 2018. p. 247–251.

ZHU, Z.; LIN, K.; JAIN, A. K.; ZHOU, J. Transfer Learning in Deep Reinforcement Learning: A Survey. 2020.

ZULKIFLI, H. **Understanding Learning Rates and How It Improves Performance in Deep Learning**. 2018. Disponível em: <https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10>. Acesso at: 6 Aug. 2022.