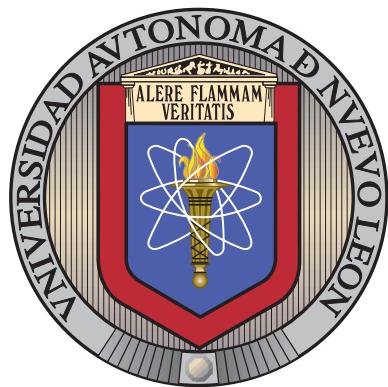


UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN ACADÉMICA



DETECCIÓN DE MELANOMA DE PIEL MEDIANTE
SEGMENTACIÓN SEMÁNTICA

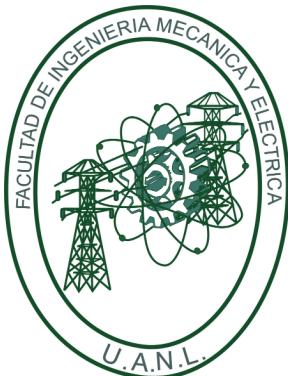
POR

MARIO ALBERTO FLORES HERNÁNDEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
INGENIERO EN MECATRÓNICA

FEBRERO 2021

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN ACADÉMICA



DETECCIÓN DE MELANOMA DE PIEL MEDIANTE
SEGMENTACIÓN SEMÁNTICA

POR

MARIO ALBERTO FLORES HERNÁNDEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE
INGENIERO EN MECATRÓNICA

FEBRERO 2021



UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN



FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

**UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SUBDIRECCIÓN ACADÉMICA**

Los miembros del Comité de Tesis recomendamos que la Tesis «Detección de melanoma de piel mediante segmentación semántica», realizada por el alumno Mario Alberto Flores Hernández, con número de matrícula 1719126, sea aceptada para su defensa como requisito parcial para obtener el grado de Ingeniero en Mecatrónica.

El Comité de Tesis

Dra. Satu Elisa Schaeffer
Asesora

Dr. Romeo Sánchez Nigenda
Revisor

Dra. Sara Elena Garza Villarreal
Revisora

Vo. Bo.

Dr. Fernando Banda Muñoz
Subdirector Académico

San Nicolás de los Garza, Nuevo León, febrero 2021

ÍNDICE GENERAL

Índice de Figuras	viii
Índice de Cuadros	ix
Notaciones	x
Agradecimientos	xi
Resumen	xii
1. Introducción	1
1.1. Hipótesis	3
1.2. Objetivos	3
1.3. Estructura de la Tesis	5
2. Antecedentes	7
2.1. Cáncer de Piel	7
2.1.1. Tipos de Cáncer de Piel	8
2.2. Redes Neuronales	9
2.2.1. Imágenes como Datos	10
2.2.2. Modelo	10
2.2.3. Evaluación y Optimización	12
2.3. Retro-propagación	13

3. Estado del Arte	14
3.1. Trabajos Similares	15
3.2. Análisis Comparativo	18
3.3. Áreas de Oportunidad	19
4. Solución Propuesta	21
4.1. Metodología	22
4.1.1. Características de los Datos de Entrada	22
4.1.2. Codificación de Características	23
4.1.3. Modelo FPN	24
4.2. Implementación de la Solución	25
4.2.1. Computación Asistida por Hardware	27
4.2.2. Módulos de la Implementación	27
4.2.3. Lectura de Entrada	28
4.2.4. Pre-procesamiento	30
4.2.5. Entrenamiento y Validación	31
4.2.6. Predicción de Máscaras	33
5. Experimentos	35
5.1. Estadísticas de los Datos de Entrada	36
5.1.1. Diseño Experimental	36
5.1.2. Resultados	37
5.1.3. Discusión	38
5.2. Evaluación del Aprendizaje	42
5.2.1. Diseño Experimental	42
5.2.2. Resultados	42
5.2.3. Discusión	43

5.3. Criterio de Jaccard	45
5.3.1. Diseño Experimental	45
5.3.2. Resultados	46
5.3.3. Discusión	47
5.4. Resumen de Experimentos	48
6. Conclusiones	50
6.1. Discusión	51
6.2. Trabajo a Futuro	52
A. Repositorio de GitHub	54
Bibliografía	58

ÍNDICE DE FIGURAS

1.1.	Ilustración de las capas de la piel y sus apéndices [20].	2
1.2.	Ejemplos de melanoma extraído de la base de datos de ISIC [6].	3
1.3.	Ejemplo de segmentación: entrada y salida.	4
2.1.	Comparación de las dimensiones entre los distintos canales de color. .	10
2.2.	Ejemplo de un perceptrón.	11
4.1.	Diagrama de flujo general de la herramienta propuesta.	22
4.2.	Representación del proceso de obtención de la matriz convolucionada.	24
4.3.	Representación de la arquitectura convolucional.	25
4.4.	Distribución de los módulos en la herramienta desarrollada.	28
4.5.	Representación del árbol de directorios de imágenes.	28
4.6.	Línea de pre-procesamiento.	31
5.1.	Conjunto de datos entrante.	38
5.2.	Histogramas de densidad de la intensidad de píxeles de la imagen entrante.	39
5.3.	Conjunto de datos después de aplicar el filtro de umbral.	40
5.4.	Histograma de densidad de la intensidad de píxeles después de aplicar el filtro de umbral.	41
5.5.	Representación de la región de sobreposición de píxeles en el coeficiente de datos.	43
5.6.	Curva de la pérdida según el coeficiente de datos en cada época. . .	44

5.7. Representación de la región de sobreposición de pixeles en el índice de Jaccard.	46
5.8. Curvas de la evaluación del criterio de Jaccard en cada época.	47

ÍNDICE DE CUADROS

3.1. Similitudes y diferencias entre los trabajos revisados; las características implementadas se representan con ✓, mientras que las no implementadas con ✗.	19
4.1. Librerías utilizadas para la implementación de la propuesta.	26
4.2. Especificaciones técnicas de los componentes.	27
5.1. Características del conjunto de datos.	37

NOTACIONES

A continuación se resumen las notaciones usadas a lo largo de este trabajo de tesis, incluyendo el símbolo, significado y equivalente en el idioma inglés.

Símbolo	Definición	Equivalente al inglés
x	dato de entrada	<i>input</i>
y	dato de salida	<i>output</i>
\hat{y}	salida estimada	<i>estimated output</i>
d	dimensión de entrada	<i>input dimension</i>
d_o	dimensión de salida	<i>output dimension</i>
n_s	número de muestras	<i>samples</i>
e_i	entrada del modelo	<i>node input</i>
p_i	peso sináptico	<i>synaptic weight</i>
ϕ	función de activación	<i>activation function</i>

AGRADECIMIENTOS

Quiero agradecer primeramente a mi madre Patricia Hernández Romero por siempre representar un ejemplo de esfuerzo y superación en cada uno de los obstáculos que se han presentado.

A mi padre Mario Alberto Flores Rosales (†), quien me enseñó las virtudes del trabajo y la disciplina.

A la Dra. Satu Elisa Schaeffer por la asesoría, paciencia y consejos otorgados durante el desarrollo de este trabajo.

Al Dr. Romeo Sánchez Nigenda y a la Dra. Sara Elena Garza Villarreal por el tiempo y comentarios otorgados para mejorar este trabajo.

A mis compañeros de la FIME y del ITESM quienes me han otorgado su tiempo para enriquecer y ampliar mis conocimientos y que además de eso, me han servido como modelos a seguir y ejemplos de superación.

RESUMEN

Mario Alberto Flores Hernández.

Candidato para obtener el grado de Ingeniero en Mecatrónica.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: DETECCIÓN DE MELANOMA DE PIEL MEDIANTE SEGMENTACIÓN SEMÁNTICA.

Número de páginas: 59.

OBJETIVOS Y MÉTODO DE ESTUDIO: El presente trabajo de tesis habla sobre la aplicación del aprendizaje profundo o también denominado *deep learning*, para la detección de melanoma de piel mediante la aplicación de las redes neuronales profundas. El melanoma es un padecimiento que se origina en las células de la piel cuando los melanocitos (las células que dan el color marrón a la piel), comienzan a crecer sin control causando estragos a quien la padece y que sin una detección y tratamiento temprano, puede aumentar exponencialmente el riesgo de fallecimiento del afectado. Una red neuronal es un modelo matemático que pretende simular el proceso de aprendizaje de las neuronas biológicas mediante el recibimiento de señales binarias o normalizadas, la asignación de un peso sináptico o importancia para cada señal y una función de activación que discrimine o realce dichas señales. Es posible desarrollar un modelo cuyo entrenamiento dé como resultado un sistema que ejecute una secuencia específica de transformaciones con el dato entrante, mediante la optimización automática de pesos sinápticos en las distintas capas del filtrado, para transformar la señal o dato entrante al valor deseado a la salida. El dato entrante, en el caso de imágenes a color, trata de tres matrices correspondientes a los tres canales de colores primarios (azul, verde, rojo), y la salida deseada trata de un mapa probabilístico correspondiente al mapa de segmentación en el cual se ilustran las regiones de la imagen mediante colores distintos, refiriéndose mediante su color a las categoría correspondiente de dicha región (tejido sano, tejido melanoma). Para obtener

un modelo que realice dicha serie de transformaciones se requiere de dos procesos: *reducción* y *reconstrucción*. La fase de *reducción* trata de reducir la dimensión de la imagen de entrada conservando su información característica, haciendo más fácil su computación en los procesos siguientes; y la fase de *reconstrucción*, como el nombre indica, corresponde a la reconstrucción de la imagen basándose en una predicción aplicada al dato previamente codificado en la fase de *reducción*. Para finalizar se realiza un escalado del mapa probabilístico obtenido de la predicción para crear la máscara de segmentación con sus correspondientes regiones categorizadas.

CONTRIBUCIONES Y CONCLUSIONES: La principal contribución del presente trabajo de tesis es la de establecer los pasos requeridos para llevar a cabo el proceso de implementación de redes neuronales profundas para realizar tareas de segmentación semántica. El lenguaje utilizado en esta implementación fue el de *Python*, debido a que sus librerías permiten desplegar ágilmente modelos neuronales y utilizar métricas que permitan evaluar y experimentar su funcionamiento. Una de las métricas utilizadas para evaluar dicha hipótesis es la del índice de Jaccard, su función es comparar un mapa binario con un mapa probabilístico obtenido por el modelo generado y determinar la similitud entre ambos.

Para la implementación se propuso el uso de la arquitectura FPN¹, la cuál es compatible con el codificador ResNet, ambos consisten en secuencias de convolución y deconvolución que realizarán las tareas de reducir las dimensiones de los datos en un solo dato multi-dimensional y de recrear el mapa probabilístico correspondiente a las computaciones del modelo. Para validar la selección de la arquitectura y codificador se estableció una serie de experimentos dependientes unos de otros en donde se determina la mejor configuración de parámetros para la creación y el entrenamiento del modelo así como la validación de las máscaras obtenidas mediante sobreposición de pixeles de ambas máscaras, mediante el coeficiente de dados y el criterio de Jaccard.

Mediante el experimento del filtro de umbral se obtuvieron datos sobre las características de las imágenes en cuanto la intensidad de los pixeles antes y después de aplicarse, y con los diferentes criterios de evaluación mencionados, se obtuvo un perfil de aprendizaje del modelo en donde se puede visualizar el principal comportamiento durante las épocas iniciales y las finales.

Firma de la asesora:

Dra. Satu Elisa Schaeffer

¹Feature Pyramid Network

CAPÍTULO 1

INTRODUCCIÓN

La piel es considerada como el órgano más grande del cuerpo humano; está compuesta por tres capas: epidermis, dermis e hipodermis, tal como se ilustra en la figura 1.1. La función principal de la piel es la de proteger al cuerpo de las hostilidades del medio ambiente, sin embargo, también nos protege de la radiación solar y los agentes externos externos como las bacterias. Otras funciones son las de contener a los órganos internos y también funciona como una interfaz que nos permite interactuar con el medio ambiente otorgándonos la capacidad de percibir sensaciones tales como el tacto y la temperatura.

Debido a la exposición continua a la radiación solar o artificial, un porcentaje de la población llega a desarrollar anomalías en las células de la piel. Una de estas anomalías es el *melanoma*, la cuál es una mutación originada en los melanocitos (células encargadas de otorgar la pigmentación marrón a la piel) en donde las células crecen sin control y se esparcen a otras partes del cuerpo ocasionando graves daños a la salud de quien la padece y que sin una detección temprana puede aumentar rápidamente el riesgo de fallecimiento [19].

Su detección temprana es imprescindible para reducir el porcentaje de morta-

lidad que esta enfermedad representa, de ahí el porqué es importante no solamente trabajar en encontrar un tratamiento, sino también (y de forma simultánea), trabajar e investigar en el uso de tecnologías emergentes que permitan su detección temprana, confiable y al alcance de la mayoría.

En los últimos años se han logrado muchos avances en cuanto al desarrollo de *software* inteligente. Una de las tecnologías que han adquirido mayor importancia es la *red neuronal*, la cual se trata de un modelo matemático que tiene la capacidad de *aprender* mediante el uso de bases de datos y mediante funciones de optimización que permite la configuración de dicho modelo y le otorga la capacidad de predecir, clasificar o construir datos ya sean futuros o desconocidos. Algunos de los sectores que se han beneficiado más de esta tecnología son: el sector automotriz (pilotos automáticos), el sector de manufactura (optimización de procesos), el sector de entretenimiento (recomendaciones personalizadas), el sector médico (diagnóstico de imágenes).

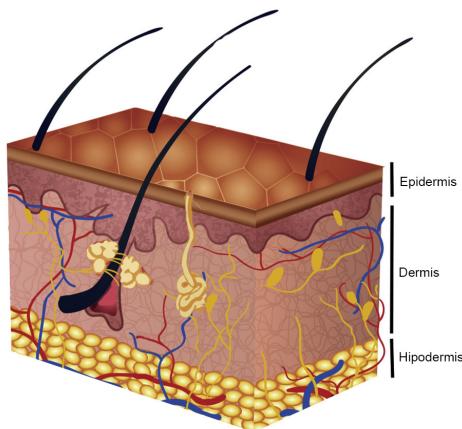


FIGURA 1.1: Ilustración de las capas de la piel y sus apéndices [20].

1.1 HIPÓTESIS

El presente trabajo de tesis tiene como hipótesis el uso de la inteligencia artificial para la detección y clasificación del melanoma de piel. Mediante la implementación de un modelo basado en el *aprendizaje profundo* es posible crear una aplicación de *software* que entrene y ajuste los parámetros de un modelo para realizar la tarea de *segmentación semántica*, la cual se trata de la transformación y predicción de pixeles para obtener la clasificación de las diferentes regiones que representa una imagen dermatológica. Y de esta manera obtener una aplicación con la capacidad de segmentar regiones dentro de las imágenes entrantes y cuya robustez pueda ser verificada a partir de una serie de experimentos.

1.2 OBJETIVOS

Primero en *objetivo general*, se habla de manera conceptual la problemática a resolver tales como las situaciones en las que podemos optimizar la solución de un problema mediante el uso de la red neuronal. Posteriormente en los *objetivos específicos* se describe de forma puctual las metas que se pretenden obtener mediante la implementación de la arquitectura de red neuronal.



FIGURA 1.2: Ejemplos de melanoma extraído de la base de datos de ISIC [6].

El *objetivo general* de este trabajo de tesis es la creación de una aplicación con la capacidad de reconocer y segmentar melanoma en la piel de forma automatizada, esto con el motivo de asistir a los médicos dedicados a esta labor a optimizar y extender la detección temprana, consiguiendo así reducir la tasa de mortalidad de este padecimiento.

El *objetivo específico*, es el de implementar un modelo de red neuronal cuya entrada sean imágenes que pueden o no contener la presencia de alguno de los tipos de cáncer de piel comunes, y como salida del modelo se obtenga un mapa de características donde de haber presencia del cáncer, éste se distinga mediante una región colorada en el espacio que abarca. El modelo debe cumplir con las siguientes características:

- El modelo debe ser capaz de trabajar con imágenes a color o en blanco y negro.
- El modelo debe contar con una función capaz de evaluar la precisión.
- El modelo debe contar con un algoritmo capaz de optimizar la precisión.
- El modelo debe ser capaz de segmentar correctamente en imágenes no utilizadas en los datos de entrenamiento.



(a) imagen entrante



(b) máscara de salida

FIGURA 1.3: Ejemplo de segmentación: entrada y salida.

En la figura 1.3a se observa como la imagen entrante presenta un caso de melanoma, mientras que la figura 1.3b se puede observar la misma imagen después de pasar por una secuencia de transformaciones. Esto se denomina *segmentación semántica* y se refiere a la acción de reconocer y señalar la presencia de una o más categorías de objetos dentro de la imagen entrante.

1.3 ESTRUCTURA DE LA TESIS

A continuación se da una breve explicación sobre el orden en el que se presentan los capítulos de este trabajo de tesis, así como una breve descripción de su contenido.

En el capítulo 2 se habla sobre los antecedentes relacionados al presente trabajo de tesis, primero se empieza definiendo la naturaleza del problema con el que se pretende tratar, después algunos conceptos clave que serán necesarios para la comprensión de la implementación propuesta tales como las dimensiones de los datos y los algoritmos de evaluación y optimización.

En el capítulo 3 se recopilan trabajos relacionados al método o problemática en cuestión, se estudia las características de dichos trabajos y se busca un punto de convergencia entre estos y el presente trabajo de tesis con el fin de comparar las áreas de oportunidad.

En el capítulo 4 se define el proceso de transformación de los datos del sistema propuesto en este trabajo de tesis, desde las características de los datos a la entrada.

El capítulo 5 describe a profundidad la implementación de la propuesta, desde la descripción de los datos utilizados para el entrenamiento del modelo y la verificación de este, la arquitectura específica utilizada

Finalmente, en el capítulo 6 se exponen los resultados obtenidos de la implementación del producto científico en el capítulo anterior, así como un análisis y conclusión final sobre los valores obtenidos en precisión y tiempo de entrenamiento.

CAPÍTULO 2

ANTECEDENTES

En este capítulo se introducen de forma teórica los conceptos relacionados con el trabajo propuesto, primero se define que es el *cáncer de piel*, cuales son los factores que influyen en el desarrollo de este padecimiento, los tipos de cáncer y las diferencias entre estos. Después algunos conceptos relacionados con las redes neuronales necesarios para un entendimiento sólido del *aprendizaje profundo*, tales como los elementos clave que conforman a la red neuronal, la manera en la que esta evalúa su precisión, y el algoritmo de optimización.

2.1 CÁNCER DE PIEL

El *cáncer de piel* es una enfermedad que suele relacionarse con la aparición de lunares o manchas que no se encontraban previamente, se pueden manifestar como manchas oscuras o rojizas, bultos y/o escamas en la superficie de la piel y afecta a todos los tonos de piel por igual. Esta enfermedad se suele desarrollar principalmente en partes del cuerpo expuestas al sol, sin embargo, también se puede desarrollar en partes que no suelen exponerse. Algunos factores como la exposición a los rayos ultravioletas (rayos UV), el uso de sustancias como el tabaco o el envejecimiento

son factores correlacionados con la aparición del *cáncer de piel*, existen dos tipos de factores: *intrínsecos* y *extrínsecos*.

Los factores *intrínsecos* son aquellos que suceden de forma interna en la piel, un ejemplo de esto es el envejecimiento cronológico, el cual es un proceso natural que consiste en degradación del colágeno, la elastina y el adelgazamiento de la epidermis debido al envejecimiento celular al paso de los años y de el efecto de algunas hormonas.

Los factores *extrínsecos* son aquellos que suceden de forma ajena al organismo como es el caso del *foto-envejecimiento* el cual sucede cuando nos encontramos expuestos a los rayos ultravioletas (UV). Este factor de envejecimiento genera lesiones en las cadenas de ácido desoxirribonucleico (ADN) debido a la oxidación y afecta la regeneración de células, al sistema inmune y a la forma en la que se regula la pigmentación [14].

2.1.1 TIPOS DE CÁNCER DE PIEL

El cáncer de piel es un padecimiento que se puede manifestar en una gran variedad de formas distintas, algunas de estas formas pueden representar un enorme riesgo para la salud de quien la padece mientras que otras pueden permanecer mucho tiempo sin afectar la calidad de vida del paciente; se clasifican principalmente en dos tipos: *benigno* y *maligno*.

Los tumores del tipo *benignos* no suelen ser considerados cáncer como tal, ya que tienen un crecimiento lento y no suelen extenderse a otras partes del cuerpo, por otro lado los tumores *malignos*, si representan un riesgo mayor ya que crecen rápidamente y suelen hacer *metástasis*, lo cual significa que migra a otras partes del

cuerpo.

2.2 REDES NEURONALES

Hasta hace algunos años el desarrollo de aplicaciones de software solía realizarse de forma robusta, por ejemplo, si deseáramos desarrollar una aplicación de reconocimiento de imágenes faciales de la forma tradicional sería necesario un grupo de expertos en dicho rubro para que definan una secuencia de transformaciones específicas para lograr esa función en donde la variabilidad de las imágenes que dicho sistema permitiría dependería únicamente de la complejidad del mismo, sin embargo, mediante las redes neuronales es posible resolver el mismo problema con la ventaja de que la variabilidad de imágenes que pueden ser reconocidas por el sistema depende de los datos utilizados para el entrenamiento. El modelo de una red neuronal *aprende* de los datos proporcionados y crea un modelo capaz de clasificar, predecir o reconstruir datos conocidos y desconocidos. Algunos de los puntos clave que conforman un sistema basado en redes neuronales son:

Datos Se debe determinar la naturaleza y la dimensión de los datos que entrarán al modelo.

Modelo Se debe crear un sistema que transforme los datos de entrada, en la salida deseada.

Evaluación El modelo debe tener la capacidad de evaluar su precisión.

Optimización El modelo debe contar con un algoritmo que optimice la precisión.

2.2.1 IMÁGENES COMO DATOS

Las imágenes se pueden definir como una matriz de $m \times n \times 1$ pixeles en el caso de las imágenes de un solo canal (blanco y negro) y en el caso de imágenes a color es de $m \times n \times k$, donde k es igual al número de canales de color que tenga la imagen, siendo tres en el caso de imágenes RGB¹, es importante definir si las imágenes ingresarán al modelo a color o en blanco y negro debido a que esto define el número de nodos de entrada de este, ya que para cada pixel debe corresponder un nodo de entrada y en el caso de imágenes a color también deberá tener un número de capas de nodos de entrada igual al número de canales que tenga la imagen.

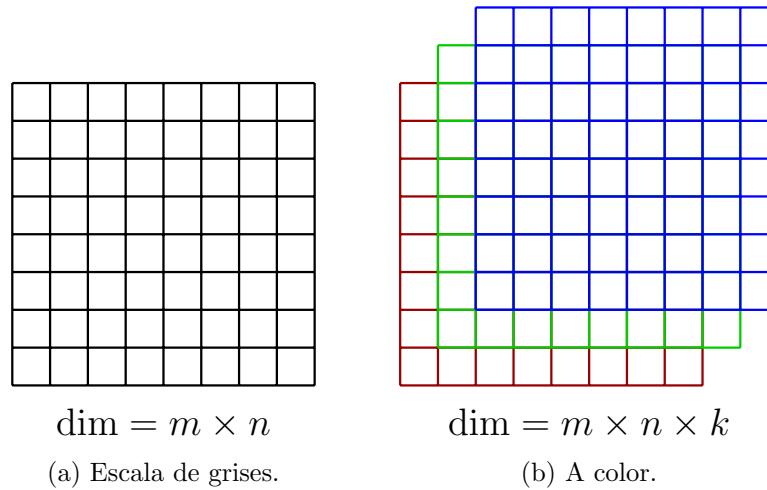


FIGURA 2.1: Comparación de las dimensiones entre los distintos canales de color.

2.2.2 MODELO

Un *modelo* se puede definir como el bloque intermedio entre los datos de entrada (*input*) y los datos de salida (*output*), es el sistema encargado de transformar los datos de entrada en la salida deseada y consta de los siguientes componentes:

¹RGB: Se refiere a los tres canales de color rojo, verde y azul, por sus siglas en inglés (*red, green, blue*).

La unidad principal que compone el modelo de la red neuronal es el perceptrón, denominado *nodo* en esta literatura. Se trata de una analogía matemática basada en el comportamiento de la neurona biológica, el nodo recibe n entradas, las cuales son multiplicadas por el peso p , se suma el producto de las entradas por sus pesos $e \times p$ y finalmente se multiplica por la función de activación ϕ .

Partiendo de lo anterior, creando arreglos con estos nodos podemos formar estructuras denominadas *redes neuronales*, las cuales cuentan con los siguientes elementos estructurales:

Capa de entrada (*input layers*) Se trata de la primera capa del modelo, en el caso de imágenes a cada pixel le corresponde un nodo de entrada. Estos nodos deben tener las mismas dimensiones que las imágenes de entrada.

Capa oculta (*hidden layer*) Las dimensiones de estos nodos pueden ser diferentes a los de entrada y tener varias capas ocultas, sin embargo esto puede afectar la complejidad y precisión del modelo.

Capa de salida (*output layer*) Esta es la capa de salida del modelo, las dimensiones de la capa de salida determinan las dimensiones del dato resultante

Función de activación (*activation*) Se trata de una función dentro de cada nodo que interactúa con el valor entrante y se multiplica por el peso.

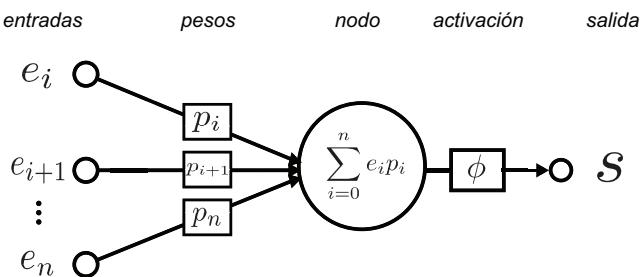


FIGURA 2.2: Ejemplo de un perceptrón.

Pesos (*weight*) Se trata de un valor entre el nodo de la capa actual y el nodo siguiente inicializado de forma aleatoria y después ajustado por un algoritmo para aproximar la salida al valor real.

Sesgo (*bias*) Se trata de un valor constante que se suma al resultado de la función de activación y el peso, de esta forma se puede ajustar que tan fácil o difícil es activar un nodo.

2.2.3 EVALUACIÓN Y OPTIMIZACIÓN

Para realizar el proceso de *aprendizaje* del modelo, primero se debe evaluar cuál es el estado actual de las predicciones. Para esto es necesario evaluar que distancia existe entre el valor predicho y el valor real, esto se puede lograr mediante una *función de pérdida* que se encargue de determinar que tanta diferencia (*loss*) existe en las estimaciones del modelo con los pesos actuales.

Un ejemplo de la función de pérdida es el de la función logarítmica del costo (*log-likelihood*), como se muestra a continuación

$$l(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=1}^q (y_j \log \hat{y}_j), \quad (2.1)$$

q representa el número de clases entre las cuales predecir, y_j representa el valor de salida real, \hat{y}_j el valor estimado de salida por el modelo y j la posición indizada de clase.

Ya que se tiene calculada la pérdida del modelo en su configuración de pesos actual, es necesario actualizar el valor de todos los pesos dentro del modelo para poder acercarnos al valor real. Aquí es donde participa el algoritmo de *gradiente*

descendiente [1] que minimiza una medida de error definido como

$$\text{error} = \frac{\partial}{\partial m} = \frac{2}{N} \sum_{i=1}^N -X_i(y_i - (mX_i + b)), \quad (2.2)$$

$$\frac{\partial}{\partial b} = \frac{2}{N} \sum_{i=1}^N (y_i - (mX_i + b)), \quad (2.3)$$

donde m y b son valores enteros que representan el peso de dos parámetros de un modelo, los cuales serán optimizados para acercar la salida del modelo a la salida real.

2.3 RETRO-PROPAGACIÓN

CAPÍTULO 3

ESTADO DEL ARTE

En este capítulo se estudia las literaturas relacionadas con el presente trabajo de tesis con el objetivo de hacer una comparativa entre distintos métodos para resolver el mismo problema, o implementaciones similares para resolver problemas distintos.

En la primera sección, *trabajos similares*, se recopilan trabajos con características relacionadas al presente trabajo de tesis, ya sea relacionados con el método o con el problema que se pretende resolver, se describe el tema que abarcan y los puntos que lo relacionan con el trabajo aquí presente.

En la segunda sección, *análisis comparativo*, se comparan las distintas características de los trabajos revisados, de esta forma podemos determinar las principales diferencias así como las ventajas y desventajas de cada trabajo. Así mismo se recopilarán todas estas diferencias en una tabla para visualizar mejor estas características.

Finalmente en las *áreas de oportunidad*, se realiza una conclusión acerca de los resultados en la tabla comparativa.

3.1 TRABAJOS SIMILARES

A continuación se mencionan los trabajos revisados en los que se puede obtener un punto de convergencia ya sea en los métodos similares utilizados para resolver problemas diferentes o en su defecto métodos distintos para resolver problemas similares.

Badrinarayanan *et al.* [2] describen la arquitectura denominada **SegNet** utilizada principalmente para la conducción autónoma y la detección de peatones, se describe como una arquitectura convolucional, con la característica de que es una jerarquía de codificador-decodificador donde a cada codificación (*encoding*) corresponde de una capa de agrupación de máximos (*max pooling*) y un decodificador. Inspirado principalmente en la arquitectura **VGG16**, con la diferencia de que el componente clave en esta arquitectura es el uso de convoluciones en lugar de la capa completamente conectada de nodos (*fully-connected-layer*) a la salida.

Ronneberger *et al.* [15] proponen uno de los primeros modelos de las redes neuronales convolucionales denominada **U-net** dirigida al análisis de imágenes del área médica. Este trabajo consiste en una arquitectura con dos trayectorias: la trayectoria de contracción de la imagen y la trayectoria de expansión. En la trayectoria de contracción de la imagen se aplica una secuencia de convoluciones de dimension 3×3 con solapado, seguido de una función de activación de rectificación linear unitaria (ReLU) y posteriormente una operación de agrupación de máximos (*max pooling*) para la compresión de la imagen (*downsampling*). Por otro lado, la trayectoria de expansión consiste de el escalado de la imagen (*upsampling*), en cada paso se aplica una capa de convolución de 2×2 a la cuál se concatena una imagen recortada del mapa de características de la trayectoria de contracción y dos convoluciones de 3×3 seguidos de una activación ReLu cada una. En total la arquitectura tiene 23

convoluciones (**U-net**).

Chen *et al.* [5] describen la arquitectura denominada Deep Lab, la cual propone un acercamiento distinto a la parte de la reconstrucción en las capas finales del modelo. Al igual que otras arquitecturas cuenta con capas de convolución, agrupación de máximos y activación de rectificación linear unitaria, sin embargo, el componente clave aquí es la denominada convolución de hoyos (*atrous convolution*), la cual es una modificación al filtro de convolución basado en la transformada de *Wavelet* para el escalado del mapa de características (*upsampling*). De esta forma se obtiene una alternativa al uso de capas de deconvolución y así se aumenta la resolución del mapa de características a la salida del modelo sin aumentar el tiempo de computación o la cantidad de parámetros.

Teichmann *et al.* [18] reducen la carga computacional en los modelos de redes neuronales convolucionales mediante la unificación de las tres tareas principales de estas: clasificación, detección y segmentación. Denominada como multi-red neuronal MultiNet se trata de una arquitectura tipo codificador-decodificador en donde el decodificador esta compuesto de tres ramas las cuales corresponden a las tareas de clasificación, detección y segmentación por lo que se realizan las tres tareas de forma simultanea partiendo de la misma entrada (*multi-tasking*) y usando la salida de cada una de las ramas para el ajuste fino de los parámetros del modelo (*fine-tuning*).

Kroner *et al.* [10] describen un modelo de codificación-decodificación contextual basado en el proceso en el que los humanos obtenemos la información espacial de los escenarios visuales complejos, mediante un mapa topográfico de las salientes. Se basa en más en el proceso biológico con el que se obtiene la información espacial que en los métodos matemáticos para calcularla, dicho lo anterior, los principales componentes para determinar una saliente son color, la intensidad y la orientación.

Kadampur y Al Riyae [9] proponen el uso de los servicios de nube en conjunto con las redes neuronales profundas y el diseño de modelos utilizando modelos pre-entrenados. Para esto utilizan la herramienta de Deep Learning Studio, la cual es un software que permite el uso de tarjetas gráficas (*GPU*) localizadas en la nube o localmente, y también permite el uso de múltiples tarjetas de forma simultanea (*multi-GPU*), la arquitectura usada para la detección de cáncer de piel en este trabajo es de una red neuronal profunda de clasificación, por lo que el modelo se codifica mediante convoluciones y se decodifica mediante capas de aplanamiento (*flatten layer*) para posteriormente entrar a una capa de nodos completamente conectada (*fully-connected-layer*) y obtener dos posibles resultados: es cáncer o no es cáncer.

Zhou *et al.* [24] hablan sobre el aprendizaje de máquina Machine Learning y las redes neuronales para la predicción teórica de nano-materiales. Primero determinaron el número de capas y la heterogeneidad del material para posteriormente entender los fenómenos visuales como el parpadeo y la emisión cuántica, concluyendo que las intensidades RGB están correlacionados al número de capas en el material del grafeno.

Luc *et al.* [12] proponen una alternativa al proceso de decodificación. Normalmente en las tareas de segmentación es necesaria la parte de codificación para realizar las predicciones a grado pixel y la decodificación para reconstruir la imagen con las categorías segmentadas. Este trabajo propone el uso de una red generativa adversaria GAN, para utilizarse como decodificador, de esta manera se obtiene un mapa de etiquetas (*map layer*) con una mayor resolución.

Jain *et al.* [8] desarrollan una herramienta de procesamiento de imágenes médicas relacionadas al melanoma, basándose en sus características físicas como: asimetría, color, bordes y diámetro. Se basa en la técnica de segmentación de imágenes mediante el uso de filtros de contraste, detección de ejes, etc. A diferencia de los otros

trabajos mencionados, este no consiste del uso de redes neuronales sino de filtrados de imagen mediante distintos kernels de convoluciones para obtener la máscara.

3.2 ANÁLISIS COMPARATIVO

En esta sección se evalúan los trabajos revisados con el trabajo propuesto en este trabajo de tesis, para esto se define una serie de características que servirán como puntos de referencia entre el trabajo propuesto y los trabajos relacionados, dichas características son las siguientes:

Modelo Tipo de arquitectura del modelo.

Clasificación El sistema puede clasificar entre distintas categorías.

Segmentación El sistema puede segmentar las imágenes.

Supervisado El sistema requiere de datos de datos de entrenamiento.

Pre-entrenamiento El sistema puede inicializarse con pesos pre-entrenados como alternativa a la inicialización con pesos aleatorios.

Evaluación El sistema cuenta con una función de evaluación y un algoritmo de optimización.

Salida Características de los datos obtenidos en la salida del modelo.

3.3 ÁREAS DE OPORTUNIDAD

En esta sección se señalan las características de los trabajos mencionados en la sección anterior y se compara con las características del método propuesto en este trabajo de tesis para obtener una comparativa sobre las áreas de oportunidad.

Para las tareas de segmentación es fundamental contar con un codificador y un decodificador eficientes, sin embargo aún más importante es tener la capacidad de evaluarse así mismo y optimizarse, a diferencia de algunas técnicas más robustas de diseño, las redes neuronales convolucionales permiten esas funcionalidades. Otro aspecto importante a considerar es la capacidad del uso de modelos pre-entrenados y re-entrenarlos utilizando datos similares, ya que incluso cuando los pesos pre-entrenados fueron obtenidos mediante bases de datos no necesariamente relacionadas con el o los objetos que nosotros queremos detectar, pero con similitudes físicas, en

CUADRO 3.1: Similitudes y diferencias entre los trabajos revisados; las características implementadas se representan con **✓**, mientras que las no implementadas con **✗**.

Trabajo	Modelo	Clasificación	Segmentación	Supervisado	Pre-entrenamiento	Evaluación	Salida
Badrinarayanan <i>et al.</i> [2]	SegNet	✓	✓	✓	✓	✓	mapa de etiquetas
Ronneberger <i>et al.</i> [15]	U-net	✓	✓	✓	✗	✓	mapa de etiquetas
Chen <i>et al.</i> [5]	DeepLab	✓	✓	✓	✗	✓	mapa de etiquetas
Teichmann <i>et al.</i> [18]	MultiNet	✓	✓	✓	✗	✓	mapa de etiquetas
Kroner <i>et al.</i> [10]	VGG16	✓	✓	✓	✓	✓	mapa de calor
Kadampur y Al Riyae [9]	CNN	✓	✗	✓	✗	✓	mapa de etiquetas
Zhou <i>et al.</i> [24]	ML / SVM	✓	✗	✓	✗	✓	etiqueta
Luc <i>et al.</i> [12]	CNN/GAN	✓	✓	✓	✓	✓	mapa de etiquetas
Jain <i>et al.</i> [8]	A.B.C.D	✗	✓	✗	✗	✗	etiqueta
Propuesta de tesis	FPN	✓	✓	✓	✓	✓	mapa de etiquetas

teoría ofrece un mejor desempeño y reduce el tiempo de entrenamiento ya que de lo contrario, se tendría que inicializar el modelo con pesos distribuidos de forma aleatoria, por lo que la precisión inicial del modelo sería cercana al cero por ciento. En este trabajo de tesis, el modelo propuesto cuenta con la capacidad de utilizar pesos pre-entrenados, el reto está en seleccionar un conjunto de pesos que mejoren la precisión en el caso de imágenes médicas, ya que si los pesos utilizados fueron obtenidos con imágenes que difieren mucho de las imágenes que se utilizaras en el presente trabajo, podría verse comprometida la precisión o el tiempo de entrenamiento.

CAPÍTULO 4

SOLUCIÓN PROPUESTA

El objetivo de este trabajo de tesis es implementar una técnica para la clasificación y segmentación de regiones de tumores del tipo melanoma en imágenes dermatológicas mediante el uso del aprendizaje profundo (*deep learning*). Para comprobar la hipótesis se desarrolló una aplicación de software cuya función es la de funcionar como entorno de trabajo mediante el cual se puedan procesar datos para el entrenamiento de modelos, trabajar con distintas arquitecturas de redes, así como modificar los parámetros del entrenamiento para hacer el ajuste fino del modelo.

En este capítulo se describe la estructura de la implementación propuesta, comenzando con la *metodología*, en donde se describe el conjunto de procedimientos por los cuales debe atravesar los datos desde su entrada como una imagen ya sea de un canal o triple canal, hasta su salida como máscara de segmentación con sus respectivas regiones clasificadas a su categoría correspondiente. Posteriormente en *implementación de la solución* se describe el lenguaje utilizado para la implementación, las librerías utilizadas; así como el análisis llevado a cabo para determinar cuál configuración de parámetros ofrece el mejor resultado y una comparativa o *benchmarking*.

4.1 METODOLOGÍA

La herramienta propuesta cuenta con tres fases principales: cargado y filtrado de datos, entrenamiento y predicción, la fase de *cargado de datos* consiste en adquirir y corregir las imágenes que serán usadas en las fases siguientes, el *entrenamiento* trata de la obtención de un modelo a partir de dos datos: la imagen real (entrada) y la máscara conocida (salida deseada) que corresponde a la región a clasificar, el proceso de entrenamiento consta de un número determinado de ciclos también denominados épocas (*epochs*) en dónde se prueban distintas configuraciones y se conserva la configuración con el mayor porcentaje de precisión. Al finalizar el entrenamiento se obtiene un modelo mediante el cuál se realiza la segunda fase, *predicción*, ahora únicamente se requiere un dato de entrada el cuál es la imagen de la cual se desconoce la clasificación de sus regiones, y como resultado se obtiene una estimación de la máscara de segmentación desconocida con sus regiones correctamente clasificadas.

4.1.1 CARACTERÍSTICAS DE LOS DATOS DE ENTRADA

Para poder llevar a cabo la creación del modelo primero es necesario tener un conjunto de datos (*dataset*) que será utilizado para el entrenamiento y la validación de la configuración de pesos sinápticos en la época presente. A partir de éste momento se referirá como *muestra* al conjunto de datos de entrada del entrenamiento del modelo; como se mencionó al comienzo del capítulo, para llevar a cabo el entre-

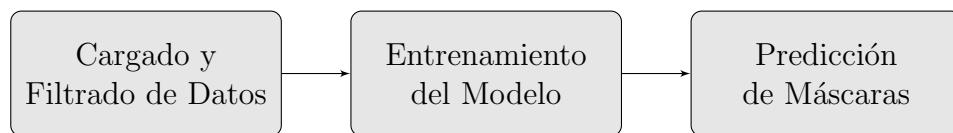


FIGURA 4.1: Diagrama de flujo general de la herramienta propuesta.

namiento es necesario otorgar al sistema dos entradas: la imagen real y la máscara conocida de las regiones ya clasificadas. Por lo tanto una muestra es el equivalente a la concatenación de una imagen dermatológica y la máscara conocida correspondiente. Con una sola muestra se puede generar un modelo sin embargo la precisión sería muy baja, generalmente entre mayor sea la cantidad de muestras se puede obtener una mejor precisión. No obstante depende mucho de la variabilidad de las imágenes, ya que si se tratara de entrenar un modelo de muchas categorías, la variabilidad haría mas complejo el modelo y por ende requeriría de un mayor número de épocas para alcanzar una precisión viable (*underfit*); así como el uso de un *dataset* muy extenso de pocas categorías y con imágenes muy similares podría causar justamente lo contrario (*overfit*), lo cuál no permitiría realizar predicciones correctamente en imágenes no tan parecidas a las usadas en el entrenamiento.

4.1.2 CODIFICACIÓN DE CARACTERÍSTICAS

Para poder realizar una predicción de la máscara segmentada, primero es necesario reducir las dimensiones de las imágenes sin afectar la información que estas representan. Para conseguir esto se utiliza un filtro de *convolución de matrices*. Una imagen es una función bidimensional $z = f(x, y)$, donde x y y son coordenadas espaciales y z es el valor de la intensidad de la imagen en el punto (x, y) , en el caso de las imágenes a color se tienen tres funciones donde cada una representa la intensidad del pixel del canal correspondiente (rojo, verde o azul) [13, p. 100]. La convolución de matrices se define como la obtención de una matriz C a partir de dos matrices A y B . La matriz $A_{m \times n}$ es el mapa de intensidad de pixeles de la imagen mientras que la matriz $B_{(2N+1) \times (2N+1)}$ es el núcleo de la convolución, mediante la ecuación 4.1 se

obtiene la matriz $C = A * B$

$$c_{ij} = \frac{1}{b} \sum_{r=1}^{2N+1} \sum_{s=1}^{2N+1} (a_i - N + r - 1, i - N + r - 1), \quad (4.1)$$

donde $b = \sum_{i,j=1}^{2N+1} b_{i,j}$, si $b = 0$ se toma $b = 1$.

El filtro de convolución es la base del proceso de codificado de la imagen, la codificación se refiere a la transformación de una imagen a un vector de características de alta dimensión (*downsampling*), una vez que se tiene el mapa de características sigue con el proceso de *upsampling*.

4.1.3 MODELO FPN

La arquitectura de red neuronal tipo red piramidal de características, por sus siglas en inglés FPN¹, se trata de una estructura que funciona en conjunto con el codificador ResNet, mientras el codificador tiene la tarea de realizar el *downsampling* de las dimensiones espaciales de la imagen de entrada y convertirlas a un mapa de característica , el decodificador hace justamente lo opuesto (*up-sampling*), a partir del mapa de características producto de la codificación, realiza una secuencia de deconvoluciones que construyen la máscara de segmentación estimada.

¹FPN: Feature Pyramid Network

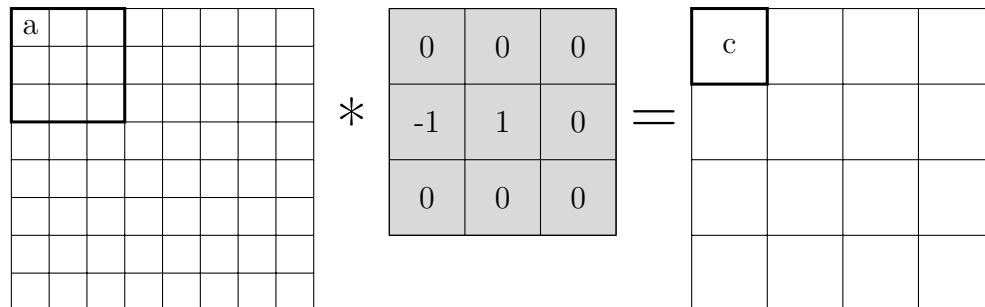


FIGURA 4.2: Representación del proceso de obtención de la matriz convolucionada.

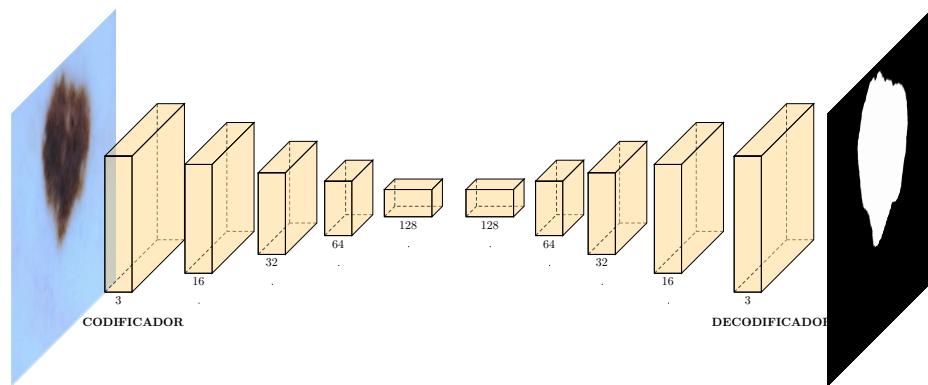


FIGURA 4.3: Representación de la arquitectura convolucional.

En la figura 4.3 se puede apreciar una arquitectura completamente convolucional en la que la imagen entrante pasa por una secuencia de transformaciones que reducen la resolución de la matriz e incrementa el numero de capas. Cuando el dato entrante se reduce, también se reduce la complejidad de las operaciones de clasificación y a partir de ese punto comienza la fase de decodificación, la cual consiste en predecir la máscara de salida en base al dato reducido.

4.2 IMPLEMENTACIÓN DE LA SOLUCIÓN

Para la implementación de la herramienta propuesta, se utilizó **Python** como lenguaje principal. **Python** es un lenguaje de alto nivel con un extenso repositorio de paquetes desarrollados por la comunidad, también es un lenguaje ligero ideal para el procesamiento de datos y de operaciones con datos de altas dimensiones (*high-dimensional data*), debido a esto último, se consideró **Python** la mejor opción para la implementación de la red neuronal profunda.

Si bien otros lenguajes tienen una mayor velocidad en la ejecución de cálculos complejos, es la facilidad con la que se pueden desplegar implementaciones complejas en relativamente pocas líneas de código lo que llevo a la conclusión de que **Python**

es la opción ideal para la implementación propuesta, como el hecho de que existen muchas documentaciones e implementaciones similares en este lenguaje.

A continuación se especifican las librerías claves utilizadas para llevar a cabo la implementación, así mismo se da una breve descripción general sobre la misma y su importancia durante el desarrollo de la propuesta.

CUADRO 4.1: Librerías utilizadas para la implementación de la propuesta.

Librería	Descripción
Torch.vision	Este paquete es una colección de módulos relacionados con las operaciones de tensores y la aceleración de cálculos mediante hardware. Se usa en conjunto con NumPY y cuenta con las arquitecturas de redes neuronales típicas, así como herramientas para el cargado de datos y la configuración de parámetros de los modelos.
Quvel Segmentation Models	Este paquete cuenta las arquitecturas de las redes neuronales de segmentación semántica típicas, se trata del marco de trabajo mediante el cuál podemos crear y configurar la arquitectura de la red para después realizar el entrenamiento.
Numpy	Esta librería fue desarrollada para habilitar el uso de vectores y matrices de grandes dimensiones y para la computación numérica en general, ya que Python originalmente no cuenta con el soporte para esto.
OpenCV	Es una librería dedicada al procesamiento de imágenes y a la visión computacional, cuenta con funciones que permiten cargar imágenes y obtener su matriz de pixeles así como aplicar efectos y transformaciones en estas.
Albumentations	Esta librería es principalmente útil para el pre-procesado de las imágenes al momento de realizar el entrenamiento, cuenta con funciones que permiten crear secuencias de transformaciones específicas para distintos subconjuntos de datos así como la capacidad de transformar imágenes a tensores.

En el cuadro 4.2 se especifican las características del equipo en el cuál fue realizado el experimento.

CUADRO 4.2: Especificaciones técnicas de los componentes.

Componente	Descripción
Procesador	AMD Ryzen 5 3600x, 3.80 GHz, de seis núcleos.
Tarjeta Gráfica	Nvidia GeForce 1050ti, 4 GB.
RAM	HyperX Fury, 16 GB, 2433 MHz.
Tarjeta Madre	Gigabyte Gaming AB-350.
Sistema Operativo	Ubuntu 20.4 64 bits

4.2.1 COMPUTACIÓN ASISTIDA POR HARDWARE

Los cálculos llevados en el proceso de entrenamiento involucran realizar operaciones con datos de altas dimensiones (*high-dimensional data*) los cuales requieren una gran cantidad de recursos de procesamiento, estas operaciones al estar relacionadas con vectores y matrices pueden ser ejecutadas rápidamente por la unidad GPU², entre mayor sea la velocidad de procesamiento de la GPU más rápido es el entrenamiento de nuevos modelos y con mayor resolución.

4.2.2 MÓDULOS DE LA IMPLEMENTACIÓN

La herramienta desarrollada en el presente trabajo de tesis se divide en cuatro módulos funcionales y un módulo central encargado de controlar la ejecución de los mismos. En la figura 4.4 se puede observar la estructura modular del software desarrollado, dentro de dichos módulos se encuentran las funciones específicas de esa categoría y en el módulo principal se encuentran los parámetros del modelo.

²GPU: Graphics Processor Unit

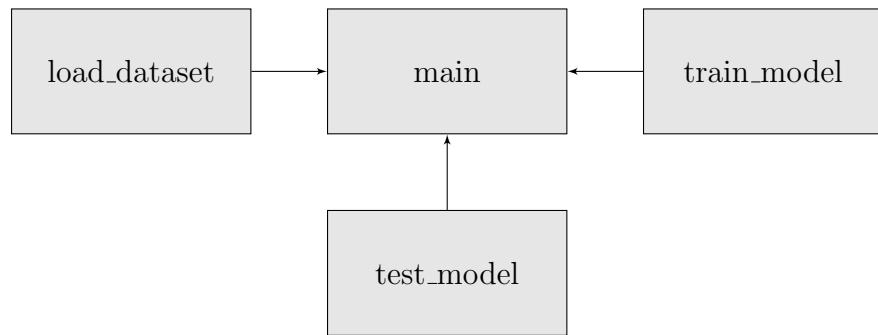


FIGURA 4.4: Distribución de los módulos en la herramienta desarrollada.

4.2.3 LECTURA DE ENTRADA

El proceso de cargado de datos y filtrado es un proceso muy importante para la ejecución correcta en las siguientes transformaciones, para el entrenamiento del modelo se utilizará una base de datos obtenida de la Asociación de Recolección de Imágenes dermatológicas, por sus siglas en inglés (*ISIC*³). La base de datos de ISIC cuenta con diez mil imágenes de entrenamiento con resoluciones distintas, con diferentes tonos de iluminación y rotación, así como su máscara de segmentación conocida y dos mil imágenes para realizar pruebas, también con resoluciones no estandarizadas y sin máscara de segmentación.

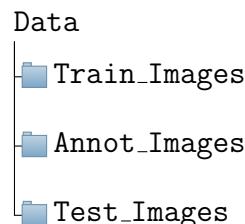


FIGURA 4.5: Representación del árbol de directorios de imágenes.

Lo primero que se desarrolló fue un asistente de cargado y filtrado de imágenes, en el código 4.1 se puede apreciar el código fuente y las funciones utilizadas para

³ISIC: International Skin Imaging Collaboration.

tal motivo. Al ser utilizado el objeto *dataset* y alimentar los argumentos con los directorios de las imágenes se obtiene como resultado un objeto de dimensión $N \times 2 \times (m \times n)$ donde N es el número de muestras cargadas y el primer elemento de la fila es la matriz de la imagen real, y el segundo la matriz de la máscara conocida.

```

1  # load_dataset.py
2  class dataset():
3      def __init__(self, images_dir, masks_dir, classes=None,):
4          # Extraccion de ids de imagenes y mascaras.
5          self.ids = listdir(images_dir)
6          self.images_fps = [os.path.join(images_dir,img_id)
7                             for img_id in self.ids]
8          self.masks_fps = [os.path.join(mask_dir,
9                                         (img_id[:-4] + '_segmentation.png'))
10                         for img_id in self.ids]
11      # Lectura de imagenes y correccion de canales.
12      def __getitem__(self,i):
13          image = cv2.imread(self.images_fps[i])
14          image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
15          mask = cv2.imread(self.masks_fps[i],0)
16          return image, mask

```

Código 4.1: Código fuente del asistente de cargado de imágenes.

```

1  # main.py.
2  import load_dataset as ld
3  train_dir = 'Data/Train_Images'
4  mask_dir = 'Data/Train_Images'
5  set_de_datos = ld.dataset(train_dir, mask_dir)
6  # Al indicar la posicion dentro del dataset se obtiene una muestra dividida en
7  # dos objetos:
8  imagen, mascara = set_de_datos[1]

```

Código 4.2: Invocación de la función contenida dentro de la clase.

En el código 4.2 se invoca la función *dataset*, la cuál tiene por argumento la dirección de las imágenes reales y las máscaras conocidas, con esto se crea el conjunto de datos que posteriormente es utilizado en la sección de entrenamiento.

4.2.4 PRE-PROCESAMIENTO

El *pre-procesamiento* se define como una secuencia de transformaciones que se aplican al momento previo del entrenamiento y afecta a un subconjunto específico de los datos. Para el diseño de la secuencia de transformaciones tanto para el subconjunto de entrenamiento como el subconjunto fueron consideradas las siguientes características.

Resolución El codificador admite resoluciones específicas, debido a las dimensiones admitidas para el filtrado de convolución de matrices.

Número de muestras La cantidad de datos disponibles para el entrenamiento afecta la precisión final, si el conjunto de datos es limitado se pueden realizar transformaciones de las imágenes presentes para obtener más datos de entrenamiento. Así como tener un conjunto muy extenso y con poca variabilidad afectaría también la capacidad de predecir imágenes generalizadas, se puede ajustar mediante el pre-procesamiento.

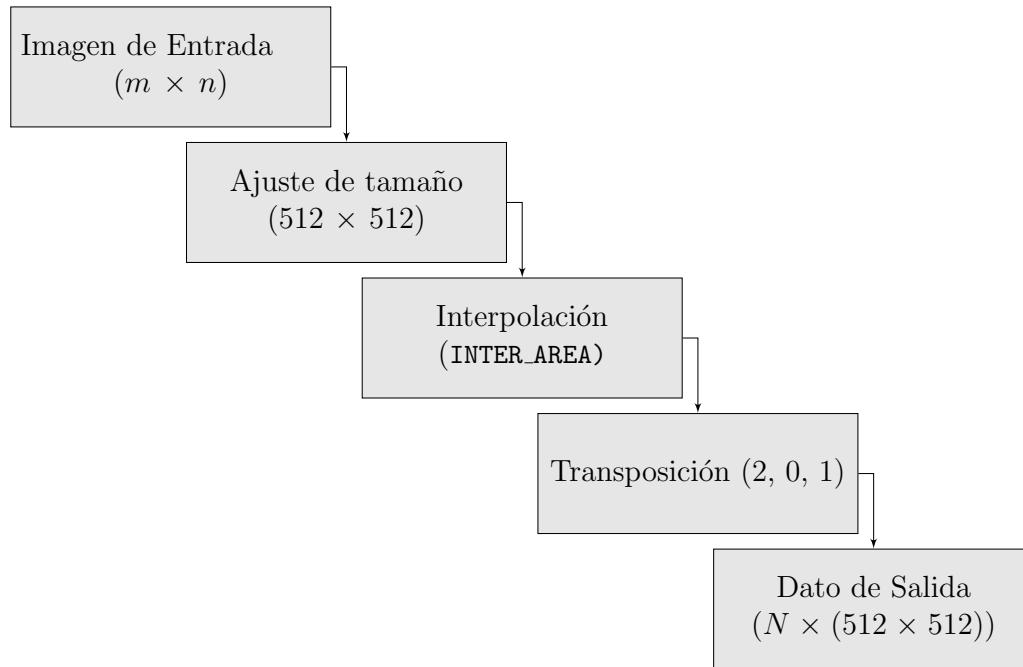


FIGURA 4.6: Línea de pre-procesamiento.

Debido a que el conjunto de datos es extenso y solo clasifica entre dos categorías, no es necesario realizar demasiadas transformaciones, no obstante es importante trasponer la matriz de entrada para convertir la matriz a tensor.

4.2.5 ENTRENAMIENTO Y VALIDACIÓN

La secuencia de entrenamiento consiste de un bucle en el cuál se condensan las imágenes en un grupo (*batch*), para ser procesadas por el codificador de forma simultanea, esto para acelerar la velocidad de entrenamiento. Sin embargo, a mayor tamaño de bache también se requiere mayor memoria disponible por la unidad procesadora de gráficos y también la precisión del entrenamiento se ve influenciado. Posteriormente se evalúa mediante el *índice de Jaccard*, como se muestra en la ecuación 5.2.

```
1 # train_model.py
```

```

2     max_score = 0
3
4     for i in range(0,40):
5         print('\n Epoch: {}'.format(i))
6         train_logs = train_epoch.run(train_loader)
7         valid_logs = valid_epoch.run(valid_loader)
8         if max_score < valid_logs['iou_score']:
9             max_score = valid_logs['iou_score']
10            torch.save(model,('Model/+' + encoder + '.pth'))
11            print('Highest Score Model Saved: {}'.format(max_score))
12
13            if i == 25:
14                optimizer.param_groups[0]['lr'] = 1e-5
15                print('decreased decoder learning rate to 1e-5')
16                lr= 0.0001 ,])

```

Código 4.3: Búcle de entrenamiento.

Para poder determinar la eficiencia durante el entrenamiento y optimizar los pesos sinápticos es necesario hacer uso de métricas, dichas métricas se aplicarán sobre el subconjunto de validación. Los datos utilizados fueron divididos en una proporción de 70 % para entrenamiento y 30 % para la validación, el subconjunto de validación no es usado durante el entrenamiento, sino qué, al terminar una época se verifica con ese subconjunto la precisión.

La librería de Yakubovskiy [23] cuenta con distintas métricas ya implementadas dentro de su API⁴, dichas métricas se pueden configurar previamente al entrenamiento. En el código 4.4 se puede apreciar la selección de la función de métrica, la función de pérdida y el optimizador de pesos del modelo.

La función de pérdida es la función mediante la cuál se evalúa el error durante el entrenamiento, mediante el dato obtenido del error y la función de optimización seleccionada se puede minimizar dicho error.

La métrica, es la función mediante la cuál se evalúa el modelo post-entrenamiento con una función que penaliza más el error que la ecuación utilizada para evaluar el

⁴Interfaz de programación por sus siglas en inglés, *Application Programming Interface*.

error durante el entrenamiento.

El optimizador, utilizando como referencia el valor de pérdida, determina si es necesario aumentar o disminuir el peso de los nodos entre convoluciones para reducir la pérdida y aumentar la precisión.

```

1 # main.py.
2
3 import segmentation_models_pytorch as smp
4
5 # Funcion de perdida (Dice Loss)
6 loss = smp.utils.losses.DiceLoss()
7
8 # Metrica de evaluacion (Jaccard Index)
9 metric = [smp.utils.metrics.IoU(threshold=0.5),]
10
11 # Funcion optimizadora (Adam)
12 optimizer = torch.optim.Adam([dict(params=model.parameters(),
13                                lr= 0.0001),])

```

Código 4.4: Ejemplo de configuración de parámetros.

4.2.6 PREDICCIÓN DE MÁSCARAS

Una vez obtenido un modelo con precisión buena (*good fit*), el objetivo es generar máscaras desconocidas a partir de las imágenes del set de pruebas. Para esto es necesario popular de nuevo un *dataset* de dimensión $N \times (m \times n)$, básicamente es un vector de matrices donde N es el número de imágenes para realizar la prueba, y $m \times n$ es la resolución de imagen de la cuál se desea obtener la máscara de segmentación.

```

1 # test_model.py
2
3 def test_model(m_path, t_path, encoder, weights, classes, device):
4     # Cargar el modelo generado
5     model = torch.load(m_path)
6
7     prep_fn = smp.encoders.get_preprocessing_fn(encoder, weights)
8
9     # Crear el subconjunto de datos para pruebas
10    vis_dataset = ds.testing_data(t_path,
11                                classes, augmentation=tfm.get_validation_augmentation())
12
13    test_dataset = ds.testing_data(t_path,
14                                classes,
15                                augmentation=tfm.get_validation_augmentation())
16
17    return vis_dataset, test_dataset

```

```

11     augmentation=tfm.get_validation_augmentation(), preprocessing=tfm.
12         get_preprocessing(prep_fn)
13
14     # Seleccionar una muestra al azar.
15
16     n = np.random.choice(len(vis_dataset))
17     vis = vis_dataset[n].astype('uint8')
18     test_image = test_dataset[n]
19
20     # Realizar una predicción de máscara con la imagen aleatoria.
21     x_tensor = torch.from_numpy(test_image).to(device).unsqueeze(0)
22     pred_mask = model.predict(x_tensor)
23     pred_mask = (pred_mask.squeeze().cpu().numpy().round())
24
25     #Visualizar la imagen real y la máscara predicha.
26
27     ds.visualize(image=vis, predicted=pred_mask)

```

Código 4.5: Código fuente del modulo de pruebas.

```

1 # main.py
2 if test_sample:
3     tsm.test_model(model_path, test_dir, ENCODER, ENCODER_WEIGHTS, CLASSES, DEVICE)

```

Código 4.6: Uso de la función definida en el modulo de pruebas.

En el código 4.5, se define la función que se encarga de cargar el modelo y cargar el subconjunto de datos para pruebas mientras que en el código 4.6 se da un ejemplo del uso de dicha función. Dando como resultado una gráfica con la imagen de entrada y la máscara estimada para tal.

Con esto finalizarían los procesos de cargado, entrenamiento y predicción de la herramienta propuesta, la línea de transformaciones se diseña en base a la cantidad de datos y las características que estas contienen. A sí mismo dichas transformaciones se llevan a cabo para adaptar las dimensiones de la imagen de entrada con el codificador, posteriormente y durante el entrenamiento se evalúa la configuración de los pesos sinápticos mediante el coeficiente de datos y se optimizan dichos pesos mediante el optimizador *adam*, el cual es una alternativa al método de gradiente descendiente y está diseñado para actualizar los pesos sinápticos en elementos de naturaleza iterativa como es el caso de las imágenes.

CAPÍTULO 5

EXPERIMENTOS

En este capítulo se presentan los experimentos llevados a cabo durante el desarrollo de la herramienta propuesta. Las siguientes secciones corresponden a diferentes experimentos llevados a cabo para validar la hipótesis del presente trabajo de tesis, cada experimento contiene tres subsecciones: *diseño experimental*, *resultados* y *discusión*.

En la subsección de *diseño experimental*, se detalla el experimento y el parámetro que se pretende definir en base a este, todos los procesos realizados por la herramienta desarrollada en el presente trabajo de tesis se ejecutan de forma secuencial, debido a esto es necesario determinar el mejor parámetro en cada paso de la transformación.

En *resultados*, se reporta e interpreta todos los resultados obtenidos mediante el experimento llevado a cabo en esa sección, cada experimento cuenta con su propia sección y se determina el mejor parámetro para realizar el experimento siguiente. Para finalizar en *discusión*, se llega una conclusión basada en los resultados obtenidos y la información que representan las gráficas.

5.1 ESTADÍSTICAS DE LOS DATOS DE ENTRADA

Este experimento tiene como objetivo determinar el codificador ideal para la arquitectura implementada basándose en la información estadística del conjunto de datos de las imágenes, también determinar las transformaciones necesarias para coincidir con las dimensiones requeridas por la arquitectura y las ofrecidas por la imagen.

5.1.1 DISEÑO EXPERIMENTAL

Las imágenes representan un mapa de características o de valores numéricos en forma de matriz, dichas características pueden ser analizadas para determinar la secuencia correcta de transformaciones en la línea de pre-procesado para acoplar correctamente las dimesiones de entrada de la imagen con las entradas disponibles del codificador.

Muestras El número de muestras del conjunto de datos puede determinar las transformaciones posteriores tanto para reducir *sobreajuste*, como para mejorar el *subajuste*.

Clases La cantidad de clases entre las que el modelo tiene que clasificar determina la complejidad computacional de éste, por lo tanto también determina transformaciones requeridas.

Valor Umbral Para determinar la complejidad de las imágenes para ser segmentadas, se puede realizar un estudio del umbral para determinar el rango específico de intensidad de pixeles en el que se encuentra el objeto.

5.1.2 RESULTADOS

Los resultados obtenidos de este experimento resultan de la aplicación de filtros de umbralización a un conjunto de datos de entrada para estudiar la forma en la que cambia la región de intensidad en la que se ubica la mayor recurrencia de píxeles.

Posteriormente se realizó un estudio del valor umbral (*thresholding*), para determinar la región característica de la intensidad de los píxeles y utilizar dicho valor dentro de las transformaciones del pre-entrenamiento. Para llevar esto a cabo primero se realizó un histograma de intensidad de píxeles del conjunto de datos entrante, a partir de dicho histograma se determinó el valor umbral y posteriormente se aplicó la función de umbral binario para dar como resultado una discretización de la intensidad de píxeles.

Para este experimento es importante considerar que una intensidad de pixel = 0, en una escala de grises significa que sería completamente negro, mientras que si pixel = 255, significa que el pixel es completamente blanco. Lo mismo aplica a las imágenes a color pero en su correspondiente canal.

Los píxeles de las imágenes de entrada tienen una forma de $m \times n$, por lo que para evaluar la intensidad de todos los píxeles es necesario aplanar la dimensión del arreglo, para esto se utilizó la función `Ravel` de NumPy, la cuál da como resultado un arreglo de una sola dimensión a partir de un arreglo de varias dimensiones o con subarreglos.

CUADRO 5.1: Características del conjunto de datos.

Dato	Valor
Muestras	10,000
Clases	2
Res. Mínima	1,024 px
Res. Máxima	2,550 px

5.1.3 DISCUSIÓN

En la serie de imágenes que se muestran en la sección de resultados se puede observar las muestras que fueron utilizadas para dicho experimento y posteriormente un histograma de densidad. En la figura 5.2 se puede observar que los pixeles se encuentran mayormente distribuidos, esto significa que clasificar dichos valores requeriría mayor complejidad de cálculos. Sin embargo, después de aplicar el filtro

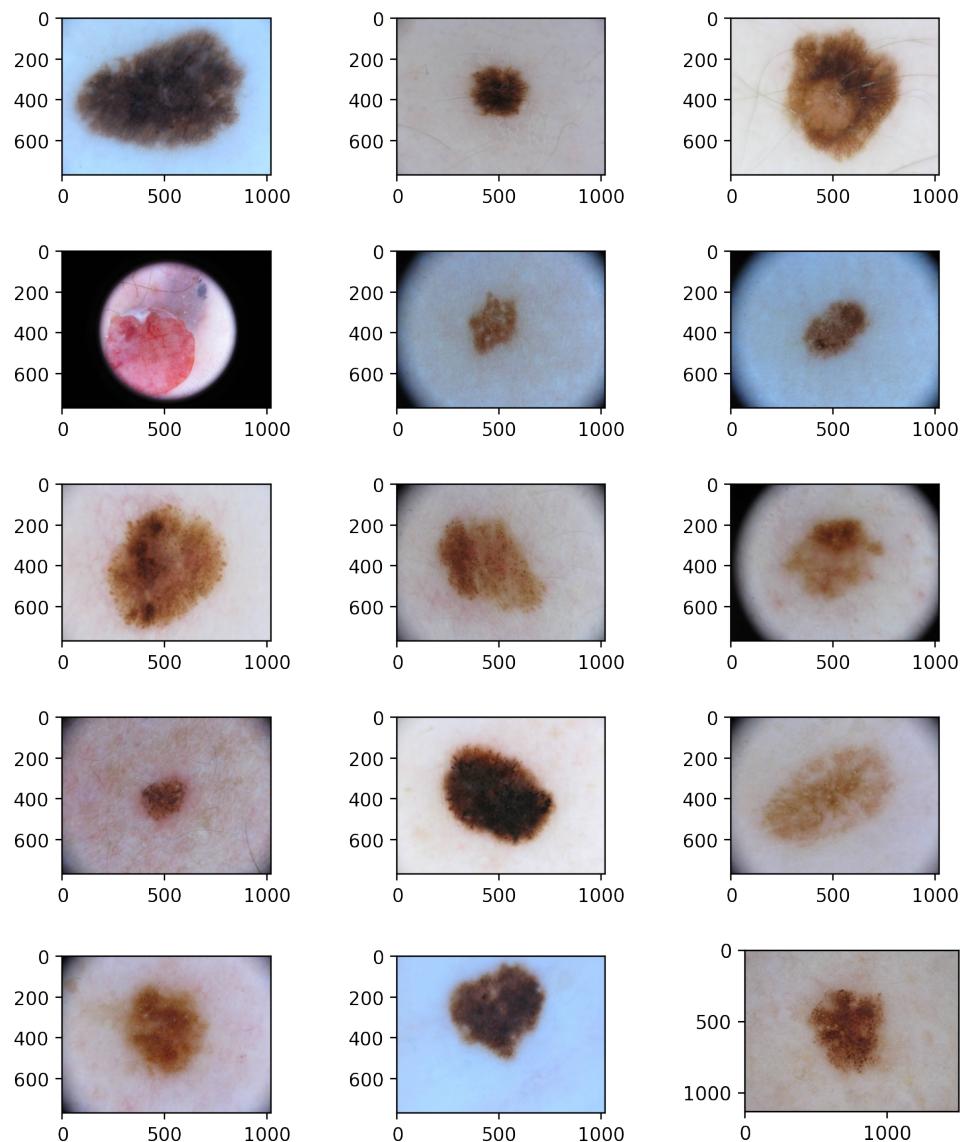


FIGURA 5.1: Conjunto de datos entrante.

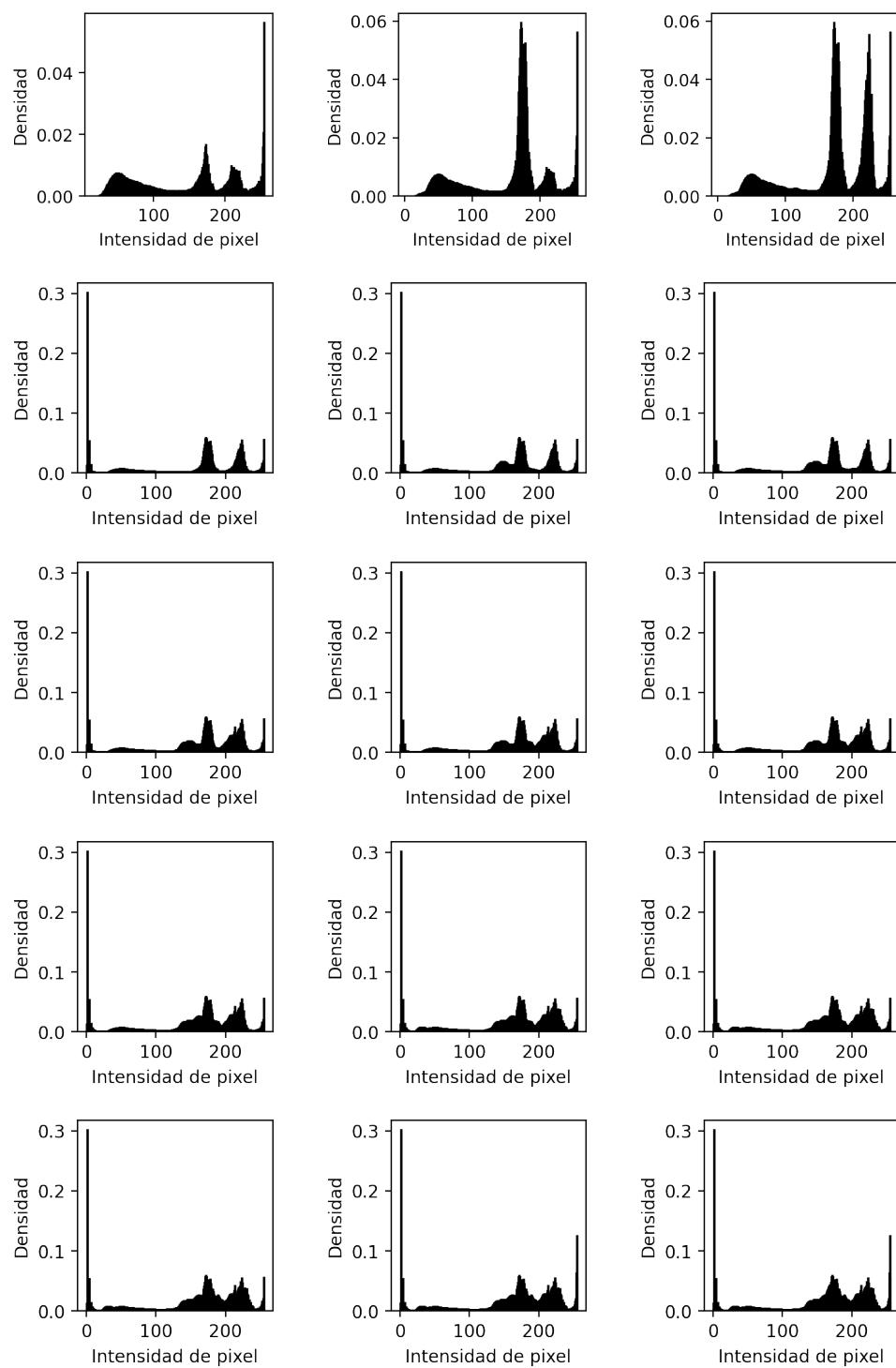


FIGURA 5.2: Histogramas de densidad de la intensidad de píxeles de la imagen entrante.

de umbralización se puede observar en la figura 5.3 que la imagen aún conserva su información relevante y en la figura 5.4 que ahora la intensidad de los píxeles no está tan distribuida, sino qué, ahora se acumulan en dos valores específicos, resultando en un cálculo más eficiente.

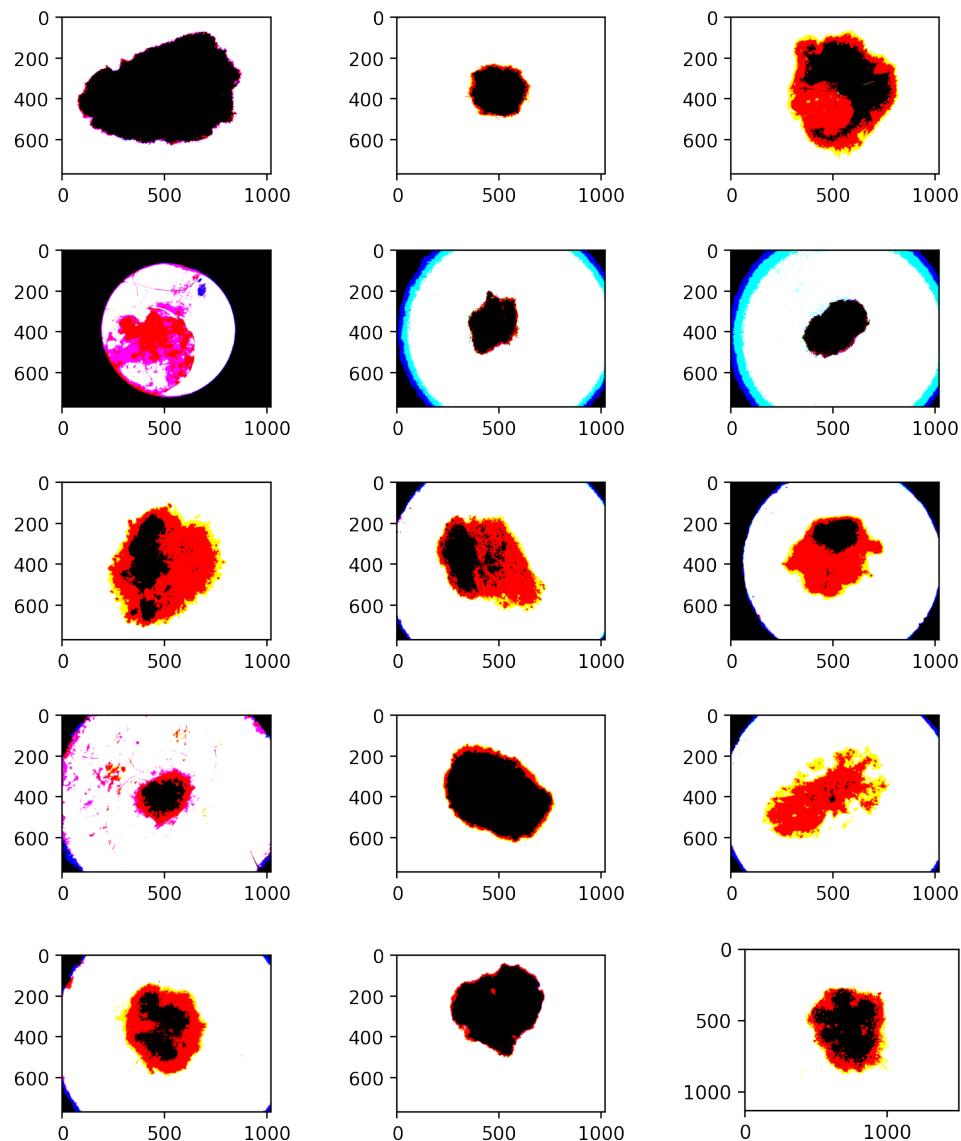


FIGURA 5.3: Conjunto de datos después de aplicar el filtro de umbral.

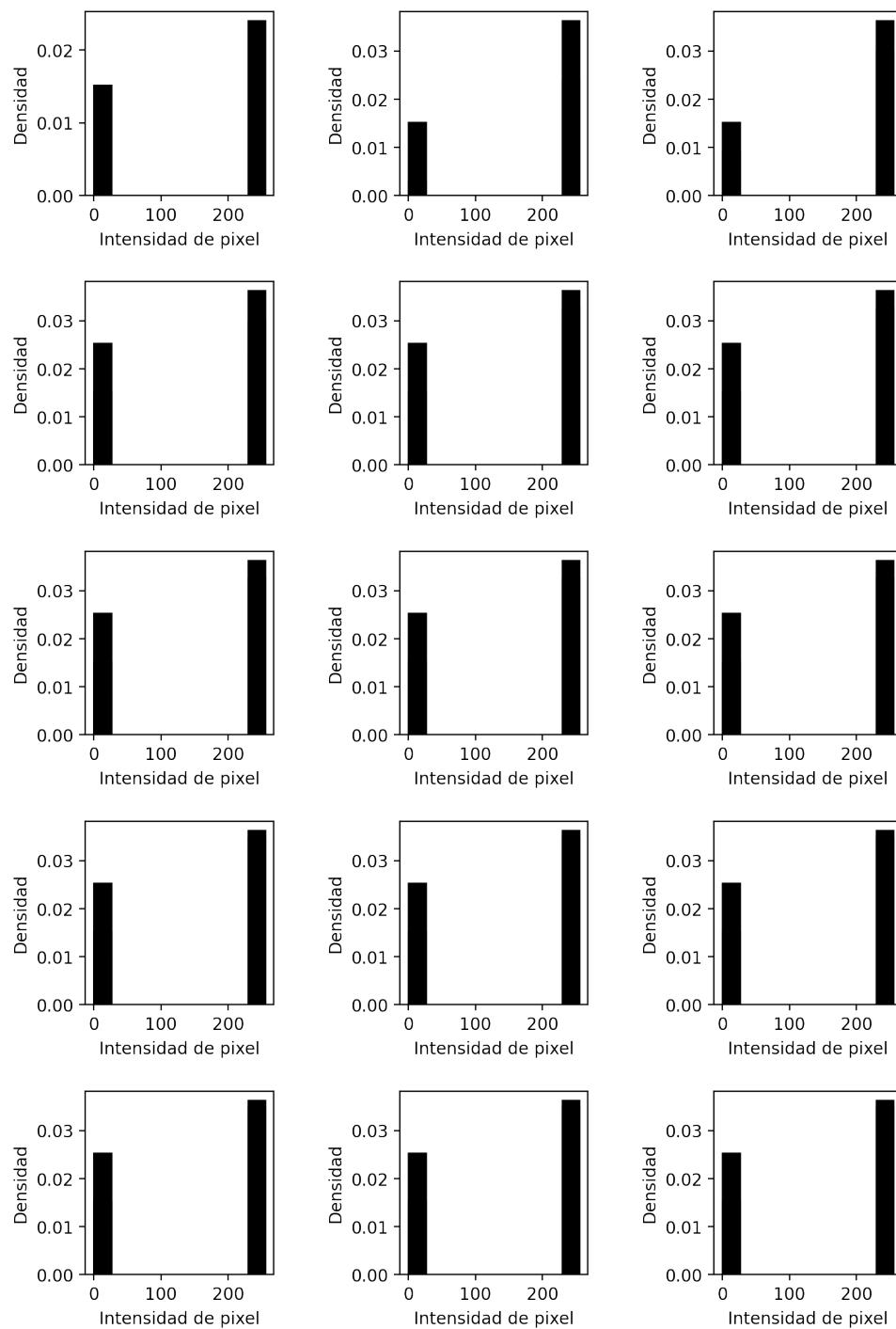


FIGURA 5.4: Histograma de densidad de la intensidad de pixeles después de aplicar el filtro de umbral.

5.2 EVALUACIÓN DEL APRENDIZAJE

Una vez seleccionado el codificador ideal, es necesario evaluar la configuración del modelo en la época presente y configuración actual, al finalizar cada ciclo de entrenamiento se realiza una validación con el porcentaje del conjunto de datos que no fue utilizado durante éste.

5.2.1 DISEÑO EXPERIMENTAL

Para determinar un valor probabilístico entre dos mapas de valores binarios es necesario utilizar un coeficiente que evalúe la similitud entre ambos conjuntos de datos, siendo el conjunto A la máscara real o *ground truth*, y el conjunto B el mapa obtenido por la computación del modelo se utilizó el criterio de dados para determinar la probabilidad, la ecuación del criterio de dados dice que la probabilidad de que dos mapas binarios sean iguales es dos veces la cantidad de píxeles sobrepuertos o idénticos entre la suma de los píxeles de ambos mapas,

$$\text{DC} = \frac{2|A \cap B|}{|A| + |B|} \quad (5.1)$$

donde A es un mapa binario de píxeles que representa la máscara real y B la máscara de segmentación obtenida por el modelo.

5.2.2 RESULTADOS

Durante el entrenamiento del modelo, se evaluó la perdida o el error mediante el coeficiente de datos, dicha evaluación fue realizada en cada iteración y los datos

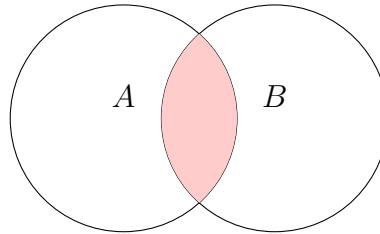


FIGURA 5.5: Representación de la región de sobreposición de píxeles en el coeficiente de datos.

fueron registrados para posteriormente graficar la curva. La curva de la pérdida debe tender a descender cuando el modelo mejora su precisión, al terminar todas las iteraciones se vuelve a pasar el modelo por el proceso de entrenamiento utilizando la configuración del modelo anterior hasta llegar a la mejor precisión posible. En la figura 5.6, se puede observar la curva en cada una de las épocas.

5.2.3 DISCUSIÓN

En la sección de resultados muestra en la figura 5.6 la curva del error según el coeficiente de datos, cada gráfica representa una época y sus iteraciones correspondientes. Como se puede observar, en la primera iteración de la primera época el error es mayor, conforme van pasando las épocas la curva se va aplanando y el error disminuye constantemente. De acuerdo a lo que indican las gráficas es posible obtener un modelo con un error mínimo a partir de la iteración numero cinco, ya que ahí es donde el error ya alcanza el mínimo obtenible y el resto de épocas no cambia significativamente.

El proceso de reducción del error no es completamente lineal, sino que, fluctúa ligeramente entre iteraciones, sin embargo, tiene un déficit constante por lo que efectivamente mediante el criterio de datos y el optimizador **Adam** el error se reduce constantemente.

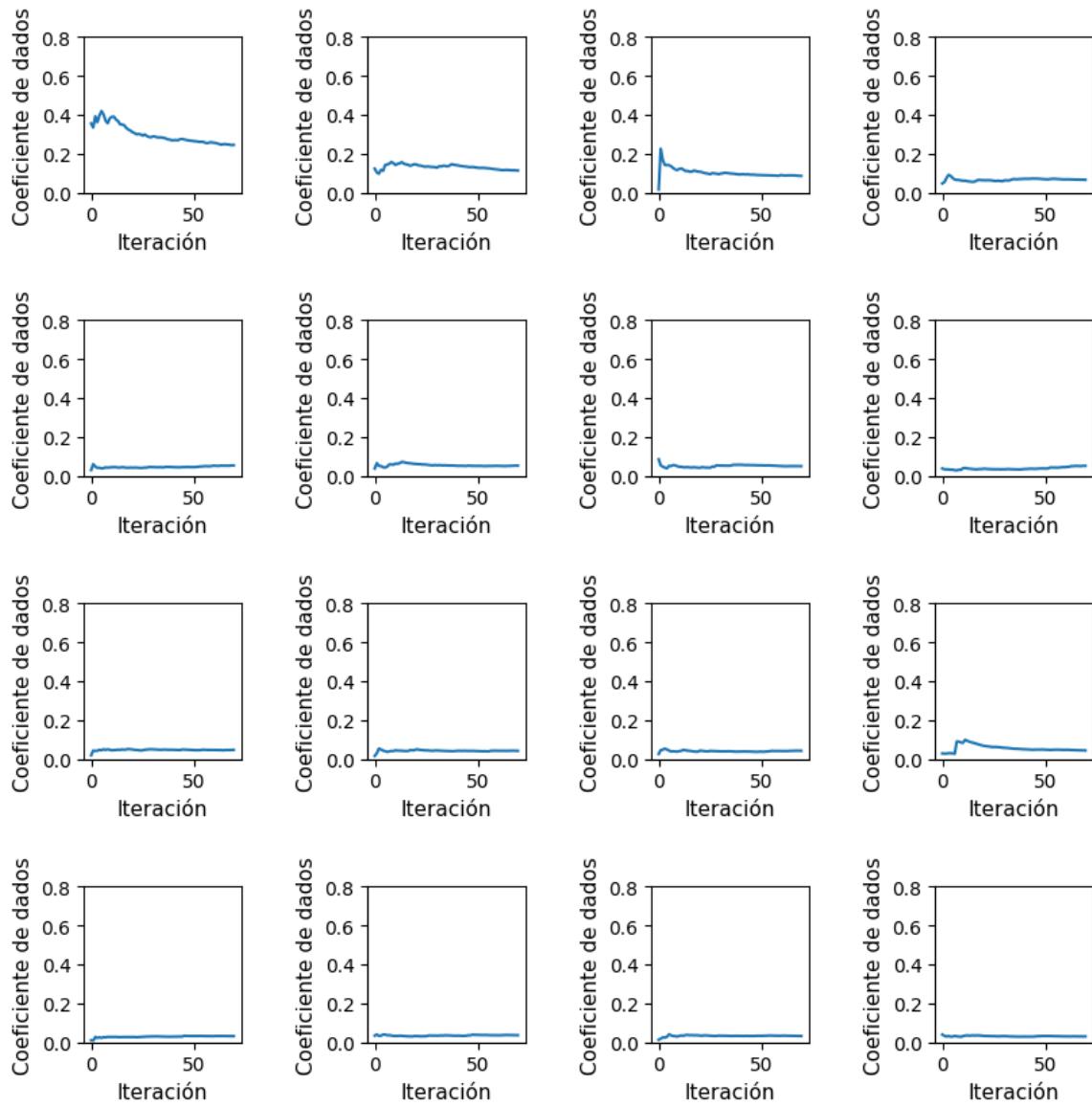


FIGURA 5.6: Curva de la pérdida según el coeficiente de datos en cada época.

5.3 CRITERIO DE JACCARD

En la sección anterior el coeficiente de dados es utilizado para evaluar la diferencia entre un mapa binario y un mapa probabilístico correspondientes a la máscara conocida y la máscara estimada durante el entrenamiento del modelo. Sin embargo para validar el resultado final es necesario utilizar una métrica muy similar que evalúe ahora la similitud y que penalice de forma estricta el error.

5.3.1 DISEÑO EXPERIMENTAL

Para determinar un porcentaje de similitud entre la máscara estimada con la máscara conocida (*ground truth*) podemos utilizar una métrica como la de la ecuación 5.2, el índice de Jaccard. Este criterio es relativamente similar al coeficiente de dados de la ecuación 5.1, mediante el solapado de ambas máscaras se promedia la coincidencia entre píxeles, sin embargo en el criterio de Jaccard *IoU* el error tiene mayor penalización, por lo que se podría decir que es una versión mas estricta de evaluación, lo cual es ideal para evaluar el modelo resultante tras cada época.

La métrica utilizada para evaluar el modelo es la del *índice de Jaccard (Intersection Over Union)* por sus siglas en inglés (*IoU*) [16] de la siguiente ecuación

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \quad (5.2)$$

donde A y B se pueden interpretar como la máscara real y la máscara estimada, y debido a que las imágenes consisten de píxeles, la ecuación anterior se puede

representar también como

$$J = \frac{1}{n} \sum_{c=1}^k W_c \sum_{i=1}^n \frac{y_i^c \hat{y}_i^c}{y_i^c + \hat{y}_i^c - y_i^c \hat{y}_i^c}, \quad (5.3)$$

donde y_i^c y \hat{y}_i^c son valores binarios y corresponden a la probabilidad de que el pixel i corresponda a la clase c de un número k de clases.

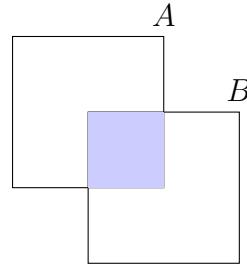


FIGURA 5.7: Representación de la región de sobreposición de píxeles en el índice de Jaccard.

La figura 5.7 representa, en la forma de un diagrama de Venn, la proporción de píxeles que coinciden (área sombreada) y el error (márgenes), es muy similar a la ecuación del coeficiente de datos sin embargo, tiene una mayor penalización en el error, siendo una mejor métrica post-entrenamiento.

5.3.2 RESULTADOS

Al diferencia de la evaluación mediante el coeficiente de datos, la cual se realiza durante el entrenamiento, la evaluación del criterio de Jaccard se realiza después del entrenamiento, específicamente en la fase de validación donde se vuelve a comparar el mapa de valores binarios real y el mapa probabilístico obtenido con el modelo al finalizar la época presente y dando una mayor penalización al error. En la figura 5.8 se puede observar la forma que la curva toma a través de las iteraciones y su evolución en cada época.

5.3.3 DISCUSIÓN

Una vez finalizado el experimento del criterio de Jaccard, se obtuvieron las gráficas correspondientes a cada época y sus iteraciones. En la figura 5.8 se puede observar en la curva el nivel de similitud, en un principio la precisión del modelo fluctúa al rededor del 75 % y conforme van avanzando las épocas, la curva se enderezza y la precisión alcanza un 96 % en la época numero seis, siendo este número la cantidad ideal de épocas de entrenamiento ya que éste es el límite de la precisión por debajo del sobreajuste. Si las épocas siguen continuando aún cuando se alcanzó la

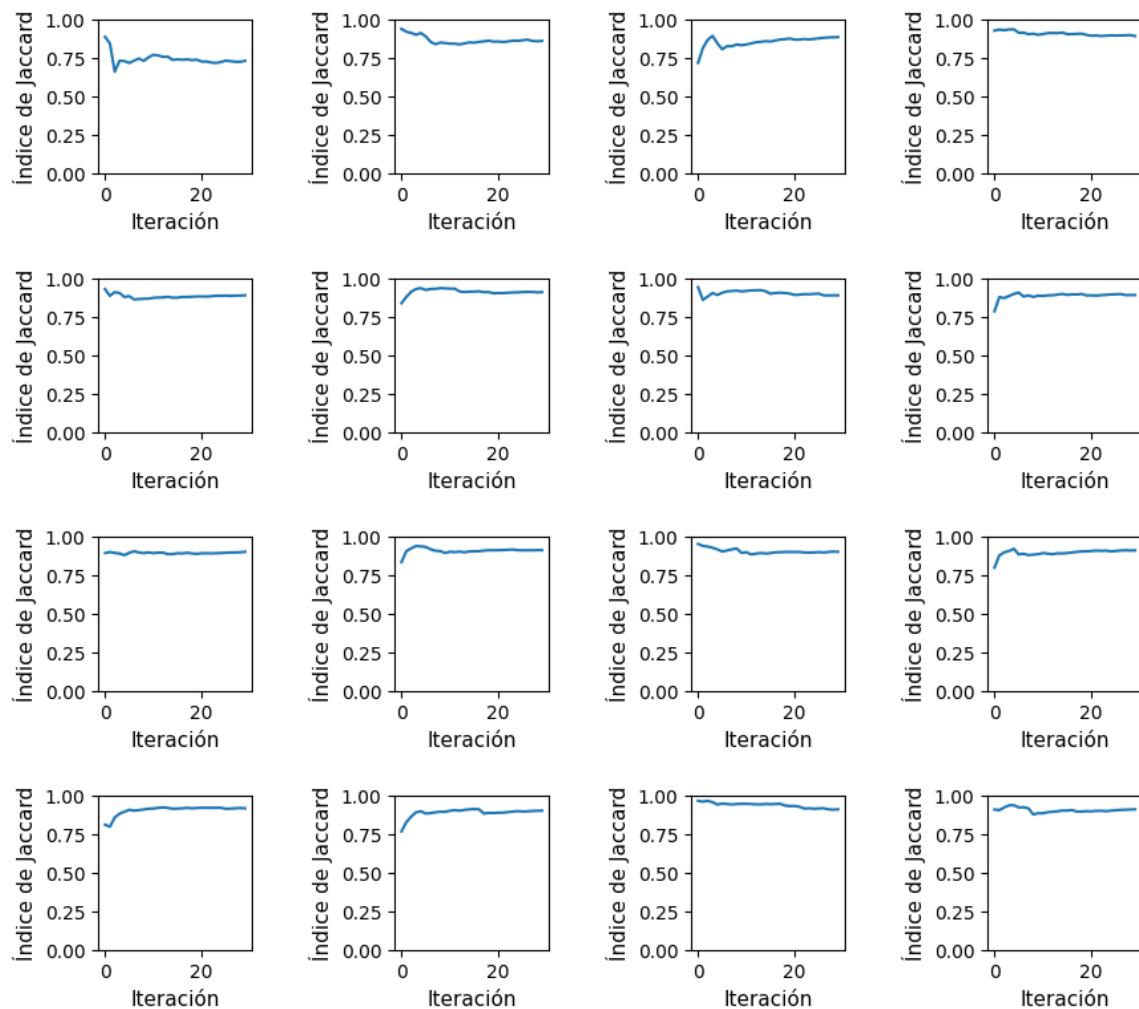


FIGURA 5.8: Curvas de la evaluación del criterio de Jaccard en cada época.

precisión máxima considerada viable (*fit*), se causaría un sobreajuste que impediría la correcta predicción de máscaras en imágenes más distintas del conjunto de datos de entrenamiento, esto sin embargo se puede arreglar mediante la aplicación de filtros aleatorios en el pre-procesado para aumentar la complejidad de la predicción y por consecuencia reducir el sobreajuste. Dichos filtros consisten en recortes aleatorios de la imagen, voltear vertical u horizontalmente imágenes al azar o un ajuste aleatorio de brillo y contraste.

5.4 RESUMEN DE EXPERIMENTOS

En el experimento de *estadísticas de los datos de entrada*, se realizó una serie de histogramas de un conjunto de imágenes para visualizar la distribución de la intensidad de los píxeles, también se aplicó un filtro de umbral para comparar la distribución antes y después. Se determinó que con un filtrado de umbral a un valor de 50 % fue suficiente para agrupar la intensidad de píxeles en dos columnas principales sin perder la información relevante de la imagen.

En el experimento de *evaluación del aprendizaje*, se realizó mediante el coeficiente de dados un registro del error en el modelo al paso de las épocas y se comprobó que en efecto, el error tiende a reducirse conforme las épocas avanzan. También se determinó que después de 6 épocas de entrenamiento el error se suele reducir casi totalmente, siendo el resto de épocas no tan relevantes para el aprendizaje.

En el experimento de *criterio de Jaccard*, se aplicó una metodología similar a la del coeficiente de dados, sin embargo, ésta fue aplicada post-entrenamiento en la parte de validación y el criterio de Jaccard penaliza aún más el error. Las gráficas mostradas en la figura 5.8 muestran que tan acertada es la máscara obtenida por el modelo en comparación con la máscara real. También se determinó que el sobreajuste

puede ser un problema en el conjunto de datos utilizado debido a la similitud entre las imágenes, por lo tanto se agregaron transformaciones aleatorias para compensar el sobreajuste.

CAPÍTULO 6

CONCLUSIONES

En este documento fue descrito el proceso de la implementación de segmentación semántica, mediante el cual es posible clasificar las regiones dentro de una imagen que corresponden a una categoría específica, siendo aquí el caso del melanoma de piel, se implementó una arquitectura de red neuronal profunda que consiste en filtros de convolución asociados a un peso sináptico el cual se ajusta de acuerdo a los criterios de evaluación, creando una linea de procesamiento que transforma la imagen entrante en un mapa probabilístico correspondiente a las regiones que la componen.

La relevancia del presente trabajo de tesis se fundamenta en la naturaleza de la tecnología que fue utilizada para la realización de las tareas de segmentación, aunque es posible realizar la misma tarea mediante una secuencia lineal de transformaciones adecuada al tipo de imágenes dermatológicas, la precisión de dicho programa no mejoraría de forma automática a menos que de forma activa se ajusten los parámetros involucrados en la transformación, siendo el número de parámetros involucrados una cantidad masiva. Mediante el uso de redes neuronales el proceso de ajuste de los parámetros se realiza de forma automática mediante los criterios mencionados en el capítulo 5, aún cuando es necesario ajustar manualmente parámetros relacionados

con el entrenamiento del modelo y de la evaluación del mismo, la cantidad de ajustes necesarios no varía en función de la complejidad de las imágenes a procesar, sino qué, va en función de la arquitectura seleccionada como recipiente para el modelo.

La implementación de la arquitectura de red neuronal convolucional fue llevada a cabo gracias a la librería de Yakubovskiy [23], cual está desarrollada en el lenguaje de **Python** y contiene las principales arquitecturas de redes neuronales ya adaptadas para la realización de tareas de segmentación semántica, el resto de la implementación consistió principalmente en establecer las líneas de pre-procesamiento de imágenes para cumplir con los requisitos de admisión para ingresar a la arquitectura de pirámide de características, obteniendo de forma satisfactoria la segmentación semántica.

6.1 DISCUSIÓN

El objetivo fué clasificar en las imágenes dermatológicas las regiones que corresponden a una de las dos posibles categorías: piel sana, melanoma. Mediante la implementación del *software* y el entrenamiento de modelos con líneas de pre-procesamiento fue posible obtener el resultado esperado con una precisión aceptable de acuerdo al coeficiente de datos y al criterio de Jaccard.

Los experimentos demuestran que la calibración se lleva a cabo conforme se repiten épocas de entrenamiento siendo aproximadamente seis épocas el valor ideal de repeticiones para obtener la precisión máxima viable.

El software desarrollado mostró resultados eficientes en las tareas de segmentación semántica con una precisión máxima del 96 %. Mediante la definición de la linea de pre-procesamiento fue posible ajustar distintos parámetros de la ejecución,

una de las principales dificultades en el entrenamiento del modelo fue la adaptación de las dimensiones de las imágenes a las dimensiones requeridas por la arquitectura, sin embargo, esto fue resuelto mediante la aplicación de una interpolación utilizando la librería de `OpenCV` y las transformación a tensor mediante `Albumentations`.

El entrenamiento en sí requiere de grandes recursos de memoria, por lo que la resolución máxima de predicción está directamente limitada por la capacidad del hardware de llevar a cabo el entrenamiento, sin embargo, una vez obtenido el modelo de segmentación semántica, independientemente de su complejidad, utilizarlo no requirió tantos recursos como al entrenarlo, siendo aún viable la utilización de este software por una gran cantidad de dispositivos en el post-entrenamiento.

Al final se obtuvo un software con la capacidad de tanto generar nuevos modelos y modificar sus parámetros de entrenamiento, como de utilizar dichos modelos para generar máscaras de segmentación semántica en imágenes dermatológicas. Así mismo, el software desarrollado en el presente trabajo de tesis provee las transformaciones requeridas para adaptar las dimensiones de los datos, independientemente de su resolución y de la cantidad de canales de color con los que cuente, dando como resultado una máscara de segmentación semántica con resolución estandarizada.

6.2 TRABAJO A FUTURO

La principal razón por la que muchos científicos estudian y desarrollan la técnica de segmentación semántica es porque mediante ésta técnica se obtiene información muy relevante sobre el objeto que se está estudiando, por ejemplo, muchos autores enfocan las redes neuronales para obtener información de medio como distinguir mediante una cámara a las personas, a los objetos, a las señales de tráfico. La razón es porque teniendo esta información es posible realizar algo como un vehículo de

conducción autónoma, y ahí es donde reside la clave del paso siguiente. Tomando únicamente el concepto de la segmentación semántica, es posible pensar en muchas posibilidades en las que pueda ser aplicada, principalmente en el área médica. Por ejemplo, podría ser aplicado para segmentar partes del cuerpo de forma externa e interna, incluso podría desarrollarse un sistema de visión con un modelo pre-entrenado para asistir a personas con ceguera parcial a detectar obstáculos en el camino.

Hablando más específicamente sobre la aplicación desarrollada en el presente trabajo de tesis, sería ideal probar con una linea de pre-procesamiento más sofisticada que permita una predicción más profunda de las categorías presentes en una imagen a una resolución mayor. Debido a las limitaciones de hardware, resulta muy complicado el entrenamiento de modelos con secuencias de transformaciones muy extensas o complejas debido a que dicha secuencia aplicaría a cada muestra del entrenamiento.

APÉNDICE A

REPOSITORIO DE GITHUB

En este apéndice se anexa la liga para acceder al código fuente completo y a los datos obtenidos como resultado de los experimentos. Para acceder al repositorio escanea el siguiente **código QR** o da clic directamente en la liga. Las librerías requeridas para la ejecución del código están descritas en el documento `requirements.txt`.



<https://github.com/Btrox148/semantic-segmentation-skin>

BIBLIOGRAFÍA

- [1] BACHMAN, D. (2007), *Advanced calculus demystified*, McGrawHill. ISBN 978-0071481212.
- [2] BADRINARAYANAN, V., A. KENDALL y R. CIPOLLA (2015), «SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation», *CoRR*, [abs/1511.00561](https://arxiv.org/abs/1511.00561), 1511.00561 [cs.CV].
- [3] BEIJING ACADEMY OF ARTIFICIAL INTELLIGENCE (2020), «Suggested Notation for Machine Learning», <https://github.com/Mayuyu/suggested-notation-for-machine-learning>.
- [4] BONI, R., C. SCHUSTER, B. NEHRHOFF y G. BURG (2002), «Epidemiology of skin cancer», *Neuroendocrinology Letters*, **23**, págs. 48–51. pmid: 12163848.
- [5] CHEN, L., G. PAPANDREOU, I. KOKKINOS, K. MURPHY y A. L. YUILLE (2016), «DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs», *CoRR*, [abs/1606.00915](https://arxiv.org/abs/1606.00915), 1606.00915 [cs.CV].
- [6] CODELLA, N. C. F., V. ROTEMBERG, P. TSCHANDL, M. E. CELEBI, S. W. DUSZA, D. GUTMAN, B. HELBA, A. KALLOO, K. LIOPYRIS, M. A. MARCHETTI, H. KITTLER y A. HALPERN (2019), «Skin Lesion Analysis Toward

- Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)», *CoRR*, **abs/1902.03368**, 1902.03368 [cs.CV].
- [7] GOYAL, M. y M. H. YAP (2017), «Multi-class Semantic Segmentation of Skin Lesions via Fully Convolutional Networks», *CoRR*, **abs/1711.10449**, 1711.10449 [cs.CV].
- [8] JAIN, S., V. JAGTAP y N. PISE (2015), «Computer Aided Melanoma Skin Cancer Detection Using Image Processing», *Procedia Computer Science*, **48**, págs. 735–740. doi: 10.1016/j.procs.2015.04.209.
- [9] KADAMPUR, M. A. y S. AL RIYAAE (2020), «Skin cancer detection: Applying a deep learning based model driven architecture in the cloud for classifying dermal cell images», *Informatics in Medicine Unlocked*, **18**, pág. 100282. doi: 10.1016/j.imu.2019.100282.
- [10] KRONER, A., M. SENDEN, K. DRIESSENS y R. GOEBEL (2020), «Contextual encoder-decoder network for visual saliency prediction», *Neural Networks*, **129**, págs. 261–270. doi: 10.1016/j.neunet.2020.05.004.
- [11] LIN, T., P. DOLLÁR, R. B. GIRSHICK, K. HE, B. HARIHARAN y S. J. BELONGIE (2016), «Feature Pyramid Networks for Object Detection», *CoRR*, **abs/1612.03144**, 1612.03144.
- [12] LUC, P., C. COUPRIE, S. CHINTALA y J. VERBEEK (2016), «Semantic Segmentation using Adversarial Networks», *CoRR*, **abs/1611.08408**, 1611.08408.
- [13] PALOMARES, F. G., J. A. M. SERRÁ y E. A. MARTÍNEZ (2016), «Aplicación de la convolución de matrices al filtrado de imágenes», *Modelling in Science Education and Learning*, **9**(1), págs. 97–108.

- [14] RITTIE, L. y G. J. FISHER (2015), «Natural and Sun-Induced Aging of Human Skin», *Cold Spring Harb Perspect Med*, doi: 10.1101/cshperspect.a015370.
- [15] RONNEBERGER, O., P. FISCHER y T. BROX (2015), «U-Net: Convolutional Networks for Biomedical Image Segmentation», *CoRR*, **abs/1505.04597**, 1505.04597 [cs.CV].
- [16] SEFERBEKOV, S. S., V. IGLOVIKOV, A. BUSLAEV y A. SHVETS (2018), «Feature Pyramid Network for Multi-Class Land Segmentation.», en *CVPR Workshops*, págs. 272–275.
- [17] SHELHAMER, E., J. LONG y T. DARRELL (2016), «Fully Convolutional Networks for Semantic Segmentation», *CoRR*, **abs/1605.06211**[cs.CV], 1605.06211 [cs.CV].
- [18] TEICHMANN, M., M. WEBER, J. M. ZÖLLNER, R. CIPOLLA y R. URTASUN (2016), «MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving», *CoRR*, **abs/1612.07695**, 1612.07695 [cs.CV].
- [19] THE AMERICAN CANCER SOCIETY MEDICAL AND EDITORIAL CONTENT TEAM (2019), «What Are Basal and Squamous Cell Skin Cancers?», URL <https://www.cancer.org/cancer/basal-and-squamous-cell-skin-cancer/about/what-is-basal-and-squamous-cell.html>.
- [20] TODOROVA, K. y A. MANDINOVA (2020), «Novel approaches for managing aged skin and nonmelanoma skin cancer», *Adv. Drug Deliv. Rev.* doi: 10.1016/j.addr.2020.06.004.
- [21] TORTORA G. AND GRABOWSKI S. (1993), «Principles of Anatomy and Physiology», URL <https://www.clinimed.co.uk/wound-care/wound-essentials/>

structure-and-function-of-the-skin#:~:text=The%20skin%20consists%
20of%20two,the%20functions%20of%20the%20skin.

- [22] WU, H., J. ZHANG, K. HUANG, K. LIANG y Y. YU (2019), «FastFCN: Rethinking dilated convolution in the backbone for semantic segmentation», *arXiv preprint arXiv: 1903.11816[cs.CV]*.
- [23] YAKUBOVSKIY, P. (2020), «Segmentation Models Pytorch», https://github.com/qubvel/segmentation_models.pytorch.
- [24] ZHOU, J., B. HUANG, Z. YAN y J.-C. G. BÜNZLI (2019), «Emerging role of machine learning in light-matter interaction», *Light: Science & Applications*, 8(1), págs. 1–7.

RESUMEN AUTOBIOGRÁFICO

Mario Alberto Flores Hernández

Candidato para obtener el grado de
Ingeniero en Mecatrónica

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

**DETECCIÓN DE MELANOMA DE PIEL MEDIANTE SEGMENTACIÓN
SEMÁNTICA**

Nací el 4 de junio de 1997 en la ciudad de Monterrey, Nuevo León. Hijo del Sr. Mario Alberto Flores Rosales y la Sra. Patricia Hernández Romero. Comencé mis estudios de Ingeniería en Mecatrónica en agosto de 2014 en la Universidad Autónoma de Nuevo León, en marzo de 2019 llevé a cabo el diplomado de Innovación Biomédica.