UNIVERSIDAD DE GUANAJUATO

CAMPUS IRAPUATO-SALAMANCA
DIVISIÓN DE INGENIERÍAS

"Estimación de los óxidos producidos por la irradiación de un láser pulsado en placas delgadas de molibdeno usando aprendizaje profundo"

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA

ELÉCTRICA

PRESENTA:

Ing. José Rodrigo Paredes Miguel

DIRECTORES:

Dr. Horacio Rostro González

Dr. David Camarena Martínez

Salamanca, Guanajuato Octubre 2024

Salamanca, Gto., a 28 de octubre del 2024.

M. en I. HERIBERTO GUTIÉRREZ MARTIN
COORDINADOR DE ASUNTOS ESCOLARES
P R E S E N T E.-

Por medio de la presente, se otorga autorización para proceder a los trámites de impresión, empastado de tesis y titulación al alumno **José Rodrigo Paredes Miguel** del ***Programa de Maestría en Ingeniería Eléctrica (Instrumentación y Sistemas Digitales)*** y cuyo número de ***NUA*** es: **146518** del cual soy director. El título de la tesis es: **Estimación de los óxidos producidos por la irradiación de un láser pulsado en placas delgadas de molibdeno usando aprendizaje profundo.**

Hago constar que he revisado dicho trabajo y he tenido comunicación con los sinodales asignados para la revisión de la tesis, por lo que no hay impedimento alguno para fijar la fecha de examen de titulación.

A T E N T A M E N T E

Dr. Horacio Rostro González
DIRECTOR DE TESIS
SECRETARIO

Dr. David Camarena Martínez
DIRECTOR DE TESIS

Dr. Mario Ibarra Manzano
PRESIDENTE

Dr. José Ruiz Pinales
VOCAL

Dedicatoria

Dedico este trabajo a mis queridos padres, la Sra. Felipa Miguel Baldomero y el Sr. Miguel Paredes Balbino, quienes han sido los pilares más importantes de mi vida. También a mis hermanos y hermanas, cuyo apoyo incondicional ha sido fundamental para mí, así como a sus hijos, mis queridos sobrinos, quienes siempre me han brindado alegría y motivación. Finalmente, agradezco profundamente a todas aquellas personas que, de una u otra forma, me apoyaron y siempre confiaron en mí.

Agradecimientos

Reitero el agradecimos a mis padres que si no fuese por ellos esto no hubiese sido posible.

Al Dr. Horacio Rostro por su gran apoyo y confianza en este proyecto.

Al Dr. David Camarena por su apoyo y dedicación para la culminación del trabajo.

A La Dra. Miroslava por la confianza de dejar este proyecto en mis manos.

A mis hermanos de la Estudiantina División de Ingenierías Salamanca.

A todos mis compañeros que de una u otra forma aportaron a terminar este trabajo.

Agradecimientos Institucionales

Expreso mi gratitud hacia la Universidad de Guanajuato, especialmente a la División de Ingenierías del Campus Irapuato-Salamanca (DICIS) por la formación académica y espacios de trabajo que se me facilitaron durante mis estudios en la institución.

Este trabajo de tesis fue realizado gracias al apoyo recibido a través del Consejo Nacional de Humanidades Ciencias y Tecnología CONAHCYT de México, bajo el CVU: 124598, número de becario: 832542.

Resumen

Este trabajo se enfoca en la aplicación de técnicas avanzadas de aprendizaje profundo, particularmente redes neuronales convolucionales, para abordar problemas de segmentación de imágenes y Redes neuronales profundas, para la predicción de diámetros de óxidos generados por la irradiación de láser ultracorto sobre placas de molibdeno. La implementación de estas técnicas permite no solo identificar y segmentar con precisión los diferentes tipos de óxidos presentes en las imágenes, sino también predecir de manera precisa los diámetros de los spots de óxido formados bajo distintas configuraciones de parámetros, como el tiempo de exposición y la fluencia del láser. El uso de redes convolucionales se ha destacado en la capacidad de procesar grandes cantidades de datos visuales y extraer patrones complejos, lo que resulta particularmente útil en este caso, donde las variaciones microscópicas en los materiales son cruciales para su análisis. A través de la segmentación, es posible identificar áreas específicas de los diferentes óxidos, lo que contribuye a una mejor comprensión de los efectos que el láser provoca sobre el material. Además, esta investigación abre la puerta a la integración de la inteligencia artificial en el ámbito de la ciencia de los materiales, permitiendo la automatización y mejora en la caracterización de los mismos. Los resultados obtenidos no solo aportan valor en términos de segmentación precisa, sino que también demuestran cómo la IA puede ser una herramienta poderosa para predecir propiedades físicas clave, como el diámetro de los óxidos. Este enfoque puede ser expandido en futuros estudios para abarcar otros aspectos del comportamiento de materiales bajo condiciones extremas, contribuyendo al desarrollo de tecnologías avanzadas en esta área.

Abstract

This work is concerned with the application of sophisticated deep learning techniques, in particular convolutional neural networks, to address the issue of image segmentation and the use of deep neural networks for the prediction of oxide diameters resulting from ultrashort laser irradiation on molybdenum plates. The implementation of these techniques allows for the accurate identification and segmentation of the different types of oxides present in the images, as well as the accurate prediction of the diameters of the oxide spots formed under different parameter settings, such as exposure time and laser fluence. The employment of convolutional networks has demonstrated an exceptional capacity to process copious quantities of visual data and extract intricate patterns, which is especially advantageous in this context, where minute variations in materials are of paramount importance for analysis. The application of segmentation techniques facilitates the identification of specific areas within the different oxides, thereby enhancing the comprehension of the laser's impact on the material. Furthermore, this research paves the way for the integration of artificial intelligence in the field of materials science, facilitating automation and enhancement in the characterization of materials. The results obtained not only provide value in terms of accurate segmentation, but also demonstrate the potential of AI as a powerful tool for predicting key physical properties, such as the diameter of oxides. This approach can be expanded in future studies to cover other aspects of material behavior under extreme conditions, contributing to the development of advanced technologies in this area.

Índice general

Índice de figuras

Capítulo 1

Introducción

Contenido

En los últimos años, la inteligencia artificial (IA) ha transformado radicalmente diversos campos de la ciencia y la industria. Áreas como la medicina, la automoción, la optimización energética y la manufactura avanzada han adoptado técnicas de aprendizaje profundo para mejorar la precisión y la velocidad en sus procesos. Estas herramientas, capaces de procesar grandes volúmenes de datos y reconocer patrones complejos, están cambiando la forma en que los investigadores y los ingenieros desarrollan nuevas tecnologías [4].

En el campo de la ciencia de materiales, la IA ha mostrado un gran potencial al acelerar la identificación, caracterización y optimización de nuevos compuestos y estructuras [5, 6]. La aplicación de técnicas de aprendizaje profundo permite la automatización de tareas complejas, como la segmentación y clasificación de microestructuras, que anteriormente requerían análisis exhaustivos y manuales [7]. Esto es particularmente relevante en la producción de materiales avanzados como los óxidos de molibdeno, cuyas propiedades físicas y químicas los hacen esenciales para aplicaciones en catálisis, semiconductores y almacenamiento de energía [8]. Se sabe que al exponer cierto tipo de materiales a un láser ultra pulsado (o continuos), producirá cambios físicos y químicos en el mismo. En estudios experimentales [9], se ha demostrado que al utilizar el método de irradiación láser (con pulsos en el orden de femtosegundos) en películas delgadas de molibdeno se estimula a la formación de óxidos de molibdeno (MoO_x). Al emplear el método láser se puede obtener un control

fino en la formación de los óxidos metálicos, en su estructura cristalina, en su estequiometría y morfología; el tiempo de formación o síntesis de los óxidos es de unos cuantos segundos a minutos. La técnica láser nos permite establecer importantes diferencias en comparación con la síntesis de óxidos metálicos formados mediante métodos convencionales.

La técnica de formación de óxidos metálicos tiene un potencial de impacto tecnológico muy importante en diversas áreas como lo son sensores, pantallas inteligentes, electrodos transparentes, semiconductores, etc. Sin embargo, el hacer estos experimentos requiere de un alta precisión, por lo que el saber las características y condiciones requeridas para producir en cierta cantidad estos óxidos es de suma importancia para la reducción de gastos (en cuanto a experimentos de prueba y error se refiere) en la producción.

En este contexto, nuestro trabajo se centra en la aplicación de técnicas de aprendizaje profundo para segmentar y caracterizar los diferentes óxidos de molibdeno formados mediante irradiación láser. El desafío radica en la precisión requerida para identificar las distintas fases de los óxidos, ya que cada una de ellas puede tener propiedades físicas y químicas únicas que afectan a su idoneidad en aplicaciones específicas. Mediante el uso de redes convolucionales, hemos desarrollado un enfoque que permite no sólo segmentar las imágenes resultantes de los experimentos de irradiación, sino también identificar patrones que contribuyan a la optimización de los procesos de producción. Este tipo de investigación representa un avance significativo al integrar la inteligencia artificial con la ciencia de materiales y la óptica láser, permitiendo una mayor comprensión de los procesos que influyen en la formación de óxidos. Además, al combinar el poder del aprendizaje profundo con métodos experimentales como la irradiación láser, nuestro enfoque interdisciplinario refuerza la colaboración entre diferentes áreas de investigación, lo que a su vez puede llevar al desarrollo de soluciones innovadoras en campos industriales clave.

1.1. Justificación

La formación de óxidos en la superficie de la capa de molibdeno es de especial interés debido a su relevancia en aplicaciones de catálisis, fotocatálisis, electrónica y almacenamiento de energía. Dado el impacto de estos óxidos en aplicaciones tan diversas, es de gran importancia segmentar y caracterizar de manera precisa los diferentes óxidos de molibdeno que se pueden generar mediante la técnica de irradiación láser. Además, sólo algunos de estos óxidos tienen aplicaciones donde su uso juegan un papel esencial como, catalizadores en reacciones como en desulfuración de combustibles fósiles y la oxidación selectiva de hidrocarburos, para fabricar dispositivos semiconductores avanzados debido a sus propiedades electrónicas únicas y han demostrado ser prometedores en la fabricación de supercondensadores y baterías de ión-litio. Esto no sólo permitiría optimizar la fabricación de materiales más eficientes y específicos para cada aplicación, sino que también contribuiría al avance de tecnologías clave en áreas críticas como la energía, la electrónica y la industria química.

La adopción del aprendizaje profundo en este contexto es crucial, ya que ofrece la capacidad de analizar datos complejos y predecir resultados, lo que, a su vez, tiene el potencial de revolucionar la optimización de procesos industriales, el desarrollo de materiales avanzados y la exploración de nuevas aplicaciones en campos como la nanotecnología y la fotónica. Esta investigación también promueve la colaboración interdisciplinaria al combinar la óptica láser, la ciencia de materiales y la informática, lo que puede dar lugar a soluciones innovadoras y avances interdisciplinarios

significativos.

1.2. Objetivos

1.2.1. Objetivo general

El propósito principal de esta investigación es desarrollar y aplicar modelos de aprendizaje profundo que permitan predecir con precisión la formación de óxidos generados por irradiación con láser pulsado en placas delgadas de molibdeno.

1.2.2. Objetivos específicos

- Recopilar datos experimentales detallados sobre la respuesta de las placas de molibdeno a la irradiación con láser.
- Realizar la segmentación manual de micrografías para crear una base de datos confiable y lista para su uso en entrenamiento.
- Entrenar las arquitecturas DeepLabV3+, U-Net, SegNet y NetLite, utilizando los datos obtenidos, para predecir cambios en las propiedades del material, incluyendo la formación de óxidos y modificaciones en su estructura cristalina.
- Realizar pruebas numéricas y análisis estadísticos sobre los resultados obtenidos de los modelos.
- Ajustar los hiperparámetros de los modelos, monitoreando métricas clave como accuracy en las etapas de entrenamiento, validación y prueba.

1.3. Antecedentes

Esta tesis parte del trabajo de la Dra. Miroslava Lara Cano, realizado en Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE) [9], donde se presenta un diseño experimental para producir diferentes tipos de óxido de molibdeno [10, 11, 12], utilizando la técnica de irradiación de luz, producida por un láser ultra pulsado sobre una placa delgada de molibdeno. A partir de un arreglo experimental (el cual se menciona más adelante) se recolectarán resultados, que en este caso son las micrografías (imágenes), los cuales serán utilizados como entradas a una red neuronal convolucional. Con esto se pretende obtener una predicción lo más acertada posible de cada uno de los óxidos que se podrían producir, debido a dos factores: tiempo de exposición y fluencia del láser.

En la literatura no se han encontrado trabajos donde se haya hecho segmentación de MoO_x producidos, ya que la técnica que se utiliza en el trabajo [9] permite la generación de diversos MoO_x realizando una sola exposición por tiempo determinado en las placas de molibdeno. La importancia de este trabajo en la segmentación de óxidos parte del interés e importancia de ellos en aplicaciones de catálisis [13], fotocatálisis [14, 15], electrónica [16], optoelectrónica [17, 18], supercondensadores y baterías de alta capacidad [19, 20, 21], por mencionar algunas.

En el campo de la catálisis, los óxidos de molibdeno desempeñan un papel crucial como catalizadores en una amplia variedad de reacciones químicas industriales. Estos materiales son particularmente efectivos en procesos como la desulfuración de combustibles fósiles, un proceso esencial para reducir las emisiones de azufre y cumplir con las normativas ambientales. La capacidad del molibdeno para formar óxidos de diferentes valencias le permite participar en reacciones redox, lo que mejora su eficacia en la remoción de azufre de los combustibles. Además, los MoO_x son utilizados en procesos de fotocatálisis, donde su capacidad para absorber luz y generar especies reactivas se emplea en la degradación de contaminantes orgánicos en el agua y el aire. Esta propiedad es particularmente útil en la eliminación de compuestos difíciles de degradar, como los residuos de pesticidas y colorantes industriales.

En la electrónica, algunos óxidos de molibdeno se utilizan para fabricar dispositivos semiconductores avanzados debido a sus propiedades electrónicas únicas, como su alta movilidad de portadores de carga y su estabilidad térmica. Los óxidos de molibdeno también juegan un papel importante en la fabricación de dispositivos optoelectrónicos, como las células solares y los LEDs orgánicos (OLEDs). En las células solares, los MoO_x se utilizan como capas de transporte de huecos debido a su alta transparencia y capacidad para mejorar la eficiencia de conversión de energía. Además, su uso en OLEDs ayuda a mejorar la vida útil y el brillo de estos dispositivos, haciéndolos más eficientes y duraderos. Además de su uso en electrocatalizadores, los óxidos de molibdeno son componentes esenciales en el desarrollo de supercondensadores y baterías avanzadas, como las de ion-litio. Estos dispositivos de almacenamiento de energía se benefician de la alta capacitancia y estabilidad cíclica de los MoO_x, lo que permite el almacenamiento eficiente de grandes cantidades de energía y una rápida liberación cuando es necesario, características cruciales para aplicaciones en vehículos eléctricos y sistemas de energía renovable.

Debido a la importancia de los MoO_x en la fabricación de algunos materiales, en este trabajo buscamos el estimar el efecto producido por la irradiación del láser ultrapulsado en placas delgadas de molibdeno, tomando en cuenta sólo la fluencia y tiempo de exposición de los pulsos en las mismas. Estos efectos conllevan a la generación de diferentes tipos de óxidos de molibdeno, así como un diámetro para cada uno de ellos. En la literatura no se ha encontrado la aplicación de técnicas de aprendizaje profundo para este problema en específico, por lo que este trabajo pretende ser una aplicación novedosa de las diferentes técnicas que la IA ofrece.

1.4. Planteamiento del problema

A pesar de los avances significativos en la aplicación de la IA en la ciencia de materiales, la segmentación y caracterización precisa de microestructuras sigue siendo un desafío crítico. Los óxidos de molibdeno, debido a su complejidad morfológica y diversidad estructural, presentan dificultades particulares en su identificación y clasificación. La variabilidad en las propiedades físicas y químicas de las diferentes fases de estos óxidos puede influir considerablemente en su desempeño en aplicaciones industriales, como catálisis y almacenamiento de energía.

Tradicionalmente, el análisis de microestructuras de materiales ha dependido de técnicas manuales y exhaustivas, que no sólo son laboriosas, sino que también son propensas a errores subjetivos. Esta dependencia de métodos convencionales limita la capacidad de los investigadores para realizar un análisis rápido y efectivo. En este contexto, el planteamiento de nuestro estudio radica en la nece-

sidad de desarrollar un enfoque automatizado y preciso para la segmentación y caracterización de óxidos de molibdeno mediante técnicas de aprendizaje profundo. La falta de un método robusto y eficiente para identificar y clasificar estas estructuras limita el avance en la investigación de nuevos materiales y su aplicación en tecnologías emergentes.

Por lo tanto, nuestro trabajo aborda esta problemática al proponer un modelo basado en redes neuronales convolucionales (CNN) que no sólo permita la segmentación automática de imágenes de microestructuras, sino que también facilite la identificación de patrones críticos que contribuyan a la optimización de los procesos de producción. La solución a este problema no sólo tiene la intención de mejorar la eficiencia en la producción de óxidos de molibdeno, sino que también puede abrir el camino hacia la integración de la inteligencia artificial en la investigación de materiales en mayor medida, ofreciendo nuevas perspectivas y aplicaciones en diversos campos industriales.

1.5. Contenido de la tesis

Este trabajo de tesis está estructurado en 6 capítulos, los cuales se describen brevemente a continuación:

Capítulo 1: Se presenta la introducción y los objetivos de la investigación, además de ofrecer una breve contextualización sobre el origen y la motivación del trabajo.

Capítulo 2: Se desarrolla el marco teórico, donde se describen los conceptos clave necesarios para una mejor comprensión de los temas abordados a lo largo del estudio.

Capítulo 3: En este capítulo se pone un énfasis particular en el molibdeno, detallando sus características y el proceso mediante el cual se obtiene el conjunto de datos utilizado en esta investigación.

Capítulo 4: Se fórmula la metodología, especificando las etapas y procedimientos llevados a cabo para el desarrollo del proyecto.

Capítulo 5: Se presentan los resultados obtenidos a lo largo del estudio, acompañados de su análisis correspondiente.

Capítulo 6: Finalmente, se exponen las conclusiones, resumiendo los principales hallazgos y aportes de la investigación.

Capítulo 2

Marco Teórico

Contenido

Este capítulo tiene como objetivo establecer los fundamentos teóricos necesarios para comprender el contexto y la importancia del problema de estudio, así como los principios y conceptos que guían el análisis e interpretación de los datos.

En primer lugar, se definen los conceptos básicos del aprendizaje automático, así como algunos de sus componentes fundamentales, como el modelo de una neurona y el perceptrón multicapa, entre otros. Posteriormente, se introducen los elementos clave que componen las redes neuronales artificiales, como el número de épocas, la tasa de aprendizaje y el proceso de retropropagación del error. Se discuten las implicaciones de cada uno de estos factores en el rendimiento de los modelos, así como las estrategias para ajustar estos hiperparámetros de manera efectiva.

En siguientes secciones, se realiza un análisis más detallado sobre las redes convolucionales, que son una extensión natural de las redes neuronales tradicionales y constituyen una herramienta

clave para el procesamiento de datos. Se describen conceptos como las capas convolucionales, las funciones de activación y las técnicas de regularización, las cuales juegan un papel crucial en la mejora del rendimiento de los modelos en tareas como la segmentación de imágenes. También se discuten los desafíos inherentes al entrenamiento de redes convolucionales profundas, como el manejo de grandes volúmenes de datos y la prevención del sobreajuste.

Finalmente, se conecta esta revisión teórica con el objetivo específico de esta investigación, que es el desarrollo y entrenamiento de modelos para la segmentación de imágenes. La comprensión profunda de estos conceptos es esencial para la implementación exitosa de los modelos propuestos y para la correcta interpretación de los resultados obtenidos, que buscan contribuir al avance en el uso de inteligencia artificial en la segmentación de imágenes, un área clave dentro del aprendizaje automático.

2.1. Aprendizaje automático

El *Aprendizaje automático* (ML, por las siglas en inglés *Maching Learning*) es, a su vez, uno de los campos de investigación de la IA que proporciona la capacidad de aprender sin ser programados directamente, es decir, que el programador no indicó la serie de pasos o reglas que se beben de seguir para lograr la tarea, si no que la hace automáticamente [22]. A través del ML, los sistemas pueden identificar patrones en grandes volúmenes de datos, hacer predicciones, y tomar decisiones basadas en esos datos.

El ML se puede dividir como se muestra a continuación:

1. **Aprendizaje Supervisado:** El aprendizaje supervisado es uno de los enfoques más comunes y ampliamente utilizados en el campo del aprendizaje automático. En el aprendizaje supervisado, un modelo se entrena utilizando un conjunto de datos etiquetados, donde cada ejemplo de entrada está emparejado con una salida deseada o etiqueta. El objetivo del modelo es aprender una función que mapee las entradas a las salidas correctas, de manera que pueda hacer predicciones precisas sobre nuevos datos no vistos. Ejemplos comunes incluyen la clasificación (como identificar correos electrónicos como spam o no spam), la regresión (como predecir el precio de una casa basándose en sus características) y la segmentación (dividir una imagen en múltiples partes o regiones, hablaremos más adelante con más detalle). En este trabajo es el que utilizaremos para generar nuestros modelos para tareas de regresión y segmentación.

2. **Aprendizaje semi-supervisado:** El aprendizaje semi-supervisado se utiliza cuando se dispone de una gran cantidad de datos no etiquetados y solo un pequeño conjunto de datos etiquetados. Este método es útil porque etiquetar manualmente grandes cantidades de datos puede ser costoso y laborioso. Con este enfoque, el modelo aprovecha tanto los datos etiquetados como los no etiquetados, usando los primeros para guiar su entrenamiento y mejorando su capacidad para generalizar a partir de la estructura de los datos no etiquetados.

3. **Aprendizaje auto-supervisado:** El aprendizaje auto-supervisado es un tipo de aprendizaje en el que el modelo se entrena para resolver una tarea generada automáticamente a partir de los datos no etiquetados. En este caso, el modelo crea sus propias etiquetas a partir de la estructura interna de los datos y aprende a resolver tareas como completar o predecir

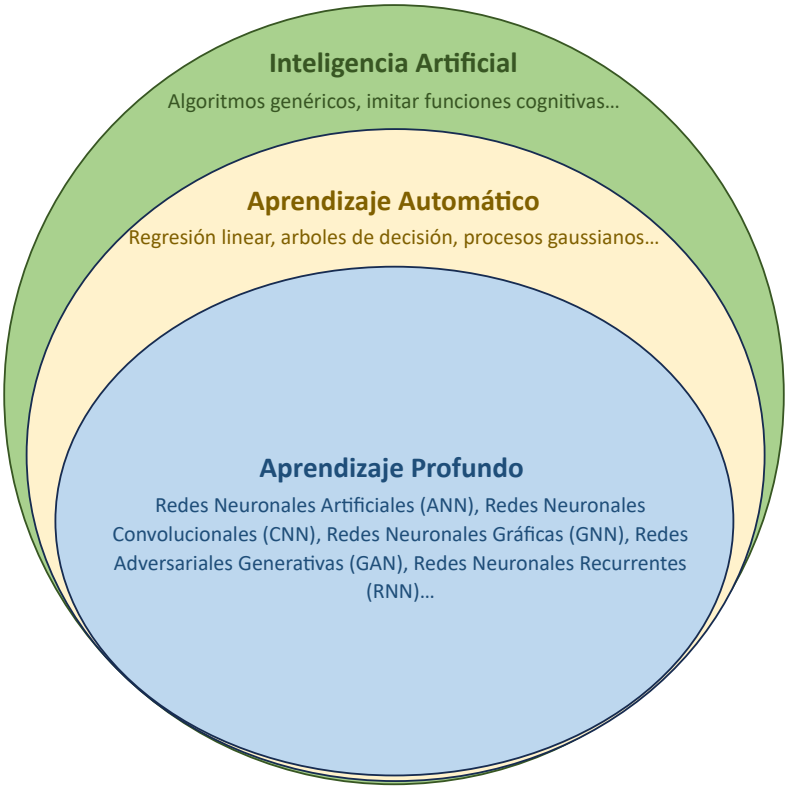


Figura 2.1: El ML es un subconjunto de la IA, que envuelve otros conceptos dentro de si, como lo son las redes neuronales artificiales.

partes faltantes de información. Los modelos que se entrenan en tareas auto-supervisadas suelen aprender representaciones generales de los datos, lo que resulta útil para muchas aplicaciones.

4. **Aprendizaje No Supervisado:** En el aprendizaje no supervisado, el modelo se entrena con datos que no están etiquetados. El objetivo es descubrir patrones ocultos o estructuras subyacentes en los datos. Ejemplos comunes incluyen el clustering (como agrupar clientes con comportamientos de compra similares) y la reducción de dimensionalidad (como reducir el número de variables en un conjunto de datos sin perder información importante).

5. **Aprendizaje por Refuerzo:** El aprendizaje por refuerzo es un enfoque donde un agente aprende a tomar decisiones interactuando con un entorno. El agente recibe recompensas o castigos según las acciones que tome, y su objetivo es maximizar la recompensa acumulada a lo largo del tiempo. Este tipo de aprendizaje es común en aplicaciones como el control robótico, la conducción autónoma y los videojuegos.

2.2. Modelo de una neurona

Una neurona es la unidad de procesamiento de información fundamental para el correcto funcionamiento de un sistema, en cuanto de una red neuronal se trata [23]. El diagrama de la Figura 2.2 muestra el modelo de una neurona, que constituye la base para el diseño de una amplia gama de redes neuronales. En resumen, se puede identificar tres elementos básicos del modelo neuronal:

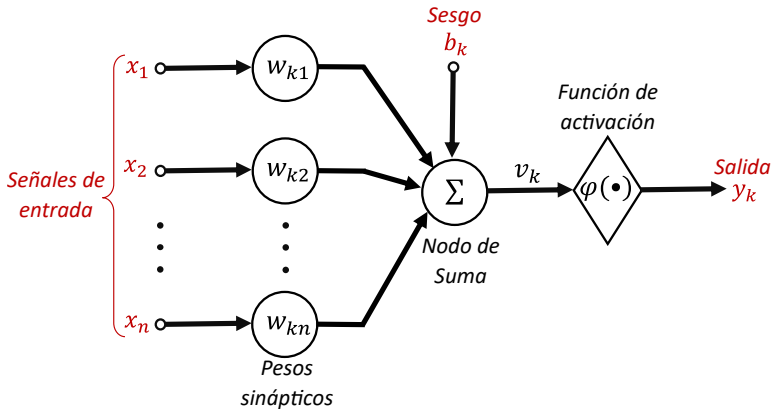


Figura 2.2: Modelo no lineal de una neurona etiquetada como k.

- Primeramente, se tiene un conjunto de *sinapsis*, o *enlaces de conexión*, las cuales enlaza la o las entradas al sumador. Cada una de ellas se caracteriza por un *peso* o *fuerza* propios. En concreto, una señal x_j a la entrada de la sinapsis j conectada a la neurona k se multiplica por el peso sináptico w_{kj}. Es importante tener en cuenta la forma en que se escriben los subíndices del peso sináptico w_{kj}. El primer subíndice en w_{kj} se refiere a la neurona en cuestión, y el segundo se refiere al número de entrada a la que se refiere el peso. Es importante aclarar que, a diferencia del peso de una sinapsis en el cerebro, el peso sináptico de una neurona artificial puede situarse en un rango que incluya valores tanto negativos como positivos.

- Posteriormente, se tiene el nodo sumador, cuya función es sumar las señales de entrada, ponderadas por las respectivas fuerzas sinápticas de la neurona; las operaciones aquí descritas constituyen un combinador lineal.

- Finalmente, se tiene una función de activación, cuya función es limitar la amplitud de la salida de una neurona. La función de activación también se denomina función de aplastamiento, ya que aplasta (limita) el rango de amplitud permisible de la señal de salida a un valor finito. Hay que destacar que normalmente, el intervalo de amplitud normalizado de la salida de una neurona se escribe como el intervalo unitario cerrado [0,1], o, alternativamente, [-1,1].

El modelo neuronal de la Figura 2.2 también incluye un sesgo aplicado externamente, denotado por b_k. El b_k tiene el efecto de aumentar o disminuir la entrada neta de la función de activación, dependiendo de si es positiva o negativa, respectivamente. En términos matemáticos, se puede describir la neurona k las ecuaciones:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

y

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

donde $x_1, x_2, \ldots, x_m$ representan las señales de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ son los pesos sinápticos asociados a la neurona k; u_k (no mostrado en la Figura 2.2) es la salida del combinador lineal en respuesta a las señales de entrada; b_k es el sesgo; $\varphi(\cdot)$ es la función de activación; y y_k es la señal

de salida de la neurona. La inclusión del sesgo b_k aplica una transformación afín a la salida u_k del combinador lineal, como se muestra en el modelo de la Figura, dado por:

$$v_k = u_k + b_k \tag{2.3}$$

Dependiendo de si el sesgo b_k es positivo o negativo, la relación entre el campo local inducido, o potencial de activación, v_k de la neurona k y la salida del combinador lineal u_k se modifica como se ilustra en la Figura 2.3. Cabe destacar que, como resultado de esta transformación afín, la gráfica de v_k en función de u_k ya no pasa por el origen.

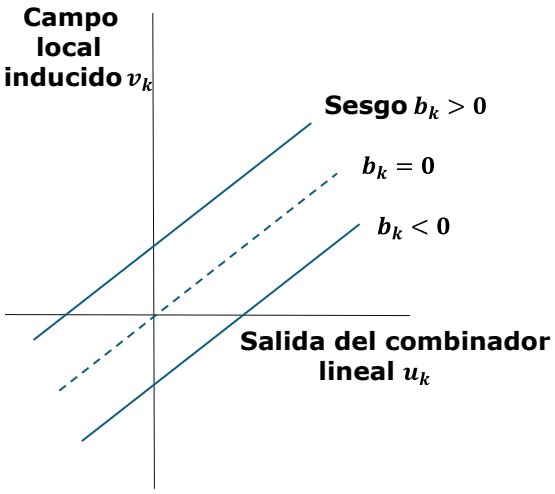


Figura 2.3: Transformación afín producida por la presencia de un sesgo, notando que $v_k = b_k$ en $u_k = 0$.

El sesgo b_k,es un parámetro externo de la neurona k. Podemos tomarla en cuenta por su presencia como en la Ec. 2.2. De forma equivalente, podemos formular la combinación de las Ecs. 2.1 a 2.3 como:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{2.4}$$

y

$$y_k = \varphi(v_k) \tag{2.5}$$

La función de activación ($\varphi(v_k)$), define la salida de la neurona en términos del campo local inducido (v). Suponiendo que utilizamos la función de activación umbral, dada por:

$$\varphi(v) = \begin{cases} 1, & \text{si } v \geq 0 \\ 0, & \text{si } v < 0 \end{cases} \tag{2.6}$$

En ingeniería, esta forma de función umbral se denomina comúnmente *función Heaviside*. En consecuencia, la salida de la neurona k que emplea dicha función umbral se expresa como:

$$y_k = \begin{cases} 1, & \text{si } v_k \geq 0 \\ 0, & \text{si } v_k < 0 \end{cases} \tag{2.7}$$

donde v_k es el campo local inducido de la neurona, dado por:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j + b_k \tag{2.8}$$

2.3. Parámetros e hiperparámetros

Las redes neuronales, como muchos otros modelos de aprendizaje automático, dependen de dos tipos fundamentales de variables que determinan su comportamiento y rendimiento: parámetros e hiperparámetros. Entender la diferencia entre estos dos conceptos es crucial para diseñar, entrenar y optimizar modelos eficaces.

Para entender la diferencia de estos dos, se debe de tener en mente que los parámetros del modelo son internos a la red neuronal, como lo son los pesos de las neuronas. Estos se estiman o aprenden a partir de muestras de entrenamiento.

Por otro lado, los hiperparámetros son variables que no se aprenden durante el entrenamiento, sino que se establecen antes de que comience el proceso de aprendizaje, establecidos por el programador. Estos valores controlan la estructura y el comportamiento del modelo durante el entrenamiento y tienen un impacto significativo en su rendimiento. Un ejemplo es la función de activación a usar o el número de capas ocultas dentro de nuestra red.

En esta sección arribaremos sólo algunos hiperparámetros fundamentales, ya que más adelante trataremos a detalle algunos otros.

2.3.1. Número de Épocas

Define la cantidad de veces que la red procesará todo el conjunto de datos de entrenamiento, conocido como número de épocas. Este valor es crucial, ya que determina cuántas oportunidades tendrá la red de ajustar sus pesos y mejorar en la detección de patrones en los datos. Un mayor número de épocas permite que la red aprenda características más complejas, aunque si es excesivo, aumenta el riesgo de sobreajuste, donde el modelo se ajusta demasiado a los datos de entrenamiento y pierde capacidad para generalizar en datos nuevos. Por otro lado, un número de épocas insuficiente puede limitar el aprendizaje, resultando en un modelo subentrenado y con un rendimiento pobre en predicciones.

Para encontrar el equilibrio adecuado, se suelen probar varios valores de épocas mediante técnicas de validación, teniendo en cuenta el tiempo de entrenamiento y los recursos computacionales disponibles. A menudo, se combina esta búsqueda con el uso de callbacks, como el de Early Stopping, que detiene el entrenamiento automáticamente cuando no se observa mejora significativa, ayudando a evitar el sobreajuste y optimizando el tiempo de entrenamiento.

2.3.2. Tamaño del lote

El tamaño del lote, mejor conocido como *batch size*, indica cuántos ejemplos de entrenamiento se utilizan para calcular la actualización de los parámetros en cada iteración. Un tamaño de lote más pequeño puede hacer que el entrenamiento sea más ruidoso pero potencialmente más efectivo, mientras que un tamaño de lote grande es más estable pero requiere más memoria.

2.3.3. Tasa de aprendizaje

La tasa de aprendizaje, mejor conocida como *Learning Rate*, es uno de los hiperparámetros más críticos en el entrenamiento de redes neuronales y otros modelos de aprendizaje automático. Define el tamaño de los pasos que el algoritmo de optimización da al actualizar los parámetros del modelo, como los pesos y sesgos, durante el proceso de aprendizaje.

Durante el entrenamiento de una red neuronal, el objetivo es minimizar una función de pérdida (loss function), que mide qué tan bien el modelo predice los datos de entrenamiento. Para minimizar esta función, se utilizan algoritmos de optimización como el Descenso del Gradiente (Gradient Descent). La tasa de aprendizaje determina la magnitud de los ajustes que se aplican a los pesos del modelo en cada iteración de este proceso. En general, si es demasiado grande se están dando pasos que permiten avanzar rápido en el proceso de aprendizaje, pero sus actualizaciones pueden terminar llevándolo a ubicaciones completamente aleatorias, y podría saltarse el mínimo. Por el otro lado, si la taza es demasiado pequeña, se harán avances constantes pero pequeños, generando así una mayor oportunidad de llegar al mínimo local de la función de pérdida, a cambio de un mayor tiempo de entrenamiento, y por ende, un mayor costo computacional.

Debido a la importancia de la tasa de aprendizaje, existen varias estrategias y técnicas para ajustarla de manera efectiva:

1. **Tasa de Aprendizaje Constante:** La tasa de aprendizaje se mantiene fija durante todo el proceso de entrenamiento. Esta es la forma más simple, pero puede no ser ideal para todos los problemas.

2. **Tasa de Aprendizaje Variable:** En muchos casos, es beneficioso disminuir la tasa de aprendizaje a medida que avanza el entrenamiento. Esto se puede lograr de diferentes maneras:

    - **Reducción escalonada:** La tasa de aprendizaje se reduce según una programación predefinida, como dividirla por la mitad cada cierto número de épocas.
    - **Ajuste adaptativo:** Técnicas como Adam ajustan la tasa de aprendizaje de manera adaptativa durante el entrenamiento, permitiendo un aprendizaje más rápido al inicio y más cuidadoso conforme se aproxima al óptimo.
    - **Ciclo de tasa de aprendizaje:** Este enfoque alterna entre diferentes tasas de aprendizaje dentro de un rango definido, lo que puede ayudar a escapar de mínimos locales y mejorar la exploración de la función de pérdida.
    - **Programación de tasa de aprendizaje:** Esta técnica implica la reducción gradual de la tasa de aprendizaje en función de alguna condición, como cuando no se observa una mejora en la función de pérdida durante un número determinado de épocas.

2.4. Perceptrón multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés *Multi-Layer Perceptron*) es un modelo de aprendizaje profundo capaz de aprender patrones complejos al entrenar con datos [23]. Es una clase de red neuronal artificial, como podemos ver en la Figura 2.4 que consiste en al menos tres

capas de nodos:

- **Capa de Entrada:** Recibe los datos de entrada, cada neurona en esta capa representa una característica del vector de entrada.
- **Capa de Salida:** Produce el resultado final, cada neurona en esta capa corresponde a una posible clase (en problemas de clasificación) o a un valor continuo (en problemas de regresión).
- **Capas Ocultas:** Estas son las capas entre la entrada y la salida. Las capas ocultas permiten que la red aprenda características intermedias que representan mejor la relación entre la entrada y la salida. Un MLP con más de una capa oculta se considera un modelo profundo, como se menciona más adelante.

Excepto por los nodos de entrada, cada nodo es una neurona que utiliza una función de activación no lineal (ver subsección 2.7), como se mostró en la subsección anterior (2.2).

El proceso de entrenamiento del MLP involucra la propagación hacia adelante y hacia atrás:

- **Propagación hacia adelante:** Los datos de entrada se procesan capa por capa hasta alcanzar la salida. El valor de salida se compara con el valor objetivo, calculando un error.
- **Propagación hacia atrás:** Este error se propaga hacia atrás a través de la red, ajustando los pesos de las conexiones utilizando el algoritmo de retropropagación del error y optimizadores como el Descenso del gradiente.

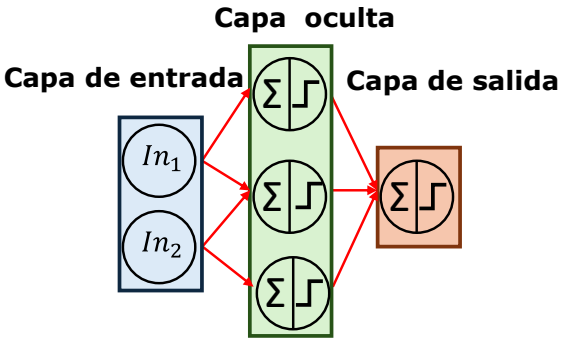


Figura 2.4: Representación gráfica de un perceptrón multicapa con dos neuronas de entrada, una capa oculta con tres neuronas y una capa de salida con dos neuronas.

2.5. Redes neuronales profundas

Las redes neuronales profundas (DNN, por las siglas en inglés *Deep Neural Network*) y el MLP a menudo se mencionan juntos, ya que ambos son tipos de redes neuronales artificiales (ANN, por las siglas en inglés *Artificial Neural Network*) utilizadas para resolver problemas complejos mediante el aprendizaje de patrones a partir de datos [24].

Aunque a menudo se usan indistintamente, el término MLP generalmente se refiere a una estructura simple con una o pocas capas ocultas y una capa de salida. DNN es un término más amplio que incluye MLPs, pero también redes con muchas capas ocultas y arquitecturas más profundas. Las DNN pueden tener muchas capas ocultas, lo que las hace más profundas y, por lo tanto, capaces de aprender representaciones más complejas. Un MLP básico típicamente tiene menos capas. Si bien ambos tipos de redes pueden usarse para tareas similares, las DNN se utilizan con frecuencia en problemas que requieren una alta capacidad de aprendizaje y generalización, como la visión por computadora y el procesamiento del lenguaje natural, donde los MLPs más simples podrían no ser tan efectivos.

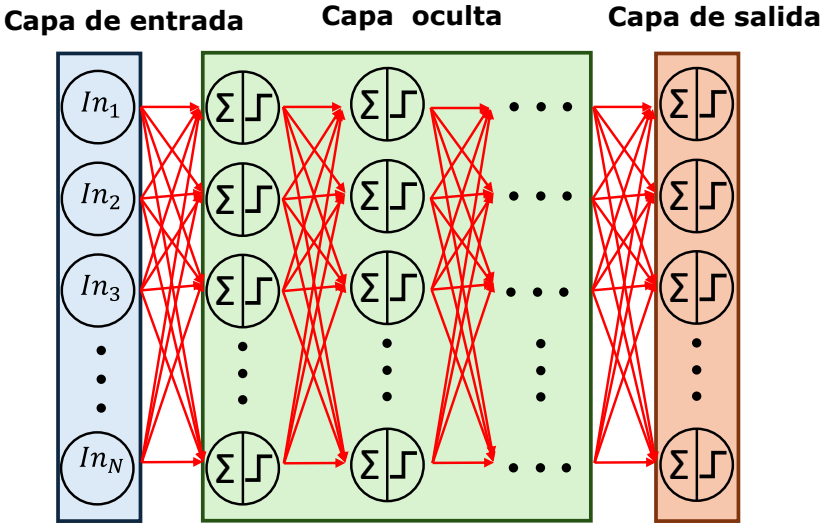


Figura 2.5: Representación gráfica de una DNN genérica con entradas, capas ocultas y salidas múltiples.

2.6. Redes neuronales convolucionales

Las redes convolucionales, comúnmente conocidas como CNN (por sus siglas en inglés, *Convolutional Neural Networks*), son un tipo de ANN que han demostrado ser eficaces en tareas de procesamiento de imágenes y reconocimiento de patrones. Se inspiran en la organización de la corteza visual del cerebro humano y se caracterizan por su capacidad para capturar y procesar características espaciales de las entradas mediante el uso de capas convolucionales.

Las CNN son muy similares a las DNN, ambas formadas por neuronas cuyos parámetros son el sesgo y peso que pueden aprender. Lo que las diferencia es que las CNN hacen la suposición explícita que las entradas son imágenes, lo que permite codificar ciertas propiedades en la arquitectura para reconocer elementos concretos de las imágenes [22].

2.6.1. Operación de convolución

Las capas convolucionales son el elemento fundamental de las redes convolucionales, diseñadas para procesar datos con una estructura en cuadrícula, como las imágenes. El objetivo principal

de estas capas es extraer características de la entrada mediante la aplicación de filtros (también conocidos como kernels) que se aprenden durante el procesamiento de la imagen. Un filtro o kernel es una matriz pequeña de pesos que se desliza sobre la imagen de entrada para extraer características específicas. Cada filtro es responsable de detectar una característica particular en la imagen, como bordes, esquinas o texturas. El tamaño del filtro es un hiperparámetro que define la altura y el ancho del filtro, donde comúnmente se utilizan tamaños pequeños como 3×3, 5×5, etc. El tamaño del filtro afecta la capacidad del modelo para capturar características locales.

Otro parámetro a considerar en la capa de convolución es la profundidad del filtro, la cual debe coincidir con la profundidad de la imagen de entrada. Por ejemplo, si la imagen de entrada es en color (RGB), el filtro tendrá una profundidad de 3 (uno para cada canal de color); por tanto una profundidad de 1 si la imagen está en escala de grises.

Al aplicar la operación de convolución genera un mapa de características por cada filtro aplicado a la imagen de entrada. Estos mapas de características representan la respuesta de la imagen a cada filtro, destacando las características detectadas. Si una capa convolucional aplica múltiples filtros, la profundidad de la salida (número de mapas de características) será igual al número de filtros. En etapas iniciales, los filtros pueden aprender a detectar bordes y texturas básicas. A medida que avanzan las capas, los filtros pueden aprender características más complejas y abstractas, como formas y patrones específicos. Para comprender de mejor forma todo lo antes mencionado, podemos ver el ejemplo de la Figura 2.6.

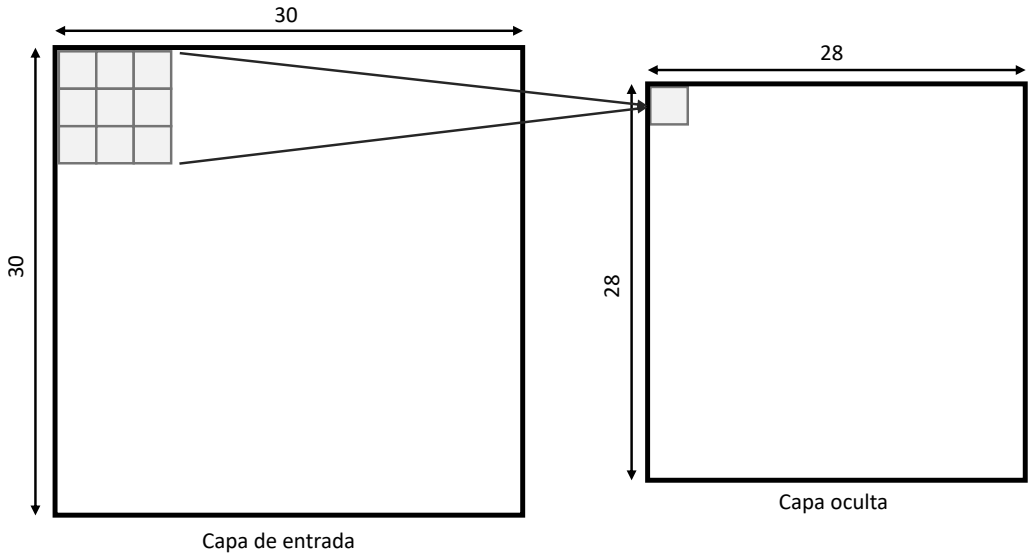


Figura 2.6: Ejemplo de aplicación del bloque de convolución. Aplicamos una ventad a de 3×3 a una imagen de 30×30, resultando en un espacio de 28×28 neuronas en la capa oculta.

En el ejemplo, suponemos que tenemos una imagen de 30×30 píxeles, que esta en escala de grises, es decir, de profundidad 1. A esta imagen le aplicamos una ventada de 3×3 (9 neuronas). Esta ventana se deslizará a lo largo de toda la imagen de entrada de izquierda a derecha y de arriba abajo, empezando por la esquina superior izquierda (como se muestra en la Figura 2.6)y posteriormente dará pasos de 1 píxel a la derecha (solapando algunos píxeles con el anterior), hasta

que llegue al otro extremo. Una vez que termine continuará dando un paso hacia abajo y hará lo mismo hasta que haya pasado por todos los píxeles.

Por cada posición de la ventana hay una neurona en la capa oculta que procesa esta información, como se muestra gráficamente en la Figura 2.6, donde tememos una entrada de 30 × 30 píxeles y una ventana de 3 × 3, lo que nos resulta en una capa oculta de 28 × 28, dado que la ventana sólo puede desplazarse 28 neuronas hacia la derecha y 28 hacia abajo antes de chocar con el extremo derecho o inferior. El valor del pesos de la neurona en la capa oculta será el producto escalar entre el filtro (o ventana) y el conjunto de 9 píxeles (3 × 3) de la capa de entrada.

Lo particular de las redes convolucionales es que se utiliza la misma matriz de pesos del filtro para todas las neuronas de la capa oculta, es decir, que tendremos que ajustar sólo 9 pesos de las neuronas del filtro (en nuestro caso en particular) para obtener la capa oculta, en lugar de ajustar 9 pesos para obtener cada neurona de la capa oculta. Además, esto reduce considerablemente el numero de parámetros que pasa de tener 3 × 3 × 28 × 28, dándonos un total de 7056 a sólo 9 del filtro de 3 × 3.

Todo lo anterior mencionado es un ejemplo básico, ya que podemos ajustar ciertos hiperparámetros, como el tamaño de paso que da el filtro sobre la imagen de entrada (llamado *stride*). Así mismo, podemos obtener una capa oculta del mismo tamaño que el de la entrada, mejorando el resultado del barrido que realiza la ventana que se va deslizando. Para obtenerlo, se aplica una técnica llamada *padding*, la cual consiste en agregar ceros al rededor de la imagen. Además, es común aplicar más de un filtro, es decir, tener más de una matriz de pesos y generar más capas ocultas a partir de una sola imagen. De esta manera, podemos obtener más características partiendo de la misma imagen. La Figura 2.7 se muestra cómo se representa, habitualmente, la capa de convolución.

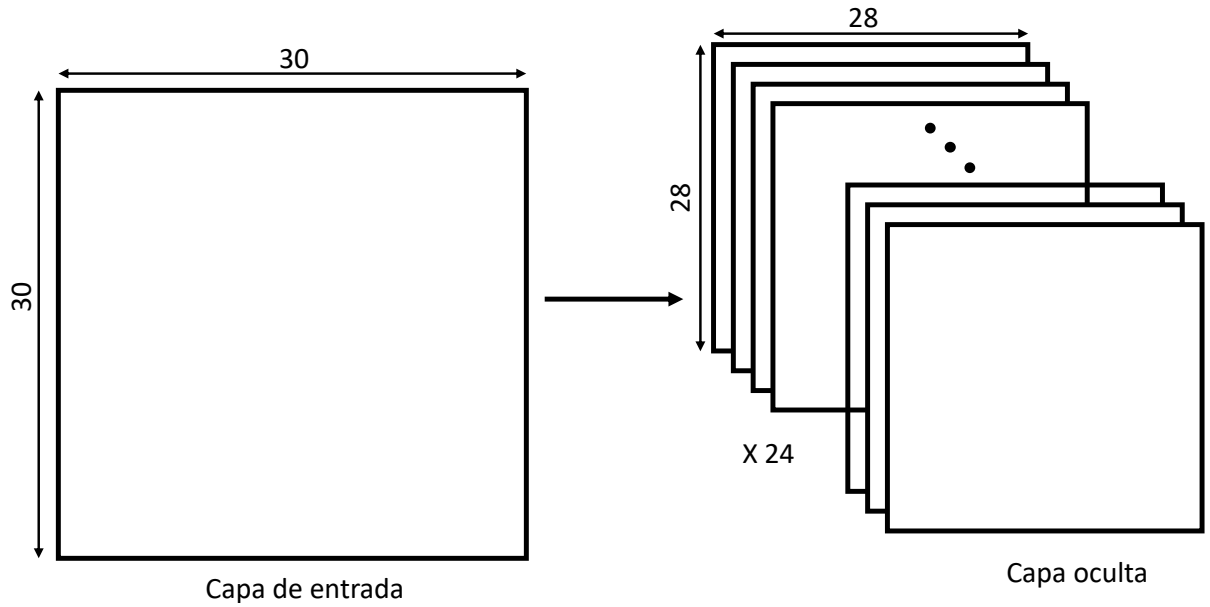


Figura 2.7: Ejemplo de capa de convolución, compuesta por 24 filtros aplicando un kernel de 3 × 3.

2.6.2. Operación de pooling

Las capas de pooling, también conocidas como capas de submuestreo, son una parte esencial de las redes convolucionales. Su función principal es reducir la dimensionalidad de los mapas de características obtenidos de las capas convolucionales, lo que disminuye la cantidad de parámetros y el costo computacional del modelo, y ayuda a prevenir el sobreajuste. Además, estas capas proporcionan cierta invarianza espacial, lo que significa que son capaces de reconocer características importantes independientemente de su ubicación exacta en la imagen. Existen varios tipos de operaciones de pooling, pero las más comunes son:

- **Max Pooling:** Selecciona el valor máximo dentro de una ventana definida que se mueve sobre el mapa de características. Este método se utiliza para capturar las características más prominentes o fuertes. Por ejemplo, en una ventana de 2×2, el valor máximo de esos cuatro píxeles se convierte en el valor del mapa de características reducido.
- **Average Pooling:** Calcula el promedio de todos los valores dentro de la ventana de la misma manera que max pooling, pero en lugar de seleccionar el máximo, toma el promedio. Por ejemplo, en una ventana de 2×2, se suma el valor de los cuatro píxeles y se divide por cuatro para obtener el valor del mapa de características reducido.
- **Global Pooling:** En lugar de usar una ventana móvil, el pooling global aplica una operación de pooling a todo el mapa de características, reduciéndolo a un sólo valor. Esto se hace generalmente hacia el final de la red para resumir toda la información espacial en cada mapa de características.

Esta capa es muy similar a la capa de convolución, ya que es como si aplicáramos una capa de convolución a la capa de convolución, a diferencia que no hay que ajustar parámetros, y que no hay solapamiento, por lo que el tamaño del pooling será la mitad de lo que es la capa de convolución, como se muestra en la Figura 2.8.

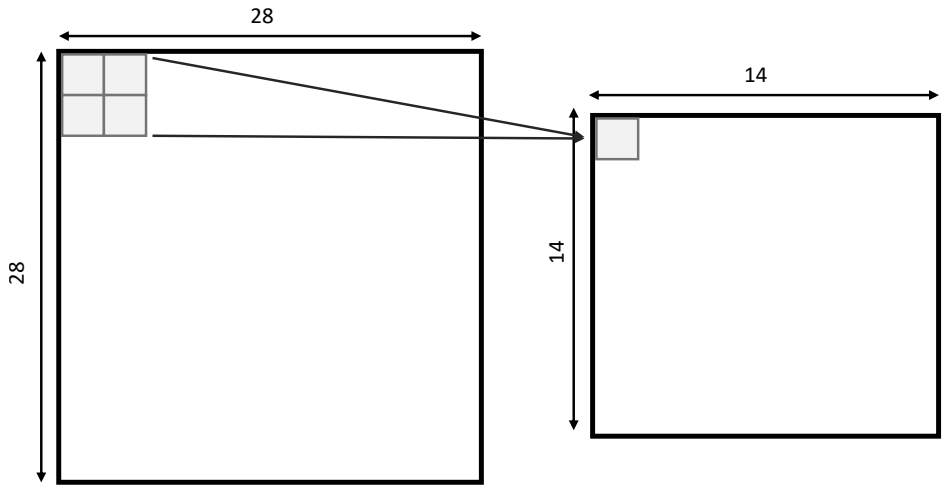


Figura 2.8: Representación de una capa de pooling construida a partir de una ventana de 2×2 que se aplica una capa de convolución de tamaño 28×28.

Debido a que la capa de convolución, regularmente, tiene más de un filtro, por cada filtro se tiene

que tener uno de pooling, por lo que una capa convolcuional con pooling se representa en la Figura 2.9.

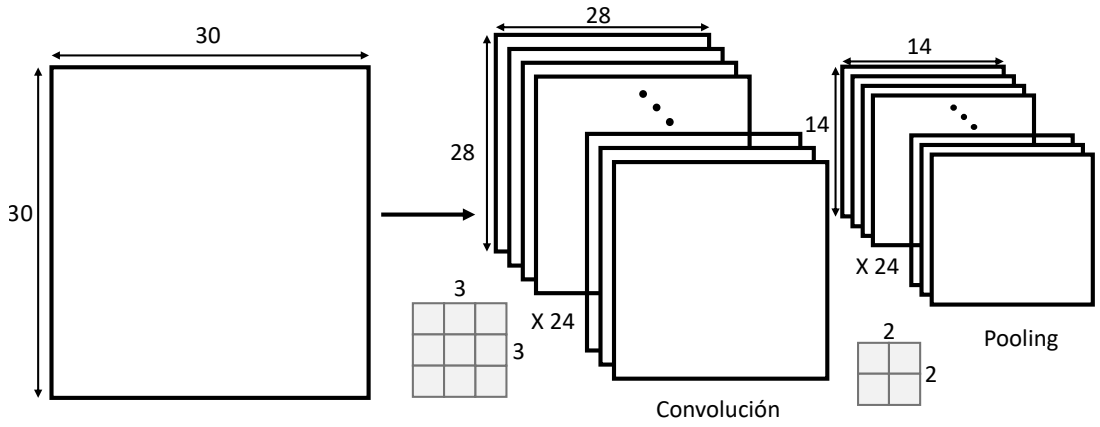


Figura 2.9: Representación al aplicarle una capa de pooling después de haber aplicado 24 filtros de convolución a una imagen de 28 × 28. La capa de pooling tendrá la misma cantidad de filtros pooling como filtros convolucionales.

2.6.3. Capas completamente Conectadas

Al final de la red, después de varias capas convolucionales y de pooling, se suelen incluir una o más capas completamente conectadas. Estas capas funcionan como en un MLP tradicional y son responsables de realizar la clasificación final o la regresión basada en las características aprendidas.

2.6.4. Normalización

La normalización se utiliza para acelerar el entrenamiento y mejorar la estabilidad del modelo. Ayuda a mantener los valores de activación y los gradientes en un rango más controlado. Además, ayuda a evitar el problema del *exploding gradient* (explosión del gradiente), que puede causar que los valores de los pesos aumenten exponencialmente durante el entrenamiento.

Existen varios tipos de normalización que se pueden aplicar en una red convolucional:

- **Normalización por lotes (batch normalization)**: En lugar de normalizar los datos solo en la entrada, la normalización por lotes se aplica a la salida de cada capa antes de la activación y se basa en cada lote de datos durante el entrenamiento.
- **Instance normalization**: Normaliza cada instancia de entrada de manera independiente y es especialmente efectiva en tareas de estilo de imagen, donde cada imagen es tratada de manera independiente.

La normalización se aplica a cada capa convolucional después de que los valores de entrada han sido procesados por la capa. El proceso de normalización consta de dos pasos:

1. **Cálculo de la media y desviación estándar:** Se calcula la media y desviación estándar de los valores de entrada para cada capa.

2. **Normalización de los valores de entrada:** Los valores de entrada se normalizan mediante la fórmula: $(x - \mu)/\sigma$, donde x es el valor de entrada, μ es la media y σ es la desviación estándar.

La normalización tiene varias ventajas en una red convolucional:

- Ayuda a evitar el problema del *exploding gradient*.
- Mejora la estabilidad del entrenamiento.
- Permite un aprendizaje más rápido y preciso.

2.6.5. Dropout

El Dropout es una técnica de regularización utilizada en redes neuronales para reducir el sobreajuste (overfitting). Fue introducida por Geoffrey Hinton y sus colaboradores [25] y se ha convertido en una práctica común en el entrenamiento de modelos profundos.

Las capas de Dropout funcionan apagando aleatoriamente un porcentaje de las neuronas en una capa durante cada paso de entrenamiento, es decir, en cada iteración algunas neuronas no contribuyen ni al proceso de activación ni al de propagación hacia atrás (backpropagation). Sin embargo, durante la fase de inferencia (cuando el modelo hace predicciones), todas las neuronas están activas y contribuyen a la salida final. Podemos describir el funcionamioento del Dropout de la siguiente manera:

1. **Apagado aleatorio de neuronas:** Durante el entrenamiento, cada neurona tiene una probabilidad p de ser apagada (puesta en 0). Esta probabilidad p es un hiperparámetro que se elige antes del entrenamiento (comúnmente se usa $p = 0.5$ en capas ocultas y $p = 0.2$ en la capa de entrada).
2. **Regularización:** Al apagar aleatoriamente neuronas, el modelo no puede depender demasiado de ninguna característica individual. Esto obliga a la red a aprender representaciones más robustas y menos propensas a sobreajustarse a los datos de entrenamiento.
3. **Escalado durante la inferencia:** Durante la inferencia, las salidas de las neuronas no se apagan. En cambio, para mantener la coherencia con el entrenamiento, las activaciones de las neuronas se escalan por el factor $1 - p$, lo que asegura que la magnitud de las activaciones sea consistente entre el entrenamiento y la inferencia.

Al evitar que el modelo dependa excesivamente de características específicas, dropout ayuda a mejorar la capacidad de generalización del modelo, lo que se traduce en un mejor rendimiento en datos no vistos. Una forma de interpretar dropout es como la combinación de muchos modelos más pequeños que comparten pesos. Durante la inferencia, todas las neuronas están activas, lo que equivale a promediar las predicciones de estos modelos más pequeños, reduciendo así la varianza del modelo.

2.6.6. Convolución transpuesta (deconvolución)

En una subsección anterior, se explicó el funcionamiento de la capa de convolución. Así como existen capas que realizan operaciones de convolución, también existe una capa encargada de llevar a cabo la operación inversa. La convolución transpuesta, también conocida como deconvolución, es una operación fundamental en arquitecturas de redes neuronales convolucionales, particularmente en aquellos modelos diseñados para tareas de segmentación de imágenes, superresolución o generación de imágenes (como los autoencoders y las redes generativas adversariales, GANs). A diferencia de la convolución convencional, cuyo objetivo es reducir la dimensión espacial de la entrada y extraer características relevantes, la convolución transpuesta invierte este proceso, ampliando las dimensiones espaciales de la entrada para generar una salida de mayor resolución.

En una convolución tradicional, un filtro (*kernel*) de dimensiones predefinidas se desplaza sobre la imagen de entrada, generando una salida de menor tamaño al combinar los valores de píxeles locales. Este proceso reduce la resolución espacial de la imagen mientras comprime la información, lo que es útil en la fase de decodificador de una red. La convolución transpuesta, en cambio, toma una representación comprimida de la imagen y aplica un kernel de manera que los valores se expanden a lo largo de las dimensiones espaciales, lo que permite aumentar la resolución de la imagen.

Una forma de visualizar el proceso es imaginar cómo una matriz de entrada es "expandida" añadiendo espacios vacíos entre sus valores, que luego son rellenados por el filtro de convolución transpuesta. Al final de este proceso, obtenemos una imagen con mayor resolución que la original. Es importante destacar que la convolución transpuesta no es simplemente una inversión de los pasos de la convolución tradicional, sino que se trata de un proceso en el que también se aprenden los valores de los filtros para generar una salida adecuada.

La convolución transpuesta, al igual que la convolución regular, depende de parámetros clave como el tamaño del filtro , el *stride* (desplazamiento) y el *padding (relleno)*. Estos parámetros controlan cuánto se expande la imagen y de qué manera se distribuyen los valores generados.

2.7. Funciones de activación

Las funciones de activación son componentes esenciales en las redes neuronales artificiales, debido que determinan si una neurona debe activarse o no al recibir una entrada. La función principal de estas es introducir no linealidad en el modelo, permitiendo a la red aprender y representar relaciones complejas que puedan tener los datos. Si no existiera esa no linealidad que proporcionan las funciones de activación, una red neuronal, por más profunda que sea, se comportaría como un modelo lineal simple. En la literatura, podemos encontrar un gran número de funciones de activación. Sin embargo, sólo se introducirán las funciones que se utilizan comúnmente, que son las mismas que se utilizarán en el presente trabajo.

- **Lineal:** La función *lineal* es la forma más simple de función de activación y se define como $f(x) = x$, donde la salida es directamente proporcional a la entrada, como se muestra en la Figura 2.10 a. Aunque, conceptualmente es sencilla y fácil de entender, su principal limitación es la falta de no linealidad. Al utilizar una función lineal como activación en todas las capas de una red neuronal, la red completa se comporta como un modelo lineal, independientemente de cuántas capas tenga. Esto significa que pierde la capacidad de aprender y representar

relaciones complejas en los datos. Por esta razón, la función lineal se usa principalmente en la capa de salida para tareas de regresión, donde el objetivo es predecir un valor continuo.

- **Sigmoide:** La función *sigmoide* o logística transforma las entradas en un rango entre 0 y 1, lo que la hace útil para problemas de clasificación binaria (Figura 2.10 b). Sin embargo, presenta problemas como el desvanecimiento del gradiente, lo que dificulta el entrenamiento de redes profundas.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

- **Tanh (tangente hiperbólica):** La función *tanh* es similar a la *sigmoide*, pero con un rango más amplio, de -1 a 1 (Figura 2.10 c). Esto puede ser útil en problemas de clasificación multi-clase.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.10}$$

- **ReLU (Rectified Linear Unit):** La función *ReLU* toma el resultado de la capa y lo pone a cero si es negativo. De esta manera, evita que los valores negativos afecten el aprendizaje del modelo (Figura 2.10 d).

$$ReLU(x) = max(0, x) \tag{2.11}$$

- **Softmax:** La función *softmax* es una extensión de la *sigmoide*, se utiliza principalmente en la capa de salida de redes neuronales para tareas de clasificación multiclase. Convierte un vector de valores en un vector de probabilidades, donde cada valor representa la probabilidad de pertenencia a una clase específica.

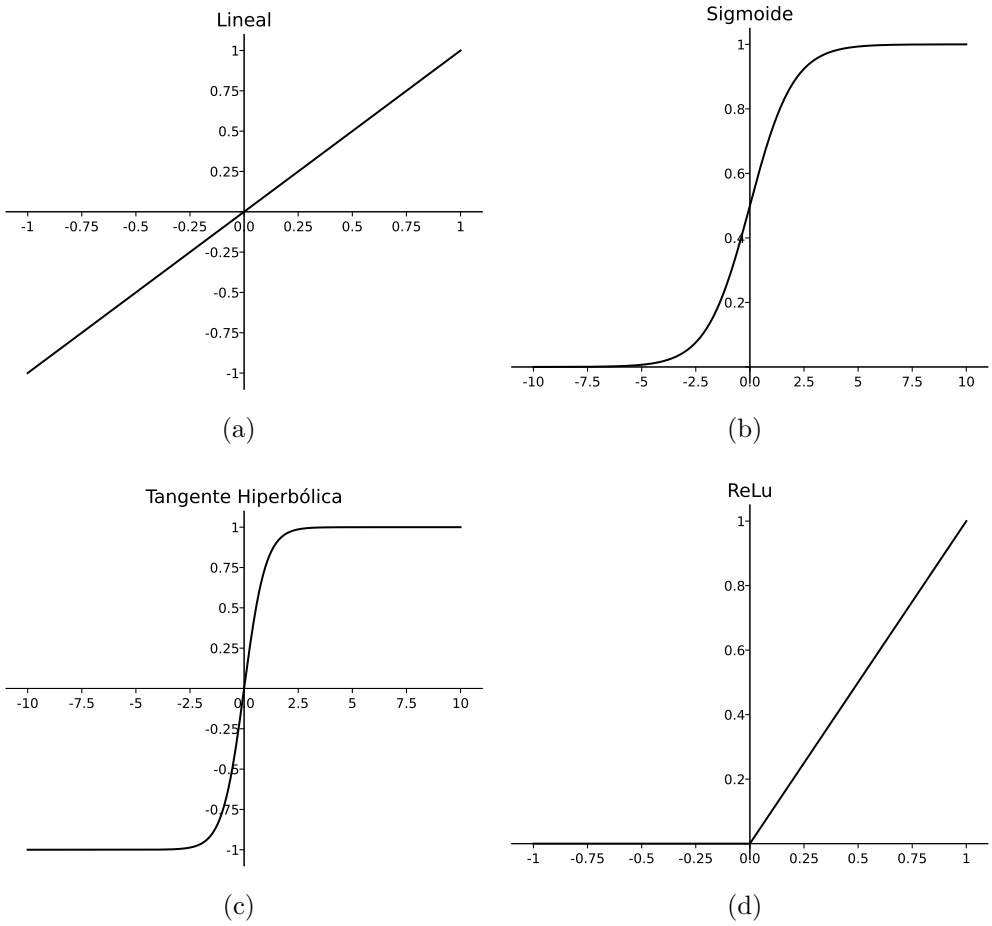
$$softmax(x) = \frac{e^{x}}{\sum_{j=1}^{k} e^{x}} \tag{2.12}$$

La elección de la función de activación puede tener un impacto significativo en el rendimiento de una red neurona, ya que cada función de activación tiene sus ventajas y desventajas:

- *ReLU* es rápida y eficiente, pero puede causar problemas si los valores negativos son importantes para el modelo.
- *Sigmoide* y *tanh* pueden ser más lentas y costosas computacionalmente, pero pueden ser útiles en problemas específicos.
- *Softmax* es ideal para problemas de clasificación multi-clase, pero puede requerir normalización adicional.

2.8. Optimizadores

En el ámbito del aprendizaje automático, un optimizador es un algoritmo diseñado para ajustar los parámetros de un modelo, como los pesos en una red neuronal, con el propósito de minimizar la función de pérdida. Esta función cuantifica la discrepancia entre las predicciones del modelo y los datos reales. El objetivo principal de un optimizador es identificar los valores óptimos de los parámetros que reducen esta pérdida, lo que a su vez mejora el rendimiento del modelo. Los optimizadores se clasifican en varias categorías según el método que emplean para actualizar los parámetros:



(a) (b) (c) (d)

Figura 2.10: Representación gráfica de la función de activación a) *lineal*, b) *sigmoide*, c) *tanh* y e) *ReLU*.

- Optimizadores basados en el gradiente, como el Descenso del Gradiente y el Descenso del Gradiente Estocástico (SGD), que utilizan el gradiente de la función de pérdida para actualizar los parámetros.
- Optimizadores que incorporan momento, como SGD con momentum y Gradiente Acelerado de Nesterov (NAG), que aceleran las actualizaciones mediante la acumulación de gradientes pasados.
- Optimizadores adaptativos, que ajustan las tasas de aprendizaje individualmente para cada parámetro, incluyendo AdaGrad (que adapta la tasa de aprendizaje según la frecuencia de actualizaciones), RMSProp (que utiliza una media móvil de los cuadrados de los gradientes), Adam (que combina los beneficios de AdaGrad y RMSProp), AdamW (una variante de Adam que incluye decaimiento de peso), Nadam (que combina Adam con el método de Nesterov) y AdaDelta (que utiliza una ventana de decaimiento para las actualizaciones). Optimizadores basados en la segunda derivada, como el Método de Newton y L-BFGS(por las siglas en inglés *Limited-memory Broyden-Fletcher-Goldfarb-Shanno*), que utilizan información de la curvatura de la función de pérdida para realizar actualizaciones más precisas [26].

2.8.1. Descenso del gradiente estocástico

El SGD, por las siglas en inglés *Stochastic Gradient Descent*, es uno de los algoritmos de optimización más utilizados en el entrenamiento de redes neuronales. Se deriva del método de Descenso del Gradiente y ha sido fundamental en la formación de modelos de aprendizaje profundo.

El objetivo del SGD es minimizar una función de pérdida, que mide qué tan bien está funcionando el modelo. La función de pérdida es una medida del error entre las predicciones del modelo y los valores reales. los parámetros del modelo se actualizan iterativamente en la dirección negativa del gradiente de la función de pérdida. La fórmula general de actualización es:

$$\theta_{t+1} = \theta_t - \eta \nabla L\left(\theta_t; x_i; y_i\right) \tag{2.13}$$

donde:

- θ_t: Parámetros del modelo en el paso t.
- η: Tasa de aprendizaje, un hiperparámetro que determina el tamaño de los pasos de actualización.
- $\nabla L\left(\theta_t; x_i; y_i\right)$: Gradiente de la función de pérdida L con respecto a los parámetros θ_t usando una muestra $(x_i; y_i)$ de los datos.

A diferencia del Descenso del Gradiente que utiliza todo el conjunto de datos para calcular el gradiente, el SGD utiliza sólo una muestra (o un minibatch) en cada iteración. Esto hace que el SGD sea más rápido y escalable en conjuntos de datos grandes, aunque introduce ruido en las actualizaciones de los parámetros.

Ventajas:

- **Eficiencia computacional:** Procesar un subconjunto de datos reduce el tiempo de cómputo en cada iteración.
- **Convergencia rápida:** Puede llegar a un mínimo más rápidamente en grandes conjuntos de datos debido a su naturaleza estocástica.
- **Escapatoria de mínimos locales:** El ruido introducido por el uso de muestras aleatorias puede ayudar a escapar de mínimos locales.

Desventajas:

- **Ruido en convergencia:** El proceso estocástico puede causar que las actualizaciones sean ruidosas, lo que puede dificultar la convergencia a un mínimo global.
- **Sensibilidad a la tasa de aprendizaje:** Una tasa de aprendizaje mal elegida puede llevar a convergencia lenta o inestabilidad.

2.8.2. AdaDelta

Adadelta es un optimizador que se basa en el concepto de adaptación de la tasa de aprendizaje, y es una extensión del método AdaGrad. Utiliza sólo la información sobre las actualizaciones más recientes, evitando la necesidad de un ajuste manual de la tasa de aprendizaje. Fue propuesto para abordar algunas de las limitaciones de AdaGrad, especialmente la rápida disminución de la tasa de aprendizaje que puede ocurrir en problemas de aprendizaje profundo.

Adadelta utiliza dos promedios móviles, uno para el gradiente y otro para las actualizaciones. A continuación se presentan las fórmulas principales:

1. **Acumulación de los cuadrados de los gradientes:**

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \tag{2.14}$$

2. **Acumulación de las actualizaciones:**

$$E[\Delta x^2]_t = \beta E[\Delta x^2]_{t-1} + (1-\beta) \Delta x_t^2 \tag{2.15}$$

3. **Actualización de parámetros:**

$$\Delta x_t = -\frac{\sqrt{E[\Delta x^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.16}$$

4. **Actualización final del parámetro:**

$$x_t = x_{t-1} + \Delta x_t \tag{2.17}$$

donde:

- β: Valor que controla el decaimiento exponencial de las tasas de aprendizaje.
- ϵ: Un pequeño número para evitar la división por cero, que se suele establecer en 10^{-6}.
- g_t: Gradiente en el tiempo t.

Ventajas:

- No requiere una tasa de aprendizaje inicial, lo que simplifica el proceso de ajuste de hiperparámetros.
- Es robusto y efectivo para problemas donde se requiere un entrenamiento más profundo.
- Utiliza sólo información de las actualizaciones más recientes, lo que puede ayudar a evitar problemas de ruido en los gradientes.

Desventajas:

- Puede ser menos eficiente en algunos problemas en comparación con optimizadores más avanzados como Adam.
- Puede converger más lentamente en algunos casos.

2.8.3. RMSProp

RMSProp (por su nombre en inglés *Root Mean Square Propagation*) es un optimizador que también adapta las tasas de aprendizaje de cada parámetro, y es especialmente útil en redes neuronales profundas. Fue propuesto para resolver algunas limitaciones de AdaGrad, que puede llevar a una rápida disminución de la tasa de aprendizaje.

Utiliza un promedio móvil para los cuadrados de los gradientes, lo que le permite adaptarse a las condiciones del optimizador a lo largo del tiempo. Agrega un término pequeño (ϵ) a la raíz cuadrada del promedio móvil para evitar divisiones por cero. RMSProp, se caracteriza por las siguientes fórmulas:

1. **Acumulación de los cuadrados de los gradientes:**

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2 \tag{2.18}$$

2. **Actualización de parámetros:**

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.19}$$

donde:

- $E[g^2]_t$: Promedio móvil de los cuadrados de los gradientes en el tiempo t.
- β: Hiperparámetro que controla el decaimiento exponencial.
- g_t: Gradiente en el tiempo t.
- θ_t: Parámetro actualizado en el tiempo t.
- α: Tasa de aprendizaje inicial.
- ϵ: Un pequeño número para evitar la división por cero.

Ventajas:

- Mantiene tasas de aprendizaje adaptativas que pueden ser más efectivas para problemas de aprendizaje profundo.
- Resuelve el problema de la disminución rápida de la tasa de aprendizaje que se presenta en AdaGrad.
- Proporciona un equilibrio entre la velocidad de convergencia y la estabilidad.

Desventajas:

- Puede ser menos efectivo en ciertos problemas en comparación con optimizadores más avanzados como Adam.
- La selección de los hiperparámetros todavía puede ser necesaria.

2.8.4. Adam

El optimizador Adam (*Adaptive Moment Estimation*) es uno de los más populares en el entrenamiento de modelos de aprendizaje automático, especialmente en redes neuronales profundas. Adam es un optimizador que combina las ventajas de dos métodos de optimización: AdaGrad y RMSProp. Utiliza el momento para acelerar el entrenamiento y mejorar la convergencia. Sus principales características es que Adam ajusta la tasa de aprendizaje para cada parámetro de forma individual, lo que permite un entrenamiento más eficiente. Utiliza tanto el promedio móvil de los gradientes como el promedio móvil de los cuadrados de los gradientes, lo que ayuda a suavizar el ruido en la dirección del gradiente. Es generalmente más rápido en converger hacia soluciones óptimas en comparación con otros optimizadores. Adam mantiene dos vectores de momentos, que se representan en las siguientes ecuaciones:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{2.20}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{2.21}$$

donde:

- m_t : Promedio móvil de los gradientes (primer momento).
- v_t: Promedio móvil de los cuadrados de los gradientes (segundo momento).
- g_t: Gradiente en el tiempo t.
- β_1 y β_1: Hiperparámetros que controlan el decaimiento exponencial.

Debido que m_t y v_t son inicializados en cero, se aplican correcciones de sesgo:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.22}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.23}$$

Ventajas:

- Rápido y efectivo en la mayoría de los casos.
- Funciona bien en problemas con gran cantidad de datos y parámetros.
- Menos necesidad de ajustar la tasa de aprendizaje manualmente.

Desventajas:

- Puede ser menos efectivo en problemas con datos muy ruidosos.
- A veces, puede converger a una solución subóptima en comparación con otros optimizadores en tareas específicas.

2.8.5. AdamW

AdamW es una variante del optimizador Adam que introduce un enfoque de regularización de pesos más efectivo. AdamW mantiene las características principales de Adam, como el promedio móvil de los gradientes y de los cuadrados de los gradientes, pero realiza una actualización adicional para la regularización de pesos. En Adam, la regularización L2 (o decaimiento de peso) se aplicaba de manera que afectaba a las actualizaciones del gradiente, lo que podía causar problemas de convergencia. AdamW separa el proceso de actualización de los pesos y la regularización, lo que permite una mejor optimización y un mejor rendimiento en la generalización.

Ha demostrado ser más efectivo en términos de generalización en comparación con Adam tradicional, especialmente en problemas donde el sobreajuste es un problema. Las fórmulas que lo caracterizan son las siguientes:

1. **Actualización de los momentos:**

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.24}$$

2. **Actualización de los cuadrados de los gradientes:**

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.25}$$

3. **Correción de sesgo:**

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.26}$$

4. **Actualización de parámetros con regularización L2:**

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t - \lambda \theta_{t-1} \tag{2.27}$$

donde:

- β_1 y β_2: Controlan el promedio móvil de los momentos y el promedio móvil de los cuadrados de los gradientes, respectivamente.
- λ: Tasa de regularización L2.

Ventajas:

- Mejora la capacidad de generalización en muchos problemas.
- Menor riesgo de sobreajuste debido a la correcta aplicación de la regularización.
- Conserva la velocidad y eficacia del optimizador Adam.

Desventajas:

- Puede requerir una búsqueda más cuidadosa de hiperparámetros.
- En algunos casos, puede ser innecesario si el modelo ya generaliza bien con Adam.

2.8.6. Nadam

Nadam (*Nesterov-accelerated Adaptive Moment Estimation*) es una variante de Adam que combina el método de Nesterov con el enfoque de Adam. Este optimizador se diseñó para mejorar la convergencia y la eficiencia al ajustar la tasa de aprendizaje adaptativa de los parámetros. Combina el promedio móvil de los gradientes (como Adam) con la técnica de aceleración de Nesterov, que proporciona un tipo de "mirada hacia adelante" en las actualizaciones de los parámetros. Nadam utiliza el término de gradiente del futuro, lo que permite un ajuste más efectivo de la dirección y la magnitud de la actualización, como lo vemos en las siguientes fórmulas:

1. **Actualización de los momentos:**

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.28}$$

2. **Actualización de los cuadrados de los gradientes:**

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.29}$$

3. **Corrección de sesgo:**

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.30}$$

4. **Actualización de parámetros con Nesterov:**

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_t + (1 - \beta_1) g_t \right) \tag{2.31}$$

Ventajas:

- Mejora la convergencia en comparación con Adam y otras variantes.
- Combina lo mejor de Adam y Nesterov, lo que puede llevar a actualizaciones más efectivas de los parámetros.

Desventajas:

- Puede ser más computacionalmente costoso debido al cálculo adicional del término de Nesterov.
- A veces puede ser menos efectivo en ciertos problemas que optimizadores más simples.

2.8.7. AdaGrad

AdaGrad (*Adaptive Gradient Algorithm*) es un optimizador que adapta la tasa de aprendizaje para cada parámetro en función de las actualizaciones anteriores. Es particularmente útil para problemas de optimización con características dispersas, ya que aumenta la tasa de aprendizaje para parámetros poco frecuentes y la disminuye para parámetros más frecuentes. Ajusta la tasa de aprendizaje de cada parámetro individualmente, lo que permite que el algoritmo tenga un enfoque más selectivo al actualizar parámetros con diferentes frecuencias. Aumenta la tasa de aprendizaje para parámetros con gradientes raros y la reduce para parámetros con gradientes frecuentes. AdaGrad se representa por las siguientes ecuaciones:

1. **Acumulación de los cuadrados de los gradientes:**

$$G_t = G_{t-1} + g_t^2 \tag{2.32}$$

2. **Actualización de parámetros:**

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{G_t} + \epsilon} g_t \tag{2.33}$$

Ventajas:

- Efectivo para problemas con características dispersas (por ejemplo, texto o datos de imágenes).
- No necesita ajustar manualmente la tasa de aprendizaje después del entrenamiento.

Desventajas:

- Puede llevar a una disminución demasiado rápida de la tasa de aprendizaje, lo que puede hacer que el modelo no converja.
- En problemas no lineales, la acumulación de gradientes puede resultar en un rendimiento subóptimo.

2.9. Funciones y métricas de evaluación

Las funciones de evaluación son herramientas fundamentales para medir la calidad y precisión de un modelo durante las etapas de entrenamiento y prueba. Estas funciones son cruciales, ya que permiten evaluar el rendimiento del modelo y ajustar sus parámetros con el fin de optimizar su desempeño. Por otro lado, las métricas de evaluación son criterios que se utilizan para medir la eficacia de un modelo una vez que ha sido entrenado. Estas métricas ofrecen una forma de cuantificar el rendimiento del modelo en tareas específicas.

Existen diversas funciones y métricas de evaluación, y a continuación se describen algunas de las más relevantes que se emplean para evaluar los modelos entrenados en este trabajo.

2.9.1. Precisión

La *precisión* es una métrica fundamental en el ámbito del aprendizaje automático y la estadística, utilizada para evaluar el rendimiento de los modelos de clasificación. Se define como la proporción de verdaderos positivos sobre el total de predicciones positivas realizadas por el modelo. En términos matemáticos, la precisión se puede expresar mediante la siguiente fórmula:

$$P = \frac{TP}{TP + FP} \tag{2.34}$$

donde:

- **TP (True Positives):** Verdaderos positivos, es decir, el número de casos que fueron correctamente clasificados como positivos.

- **FP (False Positives):** Falsos positivos, es decir, el número de casos que fueron incorrectamente clasificados como positivos.

La *precisión* es especialmente relevante en contextos donde el costo de un falso positivo es significativo, donde es crucial que el modelo mantenga una alta precisión para evitar errores que puedan afectar la salud o la seguridad de las personas.

Es importante destacar que la precisión por sí sola no proporciona una visión completa del rendimiento del modelo. Por ello, a menudo se complementa con otras métricas, como el recall (sensibilidad) y el F1-score, que ofrecen una perspectiva más equilibrada del desempeño del modelo en la detección de casos positivos y negativos.

2.9.2. Recall

La función *recall* o *sensibilidad* es una métrica de evaluación fundamental en el aprendizaje automático, especialmente en tareas de clasificación, ya que mide la capacidad del modelo para identificar correctamente todos los casos positivos. El *recall* se define como la proporción de verdaderos positivos sobre el total de casos positivos reales. La fórmula para calcular el *recall* es la siguiente::

$$S = \frac{TP}{TP + FN} \tag{2.35}$$

donde**FN (False Negatives)** son falsos negativos, es decir, el número de casos que fueron incorrectamente clasificados como negativos cuando deberían haber sido positivos. Una *sensibilidad* alta indica que el modelo es bueno para detectar las instancias reales, pero no necesariamente significa que el modelo esté clasificando correctamente. Por ejemplo, si un modelo médico tiene una *sensibilidad* del 90 %, eso significa que 90 % de los pacientes con la enfermedad en cuestión serán detectados como positivos. La *sensibilidad* es especialmente útil en problemas de detección de anomalías o outliers (valores atípicos), donde se prioriza la detección de las instancias positivas sobre la *precisión*.

2.9.3. F1-score

El F1-score es una métrica utilizada en el ámbito del aprendizaje automático para evaluar el rendimiento de modelos de clasificación, especialmente en situaciones donde hay un desequilibrio entre las clases. Combina la *precisión* y *sencibilidad* en un solo valor, proporcionando una medida más equilibrada de la eficacia del modelo en la identificación de casos positivos, y se expresa matemáticamente de la siguiente manera:

$$F = 2\frac{P * R}{P + R} \tag{2.36}$$

donde:

- **P** es la *precisión*.
- **R** es *recall*.

Un *F1-score* alto indica que el modelo está logrando un equilibrado rendimiento en términos de precisión y recall. Un *F1-score* cercano a 1 es ideal, mientras que un *F1-score* cerca de 0.5 puede indicar que el modelo está teniendo dificultades para alcanzar una buena combinación de *precisión* y *recall*.

2.9.4. Accuracy

El *accuracy* (*exactitud*)se refiere a la capacidad del modelo para clasificar correctamente todas las instancias en un conjunto de datos. En otras palabras, es la proporción de instancias que el modelo clasifica correctamente entre todas las instancias. El *accuracy* se define como la relación entre el número de instancias correctas clasificadas y el total de instancias. Su fórmula está dada por:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.37}$$

donde **TN (True Negatives)** son verdaderos negativos, es decir, el número de casos que fueron correctamente clasificados como negativos. Un *accuracy* alto indica que el modelo está clasificando correctamente la mayoría de las instancias. Sin embargo, un *accuracy* bajo puede indicar que el modelo está teniendo dificultades para clasificar correctamente algunas instancias.

2.9.5. Error cuadrático medio

El error cuadrático medio (*MSE*, por las siglas en ingles *Mean Square Error*) mide el promedio de los errores al cuadrado entre los valores predichos por el modelo y los valores reales. Cuanto menor sea el valor del MSE, más preciso es el modelo en sus predicciones. Matematicamente, se define de la siguiente manera:

$$M = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pred})^2 \tag{2.38}$$

donde:

- $\mathbf{y_i}$: Se refiere al valor real, es decir, el valor que se tiene etiquetado o establecido en conjunto de datos.
- **pred**: Es el valor predicho.

donde **n** es el número total de instancias en el conjunto de datos.

Un *MSE* bajo indica que las predicciones del modelo están cerca de los valores reales, lo que sugiere un buen rendimiento. Sin embargo, un MSE alto puede indicar que las predicciones del modelo están lejos de los valores reales, lo que sugiere un mal rendimiento.

2.9.6. Error absoluto medio

El error absoluto medio (*MAE*, por las siglas en inglés *Mean Absolut Error*) a diferencia del MSE, no eleva al cuadrado los errores, por lo que trata de manera uniforme todos los errores, independientemente de su tamaño.Matemáticamente, el MAE se define como:

$$M = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_{pred}| \tag{2.39}$$

Un *MAE* bajo indica que las predicciones del modelo están cerca de los valores reales, lo que sugiere un buen rendimiento. Sin embargo, un *MAE* alto puede indicar que las predicciones del modelo están lejos de los valores reales, lo que sugiere un mal rendimiento.

2.9.7. Coeficiente de determinación

El coeficiente de determinación, también conocido como R^2 o *R2-Score*, es una métrica que mide la proporción de la varianza de la variable dependiente que el modelo es capaz de explicar en relación con la varianza total. En otras palabras, indica qué tan bien el modelo se ajusta a los datos observados y qué porcentaje de la variabilidad de los datos logra explicar. Un valor de R^2 cercano a 1 sugiere un buen ajuste, mientras que un valor cercano a 0 indica que el modelo explica muy poca de la variación en los datos. La fórmula para el R^2 está dada por:

$$R^2 = 1 - \frac{SSE}{SST} \tag{2.40}$$

donde:

- **SSE** es la suma de los errores cuadrados entre las predicciones y los valores reales.
- **SST** es la suma de las diferencias entre los valores reales y la media de los valores reales.

Un R^2 cercano a 1 indica que el modelo explica bien la varianza de los datos, lo que sugiere un buen rendimiento. Sin embargo, un R^2 muy por de bajo de 1, puede indicar que el modelo no explica bien la variabilidad de los datos, lo que sugiere un mal rendimiento.

2.9.8. Intersección sobre la unión

La intersección sobre la unión (*IoU*, por las siglas en inglés *Intersection over Union*), también conocido como coeficiente de Jaccard, es una métrica clave en la evaluación de modelos de segmentación de imágenes, ya que mide cuán bien un modelo segmenta una clase particular (o todas las clases) en comparación con el "ground truth" (la verdad de terreno, es decir, las etiquetas reales). El *IoU* se define como la relación entre el área de intersección y el área de unión entre la máscara predicha por el modelo y la máscara verdadera. Es decir:

$$IoU = \frac{\text{Intersección}}{\text{Unión}} = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN} \tag{2.41}$$

donde:

- **Intersección:** Es el área común entre la región predicha por el modelo y la región verdadera.
- **Unión:** Es la unión total de ambas áreas, tanto la predicha como la verdadera.

donde:

- **A** representa el conjunto de píxeles en la máscara predicha (segmentada por el modelo).
- **B** representa el conjunto de píxeles en la máscara real (ground truth).

En la segmentación de imágenes, se calcula el *IoU* para cada clase (cada objeto o región de interés) en una imagen. Luego, se puede calcular el *IoU* promedio sobre todas las clases (esto se llama *mIoU* o Mean *IoU*).

En segmentación de imágenes, el *IoU* suele ser una métrica más precisa y ajustada que la precisión porque la precisión puede ser engañosa si el modelo predice demasiados píxeles de fondo correctamente pero falla en predecir los objetos correctamente.

2.9.9. Dice coefficient

El *Dice coefficient*, también conocido como *coeficiente de similaridad de Dice* (lo llamaremos *Dice*, para simplificar), es una métrica utilizada para evaluar la similitud entre dos conjuntos, especialmente en problemas de segmentación de imágenes. Se emplea ampliamente en áreas como el procesamiento de imágenes biomédicas, visión por computadora y reconocimiento de patrones. Su principal objetivo es medir qué tan bien una predicción coincide con las etiquetas reales en tareas de segmentación. Matemáticamente, se representa como de la siguiente forma:

$$Dice = \frac{2 \times |A \cap B|}{|A| + |B|} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.42}$$

Un $Dice = 1$ significa que hay una coincidencia perfecta entre la predicción y la máscara real, lo que indica que todos los píxeles segmentados están correctamente clasificados. Por otro lado, un *Dice* $= 0$ indica que no hay coincidencia alguna entre los conjuntos, es decir, ningún píxel segmentado coincide con la máscara real.

Dice es ampliamente utilizado en tareas de segmentación de imágenes médicas y otros problemas de segmentación binaria, debido a su sensibilidad en la comparación de regiones superpuestas. Una ventaja importante del Dice Coefficient es que es especialmente adecuado para clases de objetos pequeños, ya que toma en cuenta tanto la precisión como la cobertura en la región predicha.

2.9.10. Especificidad

La *especificidad* es una métrica que mide la proporción de verdaderos negativos entre todos los negativos reales. Es especialmente útil en problemas donde es importante saber cuántos de los negativos se identifican correctamente. La *especificidad* se calcula de la siguiente manera:

$$Especificidad = \frac{TN}{TN + FP} \tag{2.43}$$

Una especificidad alta indica que el modelo es efectivo para identificar la clase negativa, minimizando la cantidad de falsos positivos. Es particularmente importante en contextos donde los falsos positivos tienen consecuencias significativas (por ejemplo, pruebas médicas).

La especificidad es crucial en diversas aplicaciones, como en diagnósticos médicos, donde es esencial no sólo identificar correctamente a los enfermos (alta sensibilidad) sino también evitar clasificar incorrectamente a los sanos como enfermos.

2.10. Segmentación

La segmentación de imágenes es la tarea de encontrar grupos de píxeles que "van juntos" al dividir una imagen digital en varias regiones o segmentos, con el objetivo de simplificar la representación de la imagen y hacerla más fácil de analizar [27]. Existe una gran variedad de algoritmos que hacen estas tareas, basados en distintos métodos, como se muestran a continuación:

1. **Segmentación basada en bordes:** Se enfoca en detectar bordes dentro de la imagen, que son cambios abruptos en la intensidad de los píxeles. Los bordes generalmente corresponden a los límites entre diferentes objetos [28]. Debido a lo anterior mencionado, es difícil detectar bordes suaves, por lo que el ruido en la imagen puede causar falsos positivos.

2. **Segmentación basada en regiones:** Este enfoque agrupa píxeles que son similares en características como la intensidad o el color. Empieza con un conjunto de "semillas" y expande las regiones adyacentes que cumplen con un criterio de similitud. La imagen se divide en subregiones más pequeñas y luego se fusionan las regiones similares. Para obtener buenos resultados es necesario elegir adecuadamente los criterios de similitud y la selección inicial de semillas [29].

3. **Segmentación basada en clustering:** Los algoritmos de clustering agrupan píxeles en clusters o grupos basados en características similares. Un ejemplo de algoritmo de agrupamiento común es el *K-means Clustering*, que clasifica los píxeles en k clusters. Es importante elegir el número óptimo de clusters para obtener una buena segmentación, por lo que puede ser complicado y puede requerir ajustes [30].

4. **Segmentación semántica:** En el caso de esta técnica, descompone la imagen en regiones que corresponden a diferentes objetos o partes del entorno, asignando una etiqueta a cada píxel, por ejemplo: "person", "coche", "arbol", "piso", etc.

   Se utilizan CNNs para extraer características de alto nivel de la imagen, como bordes, texturas y patrones específicos. Estas características son fundamentales para diferenciar entre las distintas clases en la imagen. Modelo como U-Net [1] ha mostrado gran eficacia para realizar esta tarea. Más adelante se profundiza en algunos modelos, ya que requieren un enfoque más profundo para la comprensión del presente trabajo.

5. **Segmentación de instancias:** A diferencia de la segmentación semántica, no sólo identifica qué clase está presente en cada píxel, sino también distingue entre diferentes instancias de la misma clase. Un ejemplo de esta técnica es Mask R-CNN [31], que realiza segmentación de instancias asignando máscaras binarias a cada objeto detectado. Este tipo de segmentación es computacionalmente intensivo y requiere un etiquetado detallado de datos.

2.11. Redes convolucionales predefinidas

A lo largo de los años, las CNNs han experimentado un rápido desarrollo, impulsadas en gran medida por avances en arquitectura y en poder de cómputo. Aunque las CNNs se basan en principios matemáticos bien establecidos, su implementación ha variado considerablemente, lo que ha dado lugar a diversas arquitecturas optimizadas para tareas específicas. Algunas de estas arquitecturas han demostrado un rendimiento superior en competencias de visión por computadora, siendo reconocidas y utilizadas ampliamente en la investigación y la industria.

Estas redes, conocidas por su nombre propio, no sólo representan hitos en el desarrollo de las CNNs, sino que también han influido en muchas aplicaciones prácticas, desde la clasificación de imágenes hasta la segmentación de objetos y más allá. Las redes como AlexNet [32], VGG [33] y ResNet [34] han revolucionado el campo, ofreciendo mejoras significativas en precisión y eficiencia.

La segmentación de imágenes es, también, una tarea fundamental en visión por computadora, donde el objetivo es clasificar cada píxel de una imagen en una categoría específica, como se mencionó en secciones anteriores (2.10). Para enfrentar estos desafíos, se han desarrollado diversas

arquitecturas de redes convolucionales diseñadas específicamente para la segmentación. A continuación, se presentan 3 de las arquitecturas diseñadas para realizar esta tareas, las cuales serán implementadas para la segmentación que se realizará más adelante en el presente trabajo.

2.11.1. U-Net

U-Net es una de las arquitecturas más influyentes en el ámbito de la segmentación de imágenes, y fue desarrollada inicialmente para tareas de segmentación en el campo de la biomedicina. Propuesta por Olaf Ronneberger y su equipo en 2015 [1], esta red fue diseñada para superar las limitaciones de las arquitecturas tradicionales en problemas de segmentación donde los datos etiquetados suelen ser escasos, y el equilibrio entre clases es frecuentemente desbalanceado. U-Net logró destacar en aplicaciones donde la precisión en la identificación de límites y estructuras es fundamental, como en la segmentación de tejidos o células en imágenes microscópicas.

La arquitectura de U-Net sigue una estructura simétrica en forma de "u", con dos componentes principales: el encoder (codificador) y el decoder (decodificador), como se muestra en la Figura 2.11. En el codificador, la red reduce progresivamente la resolución espacial de la imagen a través de varias capas de convolución y operaciones de pooling, extrayendo características de alto nivel a medida que la imagen se hace más compacta. Esta fase del codificador es crucial para capturar las características relevantes de la imagen. Sin embargo, al reducir la resolución, se corre el riesgo de perder detalles finos importantes para una segmentación precisa. Para abordar esta limitación, U-Net introduce un decodificador que reconstruye la imagen a su resolución original utilizando capas deconvolucionales. Una de las innovaciones clave de U-Net es el uso de conexiones de salto (skip connections) entre las capas correspondientes del codificador y del decodificador. Estas conexiones permiten que las características de alta resolución de las primeras capas del codificador se transfieran directamente al decodificador, conservando así información espacial detallada que de otro modo se perdería en el proceso de downsampling (remuestreo descendente). Esta técnica no sólo mejora la precisión de la segmentación, sino que también permite que U-Net funcione bien incluso con conjuntos de datos pequeños y desbalanceados, algo común en muchas aplicaciones biomédicas.

2.11.2. SegNet

SegNet es una arquitectura diseñada específicamente para la segmentación de imágenes, desarrollada por Vijay Badrinarayanan y su equipo en 2015 [2]. Esta red surgió como una solución eficiente y escalable para la segmentación semántica, una tarea que requiere clasificar cada píxel de una imagen en una categoría. SegNet fue diseñada con el objetivo de proporcionar una red menos costosa en términos de memoria y computación, lo que la hace ideal para aplicaciones en tiempo real como la conducción autónoma, donde la eficiencia es crítica.

La arquitectura de SegNet se basa en un modelo codificador-decodificador, similar al de otras arquitecturas de segmentación como U-Net. Sin embargo, lo que diferencia a SegNet es su enfoque en la fase de decodificador. El codificador de SegNet es idéntico a las primeras capas convolucionales de redes preentrenadas populares como VGG16. El codificador consta de una serie de capas convolucionales seguidas de operaciones de MaxPooling, que reducen progresivamente la resolución de la imagen mientras extraen características de alto nivel, como se muestra en la Figura 2.12.

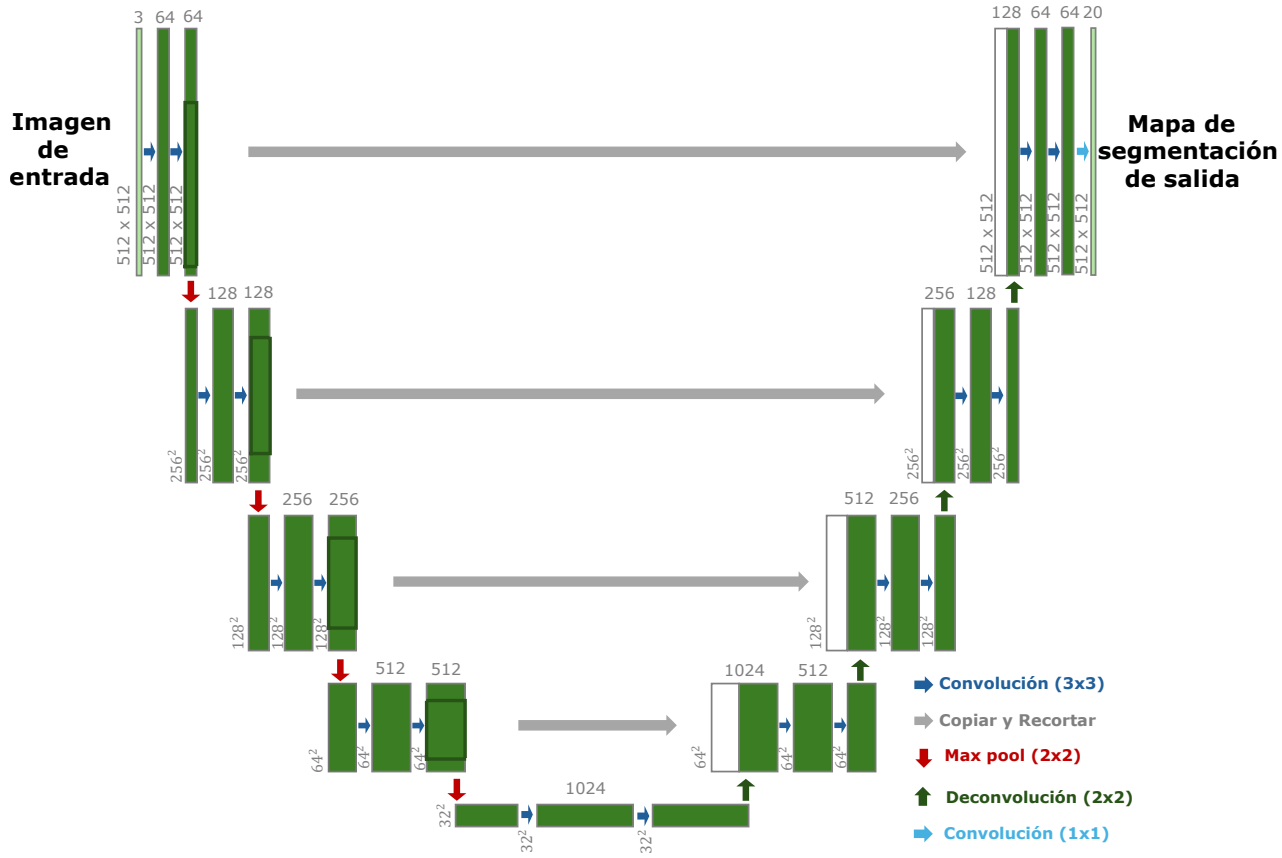


Figura 2.11: Arquitectura U-net (aplicado a nuestro problema de segmentación, basada en [1]). Cada recuadro verde corresponde a un mapa de características multicanal. El número de filtros se indica en la parte superior del recuadro. El tamaño $x-y$ se indica en el borde inferior izquierdo del recuadro. Los recuadros blancos representan mapas de características copiados. Las flechas indican las distintas operaciones.

El verdadero aporte de SegNet viene en su fase de decodificador, donde utiliza un método único para restaurar la resolución espacial de la imagen. A diferencia de otras arquitecturas que emplean técnicas como la interpolación o la deconvolución para realizar el upsampling (que se podría traducir como "remuestreo ascendente"), SegNet introduce el uso de los índices de MaxPooling obtenidos en el codificador para guiar el proceso de reconstrucción en el decodificador. Estos índices de MaxPooling contienen información sobre las posiciones espaciales de los máximos locales durante la fase de compresión en el codificador, lo que permite realizar un unpooling preciso en el decodificador. Esto es crucial, ya que permite una reconstrucción más detallada de los contornos y los límites de los objetos segmentados, sin añadir demasiada carga computacional o requerimientos de memoria adicionales.

El uso de los índices de MaxPooling es una innovación importante, ya que reduce significativamente el número de parámetros de la red en comparación con otros métodos de upsampling más costosos. Como resultado, SegNet es capaz de realizar una segmentación semántica precisa con una fracción del costo computacional de otras arquitecturas profundas. Esto hace que SegNet sea

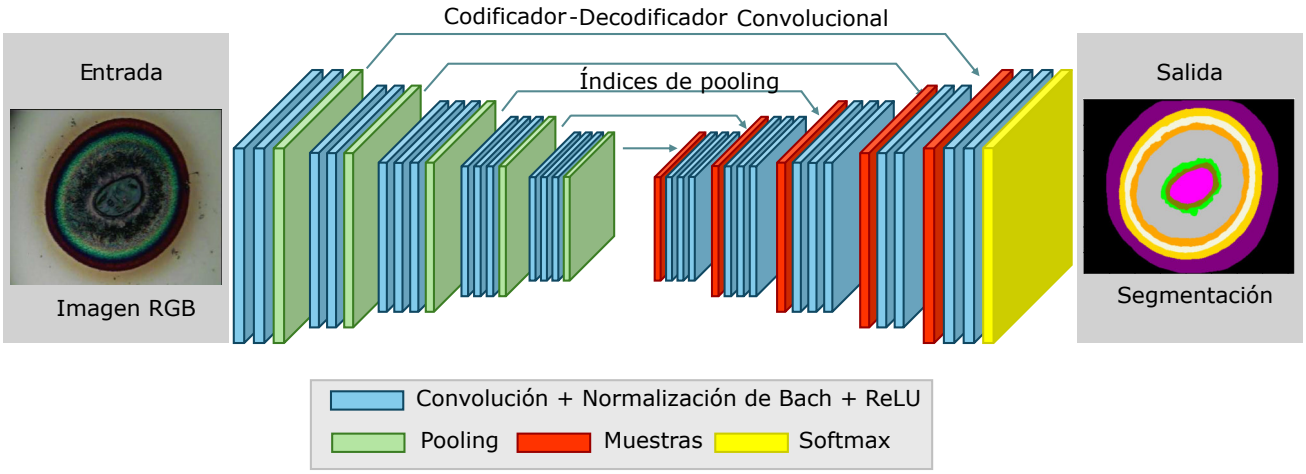


Figura 2.12: Ilustración de la arquitectura SegNet (basada en [2]). No hay capas totalmente conectadas, por lo que sólo es convolucional. Un decodificador aumenta la muestra de su entrada utilizando los índices de agrupación transferidos desde su codificador para producir un mapa o mapas de características dispersos. A continuación, realiza una convolución con un banco de filtros entrenable para densificar el mapa de características. Los mapas de características de salida finales del descodificador se introducen en un clasificador de máxima suavidad para la clasificación por píxeles.

particularmente adecuada para aplicaciones móviles y sistemas embebidos, donde la capacidad de procesamiento y memoria suelen ser limitadas.

SegNet ha demostrado ser altamente efectiva en tareas de segmentación de escenas urbanas, lo que la hace ideal para aplicaciones en la conducción autónoma, donde la identificación de carreteras, señales, peatones y otros elementos del entorno es crucial. Además, debido a su eficiencia computacional, SegNet es utilizada en otros contextos que requieren segmentación en tiempo real, como la robótica y la realidad aumentada.

2.11.3. DeepLabV3+

DeepLabV3+ es la versión mejorada de la familia DeepLab, una serie de arquitecturas desarrolladas por Google para la segmentación semántica de imágenes. Lanzada en 2018, DeepLabV3+ integra múltiples avances de sus predecesoras (DeepLabV1, V2 y V3) y se ha consolidado como una de las arquitecturas más avanzadas en segmentación de imágenes gracias a su capacidad para capturar tanto características locales como globales de una imagen, lo que le permite segmentar objetos con precisión, incluso en escenarios complejos [3].

DeepLabV3+ introduce mejoras clave sobre DeepLabV3, combinando la potencia de las convoluciones dilatadas (también conocidas como convoluciones Atrous) y un módulo codificador-decodificador que optimiza la segmentación de bordes y contornos de objetos, como se muestra en la Figura 2.13. Una de las características más importantes de DeepLabV3+ es el uso de convoluciones Atrous para expandir el campo receptivo de la red sin reducir la resolución espacial de las características extraídas. En las convoluciones tradicionales, a medida que las capas se hacen

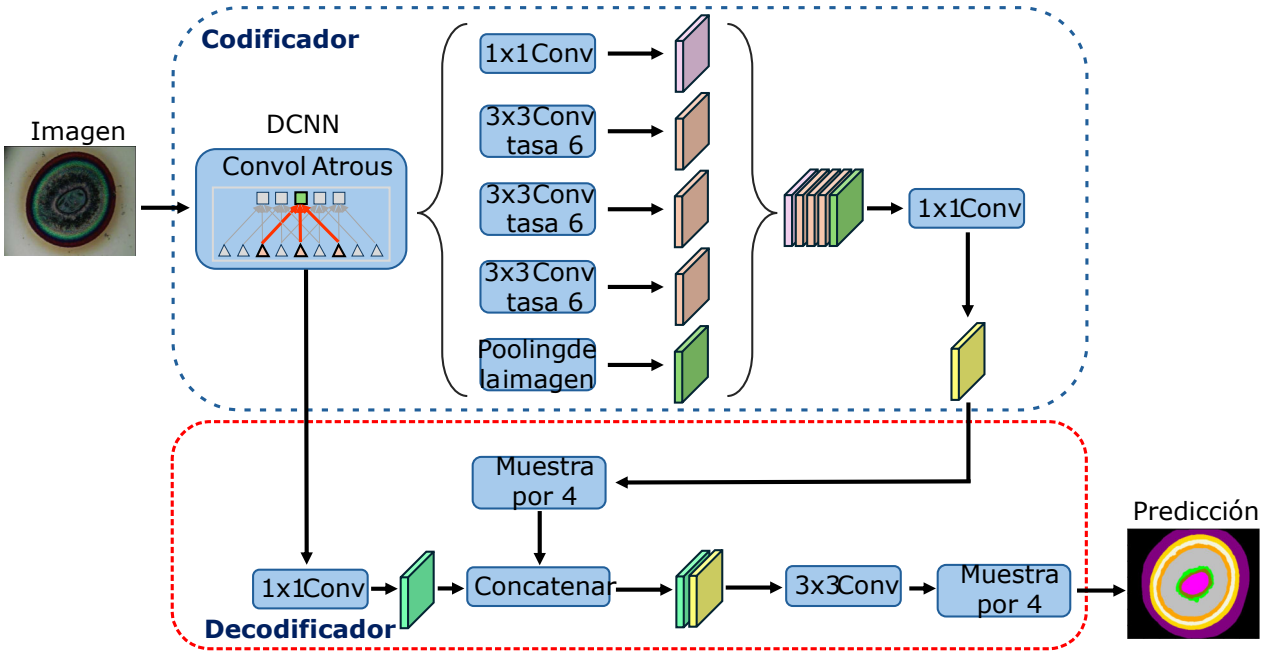


Figura 2.13: Diagrama de la arquitectura DeepLabv3+ (basada en [3]), donde amplía DeepLabv3 empleando una estructura codificador-decodificador. El módulo codificador codifica la información contextual multiescala aplicando una convolución Atrous a múltiples escalas, mientras que el módulo decodificador, sencillo pero eficaz, refina los resultados de la segmentación a lo largo de los límites de los objetos.

más profundas, el campo receptivo aumenta, pero la resolución espacial disminuye. Sin embargo, en DeepLabV3+, las convoluciones Atrous permiten que la red capture información a gran escala sin perder detalles de resolución, lo que es crucial para una segmentación precisa en objetos de diferentes tamaños. Este enfoque resulta especialmente útil cuando se trabaja con imágenes de alta resolución y escenas complejas, donde los objetos pueden variar significativamente en tamaño.

Otra innovación significativa de DeepLabV3+ es el módulo de Pooling Piramidal Espacial (PPM), que ayuda a la red a capturar información contextual en múltiples escalas. El PPM realiza un "pooling" en diferentes resoluciones y luego combina estas características para enriquecer la representación de la imagen. Al combinar información de diferentes niveles de contexto, DeepLabV3+ puede segmentar con precisión tanto objetos pequeños como grandes dentro de una misma imagen, lo que lo convierte en una red robusta para una amplia variedad de tareas de segmentación.

En cuanto al codificador, DeepLabV3+ incorpora una fase de upsampling que permite la recuperación precisa de los detalles espaciales, particularmente en los bordes de los objetos segmentados. Esta fase de upsampling incluye operaciones de convolución adicionales que refinan los contornos y los detalles más finos, garantizando una segmentación de alta calidad. La combinación del codificador con convoluciones Atrous y el codificador refinado permite que DeepLabV3+ supere los desafíos que enfrentan otras arquitecturas al segmentar objetos con formas complejas o bordes irregulares.

DeepLabV3+ ha sido utilizada con gran éxito en la segmentación semántica de escenas natu-

rales, lo que es esencial en aplicaciones como la conducción autónoma, la robótica, la realidad aumentada y la visión por computadora en tiempo real. Además, ha demostrado un rendimiento sobresaliente en benchmarks como PASCAL VOC y Cityscapes, que son estándar de referencia para la segmentación semántica de imágenes.

2.12. Validación cruzada

La validación cruzada es una técnica utilizada en el aprendizaje automático para evaluar el rendimiento de un modelo, proporcionando una medida más robusta y generalizable del desempeño que el simple uso de un conjunto de datos de entrenamiento y otro de prueba. Su principal objetivo es evitar el sobreajuste y garantizar que el modelo tenga una capacidad adecuada para generalizar a datos nuevos y no vistos.

El proceso de la validación cruzada consiste que, en lugar de dividir los datos en un único conjunto de entrenamiento y uno de prueba, la validación cruzada divide los datos en varios subconjuntos, llamados folds. El procedimiento típico es el siguiente:

1. **División de los datos en K folds:** El conjunto de datos completo se divide en K particiones o folds. Por ejemplo, en una validación cruzada de 5 particiones (K = 5), los datos se dividen en 5 conjuntos del mismo tamaño (o lo más equilibrado posible).

2. **Entrenamiento y validación iterativa:** Se realizan K iteraciones. En cada iteración, el modelo se entrena con K-1 folds y se valida con el fold restante que se apartó como conjunto de validación. Este proceso se repite hasta que cada fold ha sido utilizado como conjunto de validación una vez.

3. **Promedio del rendimiento:** Después de realizar todas las iteraciones, los resultados obtenidos en cada fold se promedian para proporcionar una estimación final del rendimiento del modelo. Este promedio proporciona una estimación más estable y representativa del comportamiento del modelo en diferentes particiones del conjunto de datos.

La validación cruzada permite el mejor uso de los datos, ya que todos los datos se usan tanto para el entrenamiento como para la validación, lo que permite aprovechar mejor la información disponible, especialmente cuando el conjunto de datos es pequeño. Al utilizarla, se reduce el sobreajuste, dado que el modelo es probado en diferentes particiones de los datos, la validación cruzada ayuda a detectar problemas de sobreajuste, ya que se evalúa el rendimiento del modelo en múltiples subconjuntos. Al promediar los resultados sobre varios folds, se obtiene una medida de rendimiento más fiable y menos dependiente de una única división de los datos.

Tipos de validación cruzada:

- **K-Fold Cross-Validation:** Es la forma más común. Como se describió anteriormente, los datos se dividen en K partes y se realizan K iteraciones de entrenamiento y validación.

- **Leave-One-Out Cross-Validation (LOO-CV):** Es un caso especial de validación cruzada donde K es igual al número de muestras en el conjunto de datos. Cada muestra se deja fuera en cada iteración, lo que genera un alto número de evaluaciones. Aunque es más precisa, es computacionalmente costosa para conjuntos de datos grandes.

- **Stratified K-Fold Cross-Validation:** Una variante de la validación cruzada tradicional, que asegura que cada fold mantiene la misma proporción de clases que el conjunto de datos original, lo que es útil en problemas de clasificación desequilibrada.

En resumen, la validación cruzada es una herramienta poderosa para evaluar modelos de aprendizaje automático, ya que ofrece una evaluación confiable y ayuda a seleccionar modelos y ajustar hiperparámetros de manera más efectiva, evitando que se tomen decisiones basadas en particiones específicas del conjunto de datos.

2.13. Aumento de datos

El aumento de datos en el aprendizaje profundo es crucial cuando se cuentan con pocos datos; ya sean imágenes, texto, etc. El aprendizaje de DNNs requiere una gran cantidad de datos, debido a que el rendimiento predictivo de los modelos que son entrenados disminuye cuando el tamaño y la calidad de la muestra son pocos. El entrenar arquitecturas DNNs con conjuntos de datos pequeños generalmente provoca un sobreajuste y poca capacidad de generalización; por lo tanto una reducción del rendimiento predictivo.

Recopilar y etiquetar datos es un proceso que lleva mucho tiempo. Además, se requiere conocimientos específicos del ámbito. Aquí es donde aumento de datos es utilizado para superar el problema del bajo tamaño de la muestra, esto, ampliando el número de datos mediante la generación de instancias sintéticas o el remuestreo. Los métodos tradicionales de aumento de datos implican transformaciones elementales de las imágenes para preservar el contenido y la categoría, como rotaciones, reflexiones, aumento de tamaño y recortes; esto en el caso de imágenes. El método seleccionado dependerá del tipo de datos que se esté tratando. Por ejemplo, en el cálculo del diámetro, se empleará la interpolación lineal, que permite estimar valores intermedios entre puntos conocidos de manera precisa. En la parte de segmentación de imágenes, se aplicarán técnicas de aumentación de datos como rotaciones, reflexiones y otras transformaciones geométricas sobre cada una de las micrografías. Estas técnicas son cruciales para mejorar la robustez del modelo y su capacidad de generalización. Los detalles específicos sobre la implementación de estas técnicas se describirán en mayor profundidad más adelante (Secciones 4.1 y 4.8).

2.13.1. Interpolación lineal

La interpolación es un método numérico, cuya finalidad es el realizar la estimación de un valor desconocido en un intervalo situado entre dos valores conocidos de una función. En la Figura 2.14 podemos ver un ejemplo de interpolación lineal, en la cual, partiendo de dos puntos conocidos podemos estimar un valor que este dentro del intervalo de los mismos. Por lo tanto, interpolar se entenderá estimar un valor desconocido en algún punto de una función, mediante el cálculo de un factor de proporcionalidad con base en los valores conocidos de la función. Para calcular este factor de proporcionalidad podemos utilizar la Ec. 2.44, que parte de la Figura 2.14.

$$f(x_i) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_i - x_0) \tag{2.44}$$

donde x_0 y $f(x_0)$ representan un punto inicial y el valor de la función en el mismo, respectivamente. $x_1 y f(x_1)$ representan un punto final y el valor de la función en el mismo, respectivamente. Finalmente, x_i y $f(x_i)$ representa el punto que queremos calcular (dentro del intervalo de los dos valores

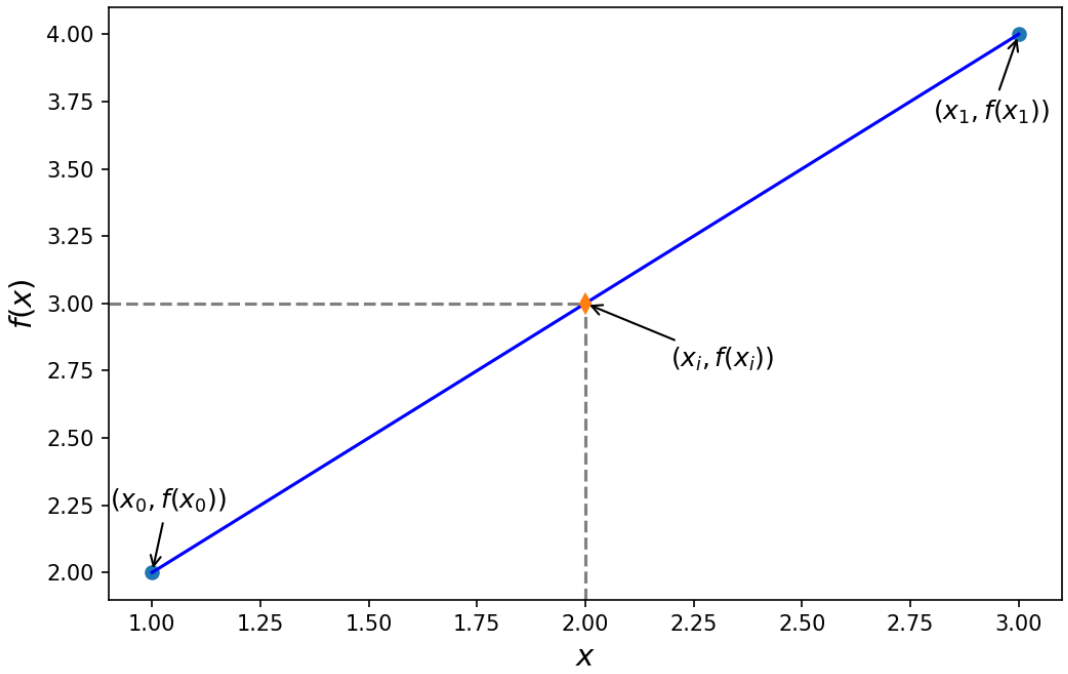


Figura 2.14: Ejemplo gráfico de interpolación lineal.

anteriores) y el valor de la función, respectivamente. Cada segmento agrupa píxeles que comparten ciertas características visuales, como color, intensidad o textura, y que pueden representar partes de objetos o estructuras específicas dentro de la imagen.

2.13.2. Técnicas comunes de aumento de datos en imágenes

En problemas de procesamiento de imágenes, las técnicas de aumento de datos pueden implicar transformaciones geométricas, cambios en la iluminación, entre otros. Aquí algunas de las técnicas más utilizadas:

1. **Rotación:** Rotar la imagen un número determinado de grados, lo que permite al modelo reconocer el objeto desde diferentes orientaciones.

2. **Desplazamientos:** Realizar traslaciones horizontales o verticales, es decir, desplazar la imagen ligeramente en ambas direcciones.

3. **Escalado (Zoom):** Aplicar un aumento o reducción del tamaño de la imagen para que el modelo aprenda a reconocer objetos de diferentes tamaños.

4. **Reflexión (Flip horizontal o vertical):** Reflejar la imagen en sentido horizontal o vertical, lo cual es útil, por ejemplo, para reconocer imágenes de objetos que pueden aparecer en ambos sentidos.

5. **Recortes Aleatorios (Random Cropping):** Recortar una parte aleatoria de la imagen, lo que obliga al modelo a centrarse en diferentes secciones de la imagen.

6. **Cambios en el Brillo y Contraste:** Variar la intensidad de brillo o contraste de la imagen para que el modelo se vuelva más robusto frente a las diferencias de iluminación.

7. **Aplicación de Ruido:** Añadir ruido gaussiano o sal y pimienta a las imágenes para hacer que el modelo sea más robusto frente a imágenes ruidosas o imperfectas.

8. **Modificación de Colores (Augmentación en el espacio de color):** Cambiar el balance de colores o realizar transformaciones en el espacio de color (como pasar de RGB a HSV o YCbCr y modificar uno de esos canales).

9. **Corte Aleatorio de Píxeles (Random Erasing):** Enmascarar (borrar) aleatoriamente una pequeña parte de la imagen para que el modelo se acostumbre a trabajar con información incompleta.

2.13.3. Aumento de datos en tiempo real

Aumento de Datos en Tiempo Real es una técnica avanzada en la que las transformaciones de los datos de entrenamiento, como las rotaciones, desplazamientos, reflejos, entre otros, se aplican dinámicamente durante el proceso de entrenamiento del modelo, en lugar de preprocesarlas de antemano. Esto significa que cada vez que el modelo entrena con un lote de imágenes, esas imágenes se modifican de forma aleatoria para generar nuevas versiones. Así, cada paso de entrenamiento recibe variaciones diferentes de las mismas imágenes base, lo que enriquece el aprendizaje del modelo sin necesidad de almacenar versiones adicionales de los datos.

2.14. Espacios de color

El espacio de color en el que se representan las imágenes puede influir significativamente en el rendimiento de los modelos de segmentación. Dependiendo del tipo de características que se desea resaltar, cambiar el espacio de color puede mejorar la precisión de la segmentación en función de la naturaleza de los datos. A continuación, se describen tres de los espacios de color más utilizados, los cuales se aplicarán en el conjunto de datos (micrografias), para analizar el desempeño de las redes convolucionales con cada uno de ellos.

2.14.1. Espacio de color RGB

El espacio de color RGB (Red, Green, Blue) es uno de los más comunes y representa las imágenes mediante una combinación aditiva de estos tres colores. Cada píxel contiene valores para cada componente, lo que permite representar una amplia gama de colores. Este espacio de color es directo y suele ser el formato predeterminado para las cámaras y muchas bases de datos de imágenes.

Una de las desventajas del espacio RGB es que las variaciones de iluminación y sombra afectan todos los canales de manera simultánea, lo que puede generar confusiones para el modelo de segmentación. Aun así, este espacio es adecuado cuando el color es una característica dominante para la diferenciación de clases en la imagen.

2.14.2. Espacio de color HSV

El espacio HSV (Hue, Saturation, Value) descompone la información de color en tres componentes: matiz (hue), saturación y valor (brillo). A diferencia de RGB, donde los tres canales están correlacionados, en HSV la información de brillo se maneja por separado, lo que permite que el modelo enfoque más en las características de color sin verse afectado directamente por las variaciones de luz.

- **Matiz (Hue):** Representa el color puro, lo que facilita la diferenciación entre diferentes colores.
- **Saturación:** Mide la intensidad del color.
- **Valor:** Describe el brillo, que puede ser útil para destacar elementos brillantes o con poca iluminación.

En segmentación, el uso de HSV puede ser beneficioso cuando se desea que las diferencias de color sean más evidentes o cuando se trata de minimizar los efectos de las variaciones de iluminación.

2.14.3. Espacio de color CIElab (Lab)

El espacio CIElab, también conocido como Lab, es perceptualmente uniforme, lo que significa que las diferencias en el espacio Lab corresponden más de cerca a cómo los humanos perciben el color. Se compone de tres componentes:

- **L:** Representa la luminosidad o claridad.
- **a:** Indica la posición entre los colores verde y rojo.
- **b:** Indica la posición entre azul y amarillo.

Lab es útil en segmentación cuando se busca una mejor separación entre los diferentes objetos o clases, ya que este espacio está diseñado para ser más preciso en términos de percepción de color. A menudo se utiliza en tareas donde los bordes entre objetos no son muy evidentes en RGB, ya que Lab facilita la identificación de esos límites.

Capítulo 3

Molibdeno

Contenido

El molibdeno es un metal de transición esencial en diversas industrias, conocido por su alta resistencia a la corrosión y su capacidad para soportar temperaturas extremas. Debido a estas propiedades, el molibdeno es ampliamente utilizado en la fabricación de aleaciones de acero, componentes electrónicos, catalizadores y en la industria aeroespacial. Además, su relevancia ha crecido en el ámbito de la investigación de materiales, especialmente en el desarrollo de tecnologías avanzadas como los recubrimientos protectores y los dispositivos semiconductores.

Uno de los aspectos críticos en la investigación sobre el molibdeno es el estudio de sus óxidos.

Los óxidos de molibdeno presentan propiedades únicas, como alta actividad catalítica y propiedades ópticas, que los hacen útiles en aplicaciones como la fotocatálisis y la fabricación de dispositivos electrónicos. Comprender la estructura y comportamiento de estos óxidos es fundamental para mejorar su rendimiento en dichas aplicaciones.

En este contexto, la segmentación de los óxidos de molibdeno en micrografías es una tarea clave en el análisis de su estructura y distribución. La segmentación permite identificar y aislar las distintas fases de los óxidos, facilitando el estudio de sus propiedades morfológicas y químicas. Al aplicar técnicas avanzadas de segmentación de imágenes, se puede obtener una caracterización más precisa y eficiente de los óxidos de molibdeno, lo que contribuye a optimizar su producción y aplicación en diversas áreas tecnológicas.

Por tanto, la segmentación de los óxidos de molibdeno no sólo es esencial para entender su comportamiento a nivel microscópico, sino que también tiene un impacto directo en la mejora de procesos industriales y en el desarrollo de nuevas tecnologías basadas en este material. En este capítulo, se abordarán los conceptos fundamentales del molibdeno y sus óxidos, junto con la técnica utilizada para su producción. Además, se explorará cómo es posible categorizar los óxidos de molibdeno en función de su fase y coloración, utilizando los datos obtenidos a través de la caracterización mediante espectroscopía Raman.

3.1. Propiedades del molibdeno

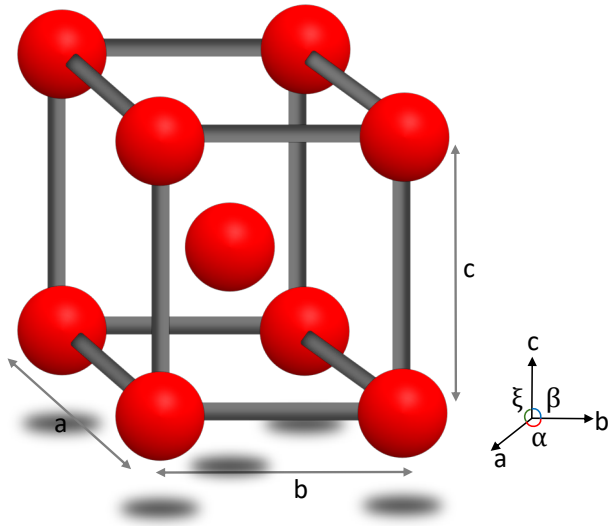


Figura 3.1: Estructura cristalina del molibdeno.

La palabra “molibdeno” (Mo, por su abreviación en la tabla periódica de los elementos) proviene del término griego *molybdos*, que significa parecido al plomo, en alusión a su color, así como a la semejanza del plomo con la molibdenita (MoS_2).

El molibdeno es un metal, cuya estructura cristalina es cúbica centrado en el cuerpo (bcc, por las siglas en inglés *body centered cubic*), por lo que sus ejes son iguales en ángulos rectos, es decir

$a = b = c$, por lo tanto $\alpha = \beta = \xi$. Su parámetro de red es $a = 3{,}147$ Å, con un ángulo entre los ejes de 90°, como se observa en la Figura 3.1.

En forma pura el molibdeno es duro y de color gris metálico, tiene una dureza Vickers de 1530 MPa, punto de fusión de 2896 K y de ebullición de 4912 K. Su coeficiente de expansión térmico es pequeño, de $5.04 \times 10^{-6}/K$, conductividad eléctrica de $0{\cdot}182 \times 10^{6}/ohm \cdot cm$ y una resistividad eléctrica de $5{\cdot}49\times\ 10^{-6} ohm \cdot cm$ (). La Tabla 3.1 presenta los parámetros termofísicos y ópticos importantes del molibdeno a temperatura ambiente.

Tabla 3.1: Propiedades termofísicas y ópticas del molibdeno.

Propiedades termofísicas del molibdeno [35]		
Parámetro	**Valor**	**Unidades**
Densidad (ρ)	10.2×10^{3}	kg/m^3
Calor específico (cp)	2.5×10^{2}	$J/(kgK)$
Conductividad térmica (Kt)	138	W/(m K)
Difusividad térmica (D)	54×10^{6}	m^2/s
Temperatura de fusión (Tf)	2896	K
Temperatura de ebullición (Te)	4912	K
Propiedades ópticas del molibdeno [36]		
Parámetro	**Valor**	**Unidades**
Coeficiente de absorción óptico ()	50×10^{6}	m^{-1}

3.2. Óxidos de molibdeno

Al incorporar oxígeno a la red cristalina, se puede conseguir distintas estequiometrías. Los óxidos de molibdeno presentan estados de oxidación 2+, 3+, 4+, 5+, 6+ [37], dependiendo de las condiciones experimentales se puede presentar una transformación de fase Mo_xO_y, donde $1 \leq x \leq 18$ y $2 \leq y \leq 52$ (en el caso de este trabajo).

Los óxidos de molibdeno poseen propiedades ópticas y electrónicas pueden ser de interés para dispositivos crómicos basados en distintos fenómenos físicos, por ejemplo, electrocromismo, en el cual un material puede exhibir un cambio de color con la aplicación de voltaje o corriente; termocromismo, donde un material cambia de color al exponerse al calor; o gasocromismo, donde el material cambia de color ante la presencia de un gas.

Actualmente, existen varias fases cristalinas del molibdeno, al utilizar técnica de irradiación de rayos X [38]. Sin embargo, para fines prácticos, en este trabajo se hará un enfoque en los óxidos: MoO_2, Mo_4O_{11}, Mo_8O_{23}, $Mo_{18}O_{52}$ y MoO_3; debido que estos óxidos son los que predominan como resultado de la irradiación de las capas de Mo con el láser de pulsos de femtosegundos.

3.3. Oxidación inducida por láser

La irradiación con láser de un metal en aire ambiental genera calor en su superficie debido al fenómeno de absorción, lo que puede provocar su oxidación por la presencia de moléculas de oxígeno. Este proceso de oxidación inducido por láser depende de factores como la longitud de onda, el tiempo de exposición, la temperatura, y si el metal es una película delgada o un material de volumen, entre otros [39].

La oxidación de un metal puede involucrar múltiples mecanismos (ver Tabla 3.2), los cuales contribuyen a la complejidad del proceso, ya que estos mecanismos pueden ocurrir de manera simultánea y afectar mutuamente el desarrollo de la oxidación [39].

Tabla 3.2: Mecanismos responsables de la oxidación láser en un metal (modificado de: [39]).

Mecanismos responsables de la oxidación láser en un metal	
Transporte del oxígeno (gas) hasta la superficie	El mecanismo se presenta durante el depósito de calor en el material
Interfaz sólido-gas Descomposición de las moléculas de oxígeno (gas) con las moléculas del material (sólido)	
Adsorción (físico-adsorción y químico-adsorción) y desorción	
Difusión del óxido	
Reacción química	El mecanismo tiene lugar durante el enfriamiento del material (proceso de solidificación)
Nucleación de los óxidos	
Crecimiento de cristales del óxido	

3.4. Coloración en películas delgadas

Cuando una capa delgada de material se oxida y es expuesta a luz blanca, puede presentar una variedad de colores visibles. Esta coloración resulta de varios fenómenos ópticos que interactúan en la superficie del material oxidado. Estos efectos ópticos, descritos en la Tabla 3.3.

En el caso de la coloración de un material debida a su composición química, conocida como *absorción selectiva*, la estructura de bandas de energía depende tanto de la composición química como

Tabla 3.3: Coloración debido a la oxidación láser en un metal (modificado de: [40]).

Coloración debido a la composición química del metal	**Coloración debido a la morfología del metal**
Absorción selectiva	Difracción
Dispersión	Interferencia

de la estructura cristalina del material. Cuando una radiación electromagnética incide sobre el material, los fotones se absorben selectivamente según la estructura de sus bandas, lo que genera un color característico en la luz reflejada o transmitida. En metales y otros materiales conductores, esta absorción ocurre debido a los electrones libres en la banda de conducción, los cuales liberan su energía mediante colisiones con la red cristalina del material.

La *dispersión* es un fenómeno en el cual el índice de refracción de un material varía según la longitud de onda de la luz que lo atraviesa. Cuando la luz incide en el material, se refracta en distintas direcciones, separando los colores que la componen y produciendo franjas de color distintas.

Para la coloración en un material debida a su morfología, es esencial considerar que la luz se comporta como una onda en los fenómenos de difracción e interferencia. En el caso de la *difracción*, la luz que incide sobre el material se desvía a causa de las estructuras periódicas presentes en su superficie, las cuales son comparables en tamaño con la longitud de onda de la luz. Esto provoca la separación de la luz en distintos colores, creando así efectos de coloración característicos.

En el fenómeno de *interferencia*, cuando la luz incide en una capa delgada, se produce una superposición de las ondas de luz reflejada dentro de esta. Este fenómeno genera un desfase que resulta en el aumento o disminución de la intensidad de la luz reflejada o transmitida. El color observado se origina por la interferencia entre las ondas reflejadas en las interfaces que separan los diferentes medios. La interferencia en una capa delgada depende de varios factores: la longitud de onda de la luz, el ángulo de incidencia, el espesor de la capa, y los índices de refracción tanto de la capa como del medio en el que se encuentran las interfaces (aire).

3.5. Irradiación en modo de exposición fija

Con el fin de obtener oxidación en la capa de molibdeno, en el trabajo [9] se realizaron irradiaciones en forma de exposiciones fijas (puntos) y exposiciones con rotaciones. Este trabajo se enfoca en el análisis de los resultados obtenidos utilizando la técnica de exposición fija, por lo que es importante destacar la forma en la que se obtuvieron dichos resultados.

Para el sistema de microprocesamiento se empleó un oscilador láser de Ti:zafiro, el cual emite pulsos de femtosegundos. Este sistema se compone de un medio activo de cristal de Ti:zafiro (Ti^{+3} : Al_2O_3), junto con varios espejos y elementos dispersivos (un par de prismas) que conforman la cavidad láser. La cavidad requiere un bombeo en modo continuo, para lo cual se utiliza un láser Nd:YAG de 532 nm (doblado en frecuencia) con potencia de bombeo alrededor de 5W, lo que permite alcanzar una potencia de salida de $\sim$ 500 mW en el oscilador. Este láser de femtosegundos opera con una longitud de onda centrada en 800 nm, produciendo pulsos a una frecuencia de repetición de 70 MHz (con una separación de 14 ns entre pulsos) y una energía por pulso de hasta 7.2 nJ, tal como se muestra en la Figura 3.2. El perfil de intensidad del haz láser es gaussiano, ligeramente elíptico, con un eje corto de 10.36 μm y un eje largo de 12.85 μm, respectivamente.

El arreglo experimental que se utilizo para inducir oxidación en las capas metálicas se muestra en la Figura 3.3. Este arreglo típico de procesamiento láser de materiales además de la fuente láser de femtosegundos incluye la óptica que transporta el haz hasta la muestra metálica (camino óptico color rojo). En los experimentos se empleó una lente de 35 mm (L1) de distancia focal para enfocar

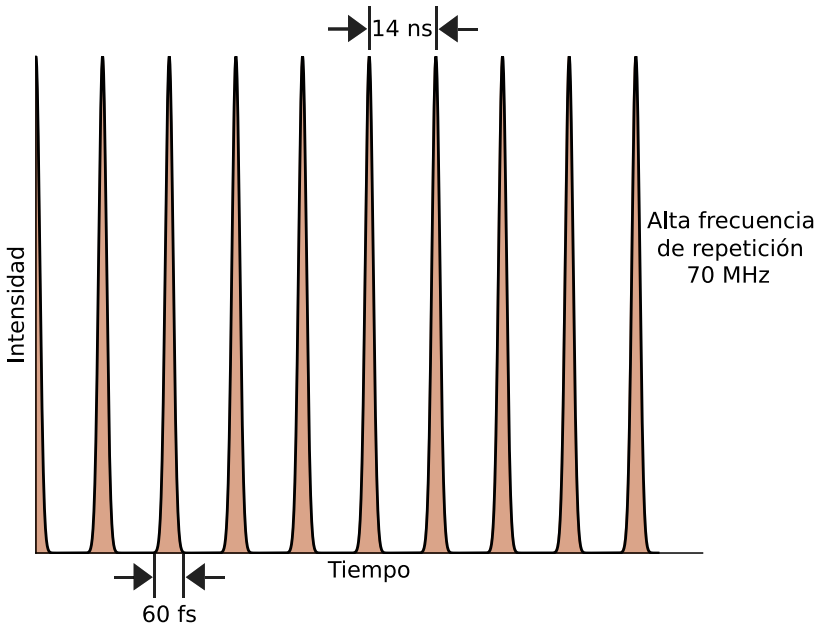


Figura 3.2: Esquema del tren de pulsos de fs a una alta frecuencia de repetición.

el haz sobre la muestra y se eligieron energías por pulso de 1.4 a 4.2 nJ. La segunda lente (L2) de 500 mm de distancia focal se utiliza en un plano de objetivo equivalente, además de un divisor de haz (DH) 90/10 y una cámara CCD que facilita el monitoreo en tiempo real de la superficie de la muestra. La cámara también sirve para validar que el láser y la superficie sobre la que impactará son ortogonales, asegurando un alineamiento preciso de 90 grados. Los tiempos de exposición se variaron desde unos cuantos segundos a varios minutos, en el intervalo de 2 a 1200 segundos.

La energía por pulso que incide en la muestra se controló por medio de un atenuador constituido por la combinación de un retardador de fase de media onda y un polarizador lineal (Glan - air) de alto contraste; el tiempo de exposición de la muestra se controló a través de un obturador. La energía por pulso incidente en la muestra se midió con un fotodiodo (Thorlabs DET210) calibrado mediante el uso de un medidor de energía (Ophir 30A-MD), el cual recibe parte del haz láser a través de la reflexión de éste en un divisor de haz de película delgada 90-10 (Thorlabs BP108) ubicado en el camino óptico como se muestra en la Figura 3.3.

Mediante una selección apropiada de fluencias y tiempos de exposición se generó una matriz de 54 puntos. Esta matriz de puntos permite estudiar la influencia de estos dos parámetros de irradiación en las características de los óxidos de molibdeno inducidos por la irradiación láser en la capa de Mo.

En la Tabla 3.4 se presentan las micrografías ópticas de las regiones transformadas en la capa de Mo, durante la serie de exposiciones fijas. Estas micrografías se obtuvieron con al ayuda de un microscopio óptico convencional, empleando un objetivo de 50X; la matriz de micrografías ópticas que se presenta está formada por seis columnas que representan la fluencia por pulso aplicada, y nueve filas que representan el tiempo de exposición empleado, lo cual significa 54 puntos irradiados con distintos parámetros de irradiación. Por simplicidad, se denota cada una de las imágenes con la abreviación S_{ij} (donde 'i' representa las columnas y 'j' las filas) a las matrices donde se presentan los resultados de transformación de la capa de molibdeno debido a la irradiación láser.

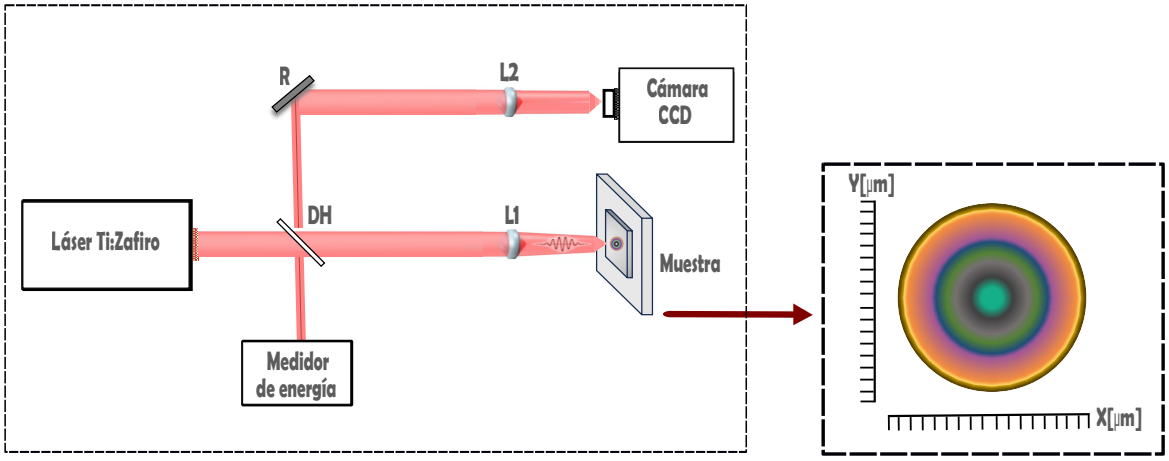


Figura 3.3: Arreglo experimental empleado para irradiar las capas delgadas de molibdeno.

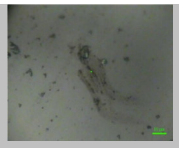
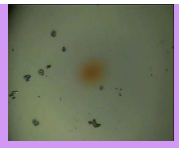
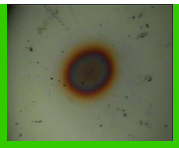
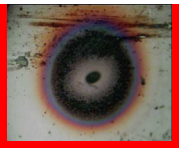
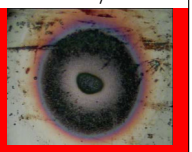
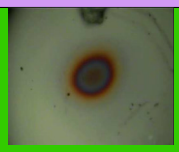
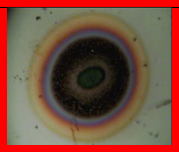
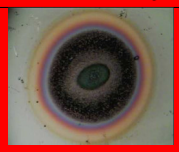
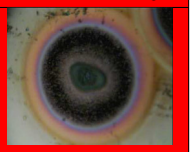
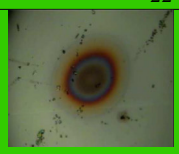
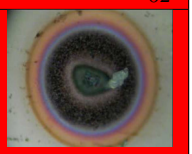
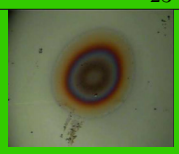
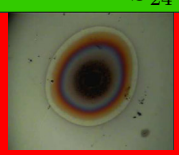
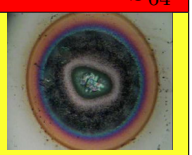
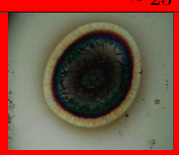
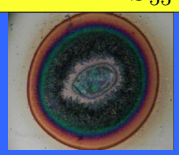
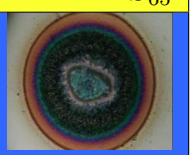
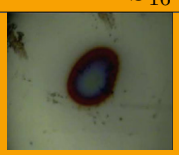
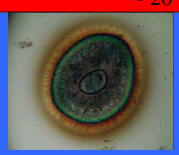
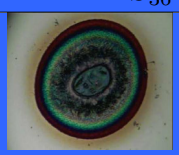
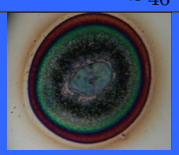
Como se puede observar en la matriz de micrografías ópticas, el efecto de la irradiación láser sobre el molibdeno se manifiesta en la formación de un patrón de anillos de distinto color. El número de anillos y los colores obtenidos en cada uno de los anillos esta correlacionado con la fluencia y el tiempo de exposición.

En la primera exposición, denotada en la matriz de micrografías ópticas como S_{11} (color gris) no se logró distinguir ópticamente algún cambio en la zona irradiada. En las regiones irradiadas correspondientes a $S_{12} - S_{15}$ y S_{21} (color lila) se observa la formación de una marca ovalada con un sólo color, llamándole café-claro. Conforme incrementa la fluencia por pulso y el tiempo de exposición, es decir, la fluencia integrada o neta, se observa la formación de anillos concéntricos en un patrón de color similar en la mayoría de las zonas irradiadas. Estos anillos de colores están constituidos de la siguiente manera: café, caféclaro, azul - verde, rojo - oscuro, café - oscuro, purpura, azul, verde, amarillo, gris - oscuro y gris - claro de la periferia al centro de la irradiación. Las zonas irradiadas $S_{16} - S_{19}$ (color naranja), muestran una tonalidad color purpura en el centro y carecen de los anillos café - claro, azul - verde, rojo - oscuro, azul y amarillo. En la región de exposiciones $S_{22} - S_{24}$ y S_{31} (color verde), se observa una tonalidad café - claro en el centro y se carece de los anillos café, azulverde, rojo-oscuro, verde, amarillo y gris-oscuro. En la región de exposiciones $S_{25} - S_{26}$, $S_{32} - S_{35}$, $S_{41} - S_{44}$, $S_{51} - S_{54}$ y $S_{61} - S_{64}$ (color rojo), se observó una tonalidad oscura en el centro de la zona irradiada, y en las exposiciones S_{45}, S_{55} y S_{65} (color amarillo), se observó una tonalidad blanca, en estas regiones se carece del anillo azul-verde, verde y amarillo. En la región de exposiciones $S_{27} - S_{29}$, $S_{36} - S_{39}$, $S_{46} - S_{49}$, $S_{56} - S_{59}$ y S_{66}-S_{69} (encerradas en color azul), se observó una tonalidad gris - pizarra en el centro de la zona irradiada y se carece del anillo purpura. El diámetro de la zona transformada por el láser alcanza hasta $\sim$100 μm (para la marca S_{69}).

3.6. Espectroscopía Raman

La espectroscopía micro - Raman es una técnica de caracterización de materiales no destructiva que permite estudiar la composición química y la estructura cristalina de diferentes regiones de

Tabla 3.4: Matriz de micrografías ópticas de las exposiciones fijas.

	1.4 mJ/cm^2	1.9 mJ/cm^2	2.5 mJ/cm^2	3.1 mJ/cm^2	3.6 mJ/cm^2	4.2 mJ/cm^2
2 s	 S_{11}	 S_{21}	 S_{31}	 S_{41}	 S_{51}	 S_{61}
10 s	 S_{12}	 S_{22}	 S_{32}	 S_{42}	 S_{52}	 S_{62}
20 s	 S_{13}	 S_{23}	 S_{33}	 S_{43}	 S_{53}	 S_{63}
30 s	 S_{14}	 S_{24}	 S_{34}	 S_{44}	 S_{54}	 S_{64}
60 s	 S_{15}	 S_{25}	 S_{35}	 S_{45}	 S_{55}	 S_{65}
180 s	 S_{16}	 S_{26}	 S_{36}	 S_{46}	 S_{56}	 S_{66}
360 s	 S_{17}	 S_{27}	 S_{37}	 S_{47}	S_{57}	S_{67}
600 s	S_{18}	S_{28}	S_{38}	S_{48}	S_{58}	S_{68}
1200 s	S_{19}	S_{29}	S_{39}	S_{49}	S_{59}	S_{69}

una muestra, con una resolución espacial en el orden de micrómetros.

El principio físico de esta técnica se basa en el efecto Raman. En este contexto, se sabe que la materia está compuesta por átomos o moléculas conectados por enlaces químicos, los cuales pueden organizarse formando redes cristalinas. Estas estructuras están en constante movimiento vibracional y rotacional, con oscilaciones que ocurren a frecuencias determinadas según la masa de las partículas involucradas y el comportamiento dinámico de los enlaces existentes. A cada uno de estos movimientos vibracionales y rotacionales de la molécula le corresponde un valor específico de energía molecular. En la Figura 3.4, se presenta un diagrama energético en el que cada estado de energía se representa mediante una línea horizontal.

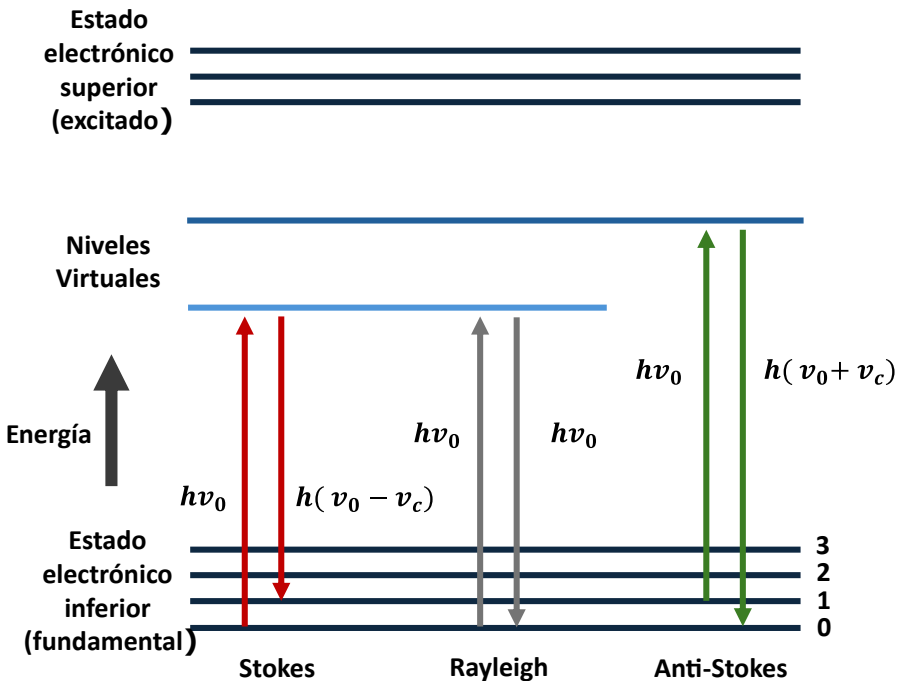


Figura 3.4: Diagrama de niveles de energía en el que las líneas horizontales representan diferentes estados vibracionales. Se ilustran las tres formas de dispersión de la radiación electromagnética.

En el efecto Raman, al incidir un haz de luz monocromático (láser) sobre un material, pueden ocurrir varios fenómenos en su superficie. Los fotones incidentes pueden reflejarse, transmitirse o dispersarse. En particular, la dispersión puede ser de tres tipos: Rayleigh (indicada por flechas verdes), Raman Stokes (flechas naranjas) y anti-Stokes (flechas azules), como se muestra en la Figura 3.5.

En la dispersión Rayleigh, o dispersión elástica de fotones, la energía del fotón incidente (hv_o) es igual al fotón dispersado (hv_o), es la parte dominante en la interacción y no aporta ninguna información sobre la composición de la muestra analizada. La mayor cantidad de luz dispersada es Rayleigh.

La dispersión Raman, o dispersión inelástica de fotones, se asocia a una diferencia en la fre-

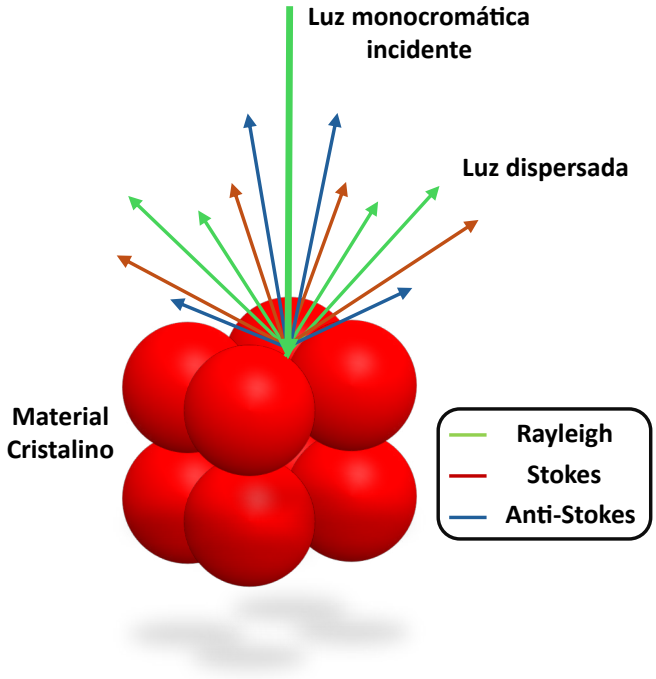


Figura 3.5: Efecto Raman.

cuencia entre el fotón incidente y el fotón dispersado. En este proceso, la luz es dispersada por la red del material que se encuentra en movimiento vibracional. En el caso de la dispersión Stokes, el fotón incidente cede energía a la molécula, de modo que el fotón dispersado tiene una energía de $h(v_o - v_c)$, y la luz experimenta un corrimiento hacia el rojo con respecto a la frecuencia de la luz incidente.

Por otro lado, en la dispersión anti-Stokes, el fotón gana energía de la molécula, lo que da lugar a un fotón dispersado con energía $h(v_o + v_c)$, y la luz se desplaza hacia el azul en el espectro respecto a la frecuencia de la luz incidente. La diferencia de energía entre los fotones incidente y dispersado está relacionada con la creación o destrucción de un fonón, es decir, la energía asociada a la transición entre estados vibracionales del material (ver Figura 3.4).

Un sistema de espectroscopía micro - Raman consiste en un láser de emisión continua (CW), que puede ser un láser He-Ne, o bien un Nd:YAG, o algún diodo láser, el cual genera el haz de luz incidente sobre la muestra. Este haz se enfoca en el área de interés mediante un objetivo convencional de microscopio. El mismo objetivo recoge la luz reflejada, que incluye tanto la dispersión Rayleigh como la dispersión Raman. La luz colectada se analiza utilizando un monocromador y se detecta con un fotomultiplicador o una cámara CCD, siendo la señal procesada a través de un software especializado.

En este caso en particular, se utilizó un sistema micro-Raman (Jobin-Yvon-Horiba, modelo LabRaman HR - 800) con una resolución espacial de aproximadamente 2 μm. La fuente de excitación fue un láser He-Ne de 632.8 nm, y para enfocar el haz láser sobre la muestra se empleó un microscopio óptico (Olympus BX41) con un objetivo de 100X, que también se utilizó para recolectar la luz dispersada y dirigirla a un espectrómetro de 80 cm de longitud focal, con una rejilla de difracción de

600 líneas/mm. La detección se llevó a cabo mediante un detector fotomultiplicador (Hamamatsu, modelo 2761).

3.7. Caracterización Raman

Como se menciono anteriormente, la espectroscopia Raman es una de las herramientas de mayor utilidad para la identificación de estructuras cristalinas en materiales, por lo tanto, se lleva a cabo la caracterización Raman de los óxidos de molibdeno obtenidos en las zonas transformadas por efecto de la irradiación láser. La tabla 3.5, presenta las posiciones de los picos que caracterizan a algunos óxidos de molibdeno y que han sido reportados en la literatura por distintos autores.

La Tabla 3.6 presenta los espectros Raman tomados en el centro de cada una de las marcas grabadas en el proceso de irradiación. Los espectros Raman se adquirieron utilizando una cintura del láser de He-Ne de 2 μm de diámetro. Esto último permite una resolución espacial suficiente para tomar varios espectros en la dirección radial de cada marca.

3.7.1. Análisis por bloque del espectro Raman

Es importante realizar un estudio espacial detallado de las características Raman, dentro de cada una de las zonas transformadas por la irradiación láser, para una mejor identificación de los mismos. Para ello, se presenta la Figura 3.6, donde se muestra un espectro representativo de cada región.

En los espectros para las marcas $S_{11} - S_{14}$ (graficados en guinda) se aprecia una banda muy poco intensa, el espectro Raman 3.6a indica la formación de óxido de molibdeno amorfo [47].

Para las marcas $S_{15} - S_{19}$, $S_{21} - S_{24}$ y S_{31} (espectros Raman graficados en azul) el espectro 3.6b muestra picos en las posiciones 208, 234, 350, 368, 462, 474, 501, 577, 593 y 748 cm^{-1}, que de acuerdo a la Tabla 3.6, pueden ser atribuidos a la fase cristalina monoclínica del dióxido de molibdeno ($m - MoO_2$).

En el rango de fluencias altas y tiempos de exposición cortos (espectros Raman graficados en verde), el espectro 3.6c muestra la aparición de bandas localizadas entre 227, 247, 309, 321, 339, 373, 395, 435, 597, 923 y 954 cm^{-1}, que se atribuyen a la aparición de la fase monoclínica Mo_8O_{23} ($\mathbf{m - Mo_8O_{23}}$) [45].

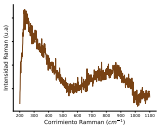
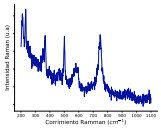
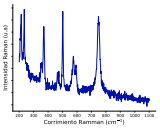
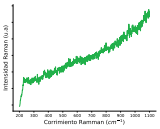
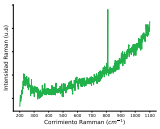
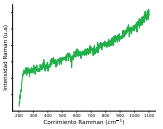
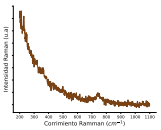
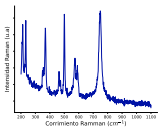
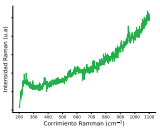
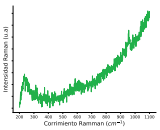
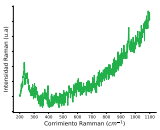
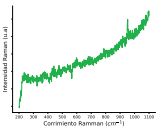
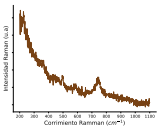
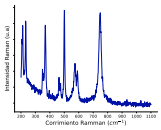
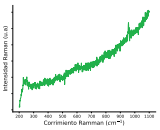
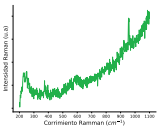
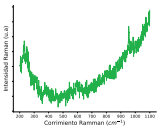
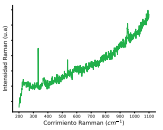
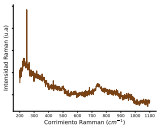
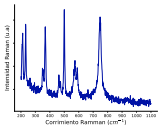
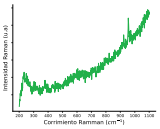
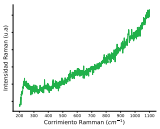
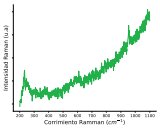
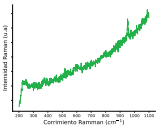
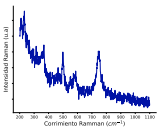
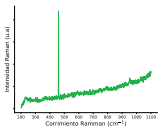
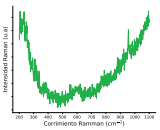
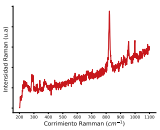
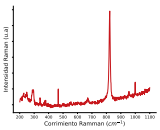
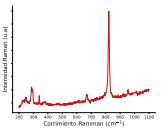
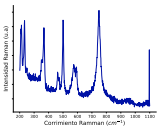
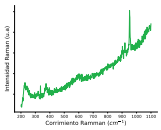
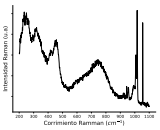
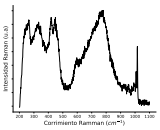
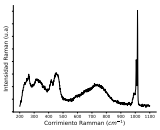
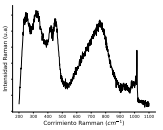
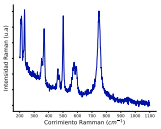
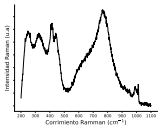
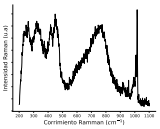
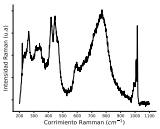
En las marcas S_{45}, S_{55}y S_{65} (espectros Raman graficados en rojo), el espectro 3.6d presenta picos localizados en 223, 246, 285, 293, 340, 370, 381, 669, 821, 955 y 997 cm^{-1} que de acuerdo a [41] et al. y [42] es asignado a la fase cristalina alfa del trióxido de molibdeno ($\alpha - MoO_3$).

Finalmente, en el caso de irradiaciones con fluencias altas y tiempos de exposición largos (espectros Raman graficados en negro), el espectro 3.6e presenta bandas y picos en las posiciones 260, 319, 345, 418, 445, 595, 762, 986, 1007 y 1014 cm^{-1} que de acuerdo a [43] muestran similitud con la fase cristalina ortorrómbica del óxido $Mo_{18}O_{52}$ ($o - Mo_{18}O_{52}$).

Tabla 3.5: Frecuencias Raman (cm^{-1}) para distintos óxidos de molibdeno.

m-MoO_2 [41]	m-MoO_2 [42]	MoO_2 [43]	o-Mo_4O_{11} [44]	o-Mo_4O_{11} [43]	m-Mo_8O_{23} [43]	m-Mo_8O_{23} [45]	o-$Mo_{18}O_{52}$ [43]	α-MoO_3 [41]	α-MoO_3 [45]	α-MoO_3 [42]	α-MoO_3 [46]
			183	171 190				117 130 159 199		115 129 158 198	148 180
203 208 228	203 208 229	208 332	206 229 264	208 253 281	208 256 282	222 245	208	219 244 285 293	220 246 290	217 245 284 291	222 253
345 363	346 350 365	353 370	306 340 380	339 359 380	309 338 352 377 394	373		338 368 379	337 378	338 365 379	316 396
461 363	459 469 496	448 473	431 452 498	413 435 454	43 480		465	470	473	472	414 492
571 589	569 588	501 572 590	568			592	547				
					668	654	622 637 679	668	668	666	690
744	744	748	744 790	714 787	709 733		761				
			835 843	837 892	811 880 899	875		821	820 875	820	
			907 985	916 963		918 951	906 931 951 972 989	997	902 921 953 996	996	900 973
							1001		1009		

Tabla 3.6: Matriz de espectros Raman de las exposiciones fijas.

	1.4 mJ/cm^2	1.9 mJ/cm^2	2.5 mJ/cm^2	3.1 mJ/cm^2	3.6 mJ/cm^2	4.2 mJ/cm^2
2 s	 S_{11}	 S_{21}	 S_{31}	 S_{41}	 S_{51}	 S_{61}
10 s	 S_{12}	 S_{22}	 S_{32}	 S_{42}	 S_{52}	 S_{62}
20 s	 S_{13}	 S_{23}	 S_{33}	 S_{43}	 S_{53}	 S_{63}
30 s	 S_{14}	 S_{24}	 S_{34}	 S_{44}	 S_{54}	 S_{64}
60 s	 S_{15}	 S_{25}	 S_{35}	 S_{45}	 S_{55}	 S_{65}
180 s	 S_{16}	 S_{26}	 S_{36}	 S_{46}	 S_{56}	 S_{66}
360 s	 S_{17}	 S_{27}	 S_{37}	 S_{47}	S_{57}	S_{67}
600 s	S_{18}	S_{28}	S_{38}	S_{48}	S_{58}	S_{68}
1200 s	S_{19}	S_{29}	S_{39}	S_{49}	S_{59}	S_{69}

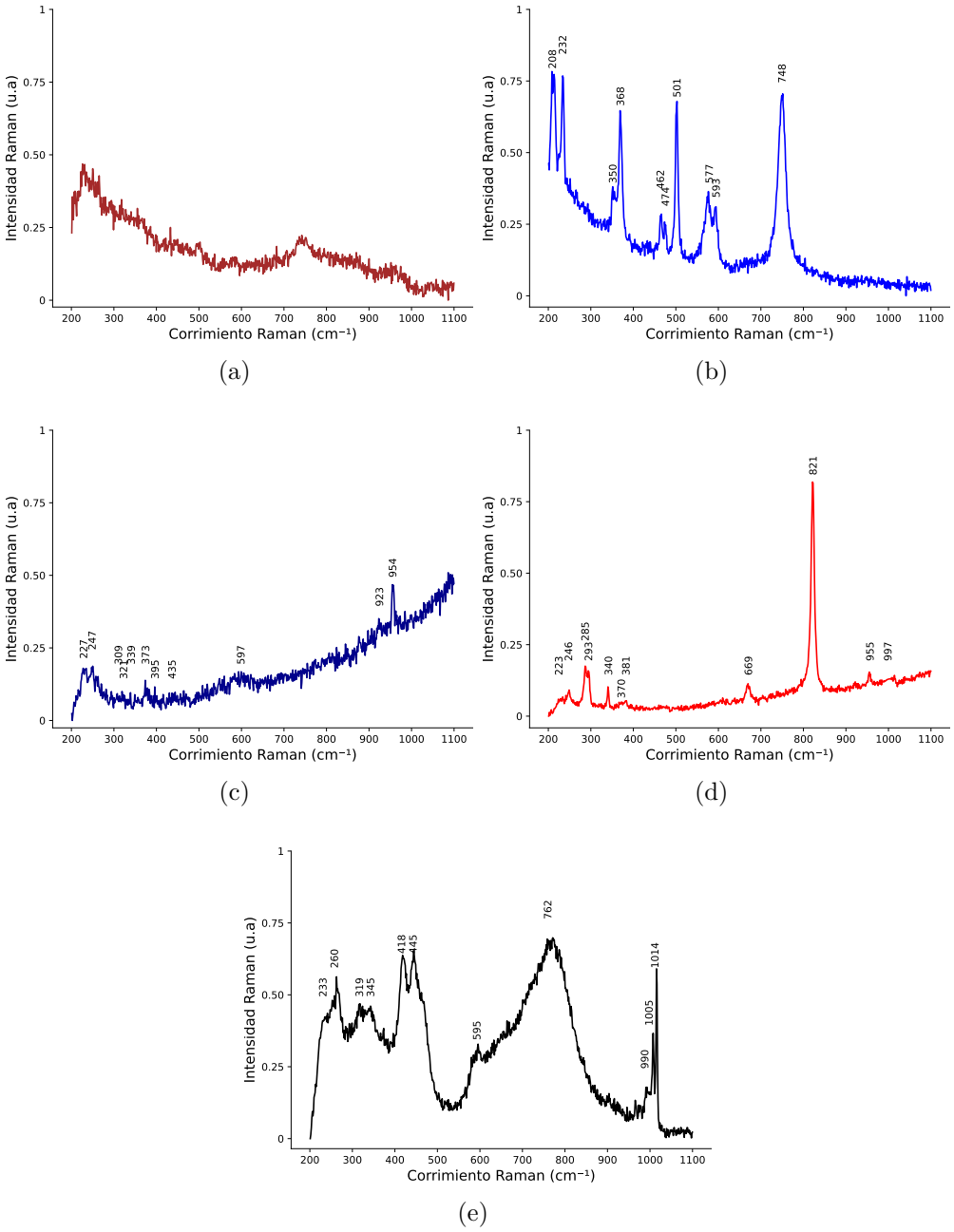


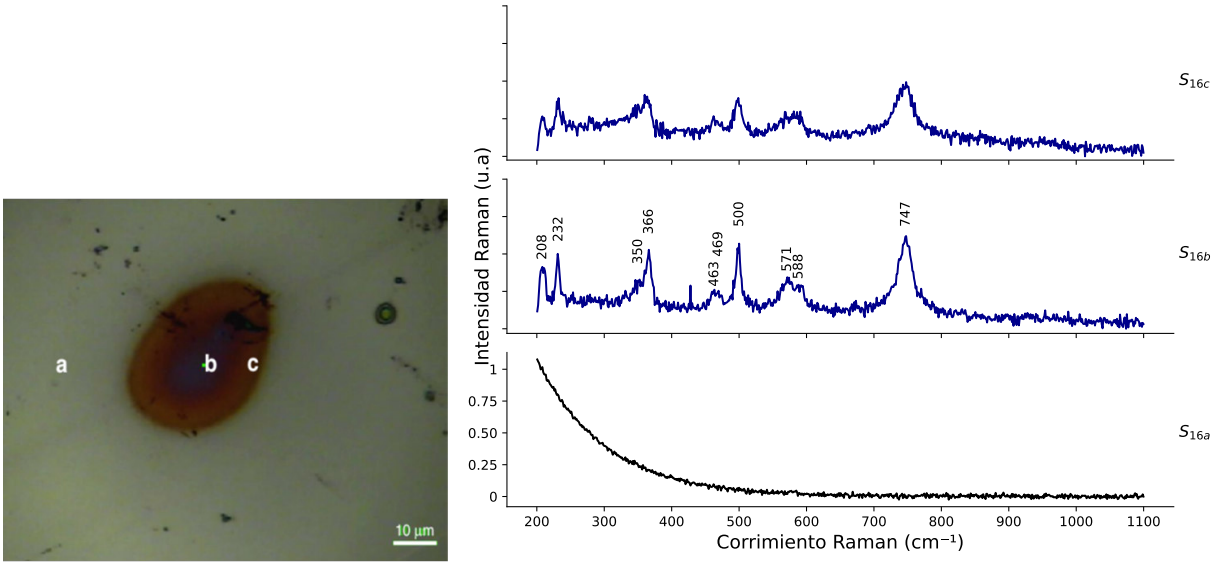
Figura 3.6: Espectros Raman representativos de matriz de espectros de la Tabla 3.6 . a) Óxido de molibdeno amorfo, b) $m - MoO_2$, c) $m - Mo_8O_{23}$, d) $\alpha - MoO_3$ y e) $o - Mo_{18}O_{52}$.

3.8. Características estructurales en la dirección radial para las exposiciones fijas

Como siguiente paso, es importante conocer los óxidos que pertenecen los diferentes anillos que puedan contener cada una de las microscopías que se muestran en la Figura 3.4. Para ello, se presentan los resultados de espectroscopía micro - Raman obtenidos en las mediciones que se realizaron en la dirección radial, para cada una de las regiones irradiadas en la capa de molibdeno. Para llevar a cabo este estudio detallado de espectroscopía micro - Raman se eligió una marca

representativa de cada bloque, ya que cada bloque presenta características similares de oxidación. Bajo este criterio a continuación se presenta el estudio detallado de las marcas S_{ij}. Se destaca que focalizando el haz láser del sistema micro - Raman se alcanza una resolución espacial de 2 μm, por lo tanto, es posible capturar el espectro Raman y analizar de manera diferenciada cada uno de los anillos que conforman la región transformada por la irradiación láser. En cada micrografía óptica se etiquetaron por medio de caracteres alfabéticos los sitios analizados por espectroscopía micro - Raman.

3.8.1. Bloque 1 ($m - MoO_2$)



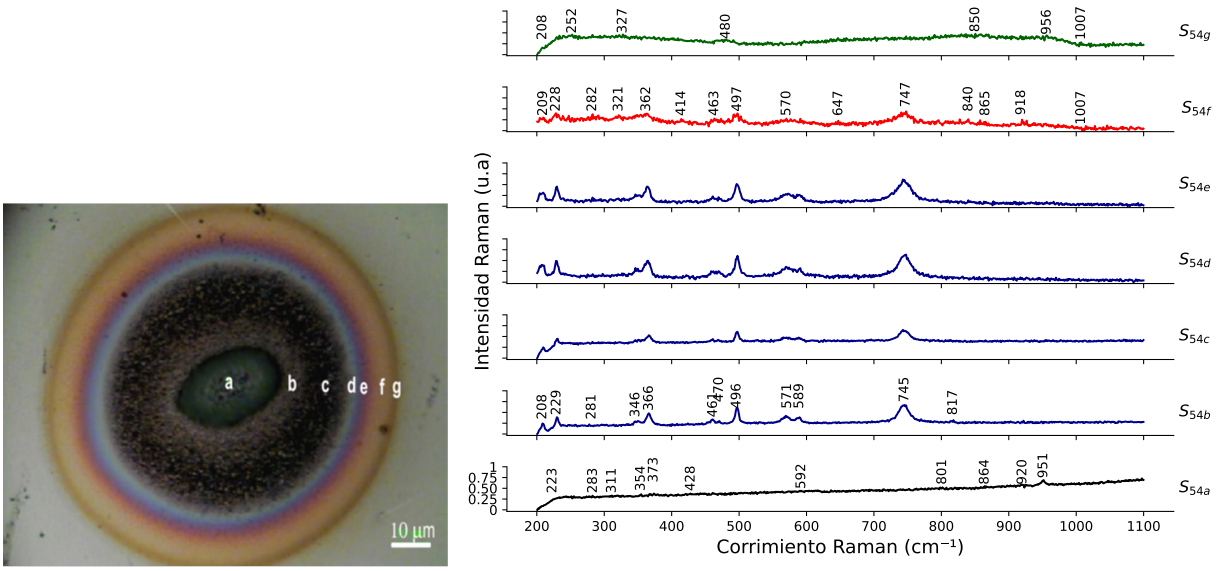
(a) (b)

Figura 3.7: Micrografía óptica y espectros Raman de las zonas marcadas para la irradiación fija S_{16}, donde se aplicó una fluencia de 1.4 mJ/cm^2 y un tiempo de exposición de 3 min.

- El espectro Raman S_{16a} presenta la ausencia de picos y corresponde a una zona no irradiada (**molibdeno metálico cristalino**).
- En el espectro Raman S_{16b} se pueden observar picos bien definidos en 208, 232, 350, 366, 463, 469, 500, 571, 588 y 747 cm^{-1}, éstos pueden ser asignados a la fase monoclínica del dióxido de molibdeno ($\mathbf{m - MoO_2}$).
- En el espectro Raman S_{16c} se observa una menor intensidad de los picos, y la posición de cada uno de éstos no se modifica. Esto corresponde a una menor cantidad o volumen de material transformado en $\mathbf{m - MoO_2}$.

3.8.2. Bloque 2 ($m - Mo_8O_{23}$)

- El espectro Raman S_{54a} presenta picos alrededor de las posiciones 233, 283, 311, 354, 373, 428, 592, 801, 864, 920 y 951 cm^{-1}. Éstos pueden ser asignados a la fase $\mathbf{m - Mo_8O_{23}}$.



(a) (b)

Figura 3.8: Micrografía óptica y espectros Raman de las zonas marcadas para la irradiación fija S_{54}, donde se aplicó una fluencia de 1.4 mJ/cm^2 y un tiempo de exposición de 30 seg.

- Los espectros S_{54b} - S_{54e} presentan picos en las posiciones 208, 229, 281, 346, 366, 461, 470, 496, 571, 589, 745 y 817 cm^{-1} que corresponden a la fase $m - MoO_2$. Además, se presentan picos localizados alrededor de 281 y 817 cm^{-1}, los cuales indican una mezcla de fases entre $\mathbf{m - MoO_2}$ y $\mathbf{m - Mo_8O_{23}}$.

- En el espectro etiquetado como S_{54f} se presentan picos en las posiciones 209, 228, 282, 321, 414, 463, 497, 570, 747, 840 y 918 cm^{-1} el cual corresponde a la fase $o - Mo_4O_{11}$, además se presentan los picos localizados alrededor de 647, 865 y 1007 cm^{-1} los cuales indican una mezcla de fases entre $\mathbf{o - Mo_4O_{11}}$ y $\mathbf{m - Mo_8O_{23}}$.

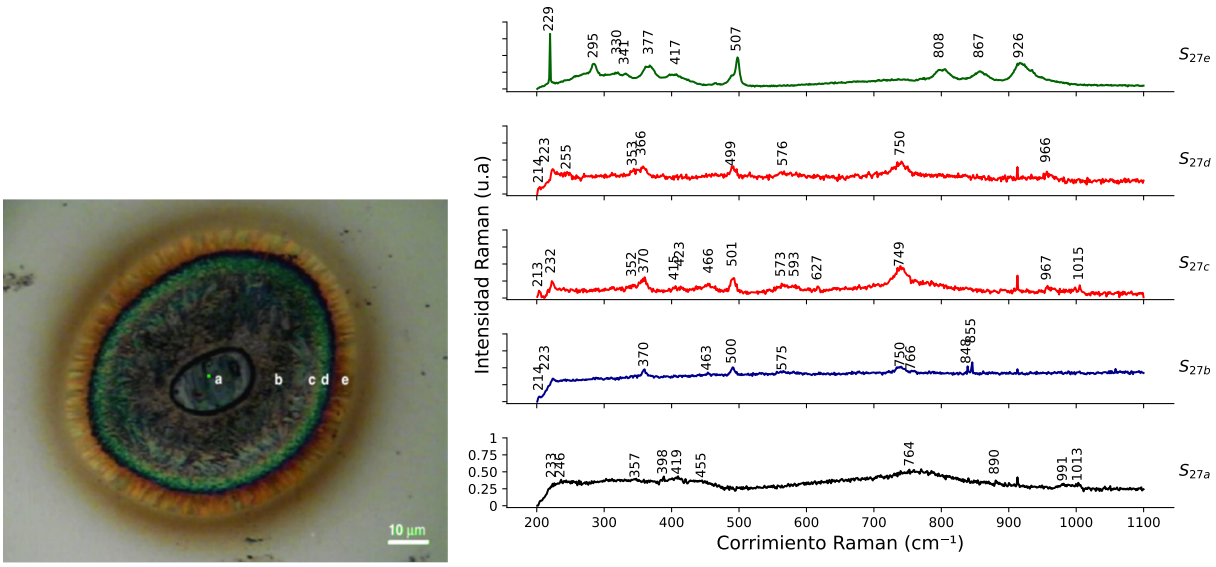
- En la periferia, el espectro S_{54g} presenta bandas ubicadas en 208, 252, 327, 480, 850, 956, 1007 cm^{-1} que corresponde a la fase $\mathbf{m - Mo_8O_{23}}$.

3.8.3. Bloque 3 ($m - Mo_{18}O_{52}$)

- En el centro de la irradiación, el espectro S_{27a} presenta picos ubicados en 233, 246, 357, 398, 419, 455, 769, 890, 991 y 1013 cm^{-1}, que pueden ser asignados a la fase $\mathbf{o - Mo_{18}O_{52}}$.

- En el espectro S_{27b}, aparecen picos en las posiciones 214, 233, 370, 463, 500, 575 y 750 cm^{-1} que pertenecen a la fase $m - MoO_2$, además en este espectro se presentan picos localizados alrededor de 766, 848 y 855 cm^{-1} los cuales indican una mezcla de fases entre $\mathbf{m - MoO_2}$ y $\mathbf{o - Mo_4O_{11}}$.

- En los anillos concéntricos (c - e) se presentan picos alrededor de las posiciones 213, 232, 352, 370, 415, 423, 466, 501, 573, 593 y 749 cm^{-1} que se pueden asignar a una mezcla de las fases entre $m - MoO_2$ y $o - Mo_4O_{11}$. Adicionalmente, se encuentran picos en las posiciones 627, 967 y 1015 cm^{-1} que pertenecen a la fase ortorrómbica $Mo_{18}O_{52}$ ($\mathbf{o - Mo_{18}O_{52}}$).

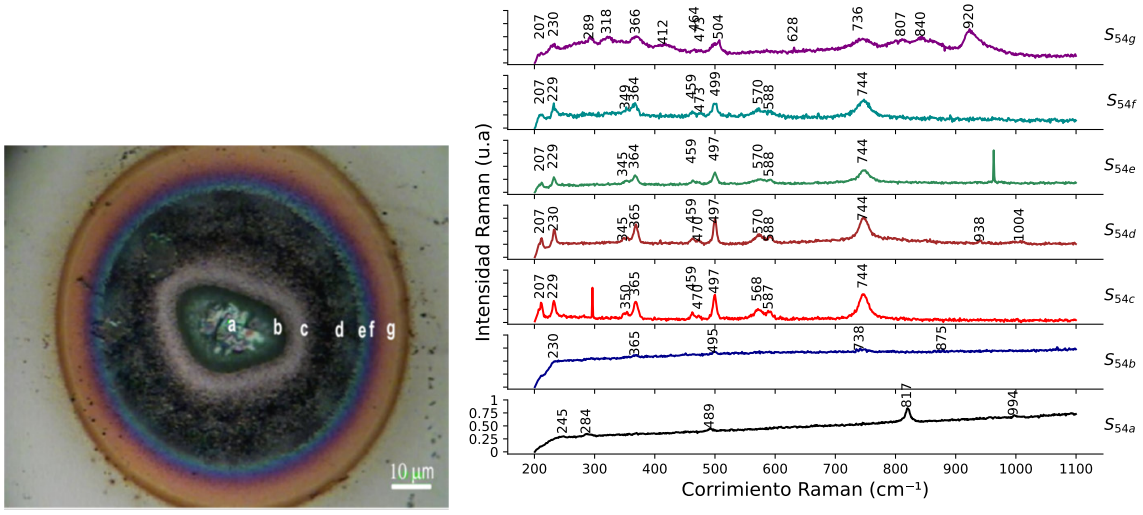


(a) (b)

Figura 3.9: Micrografía óptica y espectros Raman de las zonas marcadas para la irradiación fija S_{27}, donde se aplicó una fluencia de 1.9 mJ/cm^2 y un tiempo de exposición de 6 min.

- El espectro S_{27e} que corresponde a la periferia de la zona transformada, presenta picos localizados alrededor de 229, 295, 330, 341, 377, 417, 473, 507, 808, 867 y 926 cm^{-1} que se pueden asignar a la mezcla de fases $\mathbf{o - Mo_4O_{11}}$ y $\mathbf{m - Mo_8O_{23}}$.

3.8.4. Bloque 4 ($\alpha - MoO_3$)



(a) (b)

Figura 3.10: Micrografía óptica y espectros Raman de las zonas marcadas para la irradiación fija S_{65}, donde se aplicó una fluencia de 4.2 mJ/cm^2 y un tiempo de exposición de 1 min.

- En el espectro S_{65a} se observan picos en las posiciones de 245, 284, 489, 817 y 994 cm^{-1} los cuales se atribuyen a la fase alfa del trióxido de molibdeno ($\alpha - \mathbf{MoO_3}$).
- El espectro correspondiente a la posición S_{65b} presenta picos en 230, 365, 495, 738 y 875 cm^{-1} asociados a la fase monoclínica del dióxido de molibdeno ($\mathbf{m - MoO_2}$).
- En la región de los anillos concéntricos (c - f), se presentan picos con una mejor intensidad. En las posiciones 207, 229, 360, 365, 459, 470, 497, 568, 587 y 744 cm^{-1} los cuales corresponden a una mayor cantidad de material de la fase monoclínica del MoO_2 ($\mathbf{m - MoO_2}$).
- Para el caso del anillo (d) se presentan picos adicionales en las posiciones 938 y 1004 cm^{-1}, que indican una mezcla de fases $\mathbf{m - MoO_2}$ y $\mathbf{o - Mo_{18}O_{52}}$.
- Para el anillo (g), el espectro Raman presentó picos en las posiciones 207, 230, 289, 318, 366, 412, 464, 473, 504, 628, 736, 807, 840 y 920 cm^{-1} los cuales se pueden considerar como una mezcla de fases $\mathbf{m - MoO_2}$, $\mathbf{o - Mo_4O_{11}}$ y $\mathbf{m - Mo_8O_{23}}$.

3.9. Óxidos obtenidos mediante la técnica de exposición fija

Dadas las características descritas de cada uno de los grupos en la sección anterior, es posible realizar una clasificación de los óxidos de molibdeno de diversas maneras. En este trabajo, se ha decidido separar los óxidos tanto por su coloración como por su fase. Esta clasificación se basa en los datos presentados previamente, los cuales nos permiten establecer diferencias claras en términos visuales y estructurales.

La separación por coloración toma en cuenta las variaciones en los tonos observados en los óxidos, lo cual es relevante no sólo desde un punto de vista visual, sino también en términos de propiedades químicas y físicas que pueden correlacionarse con estos colores.

Por otro lado, la separación por su fase analiza las características morfológicas de los óxidos, lo que es crucial para entender su formación, comportamiento bajo distintas condiciones, y posibles aplicaciones en el ámbito de los materiales.

En la Tabla 3.7 se presenta una clasificación detallada basada en los criterios mencionados, proporcionando una vista clara de cómo se organizan los diferentes tipos de óxidos según estos dos enfoques.

Tabla 3.7: Tabla de clasificación de los óxidos obtenidos mediante la técnica de irradiación láser mediante exposición fija. Separados según su estructura y coloración.

Etiqueta	Coloración	Fase
Bloque 1		
0	verde grisáceo (Zona no irradiada)	Mo
1	Café claro	Óxido amorfo
2	Café - oscuro	$m - MoO_2$
3	Púrpura	$m - MoO_2$
Bloque 2		
4	Azul	$m - Mo_8O_{23}$
5	Gris claro	$m - MoO_2$ y $m - Mo_8O_{23}$
6	Gris obscuro	$m - MoO_2$ y $m - Mo_8O_{23}$
7	Azul - Claro	$m - MoO_2$ y $m - Mo_8O_{23}$
8	Rojo	$m - MoO_2$ y $m - Mo_8O_{23}$
9	Amarillo - Beige	$o - Mo_4O_{11}$ y $m - Mo_8O_{23}$
10	Café	$m - Mo_8O_{23}$
Bloque 3		
11	Azul	$m - Mo_{18}O_{52}$
12	Café grisáceo	$m - MoO_2$ y $m - Mo_4O_{11}$
13	Verde y Azul obscuro	$m - MoO_2$, $m - Mo_4O_{11}$ y $m - Mo_{18}O_{52}$
14	Amarillo - naranja	$o - Mo_4O_{11}$ y $m - Mo_8O_{23}$
Bloque 4		
15	Verde esmeralda	$\alpha - MoO_3$
16	Verde Obscuro	$m - MoO_2$
17	Gris claro	$m - MoO_2$
18	Gris oscuro	$m - MoO_2$ y $m - Mo_8O_{52}$
19	Café Claro e intenso	$m - MoO_2$, $m - Mo_8O_{52}$ y $o - Mo_4O_{11}$

Capítulo 4

Metodología

Contenido

Como se ha mencionado anteriormente, este trabajo tiene la finalidad de implementar diversas técnicas de aprendizaje profundo para predecir los efectos de la irradiación de un láser ultrapuulsado en placas delgadas de molibdeno. Este estudio se dividirá en dos etapas principales, cada una enfocada en abordar aspectos específicos de la problemática.

En la primera etapa, nos centraremos en la aplicación de redes neuronales profundas con el objetivo de determinar el diámetro de los spots de óxido generados por el haz de luz. Esta tarea implica una exploración exhaustiva de cómo varían los parámetros del láser, en particular el tiempo de exposición y la fluencia. A través de esta etapa, buscaremos establecer un modelo predictivo que no sólo sea capaz de estimar con precisión el diámetro de los spots, sino que también ofrezca

información valiosa sobre la relación entre los parámetros del láser y el tamaño del óxido resultante.

En la segunda etapa del estudio, se emplearán redes neuronales convolucionales para realizar la segmentación de los diferentes tipos de óxidos que se forman como resultado de la irradiación del láser. Este enfoque permitirá identificar y clasificar de manera precisa las variaciones en la morfología de los óxidos, correlacionando también estas observaciones con los parámetros del láser previamente mencionados. Al aplicar técnicas de segmentación, pretendemos no sólo mejorar la precisión de la clasificación, sino también profundizar en la comprensión de cómo los cambios en los parámetros del láser afectan la formación y características de los óxidos en las placas de molibdeno.

En resumen, este trabajo busca integrar el uso de redes neuronales profundas y convolucionales para ofrecer una visión integral sobre el comportamiento de los óxidos generados por la interacción del láser con el molibdeno, sentando las bases para futuras investigaciones en el campo de la física y la ingeniería de materiales.

Algo importante a destacar es que para realizar todos los entrenamientos de los modelos, tanto de regresión (redes profundas) como de segmentación (redes convolucionales), se utilizó Python como lenguaje de programación, empleando el framework TensorFlow en conjunto con Keras. Esto permite una implementación eficiente y flexible de los algoritmos, facilitando tanto el ajuste de los modelos como el manejo de los datos y las pruebas.

Caso de estudio 1: Predicción de diámetros

4.1. Aumento de los datos por interpolación

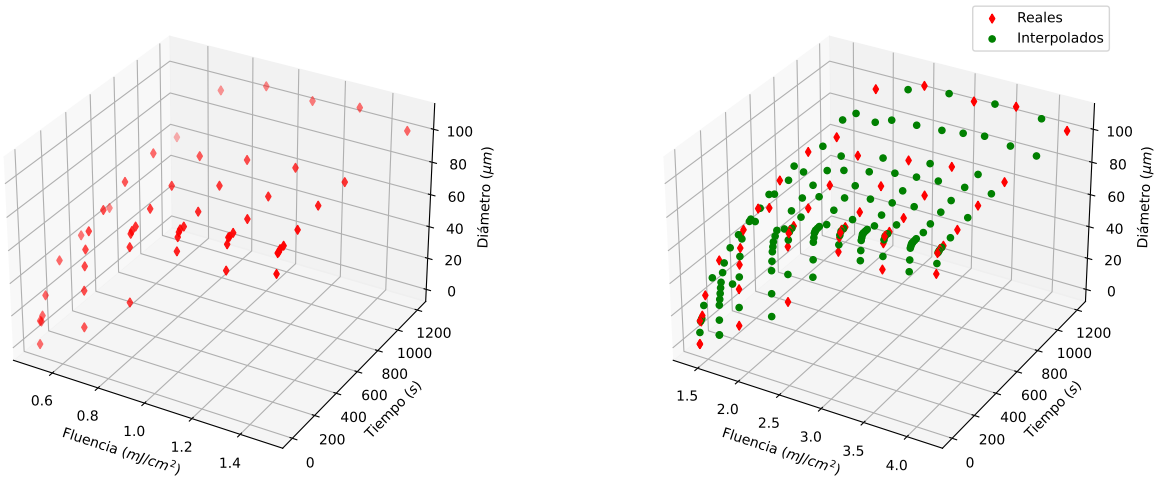
Antes de detallar la serie de pruebas que se llevarán a cabo para obtener el modelo más óptimo en la estimación de los diámetros de los óxidos, es fundamental destacar un aspecto clave del proceso. Dado que nuestro conjunto de datos es pequeño, es necesario implementar técnicas de aumento de datos para mejorar la capacidad generalizadora del modelo. Estas técnicas nos permitirán generar nuevas instancias a partir de las muestras originales, lo que ayudará a evitar el sobreajuste y asegurar que el modelo pueda desempeñarse adecuadamente cuando se enfrente a datos nuevos o no vistos.

Para este caso, el conjunto de datos son valores numéricos, los cuales corresponden a las variables: fluencia, tiempo y diámetro (Tabla 4.1). Para aumentar los datos se decidió utilizar interpolación lineal, dada por la Ec. 2.44. La interpolación lineal nos permite obtener un valor que esté en el intervalo de 2 puntos conocidos, por lo que nuestro valor a estimar sólo puede depender de 1 variable. Dados los datos podemos ver que tenemos 3 variables (fluencia, tiempo y diámetro), por lo

que podemos determinar el valor de 1 variable mientras dejamos una constante. Para ser más claro en este proceso, podemos ver la gráfica de la Figura 4.1, donde se grafican los datos de tiempo, fluencia, y diámetro en el los ejes xyz, respectivamente.

Tabla 4.1: Recopilación de datos experimentales del diámetros (μm) para diferentes valores de tiempo (s) y fluencia (mJ/cm^2).

Fluencia (mJ/cm^2)	**Tiempo** (s)								
	2	**10**	**20**	**30**	**60**	**180**	**360**	**600**	**1200**
1.4	0	14.1	13.05	16.1	26.84	41.05	45.68	48.52	59.26
1.9	19.57	41.57	55.78	65.26	74.42	80	86.42	90.1	96.1
2.5	43.68	76.31	83.57	84.63	85.47	88.84	92	96.1	105.89
3.1	82.73	90.52	92.84	93.89	93.68	95.26	100	101.57	104.21
3.6	80	94.73	98.1	97.89	98.31	99.15	101.68	104.73	107.57
4.2	86.63	98.1	98.21	99.15	99.26	101.15	104.42	104.21	101.05

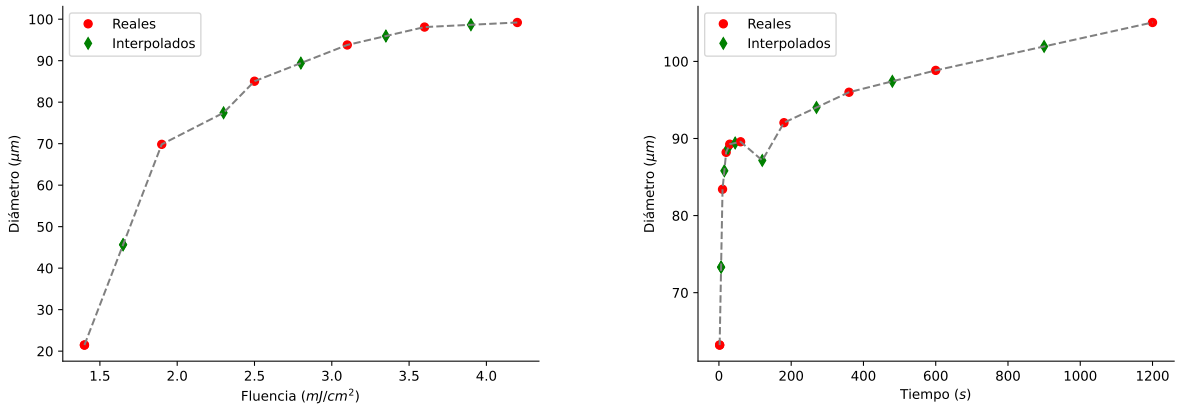


(a) (b)

Figura 4.1: Gráficas 3D del conjunto de datos. (a) Representación gráfica de los diámetros en función del tiempo (s) y fluencia (mJ/cm^2). (b) Representación gráfica del conjunto de datos una vez aplicada la interpolación lineal.

Para determinar lo valores que ese están entre los datos que tenemos tomaremos, primeramente, un tiempo constante, digamos 30 s y calculamos por interpolación el valor que está a la mitad de un punto y otro, como se muestra en la Figura 4.2b, donde podemos ver que los valores estimados con interpolación (diamantes verdes) se calculan utilizando los valores reales en cada uno de sus extremos (círculos rojos), es decir, se calcula el valor que está a la mitad de la distancia entre cada valor real. Un ejemplo de como se determina este valor, es tomar los valores de tiempo 30 s y 60 s, el valor que esta a la mitad entre estos dos valores sera 45 s, aplicamos la Ec. 2.44 y obtenemos el valor estimado del diámetro. Ahora, 45 s será aun nuevo valor de tiempo, el cual adjuntaremos

a muestro conjunto de datos.



(a) (b)

Figura 4.2: Ejemplos de interpolación lineal realizada en los datos de la tabla 4.1. (a) Ejemplo de interpolación manteniendo constante el tiempo a 30 s. (b) Otro ejemplo, ahora dejando una fluencia constante a 2.5 mJ/cm^2 .

En la Tabla 4.2 se muestran los resultados de la interpolación lineal, donde los valores dentro de los recuadros Azules. Como se mencionó, los valores se calcularon de varias formas, donde aquellos valores dentro de la tabla de color **azul** son aquellos valores que se calcularon dejando el valor de la fluencia constante, y el valor se calcula sólo variando el tiempo. En otras palabras, los calores se calculan renglón por renglón. Los valores en color **rojo** son aquéllos que se calcularon dejando el tiempo constante y variando la fluencia (columna por columna). Finalmente, los valores en color **verde** se determinaron de forma diferente, ya que estos valores se calcularon usando sólo los valores interpolados, ya que no hay valores reales en sus extremos para utilizar, por lo que para hacer una mejor aproximación se calculó la interpolación tanto con el tiempo como la fluencia constante, es decir, se utilizaron tanto los extremos de los renglones cercanos como los valores de la columna. Los valores encerrados en los recuadros rojos son los valores que se utilizaron para realizar las pruebas de las redes, esto lo veremos con más detalle en la sección 4.1.

4.2. Normalización de los datos

Antes de realizar cualquier prueba, es importante normalizar los datos que se usan para la entrada (Fluencia y tiempo), esto con la finalidad que los datos no tengan un alto rango de separación entre los valores de tiempo y fluencia. De esta forma, se obtiene un rango estándar (que va de 0 a 1) para los valores, que facilita el proceso de aprendizaje del modelo. En la Tabla 4.3 se muestra los valores reales normalizados. A partir de estos, también se normalizarán los valores que calculamos mediante la interpolación.

Tabla 4.2: Aumentos de los datos de la Tabla 4.1 utilizando interpolación lineal.

Fluencia (mJ/cm^2)	Tiempo (s)										
	1.4	1.525	1.65	1.775	1.9	2.1	2.3	2.4	2.5	2.65	2.8
2	0	4.8925	9.785	14.6775	19.57	25.5975	31.625	37.6525	43.68	53.4425	63.205
6	7.05	12.93	18.81	24.69	30.57	37.92625	45.2825	52.63875	59.995	66.6525	73.31
10	14.1	20.9675	27.835	34.7025	41.57	50.255	58.94	67.625	76.31	79.8625	83.415
15	13.575	22.35	31.125	39.9	48.675	56.49125	64.3075	72.12375	79.94	82.875	85.81
20	13.05	23.7325	34.415	45.0975	55.78	62.7275	69.675	76.6225	83.57	85.8875	88.205
25	14.575	26.06125	37.5475	49.03375	60.52	66.415	72.31	78.205	84.1	86.41625	88.7325
30	16.1	28.39	40.68	52.97	65.26	70.1025	74.945	79.7875	84.63	86.945	89.26
45	21.47	33.5625	45.655	57.7475	69.84	73.6425	77.445	81.2475	85.05	87.23375	89.4175
60	26.84	38.735	50.63	62.525	74.42	77.1825	79.945	82.7075	85.47	87.5225	89.575
120	33.945	44.76125	55.5775	66.39375	77.21	79.69625	82.1825	84.66875	87.155	88.98375	90.8125
180	41.05	50.7875	60.525	70.2625	80	82.21	84.42	86.63	88.84	90.445	92.05
270	43.365	53.32625	63.2875	73.24875	83.21	85.0125	86.815	88.6175	90.42	92.2225	94.025
360	45.68	55.865	66.05	76.235	86.42	87.815	89.21	90.605	92	94	96
480	47.1	57.39	67.68	77.97	88.26	89.7075	91.155	92.6025	94.05	95.73375	97.4175
600	48.52	58.915	69.31	79.705	90.1	91.6	93.1	94.6	96.1	97.4675	98.835
900	53.89	63.6925	73.495	83.2975	93.1	95.07375	97.0475	99.02125	100.995	101.46875	101.9425
1200	59.26	68.47	77.68	86.89	96.1	98.5475	100.995	103.4425	105.89	105.47	105.05

Fluencia (mJ/cm^2)	Tiempo (s)									
	2.95	3.1	3.225	3.35	3.475	3.6	3.75	3.9	4.05	4.2
2	72.9675	82.73	82.0475	81.365	80.6825	80	81.6575	83.315	84.9725	86.63
6	76.289375	86.625	86.81	86.995	87.18	87.365	88.615	89.865	91.115	92.365
10	86.9675	90.52	91.5725	92.625	93.6775	94.73	95.5725	96.415	97.2575	98.1
15	84.836875	91.68	92.86375	94.0475	95.23125	96.415	96.85	97.285	97.72	98.155
20	90.5225	92.84	94.155	95.47	96.785	98.1	98.1275	98.155	98.1825	98.21
25	88.10125	93.365	94.5225	95.68	96.8375	97.995	98.16625	98.3375	98.50875	98.68
30	91.575	93.89	94.89	95.89	96.89	97.89	98.205	98.52	98.835	99.15
45	89.7	93.785	94.86375	95.9425	97.02125	98.1	98.37625	98.6525	98.92875	99.205
60	91.6275	93.68	94.8375	95.995	97.1525	98.31	98.5475	98.785	99.0225	99.26
120	91.398125	94.47	95.535	96.6	97.665	98.73	99.09875	99.4675	99.83625	100.205
180	93.655	95.26	96.2325	97.205	98.1775	99.15	99.65	100.15	100.65	101.15
270	94.92625	97.63	98.32625	99.0225	99.71875	100.415	101.0075	101.6	102.1925	102.785
360	98	100	100.42	100.84	101.26	101.68	102.365	103.05	103.735	104.42
480	98.3775	100.785	101.39	101.995	102.6	103.205	103.4825	103.76	104.0375	104.315
600	100.2025	101.57	102.36	103.15	103.94	104.73	104.6	104.47	104.34	104.21
900	101.429375	102.89	103.705	104.52	105.335	106.15	105.27	104.39	103.51	102.63
1200	104.63	104.21	105.05	105.89	106.73	107.57	105.94	104.31	102.68	101.05

4.3. Modelos Propuestos

Como se mencionó anteriormente, uno de las herramientas que nos ofrece el *maching learning* don las redes neuronales, el cual, mediante una serie de datos de entrenamiento puede aprender modelos matemáticos. Para nuestro caso, utilizaremos una red neuronal profunda para entrenar un modelo, el cual, pueda predecir el diámetro de los óxidos de molibdeno generados tras ser expuesto a un láser ultrapulsado. Para este experimento se variaron dos características: tiempo y fluencia del láser, los cuales serán nuestros datos de entrada para nuestro perceptrón.

Como es sabido, no existe una regla qué nos diga que hiperparámetros resultan satisfactorios para obtener los mejores resultados, por lo que sólo queda el realizar una serie de pruebas utilizando valores diferentes. Uno de los hiperparámetros que es de gran importancia seleccionar correctamente, es la estructura que tendrá el modelo que queremos entrenar. Es importante seleccionar meticulosamente el número de capas que tendrá nuestra red, así como el número de neuronas que habrá en cada una de ellas. Para esto, se plantearon varias estructuras para estos modelos, donde se varia el número de capas y la cantidad de neuronas que hay en cada una de ellas. La Tabla 4.4 nos muestra algunos modelos que se utilizarán para hacer pruebas.

Para seleccionar un modelo, el cual nos dé el mejor resultado, debemos seleccionar cuidadosamen-

Tabla 4.3: Datos de entrada normalizados (Tiempo y fluencia).

Tiempo (s)		**Fluencia** (mJ/cm^2)	
Valor real	**Normalizado**	**Valor real**	**Normalizado**
2	0	1.4	0
6	0.0033389	1.525	0.04464286
10	0.0066778	1.65	0.08928571
15	0.01085142	1.775	0.13392857
20	0.01502504	1.9	0.17857143
25	0.01919866	2.1	0.25
30	0.02337229	2.3	0.32142857
45	0.03589316	2.4	0.35714286
60	0.04841402	2.5	0.39285714
120	0.0984975	2.65	0.44642857
180	0.14858097	2.8	0.5
270	0.22370618	2.95	0.55357143
360	0.29883139	3.1	0.60714286
480	0.39899833	3.225	0.65178571
600	0.49916528	3.35	0.69642857
900	0.74958264	3.475	0.74107143
1200	1	3.6	0.78571429
		3.75	0.83928571
		3.9	0.89285714
		4.05	0.94642857
		4.2	1

Tabla 4.4: Número de neuronas por capa de algunos modelos utilizados.

Modelos	**Número de neuronas por capa oculta**							**Número de neuronas**	**Número de Parámetros**
	1ercapa	**2^{a} capa**	**3^{a} capa**	**4^{a} capa**	**5^{a} capa**	**6^{a} capa**	**7^{a} capa**		
Modelo 1	32	64	32	*	*	*	*	128	4321
Modelo 2	64	128	64	32	*	*	*	288	18881
Modelo 3	64	128	64	32	16	*	*	304	19393
Modelo 4	128	64	32	16	*	*	*	240	11265
Modelo 5	256	128	54	32	*	*	*	470	42423
Modelo 6	350	256	128	64	32	*	*	830	134171
Modelo 7	700	525	350	175	88	36	*	1874	634379
Modelo 8	700	525	350	175	88	36	18	1892	635027

te otros hiperparámetros. No basta con sólo determinar el número de capas, si no que hay que seleccionar algunos hiperparámetros internos de cada una de las neuronas, los cuales los veremos en las siguientes secciones.

4.4. Pruebas con optimizadores

La selección del optimizador es un aspecto crucial que debemos tener en cuenta, ya que la elección correcta puede marcar una diferencia significativa en los resultados obtenidos. Un optimizador adecuado puede reducir de manera considerable el error del modelo, mientras que una elección menos apropiada podría llevar a un mayor error o a un proceso de entrenamiento más lento e ineficiente. Como ya se ha mencionado, no existe una regla universal para seleccionar el optimizador óptimo para un problema específico; sin embargo, hay recomendaciones generales que sugieren el uso de ciertos optimizadores en tareas de regresión, como es el caso de este estudio.

Dado que la elección del optimizador puede depender de factores como la naturaleza de los datos, la arquitectura del modelo y el comportamiento del entrenamiento, decidimos realizar pruebas exhaustivas con varios de los optimizadores disponibles en la biblioteca de Keras. Entre los optimizadores seleccionados se encuentran Adadelta, AdaGrad, Adam, AdamW, Nadam y RMSProp. Estos optimizadores han sido ampliamente utilizados en diferentes tareas de aprendizaje profundo y ofrecen características particulares que pueden influir en la eficiencia del aprendizaje.

Para evaluar el rendimiento de cada optimizador en nuestro modelo, se llevará a cabo un proceso de entrenamiento independiente para cada uno de ellos. Esto nos permitirá analizar de manera detallada cómo afecta cada optimizador en la convergencia del modelo, la velocidad de aprendizaje y el error final obtenido. A través de estas pruebas comparativas, se buscará determinar cuál de estos optimizadores logra el mejor equilibrio entre la precisión y el tiempo de entrenamiento, ajustándose de manera más efectiva a la naturaleza del problema de predicción de diámetros de óxidos.

Al final, el objetivo es seleccionar el optimizador que, en conjunto con el resto de los hiperparámetros del modelo, ofrezca los mejores resultados para este caso particular, permitiendo así una mejora significativa en la precisión de las predicciones.

4.5. Pruebas de funciones de activación y taza de aprendizaje

La selección de los hiperparámetros se tiene que hacer de una manera empírica, a través de experimentos de prueba y error, por lo que es necesario realizar los ajustes requeridos para obtener el mejor resultado posible. Este trabajo se podría extender demasiado explicando el porqué del elegir ciertos hiperparámetros que se usan para nuestro problema, por lo que se limita a exponer los motivos de la selección de algunos de los más relevantes.

Después de realizar una serie de experimentos, se decidió hacer el análisis en conjunto de la tasa de aprendizaje y la función de activación. Para lo siguiente, se realiza la evaluación de los modelos que ya se han estado trabajando, manteniendo el optimizador que ya se seleccionó para cada uno de ellos. Ahora, se cambiará el número de épocas y la tasa de aprendizaje (*lr*, por las siglas en inglés *learning rate*) del optimizador. Para obtener un buen resultado, es conveniente tomar en cuenta la tasa de aprendizaje del optimizador, ya que de esta dependerá el correcto o rápido aprendizaje del modelo. Cuando la *lr* es grande, el aprendizaje del modelo es más rápido, es decir, el error disminuye con el menor número de épocas. Por el contrario, si el valor de *lr* es pequeño el aprendizaje es más lento. Podría pensarse que entre más grande el valor de la *lr* es mejor. Sin embargo, el que

tenga una tasa grande podría ser una desventaja, ya que al dar pasos demasiado grandes podemos dejar pasar un valor óptimo para la neurona. Por otro lado, el tener una tasa grande nos ayuda a escapar de mínimos locales. Ahora bien, el tener una tasa baja nos ayuda a hacer un ajuste más fino para cada neurona, pero el aprendizaje es más lento, por lo que llevaría un mayor número de épocas. Para esto, en la literatura se han implementado técnicas donde la *lr* varía suavemente siguiendo un esquema triangular o exponencial [48]. Otro método consiste en hacer decrecer la *lr*, realizando un reinicio al valor máximo cada cierto número de épocas retomando la última mejor red [49]. El uso de la gran variedad de técnicas que existen para variar la *lr* puede ser una tarea exhaustiva, y conllevaría un gran estudio. Debido a esto, sólo se utilizarán 4 diferentes técnicas: escalonada, decreciente, cíclica decreciente y decreciente sin mejora.

Aunado al cambio de tasa de aprendizaje, se analizarán dos funciones de activación que podemos utilizar para cada una de las neuronas. Sólo se analizan dos de ellas, con el fin de analizar el impacto que tiene el elegir una de otra. Para esto, se aplicará la función $ReLU$ que es una de las funciones que actualmente se usan para muchos modelos de redes neuronales, y la *tanh* que no es muy utilizada actualmente.

Para esta prueba se utilizarán estas dos funciones de activación, cambiando la función para cada uno de los 8 modelos, es decir, tendremos 16 modelos (8 con la función $ReLU$ y estos mismos, pero con la función *tanh*). Así mismo, cada una de las 4 técnicas de cambio de *lr* se aplicarán para cada uno de los modelos.

4.6. Entrenamiento del mejor modelo

Una vez aplicadas todas estas pruebas, se tendrá una visión clara del rendimiento de cada uno de los modelos, considerando la variación de los diferentes hiperparámetros con los que hemos trabajado. Con base en estos resultados, se seleccionará el modelo que logre el mejor equilibrio entre eficiencia computacional y precisión. Es decir, se elegirá aquel que requiera menos recursos computacionales y, al mismo tiempo, ofrezca estimaciones más precisas, con una pérdida mínima en las predicciones.

Caso de Estudio 2: Segmentación de óxidos

4.7. Etiquetado de las micrografías

Para aplicar técnicas de segmentación con redes neuronales convolucionales (CNN), es fundamental seguir una serie de pasos precisos, ya que de ello depende el buen desempeño de la red.

Uno de los primeros pasos clave es realizar una segmentación previa de las imágenes, con el fin de generar los conjuntos de datos de entrenamiento y prueba que se usarán en la red.

En este caso particular, se realizó una segmentación manual de las imágenes, ya que no se disponía de una segmentación previa para estos óxidos. Aunque existen numerosas redes neuronales ya entrenadas para realizar segmentación automática como las ya antes mencionadas en la Sección 2.11 o modelos más genéricas como SAM (Segment Anything Model) [50], ninguna de ellas ofrecía la precisión necesaria para etiquetar correctamente los píxeles correspondientes a cada tipo de óxido en nuestras micrografías. Debido a esta limitación, fue necesario segmentar manualmente cada imagen, utilizando el software *Labelme* [51], que facilita la selección de puntos de interés para las áreas correspondientes a los óxidos. Posteriormente, se realizó un procesamiento de imágenes para rellenar dichas áreas con las etiquetas correspondientes. Este proceso manual de etiquetado permitió obtener datos más precisos y confiables, lo que resultó fundamental para el entrenamiento adecuado de la red.

En la sección 3.9, se describió el proceso de identificación de los óxidos presentes en las micrografías generadas mediante irradiación fija. Con base en este análisis, se decidió realizar una clasificación de los óxidos en función de su fase y coloración, lo cual permitió distinguir entre los diferentes tipos de materiales presentes en las imágenes. Esta clasificación está detallada en la Tabla 3.7, donde se listan los óxidos identificados.

A partir de esta clasificación, se procedió a segmentar cada micrografía, asignando los píxeles correspondientes a cada tipo de óxido. El proceso se realizó de manera manual para garantizar una mayor precisión, y los resultados finales de esta segmentación se pueden observar en la Tabla 4.5, donde se muestran las áreas asignadas a cada tipo de óxido en las imágenes segmentadas.

4.8. Aumento de imágenes para la segmentación de los óxidos

Dado que el conjunto de imágenes disponibles era limitado, se optó por aplicar técnicas de aumento de datos (data augmentation) para ampliar el número de ejemplos utilizados en el entrenamiento del modelo. Este proceso permitió generar 30 nuevas imágenes por cada imagen original, aplicando diversas transformaciones como rotaciones, ampliaciones (zoom), y técnicas de llenado para mantener la coherencia visual y contextual en las imágenes generadas. Estas técnicas no sólo aumentaron la diversidad del conjunto de datos, sino que también ayudaron a mejorar la capacidad del modelo para generalizar, ya que introdujeron variaciones que podrían representar mejor las posibles condiciones que el modelo enfrentaría en la práctica.

Las transformaciones incluyeron rotaciones en diferentes ángulos, variaciones en la escala (acercamiento o alejamiento), traslaciones de píxeles, así como pequeños cambios en el brillo y contraste para simular condiciones de iluminación variadas. De este modo, se redujo el riesgo de sobreajuste, ya que el modelo fue entrenado con imágenes más variadas y realistas.

Un ejemplo de las rotaciones aplicadas a las imágenes originales se muestra en la Tabla 4.6, donde

Tabla 4.5: Matriz de las micrografías etiquetadas.

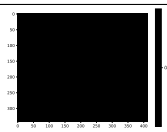
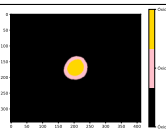
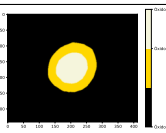
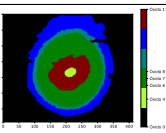
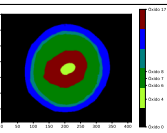
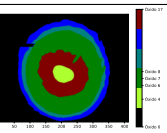
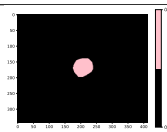
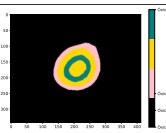
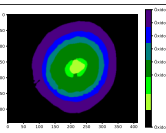
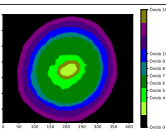
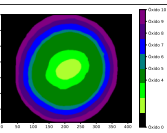
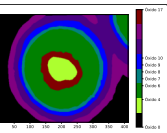
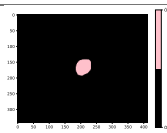
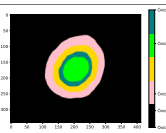
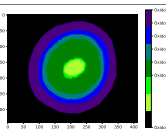
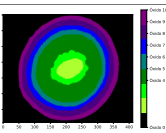
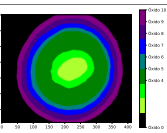
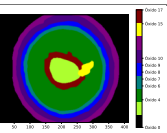
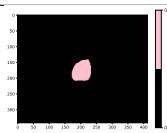
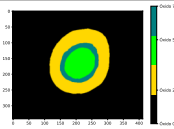
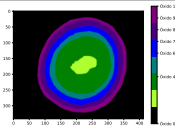
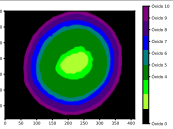
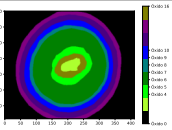
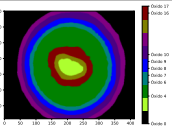
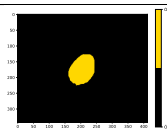
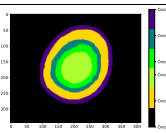
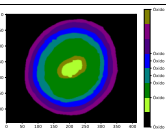
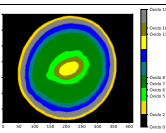
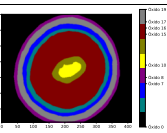
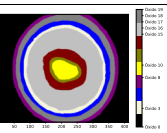
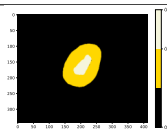
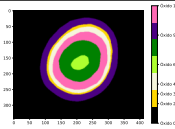
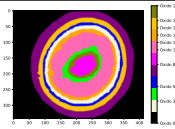
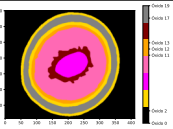
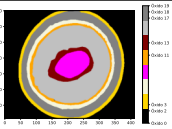
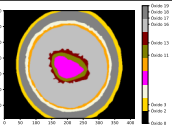
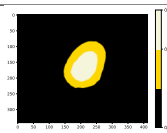
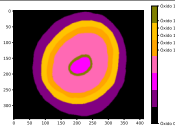
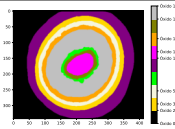
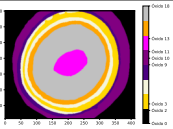
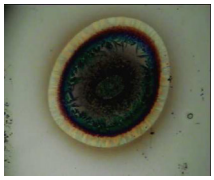
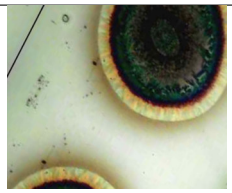
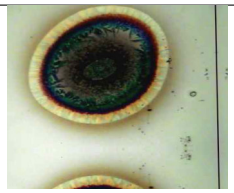
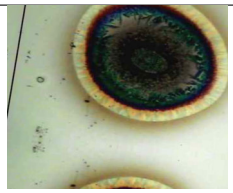
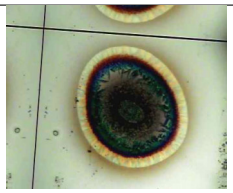
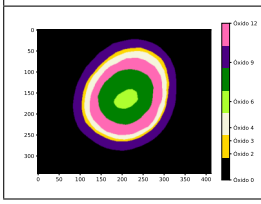
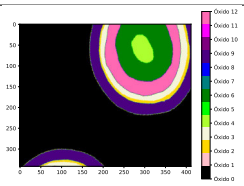
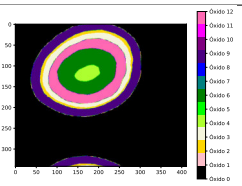
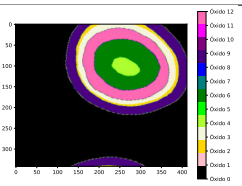
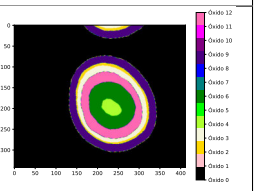
	1.4 mJ/cm^2	1.9 mJ/cm^2	2.5 mJ/cm^2	3.1 mJ/cm^2	3.6 mJ/cm^2	4.2 mJ/cm^2
2 s	 S_{11}	 S_{21}	 S_{31}	 S_{41}	 S_{51}	 S_{61}
10 s	 S_{12}	 S_{22}	 S_{32}	 S_{42}	 S_{52}	 S_{62}
20 s	 S_{13}	 S_{23}	 S_{33}	 S_{43}	 S_{53}	 S_{63}
30 s	 S_{14}	 S_{24}	 S_{34}	 S_{44}	 S_{54}	 S_{64}
60 s	 S_{15}	 S_{25}	 S_{35}	 S_{45}	 S_{55}	 S_{65}
180 s	 S_{16}	 S_{26}	 S_{36}	 S_{46}	 S_{56}	 S_{66}
360 s	 S_{17}	 S_{27}	 S_{37}	 S_{47}	S_{57}	S_{67}
600 s	S_{18}	S_{28}	S_{38}	S_{48}	S_{58}	S_{68}
1200 s	S_{19}	S_{29}	S_{39}	S_{49}	S_{59}	S_{69}

Tabla 4.6: Ejemplo de aumento de imágenes para la red neuronal convolucional. Se toma la micrografía S_{26}, y se le realizan rotaciones, tanto a la micrografía, como a la imagen etiquetada.

imagen original	Ejemplos de rotaciones aplicadas a la imagen			
				
				

se pueden observar las diferencias introducidas por estas transformaciones, tanto en las imágenes RGB, como en las máscaras correspondientes. De esta manera, se contribuye a un conjunto de datos más robusto para el entrenamiento.

4.9. NetLite

Antes de continuar con la descripción de los pasos siguientes, es importante aclarar que, para este trabajo, se ha propuesto una arquitectura de red que se espera proporcione resultados sólidos en las tareas de segmentación. La arquitectura propuesta está inspirada en U-Net, un modelo bien conocido en el campo de la segmentación de imágenes, aunque presenta algunas diferencias importantes que merecen ser destacadas. La Figura 4.3 se representa esta arquitectura, manteniendo la forma de interpretarla tal como se hace para U-Net.

Nuestra arquitectura sigue un enfoque similar, pero con algunas diferencias clave:

1. **Codificador-decodificador:** Al igual que las otras redes, esta red se basa en una etapa de codificación y otra de decodificación. La etapa de codificación se utiliza para reducir las dimensiones espaciales mientras aumenta el número de filtros, capturando características cada vez más abstractas de la imagen. La etapa de decodificación aplica operaciones de deconvolución para aumentar la resolución espacial.

2. **Bloques de convolución y max-pooling:** En lugar de las conexiones de salto utilizadas en U-Net, el modelo propuesto consiste en una secuencia de 5 bloques de convolución, cada uno con operaciones de normalización por lotes (BatchNormalization) para estabilizar y acelerar el entrenamiento, y capas de max-pooling en los primeros cuatro bloques para reducir la

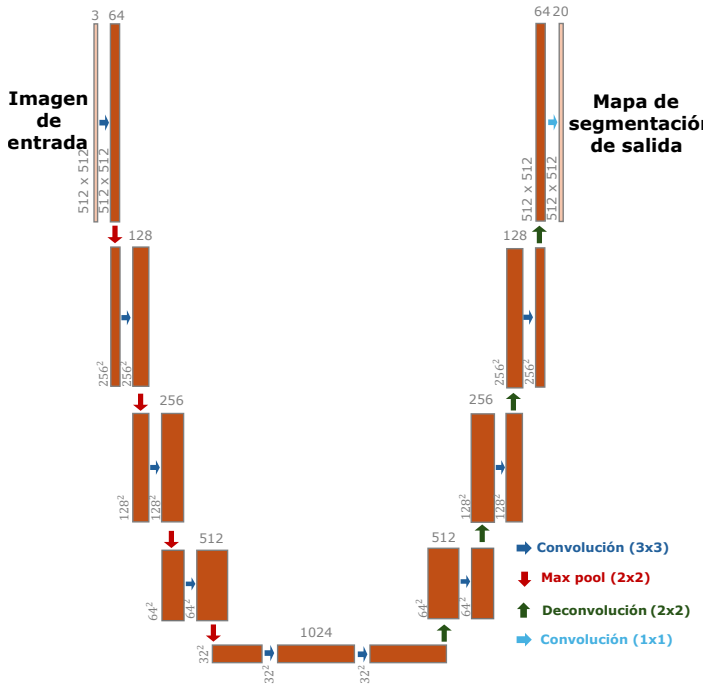


Figura 4.3: Ilustración de la arquitectura propuesta.

dimensionalidad. En el último bloque no se aplica MaxPooling, manteniendo la resolución para facilitar la reconstrucción.

3. **Bloques de deconvolución:** Posteriormente, se realiza la etapa de expansión utilizando 4 bloques de convolución traspuesta o deconvolución, cada uno con normalización por lotes y activación. Estas operaciones aumentan la resolución de las características extraídas para reconstruir la imagen segmentada, de manera similar a U-Net, pero sin las conexiones de salto entre capas correspondientes.

4. **Capa de salida:** La arquitectura finaliza con una capa de convolución con un filtro por clase de salida, y una función de activación softmax para producir un mapa de probabilidad para cada clase en la segmentación.

En comparación con U-Net, que incluye conexiones de salto directas entre la fase de contracción y expansión para mejorar la precisión de las segmentaciones, nuestra propuesta simplifica este aspecto. Así como la doble convolución que realiza U-Net, en esta red sólo se propone un bloque de convolción antes de realizar Maxpooling. Esto se hace con el fin de que la red tenga menos coste computacional, a diferencia de las otras arquitecturas que se trabajarán en este proyecto.

Se espera que este modelo mantenga un buen equilibrio entre precisión y eficiencia computacional, haciéndolo adecuado para nuestro conjunto de datos. Con esta arquitectura, buscamos obtener buenos resultados tanto en la predicción de los diámetros de los óxidos como en la segmentación precisa de los diferentes tipos de óxidos formados bajo la irradiación láser.

4.10. Pruebas de los espacios de color

Una vez aclarado el punto anterior, continuamos con la evaluación de nuestro conjunto de datos, realizando un cambio en el espacio de color. Se llevarán a cabo pruebas exhaustivas en diversos espacios de color con el fin de evaluar cuál de ellos proporciona el mejor rendimiento en la tarea de segmentación de imágenes. Para ello, se ha empleado la red propuesta, que ha sido seleccionada específicamente debido a su bajo costo computacional y capacidad para ofrecer buenos resultados sin un consumo excesivo de recursos.

Los espacios de color evaluados incluyen RGB, el cual es ampliamente utilizado en la representación de imágenes debido a su simplicidad y relación directa con la percepción visual; HSV, que separa la información de color en matices, saturación y valor, lo que puede facilitar la identificación de patrones en diferentes condiciones de iluminación; y CIELAB, un espacio diseñado para aproximar la visión humana de manera más precisa, al medir los colores de forma independiente de las condiciones de luz.

El objetivo de estas pruebas fue determinar cuál de estos espacios de color permite al modelo obtener los mejores resultados en términos de precisión en la segmentación y eficiencia computacional. Es importante señalar que, dado que el costo computacional es una limitante crucial en este proyecto, la elección del espacio de color adecuado debe equilibrar el rendimiento del modelo con el uso de recursos computacionales. Las pruebas realizadas proporcionarán información valiosa sobre el impacto de los diferentes espacios de color en el proceso de segmentación y ayudarán a seleccionar la mejor opción para el contexto específico de este trabajo.

4.11. Pruebas con distintas funciones de activación

Una vez seleccionado el espacio de color más adecuado para todo el conjunto de datos, se inicia la evaluación exhaustiva de cada una de las cuatro arquitecturas de red seleccionadas: U-Net, DeepLab, SegNet y NetLite. Este proceso implica no sólo analizar el rendimiento de cada modelo en términos de precisión y eficiencia, sino también comprender cómo cada estructura se adapta a las características específicas de los datos y los objetivos del estudio.

Durante esta primera fase de evaluación, se implementan las principales funciones de activación: *ReLU*, *tanh* y *sigmoide*. La elección de estas funciones es crucial, ya que influyen directamente en la capacidad de la red para aprender y generalizar a partir de los datos de entrenamiento. El proceso de evaluación se llevará a cabo en múltiples etapas, comenzando con la configuración de cada red y la preparación de los datos de entrada en el espacio de color seleccionado. Posteriormente, se entrenarán los modelos utilizando un conjunto de datos de entrenamiento y se validarán con un conjunto de datos separado para medir su desempeño. Las métricas de evaluación, como el accuracy, la pérdida y la intersección sobre la unión (IoU), se utilizarán para comparar la efectividad de cada modelo bajo las diferentes configuraciones de funciones de activación.

Al final de este proceso, se espera obtener información valiosa sobre las fortalezas y debilidades de cada arquitectura, lo que permitirá modificar, en caso de ser necesario, algunos hiperparámetros para mejorar le rendimiento de cada una de ellas de manera particular. Además, esta evaluación contribuirá a una mejor comprensión de cómo las diferentes funciones de activación pueden influir

en el rendimiento de las redes neuronales, proporcionando una base sólida para futuras investigaciones en este campo.

4.12. Pruebas de comparación de redes

Con base en los resultados obtenidos en la fase anterior, se busca realizar mejoras significativas en cada una de las redes evaluadas. Este proceso implicará la aplicación de las funciones de activación que hayan demostrado un mejor rendimiento en las pruebas iniciales, con el objetivo de maximizar la efectividad de cada modelo. La selección cuidadosa de estas funciones es fundamental, ya que pueden influir en la capacidad de la red para aprender patrones complejos en los datos de entrada.

En esta etapa de mejora, se llevará a cabo una comparación exhaustiva entre las diferentes arquitecturas de red. Para ello, se considerarán no sólo las funciones de activación, sino también otros hiperparámetros que impactan el desempeño de los modelos. Cada red será ajustada de manera particular, teniendo en cuenta sus características inherentes y su comportamiento durante la fase de entrenamiento. Esto incluye la optimización de parámetros como la tasa de aprendizaje, el tamaño del lote y el número de épocas, entre otros.

Además, se implementarán técnicas adicionales de regularización y aumento de datos para fortalecer el aprendizaje de cada red y mitigar posibles problemas de sobreajuste. La combinación de estas estrategias permitirá evaluar de manera más efectiva cómo cada arquitectura se adapta a los datos y cómo pueden ser mejoradas en función de sus resultados individuales.

Al final de este proceso, se espera no sólo optimizar el rendimiento de cada red, sino también obtener una comprensión más profunda de cómo las variaciones en las funciones de activación y los hiperparámetros afectan la capacidad de las redes para realizar tareas de segmentación. Esta fase no sólo contribuirá a mejorar los modelos, sino que también proporcionará información valiosa para futuros trabajos en el campo del aprendizaje profundo y su aplicación en la segmentación de imágenes.

Al final del proceso, se seleccionará una única red que cumpla de manera óptima con la tarea de segmentación, basada en el análisis de los resultados obtenidos en las pruebas. La elección se fundamentará en las métricas evaluadas, como el accuracy, la pérdida y el Intersection over Union (IoU). Estas métricas permitirán determinar la capacidad del modelo para clasificar correctamente los píxeles y ajustar bien a los datos, siendo especialmente relevante el IoU para medir la precisión en la segmentación.

4.13. Prueba de Validación cruzada

Se llevará a cabo una validación cruzada utilizando el modelo más óptimo seleccionado en las etapas anteriores. Esta técnica permitirá evaluar la robustez y generalización del modelo en diferentes subconjuntos del conjunto de datos, asegurando que los resultados obtenidos no sean un mero producto de la variabilidad del mismo. A través de la validación cruzada, se dividirá el conjunto de datos en múltiples particiones, donde el modelo será entrenado en un subconjunto y evaluado en otro. Este proceso se repetirá varias veces, lo que proporcionará métricas más confiables

sobre el rendimiento del modelo. El objetivo es confirmar que el modelo no sólo se ajusta bien a los datos de entrenamiento, sino que también mantiene un buen desempeño en datos no vistos, lo que es crucial para su aplicación práctica en la segmentación y predicción de diámetros de óxidos. Al final de este proceso, se obtendrán conclusiones más sólidas sobre la efectividad del modelo, lo que contribuirá a la validación de los resultados y a la confianza en la implementación de la inteligencia artificial en el campo de los materiales.

Capítulo 5

Resultados

Contenido

En este capítulo se presentan los resultados obtenidos a partir de las pruebas descritas en la metodología. Se realiza un análisis detallado de cada una de las pruebas ejecutadas, con el objetivo de evaluar el rendimiento de los modelos en función de los parámetros y configuraciones utilizados. Al igual que en el capítulo anterior, los resultados se han dividido en dos secciones principales: la primera corresponde a la etapa en la que se utilizaron redes neuronales profundas para la predicción de diámetros, y la segunda se centra en la aplicación de redes neuronales convolucionales para

la segmentación de los diferentes tipos de óxidos generados.

En la etapa de redes neuronales profundas, se exploran diferentes configuraciones de hiperparámetros y optimizadores, evaluando su impacto en la precisión de las predicciones y el tiempo de convergencia del modelo. Cada uno de los modelos ha sido sometido a un conjunto exhaustivo de pruebas con el fin de identificar la configuración óptima que permita realizar predicciones más precisas de los diámetros de los óxidos, mientras se mantienen los recursos computacionales en un nivel razonable.

Por otro lado, en la etapa de redes convolucionales, se presentan los resultados de la segmentación de los diferentes tipos de óxidos, analizando el comportamiento del modelo ante diferentes ajustes de parámetros y técnicas. Se realiza una comparación entre los distintos enfoques implementados para determinar cuáles ofrecen los mejores resultados en términos de precisión y eficiencia.

De esta manera, este capítulo proporciona una visión integral del rendimiento de los modelos implementados, permitiendo obtener conclusiones claras sobre el comportamiento de las redes neuronales profundas y convolucionales en el contexto de este estudio.

Caso de estudio 1: Predicción de diámetros

5.1. Selección del optimizador

Para realizar los entrenamientos, se deben de establecer hiperparámetros adicionales, por ejemplo la función de pérdida, que se deicidió establecer el *MSE*, y como métricas se utilizaron tanto *MSE* como *MAE*. Como se mencionó anteriormente, los valores de los diámetros se separaron en prueba (las celdas encerradas en rojo) y entrenamiento (todas las demás celdas). Así mismo, los datos de entrenamiento se separaron aleatoriamente en validación (10 %) y el restante para el entrenamiento. Otro dato importante es el número de épocas con el que se entrenó al modelo, ya que de este dependerá el tiempo en el que se tarde el modelo en entrenar. Para este caso, debido que sólo queremos realizar un análisis general de los optimizadores, se optó por hacer 1000 épocas, que en realidad es un número un poco alto, pero no es un gran problema, ya que el entrenamiento, al no ser un gran número de datos, no toma bastante tiempo realizar una época (aproximadamente 20 ms, dependiendo el modelo). Una vez aclarado esto, podemos realizar un análisis general de cada optimizador. La Tabla 5.1 nos muestra los MSE, MAE y R2S de los 8 modelos de la Tabla 4.4, los cuales se obtuvieron al evaluar los datos de prueba en el modelo que se obtuvo en la última época del entrenamiento. A cada uno de estos modelos se les aplicaron 6 optimizadores distintos, los cuales son posibles utilizar para este tipo de tareas, debido que existen otros, los cuales no

fueron posibles aplicar utilizando estas métricas y función de pérdida.

Tabla 5.1: Tabla de comparación del error cuadrático medio y error absoluto medio al utilizar algunos optimizadores.

	Modelo 1			Modelo 2			Modelo 3			Modelo 4		
	MSE	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**
Adadelta	8539.46	78.8	-13.76	1320.65	22.04	-1.28	6199.55	63.94	-9.71	7444.72	70.92	-11.86
Adagrad	2766.14	44.95	-3.78	246.78	12.97	0.57	260.63	13.51	0.55	250.98	12.73	0.57
Adam	130.73	8.31	0.77	81.1	5.68	0.86	294.19	11.58	0.49	113.43	9.55	0.8
AdamW	138.22	8.7	0.76	69.5	5.67	0.88	80.5	7.5	0.86	150.07	10.15	0.74
Nadam	273.73	13.73	0.53	58.47	4.25	0.9	61.31	5.37	0.89	90.67	7.34	0.84
RMSprop	154.61	11.12	0.73	277.5	9.9	0.52	130.41	8.26	0.77	257.5	11.98	0.55
	Modelo 5			**Modelo 6**			**Modelo 7**			**Modelo 8**		
	MSE	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**
Adadelta	8201.76	77	-13.17	256.17	12.29	0.56	273.11	12.85	0.53	287.26	12.95	0.5
Adagrad	228.73	11.97	0.6	284.77	14.18	0.51	223.46	12.74	0.61	207.88	11.95	0.64
Adam	59.39	6.07	0.9	171.76	8.54	0.7	83.78	6.35	0.	120.29	9.22	0.79
AdamW	87.4	6.82	0.85	147.01	8.45	0.75	74.65	4.7	0.87	113.56	7.67	0.8
Nadam	65.37	6.14	0.89	175.22	9.79	0.7	68.02	6.11	0.88	138.51	8.9	0.76
RMSprop	1119.056	20.23	-0.93	166.21	10.33	0.71	183.4	9.83	0.68	145.91	9.98	0.75

Si analizamos la Tabla 5.1, podemos ver que en los optimizadores, donde, generalmente los modelos obtienen errores más bajos son los optimizadores variantes del Adam, como es el AdamW y Nadam. Además, podemos ver que, en general, en Adadelta y el Adagrad es donde los errores son más altos. Para la siguiente prueba, el optimizador que será utilizado en cada modelo será en el que se obtuvo el menor error.

Cabe mencionar que existen otros hiperparámetros que hay que considerar para nuestro modelo, como lo es la función de activación de las neuronas, es también importante seleccionar la función que mejor se adapte para nuestro problema, en este caso, para las pruebas ya realizadas se utilizó la función $ReLU$. Se decidió utilizar esta, ya que al realizar diversas pruebas con otras funciones fue con la que se obtuvieron mejores resultados. Las pruebas realizados con las funciones de activación no se reporta, debido que podemos hacer muchas combinaciones poniendo diferentes funciones en cada capa. Sin embargo, se optó por que en todas las capas fuese la misma función.

5.2. Selección de la tasa de aprendizaje y función de activación

Hasta este punto se han encontrado algunos de los hiperparámetros con los que se han obtenido mejores resultados. Sin embargo, aún quedan algunos otros que hay que tomar en consideración, los cuales son la función de activación y la tasa de aprendizaje, los cuales se analizarán en esta sección.

5.2.1. *ReLU* como función de activación

Como se mencionó anteriormente, se utiliza la función *ReLU* como función de activación para cada una de las neuronas de cada uno de los modelos (Tabla 4.4). Aunado a esto, se utilizan los 4 técnicas de cambio de tasa mencionadas anteriormente para cada uno de estos modelos. Una aproximación al cambo de tasa lo podemos ver en la Figura 5.1. A continuación, se muestran los resultados del uso de cada una.

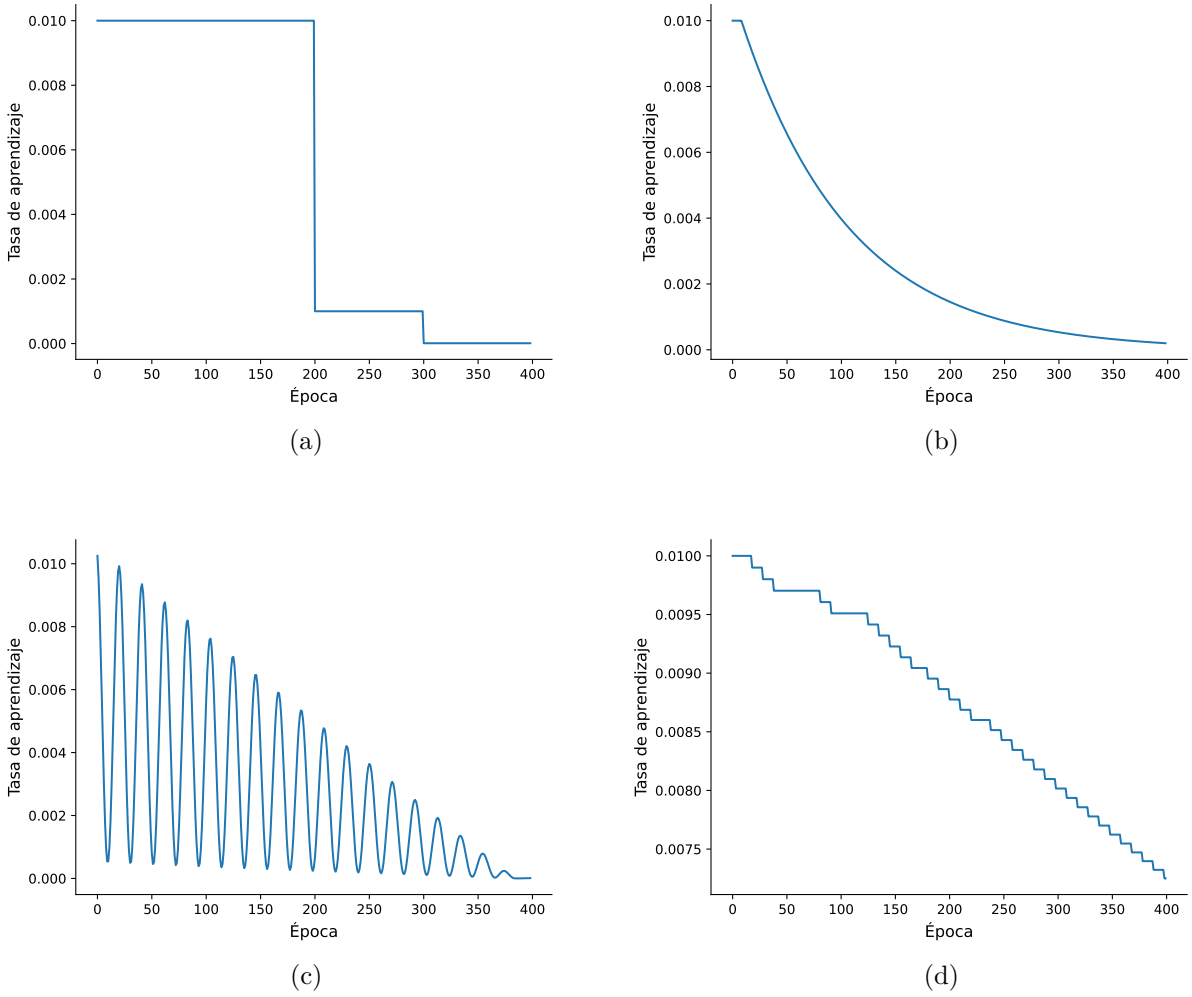


Figura 5.1: Cambios de tasa aplicados a cada uno de los 8 modelos, utilizando la función *ReLU*. (a) Cambio de tasa escalonada, aplicando cambios de .01, .001 y .00001 a 200, 100 y 100 épocas, respectivamente. (b) Tasa decreciente a un factor de .99 (a partir de la décima época) por época (400 épocas). (c) Tasa cíclica decreciente, usando una función sinusoidal a 400 épocas. (d) Ejemplo de la tasa decreciente sin mejora (cada reducción de tasa dependerá del comportamiento del modelo durante el entrenamiento).

Tasa escalonada

En esta primera forma de variar la *lr*, se optó por hacer 3 diferentes entrenamientos, disminu-

yendo en cada uno de ellos la *lr*. Primeramente, se hace un entrenamiento de 200 épocas, con $lr = 0.01$, cuyo valor se decidió establecer, debido que es el valor que *keras* le da por defecto a algunos de los optimizadores . En el segundo entrenamiento se disminuye la tasa con una razón de 10 con respecto a la anterior, es decir, $lr = 0.001$ a 100 épocas. Finalmente, se vuelve a disminuir la tasa a $lr = 0.00001$ a 100 épocas, tal cual se muestra en la Figura 5.1a. El hacer este procedimiento tiene la finalidad que en cada entrenamiento se le dé al modelo un ajuste más fino, conforme aumentan las épocas, utilizando una tasa decreciente. Cabe destacar que, el número de épocas que se utiliza en cada entrenamiento fue seleccionado tras varias pruebas, de tal manera fuesen suficientes parea alcanzar un resultado óptimo. En la Tabla 5.2 se muestra el MSE, MAE y R2S al realizar los 3 entrenamiento mencionados.

Tabla 5.2: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación, utilizando la tasa escalonada (se muestran los 3 entrenamientos).

Modelos	**1erentrenamiento** lr **= 0.01, épocas = 200**			**2° entrenamiento** lr **= 0.001, épocas = 100**			**3er entrenamiento** lr **= 0.00001, épocas = 100**		
	MSE	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**
Modelo 1	94.89	6.95	0.828	95.73	6.89	0.827	95.95	6.88	0.827
Modelo 2	20.5	3.32	0.963	15.28	2.63	0.972	13.94	2.63	0.975
Modelo 3	21.67	3.95	0.961	3.62	1.31	0.993	3.48	1.3	0.994
Modelo 4	11.35	2.69	0.979	8.96	2.1	0.984	9.07	2.06	0.984
Modelo 5	7.84	1.93	0.986	10.56	2.72	0.981	4.42	1.34	0.992
Modelo 6	20.46	3.85	0.963	11.75	2.49	0.979	10.02	2.15	0.982
Modelo 7	70.28	7.38	0.873	24.22	2.98	0.956	30.35	3.1	0.945
Modelo 8	30.65	4.66	0.945	2.51	1.15	0.995	2.03	0.93	0.996

Analizando con detenimiento la Tabla 5.2, se pude ver que no en todos los casos los errores disminuyen al hacer el segundo y tercer entrenamiento, si no que por el contrario, los errores aumentan; como lo es el caso del modelo 1, 3 y 8. Aun así, en la mayoría de los modelos siempre disminuye del primer al segundo entrenamiento, tanto el MSE como el MAE. Sin embargo, del segundo al tercer entrenamiento no disminuye el error en todos los caso, sino que aumenta. Debido a esto, no se puede establecer la regla que el tercer entrenamiento disminuye los errores con respecto al segundo. Sin embargo, se decidió dejar los tres entrenamientos, ya que al hacerlos, en algunos se obtuvo el error mínimo del modelo. Aun así, podemos ver que el mejor modelo se obtuvo al realizar el segundo entrenamiento del modelo 7, cuya celda está resaltada de verde (esto para el mejor resultado de cada técnica en cada uno de las funciones de activación) .

Esta forma de variar la *lr* no se encontró en la literatura tal cual como se implementa en este trabajo, ya que puede resultar un tanto ineficiente el implementarla de esta forma. Sin embargo, como se mostró en la Tabla 4.4, se obtuvieron excelentes resultados utilizando esta técnica.

Tasa decreciente

Como siguiente técnica, se utiliza la tasa decreciente. Esta forma de variar *lr* consiste en reducirla por cada época durante el entrenamiento, multiplicándola por un factor (menor a 1), en este

caso se utiliza un factor de 0.99. Esto quiere decir que por cada época, se reduce el 1 % de la época anterior, pero sólo a partir de la décima época. El ejemplo gráfico de esta cambio lo podemos ver en la Figura 5.1b donde se realiza este cambio a 400 épocas. El número de épocas se estableció de modo que para todos las técnicas de cambio de *lr* fuese la misma cantidad, por lo menos utilizando la función $ReLU$, ya que para el uso de *tanh* se utiliza una cantidad mayor de épocas (de lo cual se hablará más adelante).

Para el entrenamiento se utilizaron los mismos hiperparámetros que en la técnica anterior, a excepción de *lr*, que como ya se mencionó se utiliza una tasa decreciente, con un factor de .99 a partir de la décima época. Las primeras 10 épocas se mantiene con la tasa inicial establecida, que en este caso se usa una $lr = .01$. En la Tabla 5.3 se muestra el resultado de los errores y R2S, los cuales se obtuvieron al evaluar el modelo que se obtuvo finalizar las 400 épocas con los datos que se eligieron para prueba.

Tabla 5.3: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa decreciente.

Modelos	MSE	MAE	R2S
Modelo 1	175.38	10.84	0.683
Modelo 2	17.62	2.97	0.968
Modelo 3	18.93	2.92	0.966
Modelo 4	66.05	5.82	0.881
Modelo 5	20.37	3.2	0.963
Modelo 6	5.66	1.52	0.99
Modelo 7	2.94	1.15	0.995
Modelo 8	2.2	1.03	0.996

Dados los resultados de la Tabla 5.3, se puede hacer un análisis del comportamiento de los modelos, según el número de capas y el número de parámetros que van de la mano en cuanto a cantidad se refiere. Se puede ver claramente que el error disminuye conforme el número de capas aumenta, como se ve en los primeros modelos (1, 2 y 4) es donde se obtiene un mayor error. Vemos que el número de capas influye de una manera clara en el error, ya que, como vemos en la Tabla 4.4, el número de capas en el modelo 3 es mayor al que hay en el modelo 4. Usando este cambio de *lr*, se puede apreciar que mientras el número de capas y parámetros a entrenar aumente el error será menor. Esto es evidente, ya que el modelo con el menor error es el que contiene un mayor número de capas y parámetros, que corresponde al modelo 8.

Tasa cíclica decreciente

Otra técnica que se utiliza para este trabajo es la tasa cíclica, que se ha introducido en la literatura anteriormente [48], y usado en algunos otros trabajos en la literatura [52, 53]. Para este caso se trabaja usando una función sinusoidal, que conforme transcurren las épocas el pico máximo disminuye con respecto a las épocas anteriores, justo como se muestra en la Figura 5.1c. Usando este *lr* se realiza el entrenamiento de la red, utilizando los hiperparámetros seleccionados para los entrenamientos anteriores. Una vez terminado el entrenamiento se procede a evaluar los datos de

prueba, cuyos resultados ese muestran en la Tabla 5.4.

Tabla 5.4: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa cíclica decreciente.

Modelos	MSE	MAE	R2S
Modelo 1	172.86	10.78	0.687
Modelo 2	38.06	4.58	0.931
Modelo 3	12.67	2.29	0.977
Modelo 4	16.47	2.76	0.97
Modelo 5	17.54	3.04	0.968
Modelo 6	2.49	0.94	0.995
Modelo 7	2.65	1.03	0.995
Modelo 8	1.94	0.91	0.996

Analizando los datos de la Tabla, podemos ver que los resultados obtenidos son un tanto aleatorios, a diferencia de los resultados obtenidos en el entrenamiento anterior. Sin embargo, la tendencia a obtener mejor resultado con estructuras con mayor número de capa se mantiene, dado que el mejor resultado se obtuvo en el modelo con mayor número de capas y parámetros (modelo 8).

Tasa decreciente sin mejora

El siguiente y último cambio de *lr* consiste en la reducción de la misma, sólo en caso que el error del modelo no mejore (disminuya) durante cierto número de épocas consecutivas. De ser el caso que no mejore, la tasa se reducirá multiplicándose por un factor (menor a 1 y mayor a 0) que se puede establecer, dependiendo del porcentaje que se requiera reducir cuando no haya mejora. Para este caso, se utiliza una tolerancia de 10 épocas con un factor de reducción del .99, es decir, la tasa se reducirá en sólo el 1 % en comparación de la anterior.

Este tipo de cambio de *lr* es muy parecido al que se aplicó anteriormente (tasa decreciente), a diferencia que la tasa disminuirá sólo si no hay mejora en le modelo, a diferencia de la otra, que en cada época que transcurra se reducirá sin importar la mejora. La Figura 5.1d es un ejemplo del como se comporta la tasa en esta técnica, específicamente de uno de los entrenamientos del modelo 8. En la Tabla 5.5 se muestra los resultados del entrenamiento de cada modelo por 400 épocas.

Analizando los errores mostrados en la Tabla, vemos que la tendencia de los entrenamientos anteriores de obtener mejores resultados con estructuras más grandes sigue. Estos resultados evidencian la clara reducción del error mientras que el número de parámetros y el número de capas ocultas aumenta. De igual manera, vemos que el error mayor para este cambio de *lr* incluso es muy bajo, a comparación de las tablas de error anteriores. Entonces, podemos concluir que esta técnica resultó ser la más óptima para nuestro problema, utilizando $ReLU$ como función de activación. Así mismo, usando esta técnica se obtiene el mejor resultado de todos los anteriores, donde el mejor modelo es el 8, al igual que en la mayoría de los anteriores entrenamientos.

Tabla 5.5: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa decreciente sin mejora.

Modelos	MSE	MAE	R2S
Modelo 1	16.83	3.04	0.97
Modelo 2	11.6	2.93	0.979
Modelo 3	4.87	1.47	0.991
Modelo 4	8.35	2.16	0.985
Modelo 5	17.16	3.11	0.969
Modelo 6	12.79	2.88	0.977
Modelo 7	8.6	2.51	0.984
Modelo 8	9.04	2.58	0.984

5.2.2. *Tanh* como función de activación

En esta subsección, se realizará el mismo procedimiento que se hizo en la subsección anterior, ahora utilizando la función *tanh* como función de activación para cada una de las neuronas de cada uno de los modelos (Tabla 4.4). Así mismo, se utilizan las 4 técnicas de cambio de tasa mencionadas. Sin embargo, cambiamos el número de épocas en las que se entrenan los modelos, ya que al hacerse varias pruebas, resultó que para obtener un resultado óptimo es necesario aumentar el número de épocas, a diferencia de la función anterior. El comportamiento de las tasas en este caso es parecido al que se utilizó para el uso de *ReLU*, el único cambio que habrá será en el número de épocas por el que cambia *lr*, como lo muestra la Figura 5.2. A continuación, se muestran los resultados del uso de cada una.

Tasa escalonada

A diferencia de la función de activación que se estudió con anterioridad (*ReLU*), que se decidió trabajar con sólo 400 épocas en total (divididas en 200, 100 y 100 en cada uno de los 3 entrenamientos), se aumentó el número de épocas para el primer y segundo entrenamiento, de tal forma que el primer entrenamiento consta de 500 épocas, con $lr = 0.01$. En el segundo entrenamiento se estableció una $lr = 0.001$ a 300 épocas. Finalmente, el último entrenamiento se mantiene con una tasa a $lr = 0.00001$, a tan sólo 100 épocas, tal cual se muestra en la Figura 5.2a. En la Tabla 5.6 se muestra el MSE, MAE y R2S al realizar los 3 entrenamiento mencionados usando la función *tanh*.

Si analizamos con detenimiento en cada uno de los resultados de los modelos, se aprecia con claridad el aumento en el error mientras el número de capas y parámetros aumenta. Este comportamiento es justamente lo contrario al que se ve utilizando *ReLU*, donde el error disminuye si la densidad de la estructura aumenta. Así mismo, en todos los modelos el error del segundo entrenamiento disminuye con respecto al primero. En la mayoría de los casos, el error del entrenamiento del tercer entrenamiento es menor al del segundo, con excepción del modelo 5. Para este caso, el modelo con el que se obtuvo un error menor es el primero, tras haber realizado el tercer entrenamiento.

Tasa decreciente

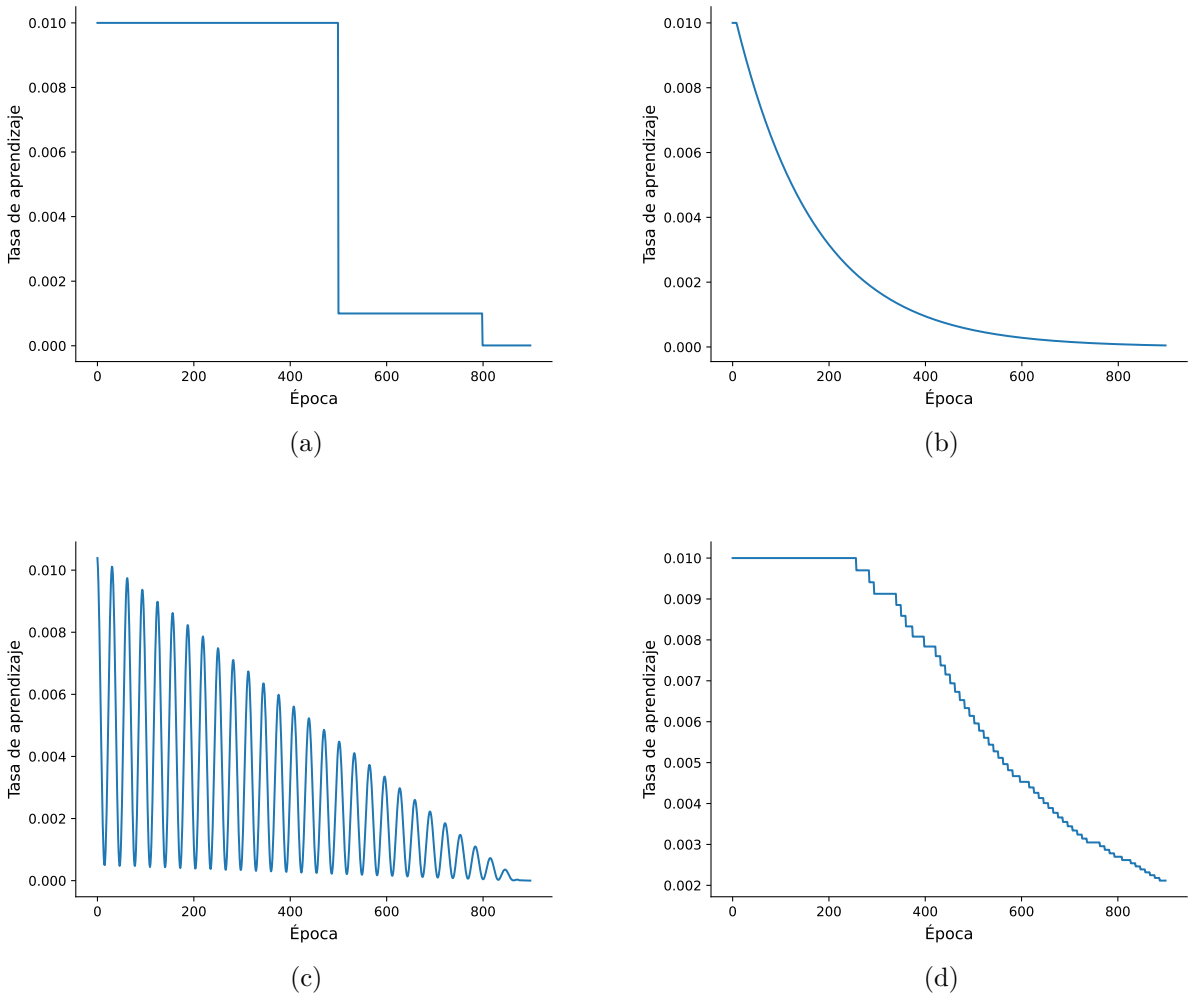


Figura 5.2: Cambios de tasa aplicados a cada uno de los 8 modelos, utilizando la función *tanh*. (a) Cambio de tasa escalonada, aplicando cambios de .01, .001 y .00001 a 500, 300 y 100 épocas, respectivamente. (b) Tasa decreciente a un factor de .99 (a partir de la décima época) por época. (c) Tasa cíclica decreciente, usando una función sinusoidal a 900 épocas. (d) Ejemplo de la tasa decreciente sin mejora (cada reducción de tasa dependerá del comportamiento del modelo durante el entrenamiento).

Ahora realizamos el entrenamiento con la tasa decreciente (ver Figura 5.2b), respetando los hiperparámetros ya seleccionados con anterioridad.

Analizando los resultados mostrandos en la Tabla 5.7, vemos que en la mayoría de los modelos, los resultados de los errores son muy grandes, manteniendo el mismo rango de error, a excepción del primer modelo, donde se encuentra el entrenamiento con el error menor al utilizar *lr* decreciente.

Tasa cíclica decreciente

Ahora, se procede a realizar el entrenamiento con *lr* cíclica, que se muestra en la Figura 5.2c.

Tabla 5.6: Tabla de comparación del error cuadrático medio, error absoluto medio durante y coeficiente de determinación, utilizando la tasa escalonada (se muestran los 3 entrenamientos).

Modelos	**1er entrenamiento** lr **= 0.01, épocas = 500**			**2o entrenamiento** lr **= 0.001, épocas = 300**			**3er entrenamiento** lr **= 0.00001, épocas = 100**		
	MSE	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**	**MSE**	**MAE**	**R2S**
Modelo 1	4.82	1.62	0.991	1.92	1.01	0.997	1.79	0.96	0.997
Modelo 2	561.43	19.08	-0.015	13.04	2.7	0.976	12.46	2.67	0.977
Modelo 3	565.7	19.36	-0.023	21.13	3.51	0.962	20	3.47	0.964
Modelo 4	579.79	20.03	-0.048	70.88	5.93	0.872	62.06	6.22	0.888
Modelo 5	575.52	19.85	-0.041	26.25	3.85	0.953	24.1	3.72	0.956
Modelo 6	564.95	19.31	-0.021	72.78	5.94	0.868	66.79	6.17	0.879
Modelo 7	570.17	19.6	-0.031	113.42	8.79	0.795	101.7	7.63	0.816
Modelo 8	575.94	19.87	-0.041	166.4	8.68	0.699	91.07	6.93	0.835

Tabla 5.7: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa decreciente.

Modelos	MSE	MAE	R2S
Modelo 1	5.73	1.75	0.99
Modelo 2	563.27	19.21	-0.018
Modelo 3	54.24	5.6	0.902
Modelo 4	25.95	4.12	0.953
Modelo 5	565.16	19.32	-0.022
Modelo 6	567.22	19.44	-0.025
Modelo 7	566.31	19.39	-0.024
Modelo 8	565.11	19.32	-0.022

Tabla 5.8: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa cíclica decreciente.

Modelos	MSE	MAE	R2S
Modelo 1	2.8	1.13	0.995
Modelo 2	56.51	5.38	0.898
Modelo 3	569.18	19.55	-0.029
Modelo 4	4.21	1.43	0.992
Modelo 5	7.74	2.02	0.986
Modelo 6	7.44	2.01	0.987
Modelo 7	567.07	19.44	-0.025
Modelo 8	567.56	19.46	-0.026

Al comparar la Tabla 5.8 con la 5.7, vemos que mantienen cierta relación, donde los errores con modelos demasiado grandes son muy grandes, mientras que el modelo con menor numero de parámetros son más pequeños sus errores.

Tasa decreciente sin mejora

Al igual que los entrenamientos anteriores, al utilizar la técnica de *lr* sin mejora, se obtiene el mejor resultado en estructuras pequeñas. Para este, como en los caso anteriores, el modelo con el mejor resultado es el primer modelo, mientras que los demás errores son muy grandes.

Tabla 5.9: Tabla de comparación del error cuadrático medio, error absoluto medio y coeficiente de determinación utilizando la tasa decreciente sin mejora.

Modelos	MSE	MAE	R2S
Modelo 1	5.73	1.75	0.99
Modelo 2	563.27	19.21	-0.018
Modelo 3	54.24	5.6	0.902
Modelo 4	25.95	4.12	0.953
Modelo 5	565.16	19.32	-0.022
Modelo 6	567.22	19.44	-0.025
Modelo 7	566.31	19.39	-0.024
Modelo 8	565.11	19.32	-0.022

5.2.3. Análisis de los resultados

Al analizar los resultados obtenidos utilizando la función de activación *ReLU*, se observa que, independientemente de la tasa de aprendizaje empleada, los modelos con un mayor número de capas y parámetros (modelos 7 y 8) presentan errores de menor magnitud. Los valores de MSE mínimos para estos modelos oscilan entre 4.87 y 1.94, en contraste con los modelos que tienen un menor número de parámetros, cuyos MSE mínimos superan los 10. Esto sugiere que una mayor complejidad del modelo contribuye a mejorar el rendimiento en términos de precisión. Por el contrario, al utilizar *tanh* los modelos con el menor error son aquéllos que tienen el menor número de capas (modelo 1 y 2),con MSE minimos que oscilan entre 5.73 y 1.78 .

Al analizar las distintas técnicas de ajuste de la tasa de aprendizaje, observamos que al utilizar la función de activación *ReLU*, el mejor resultado se obtiene con la tasa cíclica decreciente, aplicada en el modelo 8. En este caso, los valores de las métricas alcanzadas fueron un MSE de 1.94, MAE de 0.91 y R2S de 0.996. Por otro lado, al emplear la función de activación *tanh*, el mejor desempeño se consiguió con la tasa de aprendizaje escalonada, aplicada en el tercer entrenamiento del modelo 1, obteniendo un MSE de 1.79, MAE de 0.96 y R2s de 0.997.

Es claro que siempre se busca utilizar la menor cantidad de recursos para obtener los mejores resultados. En el caso de estudio de este problema, se pudo obtener excelentes resultados utilizando pocas neuronas, usando como función de activación *tanh*, al realizar 3 entrenamientos con *lr* 0.01, 0.001 y 0.00001 para 500, 300 y 100 épocas respectivamente.

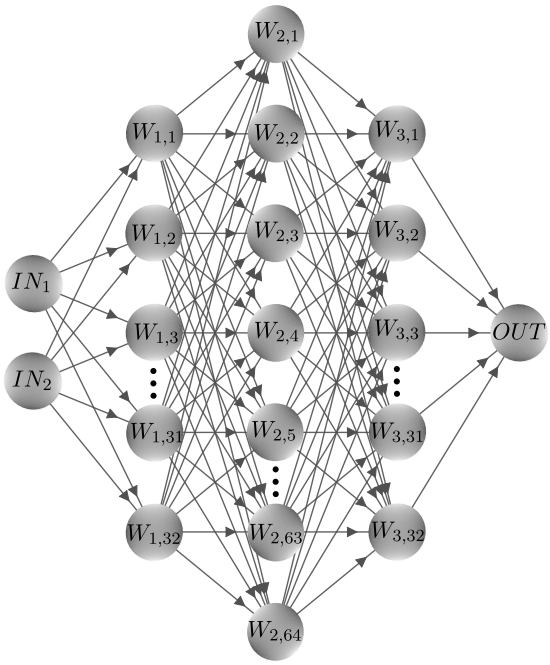


Figura 5.3: Representación gráfica de la estructura del modelo 1.

5.3. Entrenamiento del mejor modelo

Anteriormente, se realizó el entrenamiento de todos lo modelos, utilizando cada una de las técnicas para el cambio de tasa que se plantearon en este trabajo. De estos entrenamientos se obtuvo el modelo más óptimo, el cual resultó ser el modelo 1 con el uso de *tanh*, cuya estructura se muestra en la Figura 5.3. El valor de los errores al evaluar los datos de prueba se muestran en la Tabla 5.6, cuyo mejor resultado se encuentra resaltada la celda de verde. Además de analizar el error del modelo al terminar cada uno de los entrenamientos, es interesante observar la variación del valor del error conforme avanzan las épocas durante cada uno de los entrenamientos. En la Figura 5.4 podemos ver gráficamente el comportamiento del MAE durante los procesos de entrenamiento y validación del modelo con mejor desempeño (modelo 1), donde la línea anaranjada representa el MAE que se obtienen utilizando sólo los datos de entrenamiento, y la azul, el MAE al evaluar los valores que se separaron para realizar la validación del modelo obtenido en cada una de las épocas.

Analizando el primer entrenamiento (Figura 5.4a) se puede ver que el error de entrenamiento disminuye considerablemente hasta las 350 épocas, aproximadamente. A partir de ahí, el error sigue disminuyendo muy poco conforme las épocas avanzan, pero aún así sigue disminuyendo. En el segundo entrenamiento (Figura 5.4b) se aprecia de mejor forma que, tanto el error de validación como el de entrenamiento disminuyen conforme las épocas avanzan; eso quiere decir que el haber bajado la *lr* del optimizador ayudó a sacar el modelo de mínimos locales. En el tercer entrenamiento (Figura 5.4c) se observa que sólo en las primeras 20 épocas hay una disminución del error seguido de eso el error es constante, por lo que se aprecia que ya no hay mejora alguna en el modelo a partir de ahí. A pesar de que ya no hay mejora, el modelo no comienza a tener sobreajuste, esto puede ser debido a la baja tasa que se maneja en el ultimo entrenamiento.

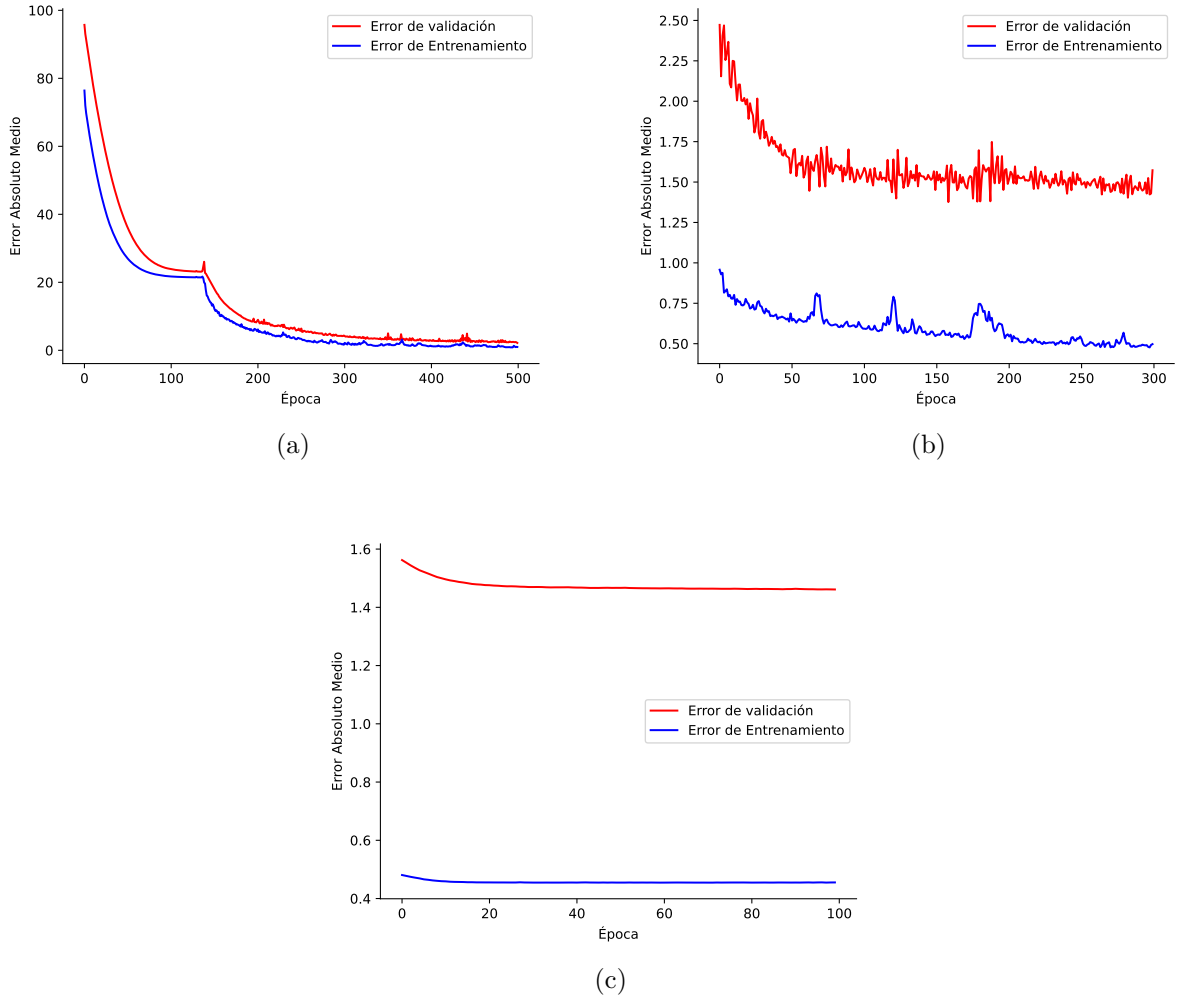


Figura 5.4: Gráficas del MAE del modelo 1 (modelos seleccionado) que se obtuvo durante el entrenamiento y con los datos de validación para cada época durante el (a) primer, (b) segundo, y (c) tercer entrenamiento.

5.4. Evaluación de los datos de prueba

Anteriormente, se obtuvo el MAE y MSE al evaluar los datos de prueba en el último modelo obtenido del tercer entrenamiento. Estas métricas miden la distancia de separación que hay entre el valor estimado con el real de cada uno de los datos, a parir de esas distancias se obtiene un promedio, y ese es el error medio. La Figura 5.5 muestra una comparación del diámetro real de los datos de prueba con los que estima el modelo, ayudando a tener una mejor idea de cuán distinto es el valor real de aquél que se obtuvo por estimación del modelo.

Se observa que, en general, las aproximaciones son precisas, e incluso exactamente las mismas, como vemos en los datos 3 - 10, donde los puntos prácticamente se empalman. De igual forma, la distancia entre los valores reales y estimados de los puntos 1 y 11 no están tan alejados, tomando en cuenta que las distancias están en le orden de micrómetros. La diferencia más grande la podemos ver en el dato 1, donde la distancia es de 3.3 μm. Finalmente, se puede concluir que, en general,

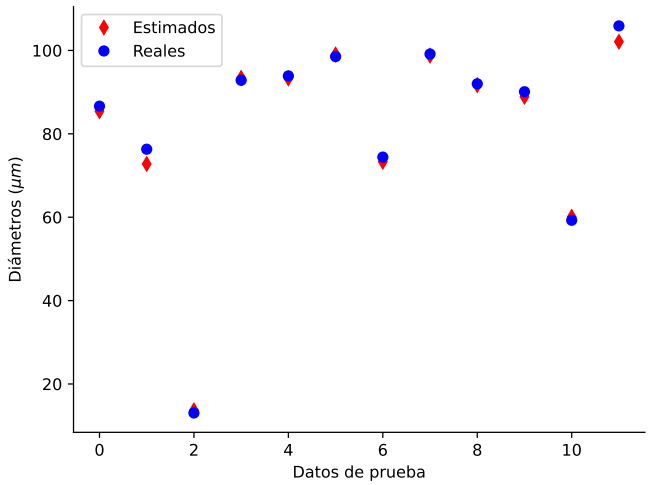


Figura 5.5: Comparación del diámetro estimado con el modelo 8 con el valor real utilizando los datos de prueba (que no había visto el modelo antes).

el modelo estima con precisión el valor de los diámetros, obteniendo el mínimo error en cada una de sus predicciones.

Caso de estudio 2: Segmentación de óxidos

5.5. Selección del espacio de color

El espacio de color puede influir significativamente en el rendimiento de los modelos de segmentación, es por eso que se realizaron pruebas para analizar el rendimiento de la red neuronal que se desarrolló para este proyecto (ver sección 4.9), utilizando los espacios de color RGB, HSV y CIELAB. En la Figura 5.6 podemos ver la comparación de las curvas de aprendizaje durante el entrenamiento del modelo por 50 épocas. Así mismo, en la Tabla 5.10 podemos ver el resultado del modelo entrenado.

Analizando la Tabla 5.10, podemos ver que el modelo en RGB muestra una precisión bastante consistente entre entrenamiento, validación y prueba. Aunque la pérdida de validación es mayor que la de entrenamiento, lo que podría indicar algo de sobreajuste, los resultados son sólidos en general, especialmente en la prueba. Además, destacar que precisión en prueba (71.59 %) es la más

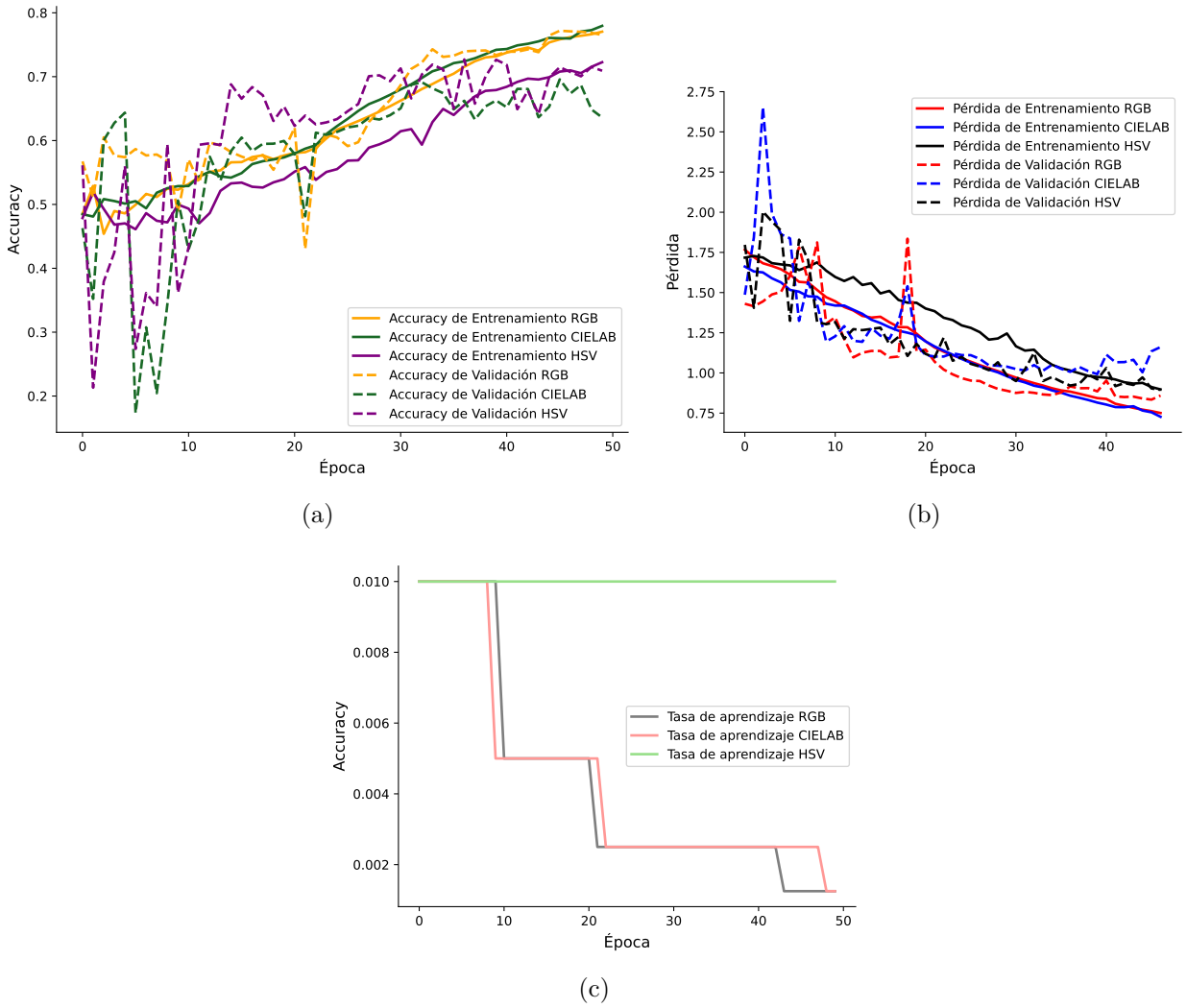


Figura 5.6: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de la arquitectura NetLite.

Tabla 5.10: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura NetLite utilizando los espacios de color RGB, HSV y CIELAB.

Métricas	**RGB**	**HSV**	**CIELAB**
Accuracy de entrenamiento	77. 04 %	72.26 %	77.96 %
Accuracy de validación	76.53 %	70.93 %	63.61 %
Pérdida de entrenamiento	0.7505	0.8974	0.7270
Pérdida de validación	0.8593	0.8948	1.1609
Accuracy prueba	71.59 %	64.97 %	60.61 %
IoU promedio	0.1946	0.1400 %	0.1358 %

alta entre los tres espacios de color.

El modelo en HSV tiene una precisión algo menor en comparación con RGB en todos los aspectos, y la pérdida es más alta tanto en entrenamiento como en validación. Sin embargo, la diferencia entre las métricas de entrenamiento y validación es más pequeña, lo que puede indicar un mejor ajuste, pero a costa de una menor capacidad de generalización.

El modelo en CIELAB muestra un gran desajuste entre la precisión de entrenamiento (77.96 %) y validación (63.61 %), lo cual es una señal de sobreajuste severo. La alta pérdida de validación también sugiere que el modelo no está generalizando bien a este espacio de color, y la precisión de prueba es la más baja de los tres, lo que confirma este problema.

Las curvas de todas las funciones de activación nos muestran que a lo largo de las épocas el rendimiento del modelo va mejorando, aunque un poco lento, sugiere que hay que cambiar algunos hiperparámetros de la red para mejorara su rendimiento, lo cual se analizará con más detalle en secciones posteriores.

En conclusión, RGB sigue siendo el mejor espacio de color en términos de rendimiento global, con la mejor precisión de prueba y una pérdida razonable. HSV tiene un desempeño aceptable, pero no supera a RGB en generalización. CIELAB parece ser el peor de los tres, ya que muestra un alto grado de sobreajuste y una baja precisión de prueba. Finalmente, se decidió seguir trabajando con el conjunto de datos en RGB para aplicar a las demás arquitecturas.

5.6. Selección de función de activación para cada arquitectura

En este trabajo se le da un especial enfoque al uso de las funciones de activación, debido que la selección correcta de esta dependerá un modelo óptimo utilizando los recurso mínimos. A continuación se presentan cada una de las arquitecturas, intercambiando sus funciones de activación entre *ReLU*, *sigmoide* y *tanh*.

5.6.1. Función de activación para U-Net

Tabla 5.11: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura U-Net intercambiando las funciones de activación a *ReLU*, *sigmoide* y *tanh*.

Métricas	***ReLU***	***sigmoide***	***tanh***
Accuracy de entrenamiento	72.43 %	48.75 %	48.99 %
Accuracy de validación	55.21 %	56.74 %	56.74 %
Pérdida de entrenamiento	1.4600	2.1263	2.1307
Pérdida de validación	1.9206	2.0525	2.0415
Accuracy prueba	52.24 %	52.24 %	52.24 %
IoU promedio	0.0994	0.0994	0.0994

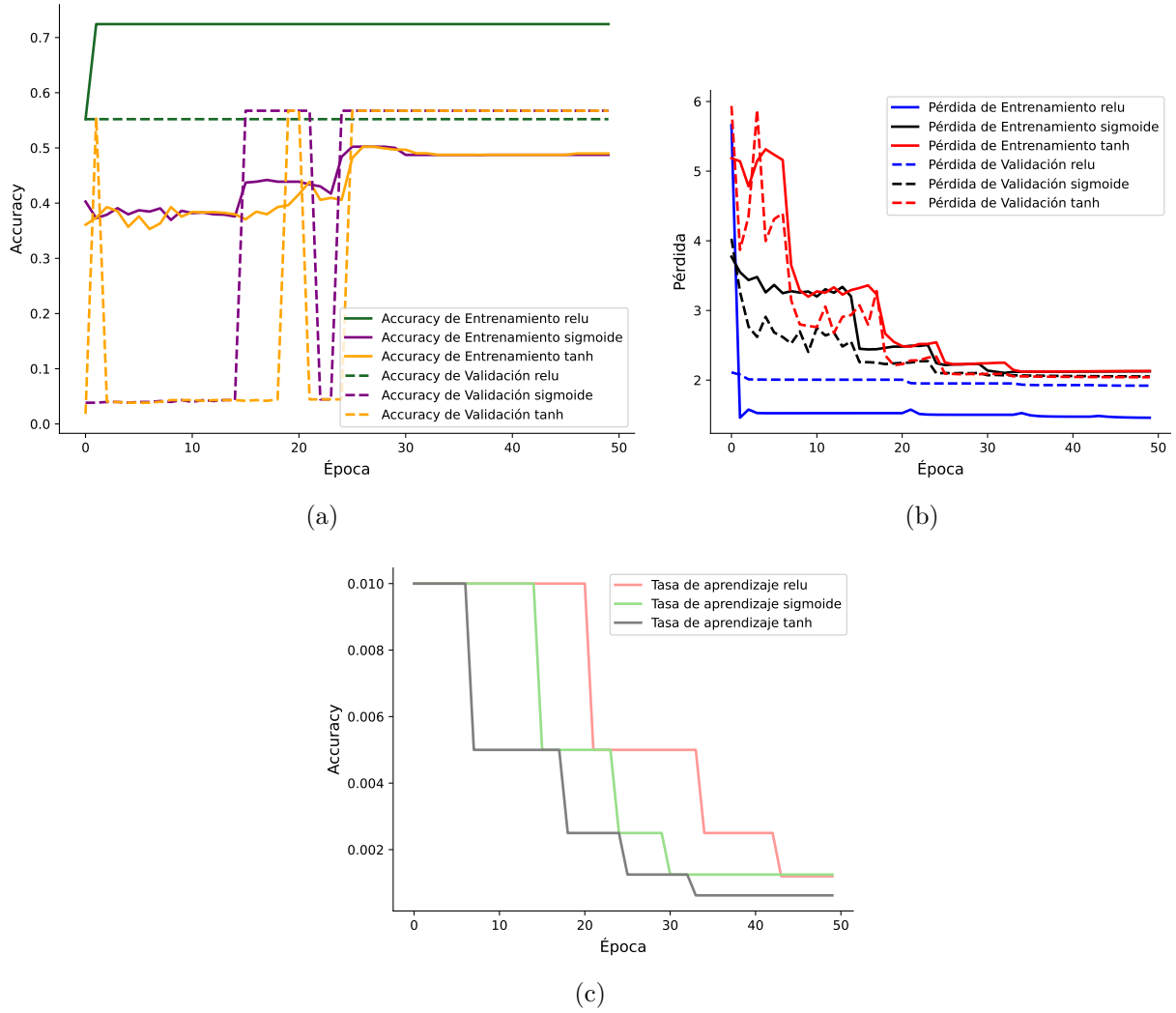


Figura 5.7: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de la arquitectura U-Net.

Aquí, *ReLU* tiene una alta precisión de entrenamiento (72.43 %), pero la precisión cae significativamente en la validación y prueba (55.21 % y 52.24 %, respectivamente). Analizando su gráfica, vemos que el accuracy de validación y entrenamiento se mantienen constantes a lo largo de las 50 épocas, lo que nos sugiere que el aprendizaje ha llegado a un punto donde no puede mejorar su rendimiento en la tarea de clasificación. Esto nos indica que se necesita realizar un ajuste de hiperparámetros para la mejora del rendimiento del modelo.

El accuracy de entrenamiento con *sigmoide* es bastante bajo (48.75 %), lo que indica que el modelo tiene dificultades para aprender patrones en los datos de entrenamiento. Sin embargo, la precisión de validación y prueba son ligeramente mejores que en *ReLU*, pero aún no son ideales. La pérdida es alta tanto en entrenamiento como en validación, lo que sugiere que el modelo no está aprendiendo de manera eficiente, aunque a diferencia de la curva de aprendizaje de *ReLU*, esta sí se ve una mejora en accuracy de entrenamiento, por lo menos a la época 30. Sin embargo, el accuracy de validación también se mantiene constante por largos intervalos, lo que sugiere un límite de aprendizaje tal como se presenta con las otras funciones de activación.

Finalmente, *tanh* tiene un comportamiento muy similar al de *sigmoide*, con una precisión baja en el entrenamiento (0.4899) pero mejor rendimiento en la validación y prueba (0.5674 y 0.5213). La pérdida es igualmente alta, lo que sugiere que el modelo está subentrenado o que la activación no es la más adecuada para la tarea.

En conclusión, *ReLU* tiene la mejor precisión de entrenamiento, pero sufre de un notable sobreajuste, lo que resulta en una peor generalización. *sigmoide* y *tanh* muestran comportamientos muy similares, con mejor generalización, pero ambas activaciones parecen no estar aprendiendo de manera eficiente, dado el bajo rendimiento en entrenamiento y la alta pérdida. Para este caso, se seguirá trabajando con *ReLU*, que a pesar de no haber tenido mejora durante las 50 épocas fue la que se obtuvo un mejor resultado desde las primeras. Haciendo unos ajustes se podrá obtener un mejor rendimiento del modelo con la función *ReLU*.

5.6.2. Función de activación para SegNet

Tabla 5.12: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura SegNet intercambiando las funciones de activación a *ReLU*, *sigmoide* y *tanh*.

Métricas	***ReLU***	***sigmoide***	***tanh***
Accuracy de entrenamiento	53.62 %	50.63 %	51.15 %
Accuracy de validación	55.44 %	55.48 %	55.81 %
Pérdida de entrenamiento	1.6578	1.9611	1.9338
Pérdida de validación	1.9100	1.8779	1.8525
Accuracy prueba	52.24 %	52.91 %	55.36 %
IoU promedio	0.0970	0.0766	0.0690

Observando la Tabla 5.12, podemos concluir que *ReLU* tiene la mejor precisión de entrenamiento, pero la precisión de prueba es la más baja, lo que podría sugerir un sobreajuste (mejora en entrenamiento, pero disminuye en prueba). La pérdida de validación también es la más alta, lo que refuerza este punto.

Por otro lado tenemos la *sigmoide*, cuya accuracy de validación es ligeramente superior a la de *ReLU*, y la pérdida de validación es menor, lo que indica un mejor ajuste para el conjunto de validación. Sin embargo, la precisión de entrenamiento es la más baja, lo que podría indicar que le cuesta aprender patrones en los datos de entrenamiento.

Finalmente, *tanh* tiene la mejor precisión de validación y la menor pérdida de validación, lo que sugiere que está generalizando mejor que las otras dos activaciones. La precisión de entrenamiento no es la más alta, pero la consistencia entre las métricas de entrenamiento y validación indica un buen balance.

Basado en las métricas, podríamos decir que tanh parece ser la mejor opción ya que tiene la mejor precisión de validación y la menor pérdida de validación, lo que sugiere que está generalizando

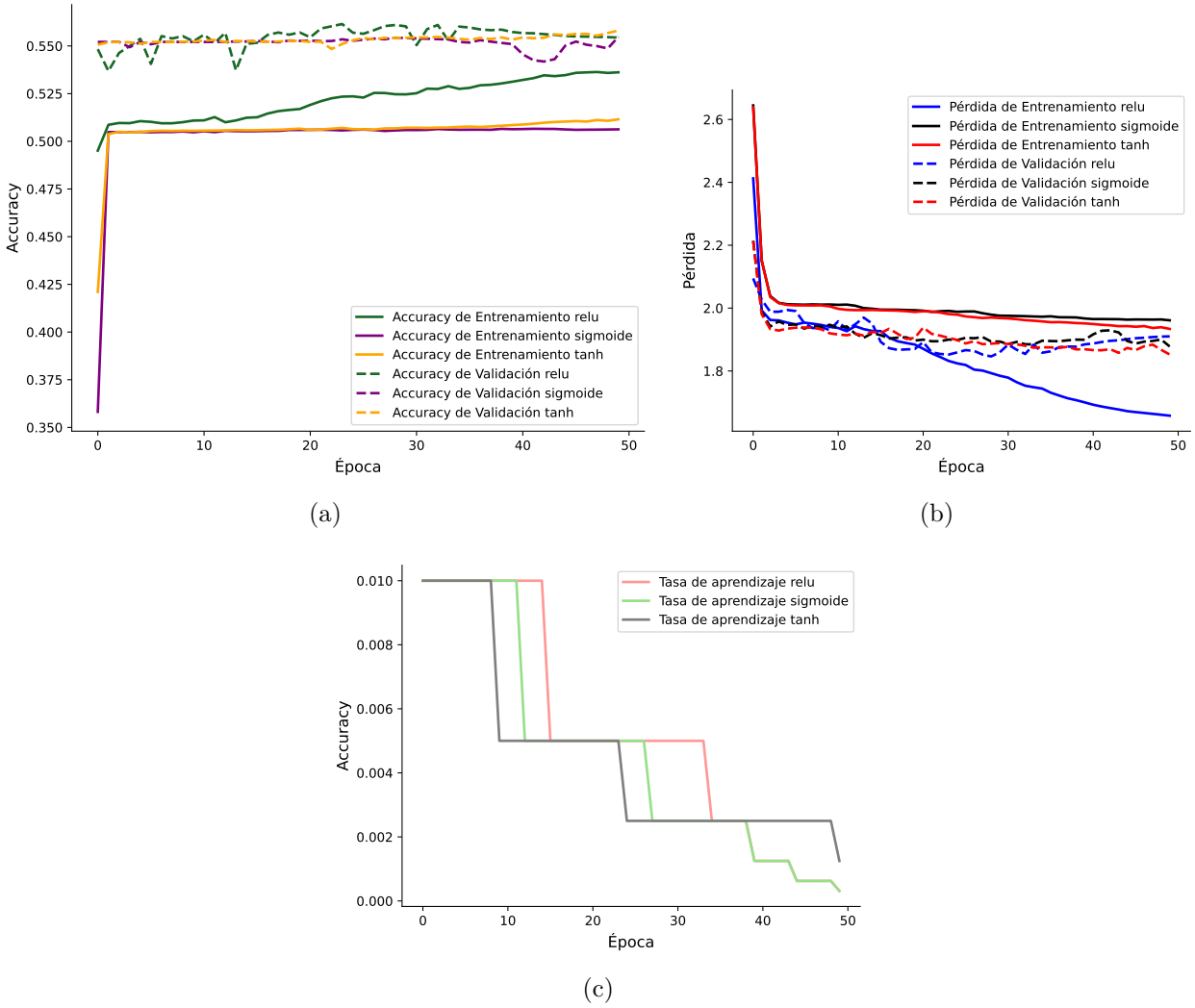


Figura 5.8: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de la arquitectura SegNet.

mejor. Sin embargo, observando las curvas, vemos que *ReLU* es la que tiene mejoras considerables durante el entrenamiento, pudiendo obtener un aprendizaje más fluido ajustando algunas métricas del modelo, por lo que seguiremos trabajando con *ReLU* para mejorar el modelo.

5.6.3. Función de activación para DeepLabV3+

Si analizamos la Tabla 5.13, podemos ver que en la función *ReLU* la precisión de entrenamiento es bastante buena (83.36 %), pero la precisión de validación es significativamente más baja (56.24 %). Esto sugiere que el modelo está sufriendo de sobreajuste. Además, la pérdida de validación es muy alta, lo que indica que el modelo no generaliza bien y la gráfica de la Figura 5.9 refuerza lo antes dicho, ya que el accuaracy de validación se mantiene constante durante todo el entrenamiento.

La función *tanh* muestra la mejor precisión de entrenamiento (89.84 %), pero al igual que *sigmoide*, la precisión de validación es baja (55.86 %). Sin embargo, tiene la menor pérdida de validación, lo

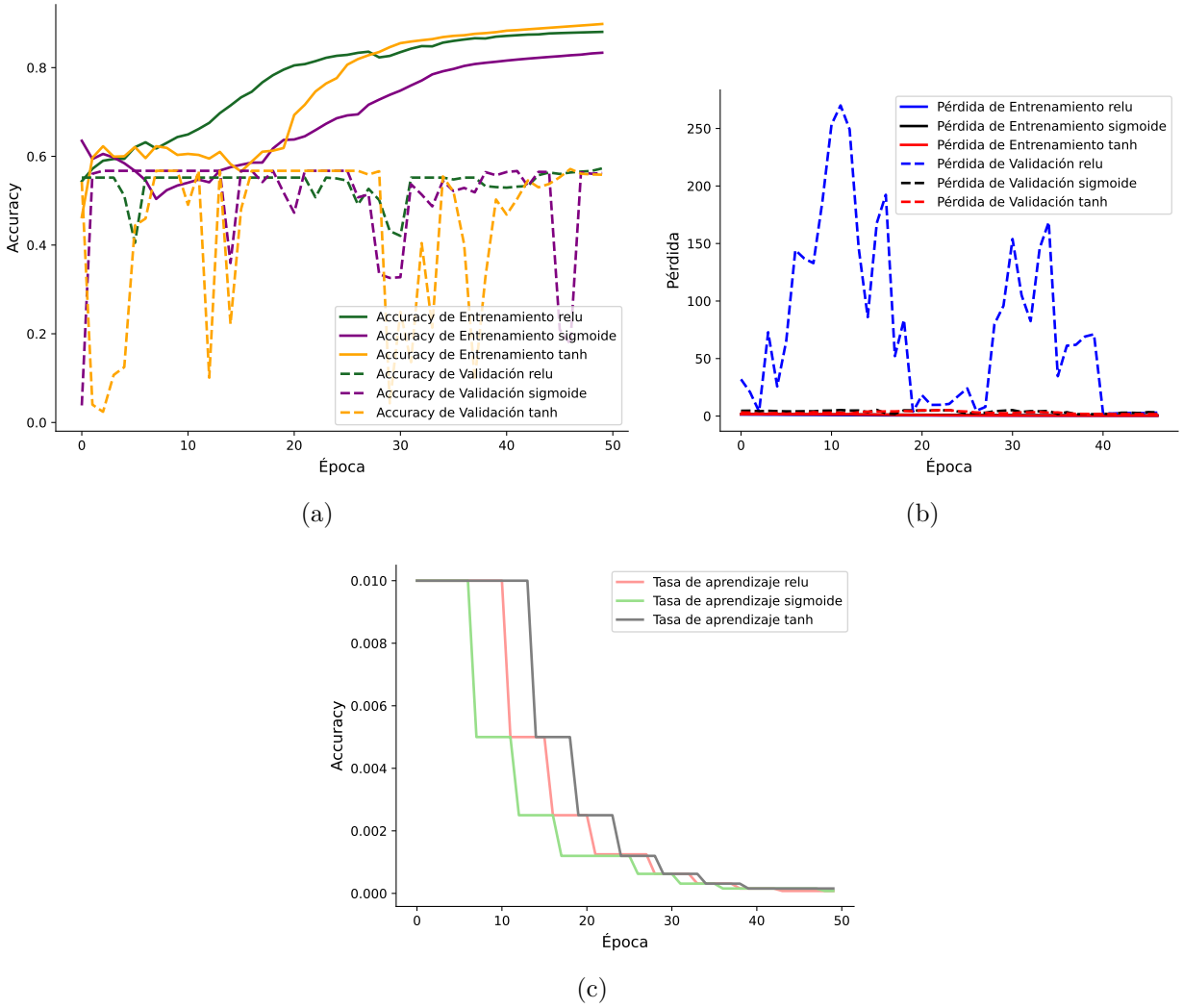


Figura 5.9: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de la arquitectura DeepLabV3+.

Tabla 5.13: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura DeepLabV3+ intercambiando las funciones de activación a *ReLU*, *sigmoide* y *tanh*.

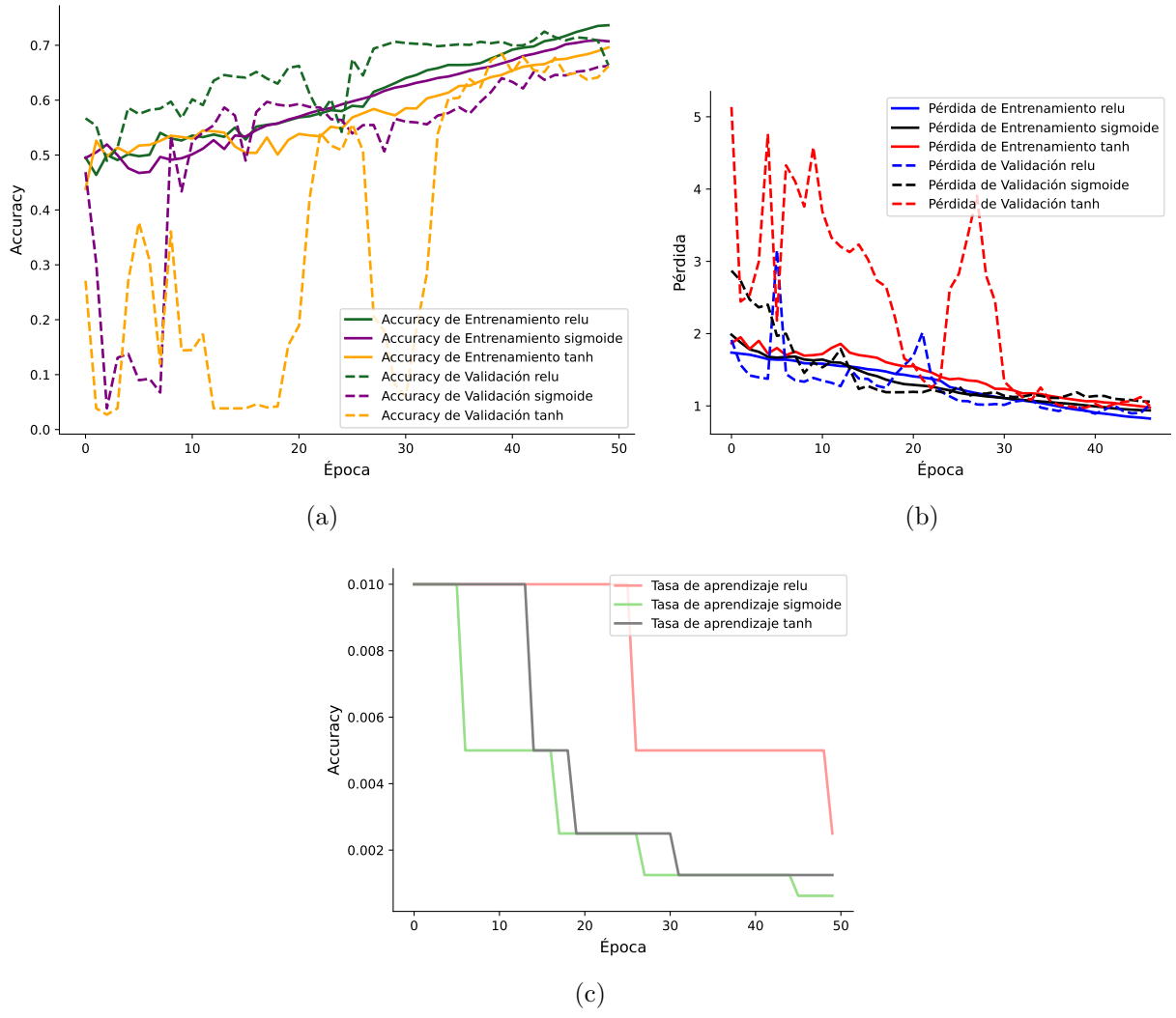
Métricas	***ReLU***	***sigmoide***	***tanh***
Accuracy de entrenamiento	88.06 %	83.36 %	89.84 %
Accuracy de validación	57.29 %	56.24 %	55.86 %
Pérdida de entrenamiento	0.3637	0.5724	0.3472
Pérdida de validación	2.9764	3.1340	1.7248
Accuracy prueba	53.72 %	52.02 %	59.61 %
IoU promedio	0.0629	0.0994	0.1634

que sugiere que está aprendiendo patrones en el entrenamiento de manera más efectiva. A pesar de esto, la diferencia entre las métricas de entrenamiento y validación también indica un cierto grado

de sobreajuste. La precisión de prueba es la mejor de las tres (59.61 %).

En conclusión, *ReLU* tiene una buena precisión de entrenamiento (88.06 %) y una precisión de validación aceptable (57.29 %), pero también muestra una diferencia significativa entre entrenamiento y validación. La pérdida de validación es alta, pero no tanto como en el caso de *sigmoide*. La mejor precisión de prueba y la menor pérdida de validación la encontramos en *tanh*, lo que sugiere un mejor balance general entre ajuste y generalización, como no lo indica IoU, que es el mayor a comparación de las otras, por lo que se continuará usando esta para las siguientes pruebas.

5.6.4. Función de activación para NetLite



(a) (b)

(c)

Figura 5.10: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de la arquitectura NetLite.

La precisión de entrenamiento en *ReLU* es decente (70.73 %) y la de validación es razonable (66.27 %). La pérdida de validación es algo menor que la de entrenamiento, lo que sugiere un ajuste moderado. Sin embargo, la precisión de prueba (58.41 %) es un poco baja en comparación con las otras activaciones.

Tabla 5.14: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura NetLite intercambiando las funciones de activación a *ReLU*, *sigmoide* y *tanh*.

Métricas	***ReLU***	***sigmoide***	***tanh***
Accuracy de entrenamiento	77.04 %	70.73 %	69.62 %
Accuracy de validación	76.53 %	66.27 %	66.19 %
Pérdida de entrenamiento	0.7505	0.9386	0.9774
Pérdida de validación	0.8593	1.0590	0.9990
Accuracy prueba	71.59 %	58.41 %	64.14 %
IoU promedio	0.1946	0.1255	0.1425

La precisión de entrenamiento de *tanh* es un poco más baja que la de *sigmoide*, pero la precisión de validación es casi igual. Además, tiene una pérdida de validación que es ligeramente mejor que su pérdida de entrenamiento, lo que sugiere que está generalizando razonablemente bien. La precisión de prueba (64.14 %) es la mejor entre las tres activaciones.

En conclusión, *ReLU* tiene la mejor precisión de entrenamiento y validación (77.04 % y 76.53 %, respectivamente). La pérdida de validación es la más baja entre las tres, lo que indica que está ajustándose bien a los datos. Además, la precisión de prueba (71.59 %) es la más alta de todos los modelos. *ReLU* se destaca como la mejor opción en este conjunto de resultados, con la mejor precisión de prueba y las métricas más equilibradas entre entrenamiento y validación. Cabe destacar que, hasta ahora, este modelo ha presentado los mejores resultados en cuanto a curvas de aprendizaje se refiere, ya que conserva un equilibrio entre validación y entrenamiento, a diferencias de los modelos anteriores.

5.7. Selección del modelo

Una vez identificada la función de activación con mejor rendimiento para cada uno de los modelos, se procedió a entrenar nuevamente cada una de las arquitecturas, ahora utilizando las funciones óptimas seleccionadas. El entrenamiento anterior, además de evaluar el impacto de las funciones de activación en el desempeño de los modelos, se diseñó con el propósito de realizar un análisis exhaustivo de la estructura de cada red, lo que permitió obtener una mejor comprensión de cómo los distintos hiperparámetros afectan el rendimiento.

En esta nueva fase de entrenamiento, se llevó a cabo un número exhaustivo de pruebas con el objetivo de garantizar mejoras significativas en cada una de las arquitecturas. Para ello, se realizaron variaciones en varios parámetros clave, como las tasas de aprendizaje y los esquemas de regularización, entre otros, mientras que se mantuvo intacta la estructura base de los modelos. Este enfoque permitió optimizar el rendimiento de cada arquitectura, asegurando que se lograra un equilibrio entre el costo computacional y la precisión en la segmentación de óxidos, mejorando los resultados de forma sistemática en cada uno de los modelos probados. En la Figura 5.11 podemos ver las gráficas, y en la tabla 5.15 podemos ver las métricas de evaluación de cada una de las arquitecturas.

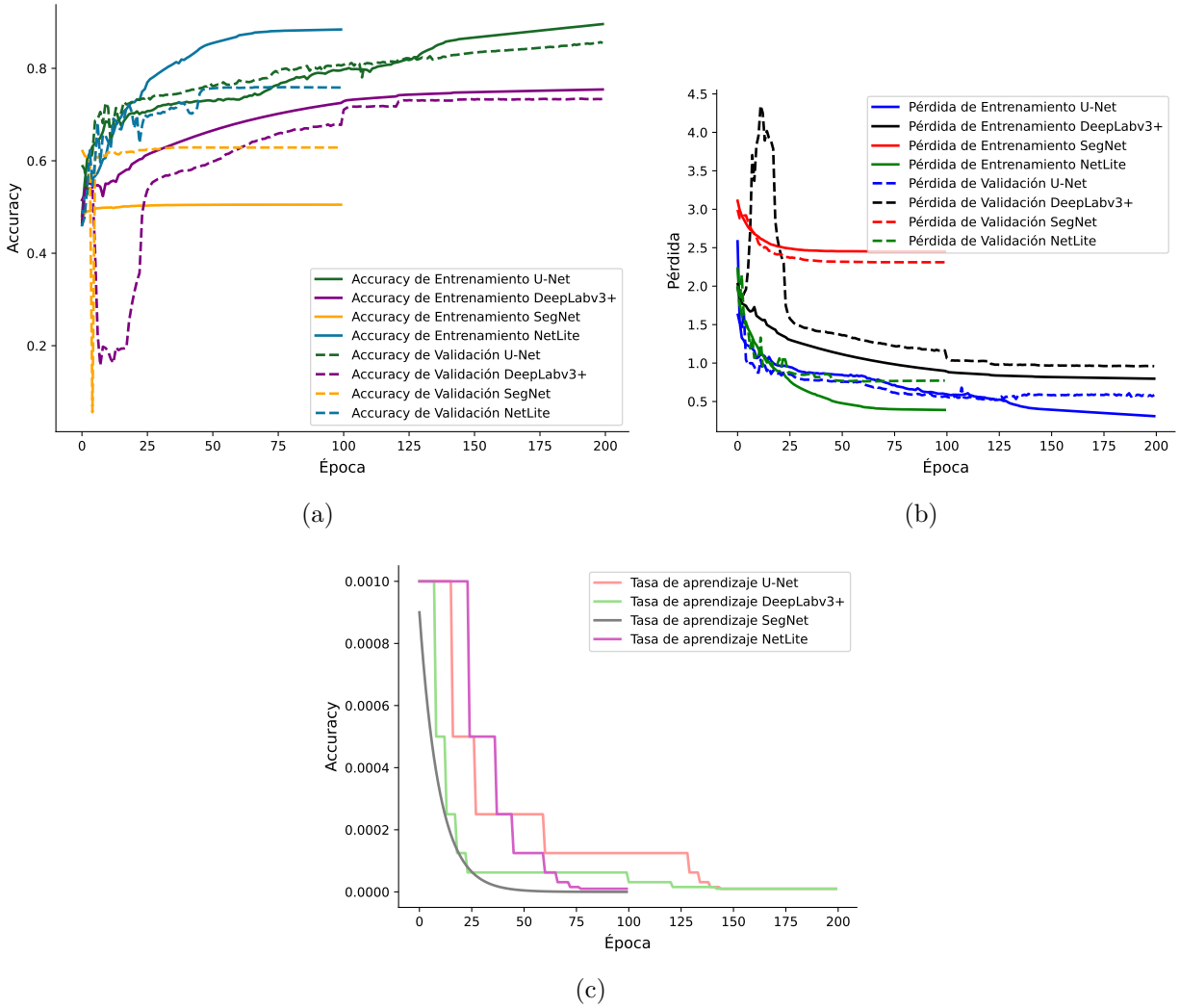


Figura 5.11: Curvas de a) accuracy, b) pérdida, y c) tasa de aprendizaje durante el entrenamiento de las 4 arquitecturas.

Tabla 5.15: Tabla de comparación del accuracy, pérdida e IoU de la arquitectura para las 4 arquitecturas.

Métricas	**SegNet**	**DeepLabv3+**	**NetLite**	**U-Net**
Accuracy de entrenamiento	50.50 %	75.43 %	88.41 %	89.58 %
Accuracy de validación	62.87 %	73.39 %	75.84 %	85.53 %
Pérdida de entrenamiento	2.4513	0.7968	0.3900	0.3098
Pérdida de validación	2.3114	0.9593	0.7719	0.5855
Accuracy prueba	51.95 %	66.73 %	71.77 %	80.29 %
IoU promedio	0.0272	0.2024	0.2641	0.4384

Al analizar el rendimiento de SegNet, se observa que es el modelo con el desempeño más bajo entre las redes probadas. A lo largo de las épocas de entrenamiento, SegNet mostró una tendencia plana en cuanto a su valor de accuracy, tanto en entrenamiento como en la de validación. Este comportamiento indica que, independientemente del número de épocas, el modelo no logra aprender de manera significativa o mejorar su rendimiento en el conjunto de validación, lo que sugiere una posible limitación en su capacidad de generalización para este problema en particular. A diferencia de otros modelos como DeepLabV3+ y U-Net, que mostraron mejoras a lo largo de las épocas, SegNet se mantuvo estancado en un mismo valor de accuracy desde las primeras etapas del entrenamiento. Dada esta situación, se tomó la decisión de limitar el entrenamiento de SegNet a 100 épocas, ya que no había indicios de que un mayor número de iteraciones mejorara su desempeño. Continuar el entrenamiento habría sido ineficiente, tanto en términos de tiempo como de recursos computacionales, debido a la falta de progreso observado.

Al igual que en el caso de SegNet, NetLite también fue entrenada durante un total de 100 épocas. Aunque su curva de aprendizaje muestra un ascenso rápido al inicio, indicando que el modelo aprende de manera eficiente en las primeras fases del entrenamiento, a partir de la época 70 aproximadamente, el accuracy alcanza un valor estable y no presenta mejoras significativas. Esto sugiere que el modelo llega rápidamente a un punto en el que deja de aprender nuevas características relevantes, alcanzando su límite de capacidad de generalización. A pesar de realizar pruebas con un mayor número de épocas, el rendimiento de NetLite no mejora. De hecho, después de la época 100, el accuracy comienza a disminuir, lo que evidencia un caso claro de sobreajuste, donde el modelo empieza a adaptarse demasiado al conjunto de entrenamiento a expensas de su desempeño en el conjunto de validación. Adicionalmente, se observa un aumento en la pérdida, tanto en la fase de validación como en la fase de prueba, lo que confirma este comportamiento no deseado.

DeepLabV3+ también muestra una tendencia de mejora continua a lo largo de las épocas, pero con un ritmo de aprendizaje más lento en comparación con otras arquitecturas como NetLite y U-Net. Aunque se mantuvo en entrenamiento durante 200 épocas, su progreso fue constante pero gradual, lo que indica que la red tiene el potencial de seguir mejorando, aunque a un ritmo reducido. Esto también se puede deber a que se utilizó *tanh* como función de activación, que requiere mayor coste computacional, aunque no debería ser tan considerable este coste a diferencia de $ReLU$, por ejemplo. Otro motivo se puede deber a que *tanh* puede sufrir del problema de desvanecimiento del gradiente, donde los gradientes pueden volverse extremadamente pequeños, ralentizando el entrenamiento, lo cual puede llevar a requerir más tiempo y recursos computacionales para lograr una buena convergencia.

Finalmente, U-Net demuestra ser la arquitectura más efectiva en este análisis. Aunque durante las primeras 150 épocas su rendimiento en entrenamiento se mantiene por debajo de NetLite, es en la fase de validación donde U-Net comienza a sobresalir a partir de la época 50. Este comportamiento sugiere que, a diferencia de NetLite, U-Net tiene una mayor capacidad de generalización, es decir, es más eficaz para adaptarse a datos que no ha visto previamente. Al final de las 200 épocas, U-Net no sólo alcanza, sino que supera el rendimiento de las demás arquitecturas, incluyendo a NetLite. Aunque le toma más tiempo alcanzar los resultados de NetLite, su capacidad de mejora continúa, mientras que NetLite llega a un punto donde su aprendizaje se estanca. Este rendimiento sostenido en U-Net demuestra su potencial para seguir aprendiendo y obtener mejores resultados, mientras que NetLite alcanza rápidamente su límite.

Debido a esta capacidad superior de U-Net para seguir mejorando a lo largo del tiempo y su mejor generalización en comparación con otras arquitecturas, se decidió utilizar esta red para llevar a cabo la tarea de segmentación en nuestro conjunto de datos. Esto asegura que el modelo seleccionado no sólo sea preciso en el entrenamiento, sino que también sea robusto en su capacidad para manejar datos nuevos y obtener mejores resultados en la segmentación.

5.8. Validación cruzada

Finalmente, con el objetivo de comprobar que el modelo no sólo se ajusta bien a los datos de entrenamiento, sino que también mantiene un buen desempeño en datos no vistos, se implementó la validación cruzada. Este proceso consistió en dividir las imágenes y sus respectivas máscaras de entrenamiento en 5 grupos o folds. Es importante destacar que las imágenes aumentadas de cada imagen original se mantuvieron dentro del mismo grupo, evitando así que las versiones aumentadas aparecieran tanto en los datos de entrenamiento como en los de validación, lo que podría causar una evaluación incorrecta del rendimiento del modelo.

Una vez que esta división estuvo lista, se procedió con el entrenamiento del modelo en cada uno de los 5 grupos. Al finalizar el entrenamiento de cada folder, se calcularon las métricas relevantes como accuracy, pérdida e IoU para evaluar el rendimiento, como se hizo en las pruebas anteriores. Además de las métricas que ya hemos calculado anteriormente, se añadió Dice promedio, Precisión, Recall, F1-Score y Especificidad. Finalmente, se obtuvo el promedio de los resultados de cada métrica en los diferentes folds, proporcionando una evaluación más robusta del modelo. Estos promedios se presentan en la Tabla 5.16.

Tabla 5.16: Tabla de métricas al realizar validación cruzada de la arquitectura U-Net en la tarea de segmentación de óxidos.

Métricas	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Valor Promedio**
Accuracy de entrenamiento	94.64 %	94.30 %	94.63 %	94.60 %	94.95 %	94.62 %
Accuracy de validación	93.28 %	95.85 %	94.26 %	93.98 %	91.65 %	93.8 %
Pérdida de entrenamiento	0.1065	0.1035	0.0970	0.1042	0.0934	0.1009
Pérdida de validación	0.1547	0.0732	0.1214	0.1379	0.2170	0.1408
Accuracy prueba	83.47 %	84.22 %	84.12 %	83.42 %	83.70 %	83.60 %
IoU promedio	0.4915	0.5016	0.5046	0.4966	0.4903	0.4969
Dice promedio	0.6249	0.6370	0.6396	0.6306	0.6254	0.6315
Precisión	0.8637	0.8674	0.8664	0.8649	0.8595	0.8643
Recall	0.8347	0.8422	0.8669	0.8342	0.8370	0.8430
F1-Score	0.8292	0.8385	0.8412	0.8292	0.8311	0.8338
Especificidad	0.9908	0.9912	0.9911	0.9907	0.9908	0.9909

5.9. Análisis de los resultados

E accuracy de entrenamiento (94.62 %) y precisión de validación (93.8 %) muestran valores cercanos entre sí, lo cual es un buen indicio, ya que sugiere que el no está sobreajustando de manera significativa. El modelo logra aprender bien durante el entrenamiento y generaliza bastante bien en los datos de validación. La pérdida de entrenamiento (0.1009) y pérdida de validación (0.1408) indica que la validación es algo mayor que la de entrenamiento, lo que es normal, sigue dentro de un rango aceptable. Un pequeño aumento en la pérdida de validación es natural cuando el modelo se enfrenta a datos no vistos. La Precisión de prueba (83.60 %) nos indica que el modelo mantiene un rendimiento razonable en datos completamente nuevos (de prueba), aunque se observa una ligera caída con respecto a la precisión de validación. Esto podría indicar que, aunque el modelo generaliza bien, todavía hay margen para mejorar la robustez de su desempeño en datos externos.

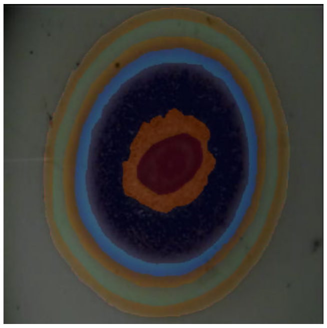
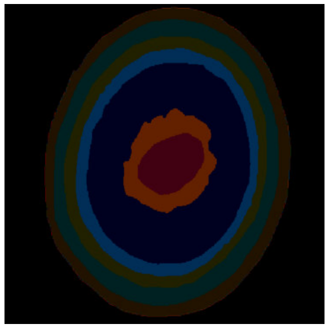
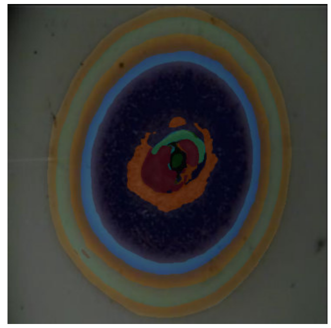
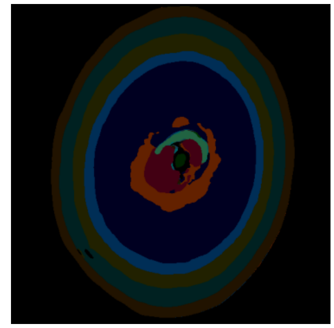
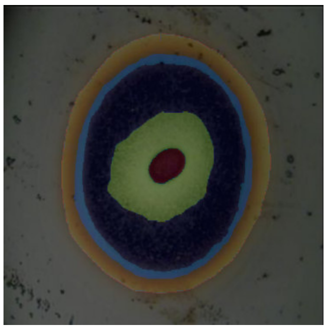
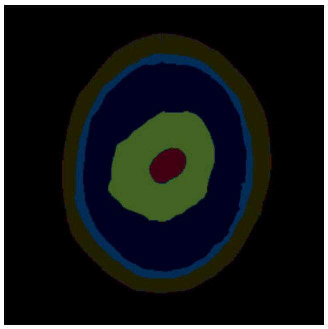
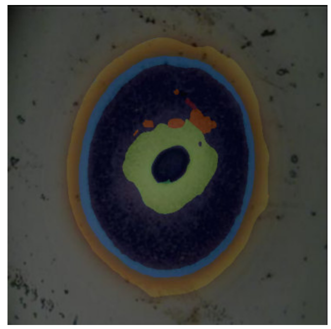
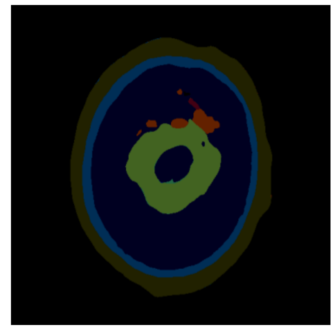
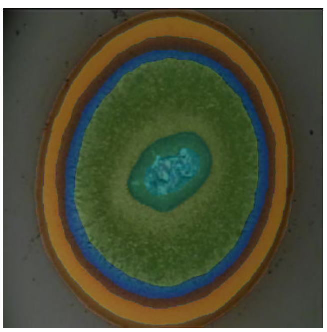
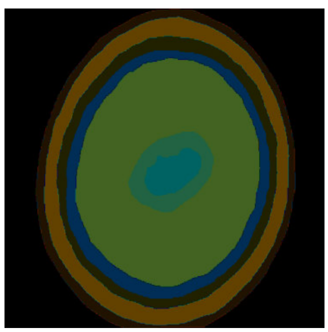
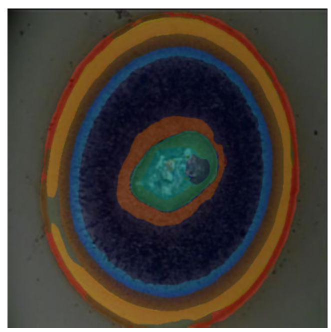
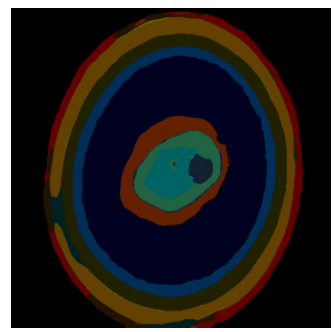
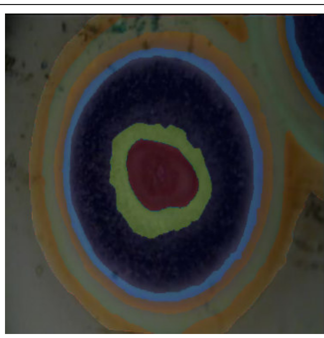
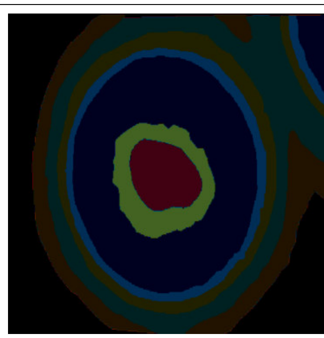
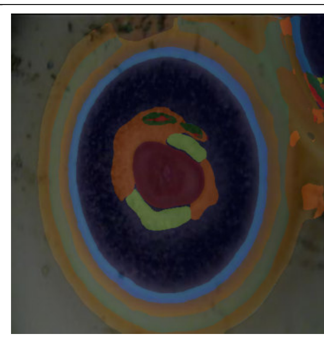
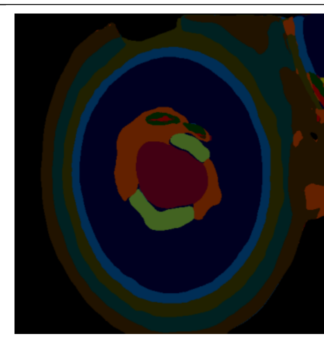
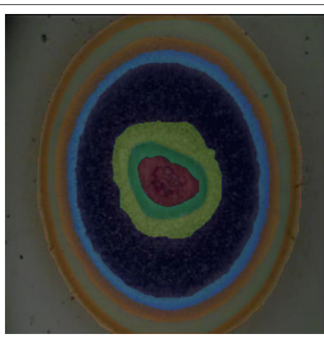
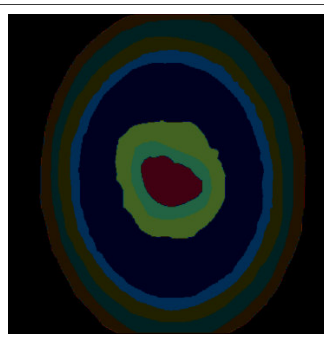
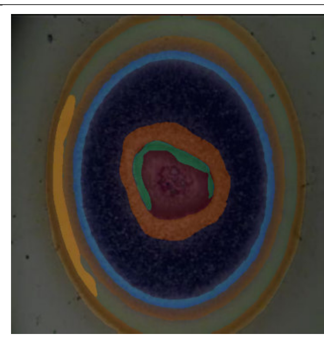
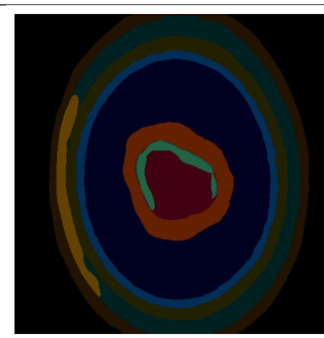
IoU promedio (0.4969) y Dice promedio (0.6315) son valores importantes en tareas de segmentación. Un IoU promedio de 0.4969 sugiere que el modelo tiene un desempeño moderado ya que un valor cercano a 0.75 indicaría un buen desempeño. De igual forma, el Dice promedio (0.6315) indica un desempeño intermedio, ya que un valor cercano a 1 sería el ideal.

La Precisión (0.8643) indica que el modelo es capaz de identificar correctamente una gran proporción de los píxeles predichos como pertenecientes a la clase correcta. Un valor alto de precisión es positivo, pero puede estar indicando un cierto sesgo hacia predecir clases más comunes. Por su parte, Recall (0.8430) indica que logra identificar correctamente una buena proporción de los píxeles de las clases verdaderas. Un balance entre precisión y recall es deseable.

Al combinar precisión y recall, el F1-Score (0.8338) indica que el balance entre ambas métricas es bueno, aunque se podría mejorar. Finalmente tenemos la Especificidad (0.9909), lo cual nos indica un alta especificidad, lo se traduce a que el modelo es muy preciso al predecir los píxeles de las clases negativas (no pertenecientes a las clases de interés). Este resultado es muy positivo, lo que indica que el modelo comete pocos falsos positivos.

Filialmente, en la Figura 5.17, vemos algunas fe las imágenes, resultado de las predicciones del modelo del Fold 3, utilizando la arquitectura U-Net.

Tabla 5.17: Resultados de la segmentación de algunas de las imágenes de prueba, utilizando el modelo del Fold 3.

Imagen original	**Máscara original**	**Imagen predicha**	**Máscara predicha**
			
			
			
			
			

Capítulo 6

Conclusiones

El presente trabajo de investigación logró cumplir el objetivo principal de desarrollar y aplicar modelos de aprendizaje profundo para predecir con precisión los óxidos resultantes de la irradiación con láser pulsado en placas delgadas de molibdeno.

En este trabajo, se aplicaron exitosamente redes neuronales profundas para la estimación de los diámetros de los óxidos resultantes de la irradiación fija en películas de molibdeno. Se llevó a cabo un estudio en el que se observó una variación en el diámetro de los círculos de óxidos en función del tiempo de exposición y la fluencia del láser en las muestras de molibdeno. Estos datos experimentales se utilizaron para entrenar una serie de arquitecturas de redes neuronales, realizando pruebas exhaustivas para obtener el modelo más óptimo, balanceando precisión y eficiencia en el uso de recursos.

El proceso incluyó la experimentación con diferentes optimizadores, cambios en la arquitectura de las redes, variación en el número de parámetros y épocas de entrenamiento, así como la evaluación de distintas funciones de activación, entre otros factores. Los resultados obtenidos fueron no sólo interesantes, sino también excelentes, demostrando que la regresión basada en redes neuronales puede ser muy efectiva para la estimación de diámetros. Un hallazgo significativo fue el impacto de la función de activación en la optimización de tareas específicas como la regresión. Los resultados muestran que, al utilizar la función de activación $ReLU$, se obtuvieron buenos resultados con redes de gran profundidad, es decir, con muchas capas y neuronas. Sin embargo, al cambiar la función de activación a *tanh*, se requería un menor número de parámetros para alcanzar un rendimiento similar. De este modo, una arquitectura menos densa, pero con la función de activación *tanh*, produjo resultados comparables a los de redes más complejas que utilizaban $ReLU$, optimizando así los recursos computacionales.

Por otro lado, a través de la recopilación exhaustiva de datos experimentales sobre la respuesta del molibdeno ante la irradiación láser, se estableció una base sólida para entrenar las arquitecturas neuronales. Mediante un análisis exhaustivo de la literatura y el uso de técnica de la microscopía Raman, se logró etiquetar de manera precisa las micrografías, identificando las fases de los óxidos presentes en cada una de ellas. Basándonos en los datos obtenidos de la microscopía Raman y en el conocimiento existente sobre los óxidos de molibdeno, se identificaron con precisión las fases de los óxidos MoO_2, Mo_4O_{11}, Mo_8O_{23}, $Mo_{18}O_{52}$ y MoO_3. El experimento que se llevó a

cabo irradiando de manera fija las placas de molibdeno con un láser ultrarrápido, lo que permitió observar la presencia de estos óxidos tanto en estado puro como en combinación con una o dos fases adicionales. Además, se identificaron distintas coloraciones dentro de una misma fase, lo que sugiere la posibilidad de variaciones en la estructura o concentración del óxido en cuestión. Este análisis detallado permitió la correcta clasificación y etiquetado de las micrografías, lo que fue un paso crucial para aplicar posteriormente técnicas de segmentación de imágenes.

Para realizar la segmentación de las micrografías, se probaron tres arquitecturas bien conocidas: SEGNET, DeepLabV3+ y U-Net, además de una variación de esta última, optimizada para ser más ligera en términos computacionales. Esta variante fue denominada NetLite, en referencia a su similitud con U-Net, pero con una estructura más simplificada para reducir el costo computacional.

Se llevó a cabo un exhaustivo proceso de entrenamiento para cada una de estas arquitecturas, con el objetivo de encontrar los parámetros más óptimos. Se ajustaron diversos hiperparámetros, tales como la función de activación, la tasa de aprendizaje, el optimizador, y se realizaron pruebas utilizando diferentes espacios de color, entre otros ajustes. El propósito de esta tesis era que NetLite destacara en la tarea de segmentación, dado que, en las primeras épocas de entrenamiento, superaba por un amplio margen a las otras tres redes en cuanto a velocidad de aprendizaje. Sin embargo, a pesar de su rápido entrenamiento, NetLite alcanzó un punto de estancamiento en el que dejó de mejorar con el aumento de las épocas. A pesar de realizar numerosas pruebas y ajustes, esta red llegaba a un máximo en su capacidad de aprendizaje. Por el contrario, U-Net, aunque más lenta, logró superar a NetLite en precisión (accuracy) y reducir significativamente el error. Por esta razón, U-Net fue finalmente seleccionada para realizar la tarea de segmentación en las micrografías experimentales, ya que ofreció el mejor rendimiento global, tanto en precisión como en generalización.

Finalmente, al realizar la validación cruzada, se comprobó que los resultados no sólo dependían de la división inicial del conjunto de datos, sino que, al aplicar este enfoque, fue posible evaluar de manera más precisa la capacidad del modelo para generalizar. En particular, al utilizar la arquitectura U-Net, se observó que el modelo mantenía un buen desempeño en la tarea de segmentación a lo largo de los diferentes pliegues de la validación. Este proceso permitió confirmar que U-Net no estaba sobreajustado a un sólo subconjunto de datos, sino que mostraba consistencia y robustez en su capacidad para segmentar correctamente, independientemente de cómo se organizaran los datos en cada iteración. Así, la validación cruzada confirmó la eficacia de U-Net para esta tarea, asegurando que el modelo podría generalizar bien en datos nuevos.

En conclusión, este estudio contribuye al campo de la caracterización de materiales mediante el uso de inteligencia artificial, demostrando que el aprendizaje profundo es una herramienta eficaz para modelar y predecir los efectos de la irradiación láser en superficies delgadas. A futuro, el trabajo podría ampliarse con la implementación de técnicas adicionales de procesamiento de imágenes o la incorporación de datos experimentales más diversos para mejorar la precisión de las predicciones. Además, este trabajo no sólo avanza en las técnicas de segmentación y predicción en materiales, sino que también fortalece el vínculo entre el aprendizaje automático y la ciencia de materiales, facilitando el desarrollo de modelos predictivos más precisos y eficientes, y abriendo nuevas oportunidades de investigación e innovación en materiales avanzados.

Bibliografía

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. pages 234–241, 2015.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. September 2018.

[4] Fan Zhang, Albert PC Chan, Amos Darko, Zhengyi Chen, and Dezhi Li. Integrated applications of building information modeling and artificial intelligence techniques in the aec/fm industry. *Automation in Construction*, 139:104289, 2022.

[5] Jing Wei, Xuan Chu, Xiang-Yu Sun, Kun Xu, Hui-Xiong Deng, Jigen Chen, Zhongming Wei, and Ming Lei. Machine learning in materials science. *InfoMat*, 1(3):338–358, 2019.

[6] Dane Morgan and Ryan Jacobs. Opportunities and challenges for machine learning in materials science. *Annual Review of Materials Research*, 50(1):71–103, 2020.

[7] Tim Mueller, Aaron Gilad Kusne, and Rampi Ramprasad. Machine learning in materials science: Recent progress and emerging applications. *Reviews in computational chemistry*, 29:186–273, 2016.

[8] Chengxiang Huang, Wei Zhang, and Weitao Zheng. The debut and spreading the landscape for excellent vacancies-promoted electrochemical energy storage of nano-architected molybdenum oxides. *Materials Today Energy*, 30:101154, 2022.

[9] Miroslava Cano-Lara. Óxidos de molibdeno inducidos por irradiación láser de pulsos ultracortos, 2013.

[10] M. Cano-Lara, S. Camacho-López, A. Esparza-García, and M. A. Camacho-López. Laser-induced molybdenum oxide formation by low energy (nj)-high repetition rate (mhz) femtosecond pulses. *Optical Materials*, 33:1648–1653, 2011.

[11] Santiago Camacho-López, Israel O. Perez-López, Miroslava Cano-Lara, Alejandro Esparza-Garcia, M. Carmen Maya-Sanchez, J. Apolinar Reynoso-Hernandez, and Marco Camacho-Lopez. Laser fluence dependence of the electrical properties of moo2 formed by high repetition femtosecond laser pulses. *Physica Status Solidi (A) Applications and Materials Science*, 215, 10 2018.

[12] Santiago Camacho-López, Miroslava Cano-Lara, and Marco Camacho-López. Fast growth of multi-phase moox synthesized by laser direct writing using femtosecond pulses. *Crystals*, 10:1–18, 7 2020.

[13] Wei Hua, Huan-Huan Sun, Fei Xu, and Jian-Gan Wang. A review and perspective on molybdenum-based electrocatalysts for hydrogen evolution reaction. *Rare Metals*, 39:335–351, 2020.

[14] Vandna Rani, Monika Malhotra, Shilpa Patial, Sonali Sharma, Pardeep Singh, Aftab Aslam Parwaz Khan, Sourbh Thakur, Pankaj Raizada, Tansir Ahamad, and Abdullah M Asiri. Formulation strategies for the photocatalytic h2 evolution and photodegradation using moo3-based z-scheme photocatalysts. *Materials Chemistry and Physics*, 299:127454, 2023.

[15] Haiyang Yu, Yuzhi Liu, Shibo Cong, Shuai Xia, and Donglei Zou. Review of mo-based materials in heterogeneous catalytic oxidation for wastewater purification. *Separation and Purification Technology*, 312:123345, 2023.

[16] Chenggong Wang, Irfan Irfan, Xiaoliang Liu, and Yongli Gao. Role of molybdenum oxide for organic electronics: Surface analytical studies. *Journal of Vacuum Science & Technology B*, 32(4), 2014.

[17] Zhichao He, Yang Liu, Shuping Lin, Sihan Shi, ShuLong Sun, Jinbo Pang, Zhiqiang Zhou, Yun Sun, and Wei Liu. Energy band alignment in molybdenum oxide/cu (in, ga) se2 interface for high-efficiency ultrathin cu (in, ga) se2 solar cells from low-temperature growth. *ACS Applied Energy Materials*, 3(4):3408–3414, 2020.

[18] André L F. Cauduro, Roberto Dos Reis, Gong Chen, Andreas K Schmid, Christophe Méthivier, Horst-Gunter Rubahn, Léo Bossard-Giannesini, Hervé Cruguel, Nadine Witkowski, and Morten Madsen. Crystalline molybdenum oxide thin-films for application as interfacial layers in optoelectronic devices. *ACS Applied Materials & Interfaces*, 9(8):7717–7724, 2017.

[19] Ahmed Afif, Sheikh MH Rahman, Atia Tasfiah Azad, Juliana Zaini, Md Aminul Islan, and Abul Kalam Azad. Advanced materials and technologies for hybrid supercapacitors for energy storage–a review. *Journal of Energy Storage*, 25:100852, 2019.

[20] Wei Zhang, Bo Wang, Hao Luo, Fan Jin, Tingting Ruan, and Dianlong Wang. Moo2 nanobelts modified with an mof-derived carbon layer for high performance lithium-ion battery anodes. *Journal of Alloys and Compounds*, 803:664–670, 2019.

[21] Si-Qi Wang, Xiang Cai, Yu Song, Xiaoqi Sun, and Xiao-Xia Liu. Vo_x@ mo o_3 nanorod composite for high-performance supercapacitors. *Advanced Functional Materials*, 28(37):1803901, 2018.

[22] Jordi Torres. *DEEP LEARNING Introducción práctica con Keras*. Lulu. com, 2020.

[23] Simon S Haykin et al. *Neural networks and learning machines*. New York: Prentice Hall, 2009.

[24] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[27] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[28] M. Brejl and M. Sonka. Edge-based image segmentation: machine learning from examples. 2:814–819, 1998.

[29] Zhongwu Wang, John R. Jensen, and Jungho Im. An automatic region-based image segmentation algorithm for remote sensing applications. *Environmental Modelling & Software*, 25(10):1149–1165, 2010.

[30] Bharath Sathya and R Manavalan. Image segmentation by clustering methods: performance analysis. *International Journal of Computer Applications*, 29(11):27–32, 2011.

[31] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. pages 2961–2969, 2017.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[35] J Hermann, M Benfarah, S Bruneau, E Axente, G Coustillier, T Itina, J-F Guillemoles, and P Alloncle. Comparative investigation of solar cell thin film processing using nanosecond and femtosecond lasers. *Journal of Physics D: Applied Physics*, 39(3):453, jan 2006.

[36] J. Sotrop, M. Domke, A. Kersch, and H.P. Huber. Simulation of the melting volume in thin molybdenum films as a function of the laser pulse duration. *Physics Procedia*, 41:520–523, 2013. Lasers in Manufacturing (LiM 2013).

[37] Jeppe Byskov-Nielsen. *Short-pulse laser ablation of metals: Fundamentals and applications for micro-mechanical interlocking*. 2010.

[38] Arne Magnéli, Georg Andersson, Birgitta Blomberg, and Lars Kihlborg. Identification of molybdenum and tungsten oxides by x-ray powder patterns. *Analytical chemistry*, 24(12):1998–2000, 1952.

[39] M Wautelet. Laser-assisted reaction of metals with oxygen. *Applied Physics A*, 50:131–139, 1990.

[40] Ángel Pérez del Pino. *Coloración del titanio mediante el tratamiento superficial de oxidación por láser*. Universitat de Barcelona, 2003.

[41] P. A. Spevack and N. S. McIntyre. Thermal reduction of molybdenum trioxide. *The Journal of Physical Chemistry*, 96(22):9029–9035, 1992.

[42] M.A. Camacho-López, L. Escobar-Alarcón, M. Picquart, R. Arroyo, G. Córdoba, and E. Haro-Poniatowski. Micro-raman study of the $m - moo_2$ to $\alpha - moo_3$ transformation induced by cw-laser irradiation. *Optical Materials*, 33(3):480–484, 2011.

[43] Blume Andreas. Synthese und strukturelle untersuchungen von molybdän, vanadium und wolframoxiden als referenzverbindungen für die heterogene katalyse, March 2004.

[44] Dietere Martin. In situ resonance raman studies of molybdenum oxide based selective oxidation catalysts, March 2001.

[45] I Navas, R Vinodkumar, K J Lethy, A P Detty, V Ganesan, V Sathe, and V P Mahadevan Pillai. Growth and characterization of molybdenum oxide nanorods by rf magnetron sputtering and subsequent annealing. *Journal of Physics D: Applied Physics*, 42(17):175305, aug 2009.

[46] HJ Lunk, TA Frisk, H Hartl, IG Shenderovich, D Mauder, M Feist, MJG Fait, R Eckelt, MA Hartl, and LL Daemen. What's behind hexagonal molybdenum oxide?, 2009.

[47] Alfonso José Vázquez Vaamonde, Juan J Damborenea González, and JJ de Damborenea. *Ciencia e ingeniería de la superficie de los materiales metálicos*, volume 31. Editorial CSIC-CSIC Press, 2000.

[48] Leslie N. Smith. Cyclical learning rates for training neural networks. pages 464–472, 2017.

[49] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[50] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr DollÃ¡r, and Ross Girshick. Segment anything, 2023.

[51] wkentaro. Github - wkentaro/labelme: Image polygonal annotation with python (polygon, rectangle, circle, line, point and image-level flag annotation)., Jun 2024.

[52] Jiaqi Li and Xiaodong Yang. A cyclical learning rate method in deep learning training. pages 1–5, 2020.

[53] Ralf Gulde, Marc Tuscher, Akos Csiszar, Oliver Riedel, and Alexander Verl. Deep reinforcement learning using cyclical learning rates. pages 32–35, 2020.